Java™ Programming Language

SL-275

Copyright 2000 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303, U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun Logo, Java, JavaOS, JavaSoft, JVM, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

Netscape Navigator is a trademark of Netscape Communications Corporation.

UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

The OPEN LOOK and Sun Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g) (2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015 (b)(6/95) and DFAR 227.7202-3(a).

X Window System is a product of the X Consortium, Inc.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Preface

About This Course

Course Goals

This course provides you with knowledge and skills to:

- Program and run advanced Java™ applications and applets
- Help you prepare for the Sun Microsystems™ Certified Java Programmer and Developer examinations

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Course Overview

This course covers the following areas:

- Syntax of the Java programming language
- Object-oriented concepts as they apply to the Java programming language
- Graphical user interface (GUI) programming
- Applet creation
- Multithreading
- Networking

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Course Map

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, April 2000, Revision D

Module-by-Module Overview

- Module 1 – "Getting Started"
- Module 2 – "Object-Oriented Programming"
- Module 3 – "Identifiers, Keywords, and Types"
- Module 4 – "Expressions and Flow Control"
- Module 5 – "Arrays"
- Module 6 – "Inheritance"
- Module 7 – "Advanced Class Features"
- Module 8 – "Exceptions"
- Module 9 – "Text-Based Applications"

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Module-by-Module Overview

- Module 10 – "Building Java GUIs"
- Module 11 – "GUI Event Handling"
- Module 12 – "Introduction to Java Applets"
- Module 13 – "GUI-Based Applications"
- Module 14 – "Threads"
- Module 15 – "Advanced I/O Streams"
- Module 16 – "Networking"

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Course Objectives

- Describe key language features
- Compile and run a Java application
- Understand and use the online hypertext Java technology documentation
- Describe language syntactic elements and constructs
- Understand the object-oriented paradigm
- Use object-oriented features of Java
- Understand and use exceptions
- Understand and use the Collections API
- Read and write to files

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, April 2000, Revision D

Course Objectives

- Develop a graphical user interface
- Describe the Java technology Abstract Window Toolkit (AWT)
- Develop a program to take input from a GUI
- Understand event handling
- Develop Java applets
- Understand and use the `java.io` package
- Understand the basics of multithreading
- Develop multithreaded Java applications and applets
- Develop Java client and server programs using Transmission Control Protocol/Internet Protocol

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, April 2000, Revision D

Skills Gained by Module

Skills Gained	Module															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Describe key language features	■															
Compile and run a Java application	■															
Understand and use the online hypertext Java technology documentation		▒							■							
Describe language syntactic elements and constructs		■	■	■	■	■										
Understand the object-oriented paradigm		■	▒			■	■									
Use object-oriented features of Java		■	▒			■	■									
Understand and use exceptions								■								
Understand and use the Collections API									■							
Read and write to files									■							
Develop a GUI										■	■	▒	■			
Describe the Java technology platform's Abstract Window Toolkit										■	■		■			
Create a program to take input from a graphical user interface											■		■			
Understand event handling											■					
Develop Java applets												■	▒			
Understand and use the `java.io` package														■		
Understand the basics of multithreading															■	
Develop multithreaded Java applications and applets															■	
Develop Java client and server programs using TCP/IP																■

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services, April 2000, Revision D

Guidelines for Module Pacing

Module	Day 1	Day 2	Day 3	Day 4	Day 5
About This Course	A.M.				
Module 1 – "Getting Started"	A.M.				
Module 2 – "Object-Oriented Programming"	P.M.				
Module 3 – "Identifiers, Keywords, and Types"	P.M.				
Module 4 – "Expressions and Flow Control"		A.M.			
Module 5 – "Arrays"		A.M.			
Module 6 – "Inheritance"		P.M.			
Module 7 – "Advanced Class Features"			A.M.		
Module 8 – "Exceptions"			A.M.		
Module 9 – "Text-Based Applications"			P.M.		
Module 10 – "Building Java GUIs"				A.M.	
Module 11 – "GUI Event Handling"				A.M.	
Module 12 – "Introduction to Java Applets"				P.M.	
Module 13 – "GUI-Based Applications"				P.M.	
Module 14 – "Threads"					A.M.
Module 15 – "Advanced I/O Streams"					P.M.
Module 16 – "Networking"					P.M.

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Topics Not Covered

- General programming concepts. This is not a course for people who have never programmed before.
- General object-oriented concepts.

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

How Prepared Are You?

Before attending this course, you should have completed:

- SL-110: *Java Programming For Non-Programmers*

or have:

- Created compiled programs with C or C++
- Created and edited text files using a text editor
- Used a World Wide Web (WWW) browser, such as Netscape Navigator™

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Introductions

- Name
- Company affiliation
- Title, function, and job responsibility
- Programming experience
- Reasons for enrolling in this course
- Expectations for this course

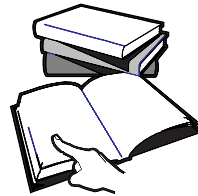
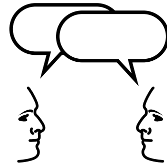
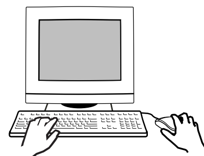
Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

How to Use Course Materials

- Course Map
- Relevance
- Overhead Image
- Lecture
- Exercise
- Check Your Progress
- Think Beyond

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Course Icons

- Reference 
- Discussion 
- Exercise 

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Typographical Conventions

- `Courier` – Commands, files and directories, and on-screen computer output
- **`Courier bold`** – Input you type
- *`Courier italic`* – Variables and command-line placeholders
- *Palatino italics* – Book titles, new words or terms, and words that are emphasized

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Module 1

Getting Started

Objectives

- Describe key features of Java technology
- Define the terms *class* and *applications*
- Write, compile, and run a simple Java application
- Describe the Java virtual machine's (JVM™) function
- Describe how garbage collection works
- List the three tasks performed by the Java platform that handle code security

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Relevance

- Is the Java programming language a complete language or is it just useful for writing programs for the Web?
- Why is another programming language needed?
- How does the Java technology platform improve on other language platforms?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

What Is the Java Technology?

- Java technology is:
  - A programming language
  - A development environment
  - An application environment
  - A deployment environment
- It is similar in syntax to C++; similar in semantics to SmallTalk
- It is used for developing both *applets* and *applications*

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Primary Goals of the Java Technology

- Provides an easy-to-use language by:
  - Avoiding the pitfalls of other languages
  - Being object-oriented
  - Enabling users to create streamlined and clear code

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Primary Goals of the Java Technology

- Provides an interpreted environment for:
  - Improved speed of development
  - Code portability
- Enables users to run more than one thread of activity
- Loads classes dynamically, that is, at the time they are actually needed
- Supports dynamically changing programs during runtime by loading classes from disparate sources
- Furnishes better security

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Primary Goals of the Java Technology

The following features fulfill these goals:

- The Java virtual machine (JVM)
- Garbage collection
- Code security

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

A Basic Java Application

TestGreeting.java

```
1   //
2   // Sample "Hello World" application
3   //
4   public class TestGreeting{
5     public static void main (String args[]) {
6       Greeting hello = new Greeting("Hello");
7       hello.greet("World");
8     }
9   }
```

Greeting.java

```
1   // The Greeting class declaration
2   public class Greeting {
3     private String salutation;
4
5     public Greeting(String s) {
6       salutation = s;
7     }
8
9     public void greet(String whom) {
10      System.out.println(salutation + " " + whom);
11    }
12  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Compiling and Running `TestGreeting`

- Compiling `TestGreeting.java`

```
javac TestGreeting.java
```

- `Greeting.java` is compiled automatically
- Running an application

```
java TestGreeting
```

- Locating common compile and runtime errors

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Compile-Time Errors

- `javac: Command not found`
- `Greeting.java:10: Method printl(java.lang.String) not found in class java.io.PrintStream.`
  ```
  System.out.printl^(salutation + " " + whom);
  ```
- `TestGreet.java:4: Public class TestGreeting must be defined in a file called "TestGreeting.java".`

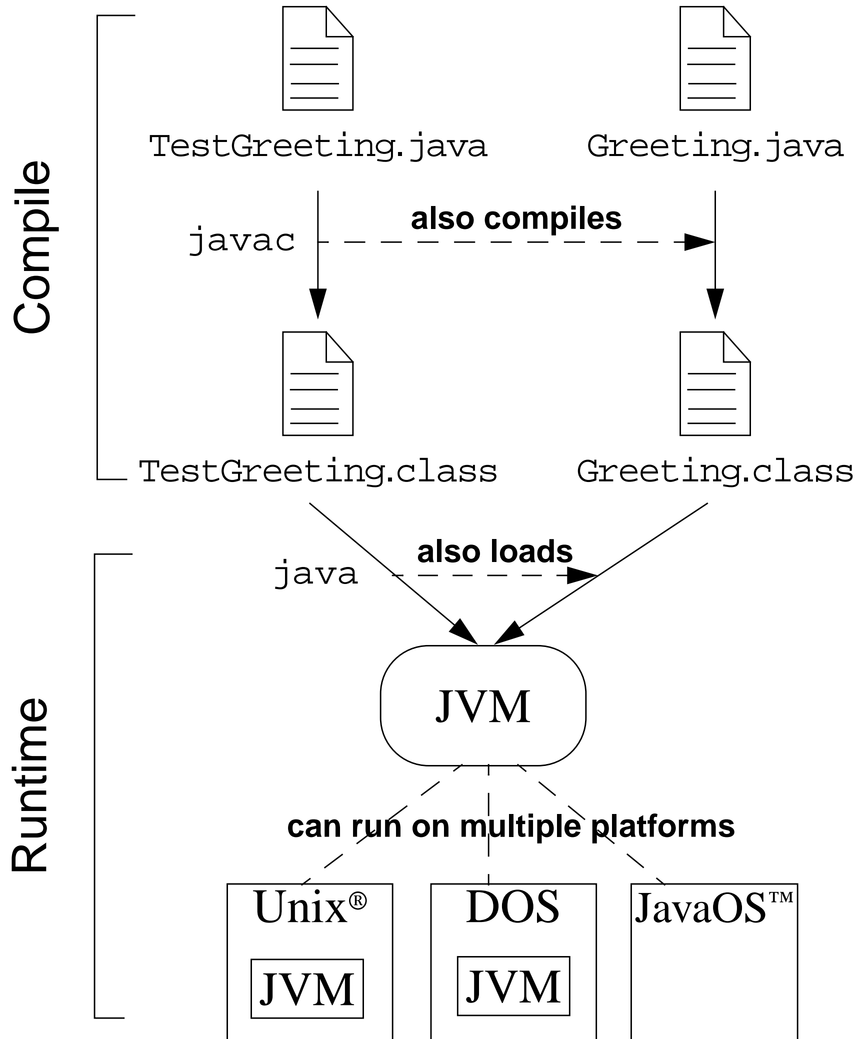
Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Runtime Errors

- `Can’t find class TestGreeting`
- `Exception in thread "main" java.lang.NoSuchMethodError: main`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Java Runtime Environment



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The Java Virtual Machine

- Provides hardware platform specifications
- Reads compiled byte codes that are platform independent
- Is implemented as software or hardware
- Is implemented in a Java technology development tool or a Web browser

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The Java Virtual Machine

- JVM provides definitions for the:
  - Instruction set (central processing unit [CPU])
  - Register set
  - Class file format
  - Stack
  - Garbage-collected heap
  - Memory area

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The Java Virtual Machine

- The majority of type checking is done when the code is compiled.
- Every Sun Microsystems approved implementation of the JVM must be able to run any compliant class file.

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

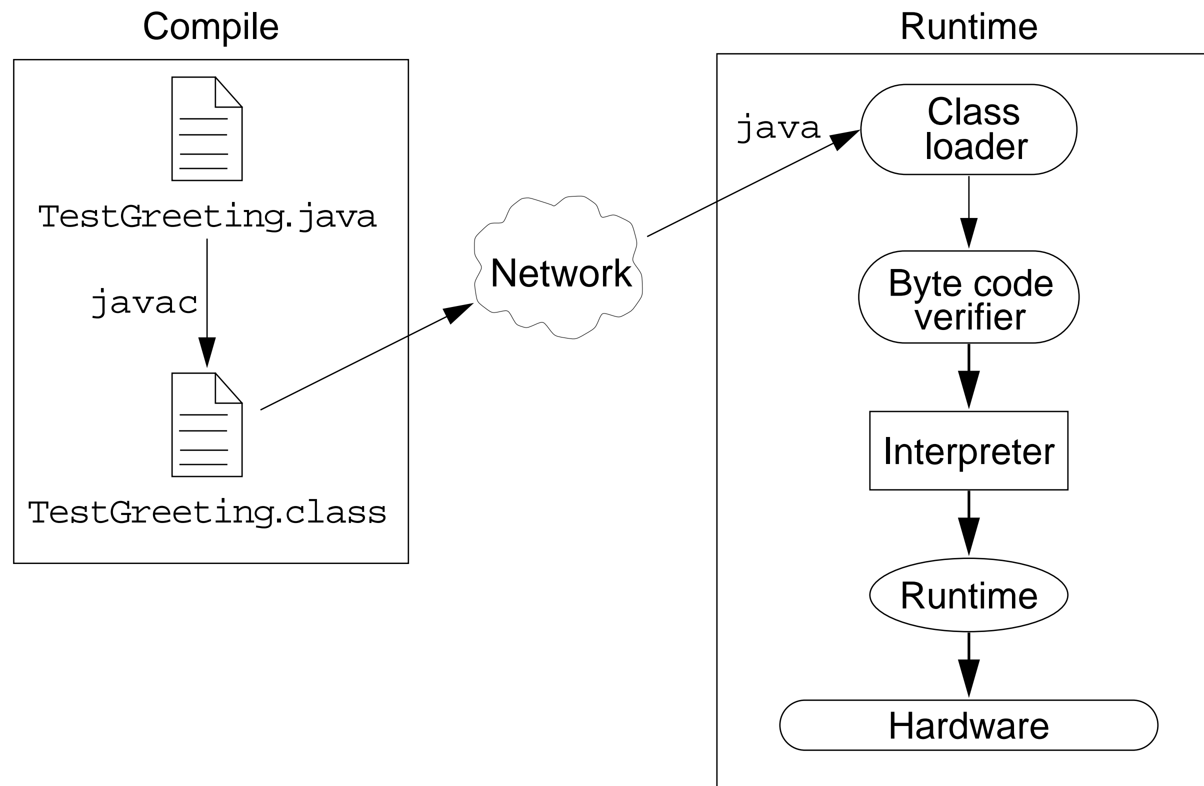
Garbage Collection

- Allocated memory that is no longer needed should be deallocated
- In other languages, deallocation is the programmer’s responsibility
- The Java programming language provides a system-level thread to track memory allocation
- Garbage collection:
  - Checks for and frees memory no longer needed
  - Is done automatically
  - Can vary dramatically across JVM implementations

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

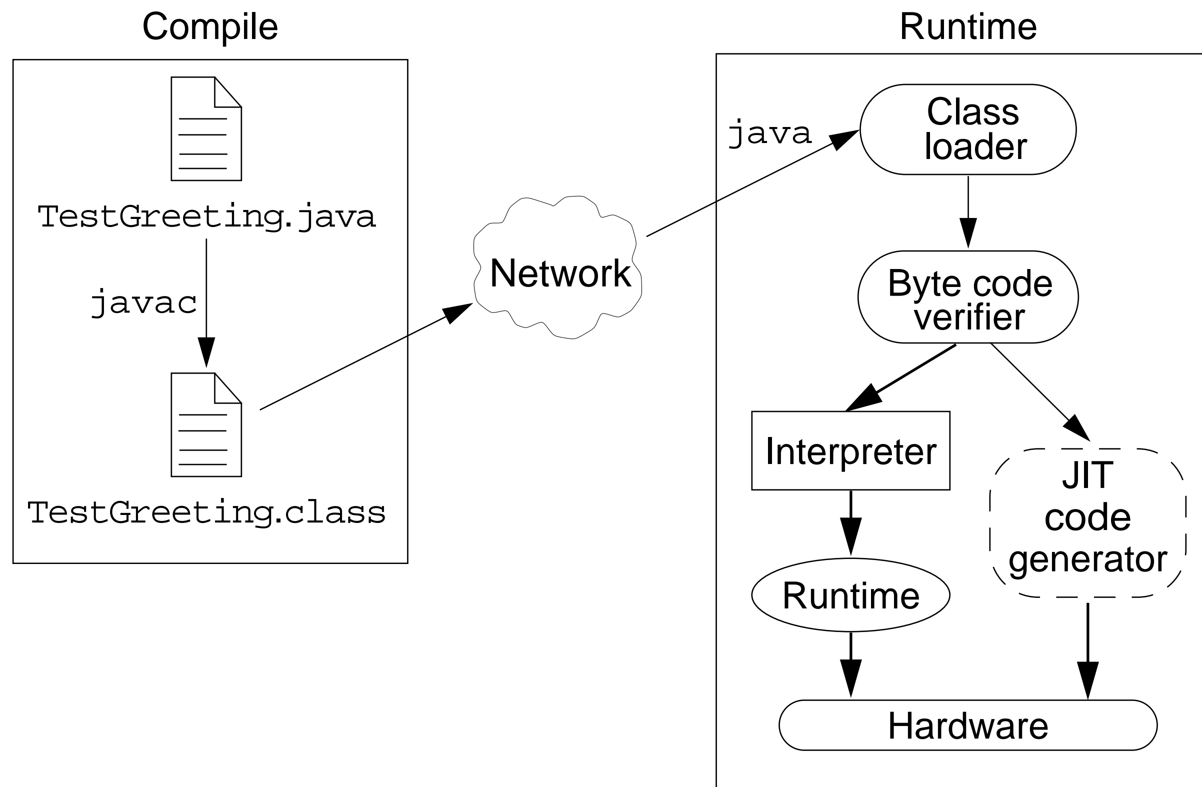
Code Security

The Java application environment performs as follows:



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Just-In-Time Code Generator



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Java Runtime Environment

- Performs three main tasks:
  - Loads code
  - Verifies code
  - Executes code

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Class Loader

- Loads all classes necessary for the execution of a program
- Maintains classes of the local file system in separate "namespaces"
- Prevents spoofing

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Bytecode Verifier

Ensures that:

- The code adheres to the JVM specification
- The code does not violate system integrity
- The code causes no operand stack overflows or underflows
- The parameter types for all operational code are correct
- No illegal data conversions (the conversion of integers to pointers) have occurred

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Performing Basic Java Tasks

- Exercise objectives:
  - Write, compile, and run a simple application
- Tasks:
  - Explore compile and runtime errors
  - Create a Java application

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Describe key features of Java technology
- Define the terms *class* and *applications*
- Write, compile, and run a simple Java application
- Describe the Java virtual machine's (JVM) function
- Describe how garbage collection works
- List the three tasks performed by the Java platform that handle code security

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Think Beyond

- How can you benefit from using this programming language in your work environment?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Module 2

Object-Oriented Programming

Objectives

- Define modeling concepts: *abstraction*, *encapsulation*, and *packages*
- Discuss why Java application code is reusable
- Define *class*, *member*, *attribute*, *method*, *constructor*, and *package*
- Use the access modifiers `private` and `public` as appropriate for the guidelines of encapsulation
- Invoke a method on a particular object

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Objectives

- In a Java program, identify the following:
  - The `package` statement
  - The `import` statements
  - Classes, methods, and attributes
  - Constructors
- Use the Java technology application programming interface (API) online documentation

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Relevance

- What is your understanding of software analysis and design?
- What is your understanding of design and code reuse?
- What features does the Java programming language possess to make it an object-oriented language?
- What does the term *object-oriented* really mean?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Software Engineering

Toolkits / Frameworks / Object APIs (90's - up)				
Java 2 SDK	AWT / Swing	Jini	Java Beans	JDBC

Object-Oriented Languages (80's - up)					
SELF	Smalltalk	Common Lisp Object System	Effiel	C++	Java

Libraries / Functional APIs (60's - early 80's)				
NASTRAN	TCP/IP	ISAM	X-Windows	OpenLook

High-Level Languages (50's - up)				Operating Systems (60's - up)			
Fortran	LISP	C	COBOL	OS/360	UNIX	MacOS	MS-Windows

Machine Code (late 40's - up)

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Analysis and Design

- Analysis describes *what* the system needs to do
  - Modeling the real-world: actors and activities, objects and behaviors
- Design describes *how* the system does it
  - Modeling the relationships and interactions between objects and actors in the system
  - Finding useful abstractions to help simplify the problem or solution

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Abstraction

- Functions – Write an algorithm once to be used in many situations
- Objects – Group a related set of attributes and behaviors into a class
- Frameworks and APIs – Large groups of objects that support a complex activity
  - Frameworks can be used "as is" or be modified to extend the basic behavior

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Classes as Blueprints for Objects

- In manufacturing, a blueprint is a description of a device from which many physical devices are constructed
- In software, a class is a description of an object:
  - A class describes the data that each object includes
  - A class describes the behaviors that each object exhibits
- In Java, classes support three key features of OOP:
  - encapsulation
  - inheritance
  - polymorphism

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Declaring Java Classes

- Basic syntax of a Java class:

```
<class_declaration> ::=
  <modifier> class <name> {
    <attribute_declaration>*
    <constructor_declaration>*
    <method_declaration>*
  }
```

- Example:

```
public class Vehicle {
  private double maxLoad;
  public void setMaxLoad(double value) {
    maxLoad = value;
  }
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Declaring Attributes

- Basic syntax of an attribute:

```
<attribute_declaration> ::=
  <modifier> <type> <name> [= <default_value>];

<type> ::= byte | short | int | long | char
           float | double | boolean | <class>
```

- Examples:

```
public class Foo {
  public  int     x;
  private float   y = 10000.0F;
  private String name = "Fred Flintstone";
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Declaring Methods

- Basic syntax of a method:

```
<method_declaration> ::=
   <modifier> <return_type> <name> (<parameter>*) {
      <statement>*
   }
<parameter> ::= <parameter_type> <parameter_name>,
```

- Examples:

```
public class Thing {
   private int x;
   public int getX() {
      return x;
   }
   public void setX(int new_x) {
      x = new_x;
   }
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Accessing Object Members

- The "dot" notation: *<object>.<member>*
- This is used to access object members including attributes and methods
- Examples:

```
thing1.setX(47);
thing1.x = 47;   // only permissible if x is public
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Information Hiding

The Problem:

MyDate
+day : int +month : int +year : int

Client code has direct access to internal data:

```
MyDate d = new MyDate()

d.day = 32;
// invalid day

d.month = 2; d.day = 30;
// plausible but wrong

d.day = d.day + 1;
// no check for wrap around
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Information Hiding

The Solution:

MyDate
-day : int -month : int -year : int
+getDay() : int +getMonth() : int +getYear() : int +setDay(d : int) : boolean +setMonth(m : int) : boolean +setYear(y : int) : boolean -validDay(d: int) : boolean

verify days in month

Client code must use setters/getters to access internal data:

```
MyDate d = new MyDate()

d.setDay(32);
// invalid day, returns false

d.setMonth(2);
d.setDay(30);
// plausible but wrong, setDay returns false

d.setDay(d.getDay() + 1);
// this will return false if wrap around
// needs to occur
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Encapsulation

- Hides the implementation details of a class
- Forces the user to use an interface to access data
- Makes the code more maintainable

MyDate
-date : long
+getDay() : int +getMonth() : int +getYear() : int +setDay(d : int) : boolean +setMonth(m : int) : boolean +setYear(y : int) : boolean -validDay(d: int) : boolean

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Declaring Constructors

- Basic syntax of a constructor:

```
<constructor_declaration> ::=
   <modifier> <class_name> (<parameter>*) {
      <statement>*
   }
```

- Examples:

```
public class Thing {
   private int x;
   public Thing() {
      x = 47;
   }
   public Thing(int new_x) {
      x = new_x;
   }
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Declaring Constructors

```
public class Thing {
  private int x;
  public Thing() {
    x = 47;
  }
  public Thing(int new_x) {
    x = new_x;
  }
  public int getX() {
    return x;
  }
  public void setX(int new_x) {
    x = new_x;
  }
}
```

Example usage:

```
public class TestThing {
  public static void main(String[] args) {
    Thing  thing1 = new Thing();
    Thing  thing2 = new Thing(42);

    System.out.println("thing1.x is " + thing1.getX());
    System.out.println("thing2.x is " + thing2.getX());
  }
}
```

Output:

```
thing1.x is 47
thing2.x is 42
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The Default Constructor

- There is always at least one constructor in every class
- If the writer does not supply any constructors, the default constructor will be present automatically
  - The default constructor takes no arguments
  - The default constructor has no body
- Enables you to create object instances with `new` *`Xxx`*`()` without having to write a constructor

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Source File Layout

- Basic syntax of a Java source file:

```
<source_file> ::=
  [<package_declaration>]
  <import_declaration>*
  <class_declaration>+
```

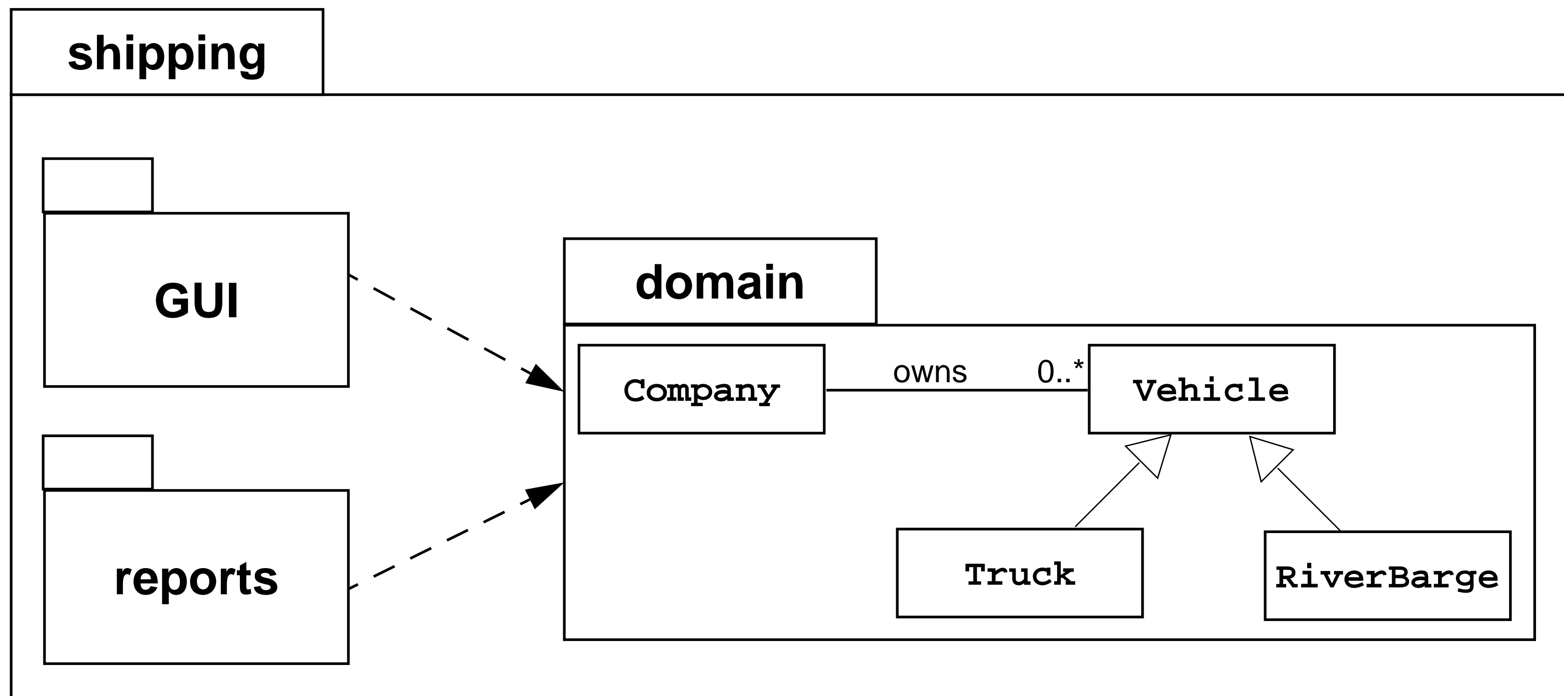
- Example, the `VehicleCapacityReport.java` file:

```
package shipping.reports.Web;

import shipping.domain.*;
import java.util.List;
import java.io.*;

public class VehicleCapacityReport {
  private List  vehicles;
  public void generateReport(Writer output) {...}
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Software Packages

- Packages help manage large software systems
- Packages can contain classes and sub-packages



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The package Statement

- Basic syntax of the package statement:

```
<package_declaration> ::=
  package <top_pkg_name>[.<sub_pkg_name>]*;
```

- Example:

```
package shipping.reports.Web;
```

- Specify the package declaration at the beginning of the source file
- Only one package declaration per source file
- If no package is declared, then the class "belongs" to the default package
- Package names must be hierarchical and separated by dots

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The `import` Statement

- Basic syntax of the package statement:

```
<import_declaration> ::=
  import <pkg_name>[.<sub_pkg_name>]*.<class_name | *>;
```

- Examples:

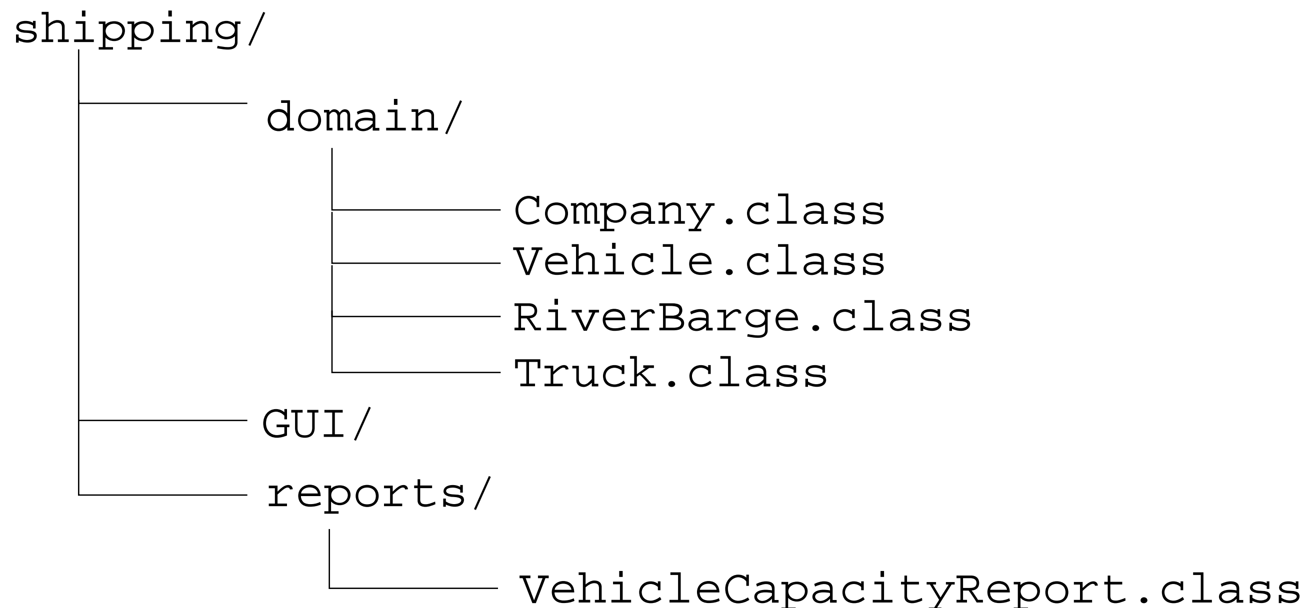
```
import shipping.domain.*;
import java.util.List;
import java.io.*;
```

- Precedes all class declarations
- Tells the compiler where to find classes to use

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

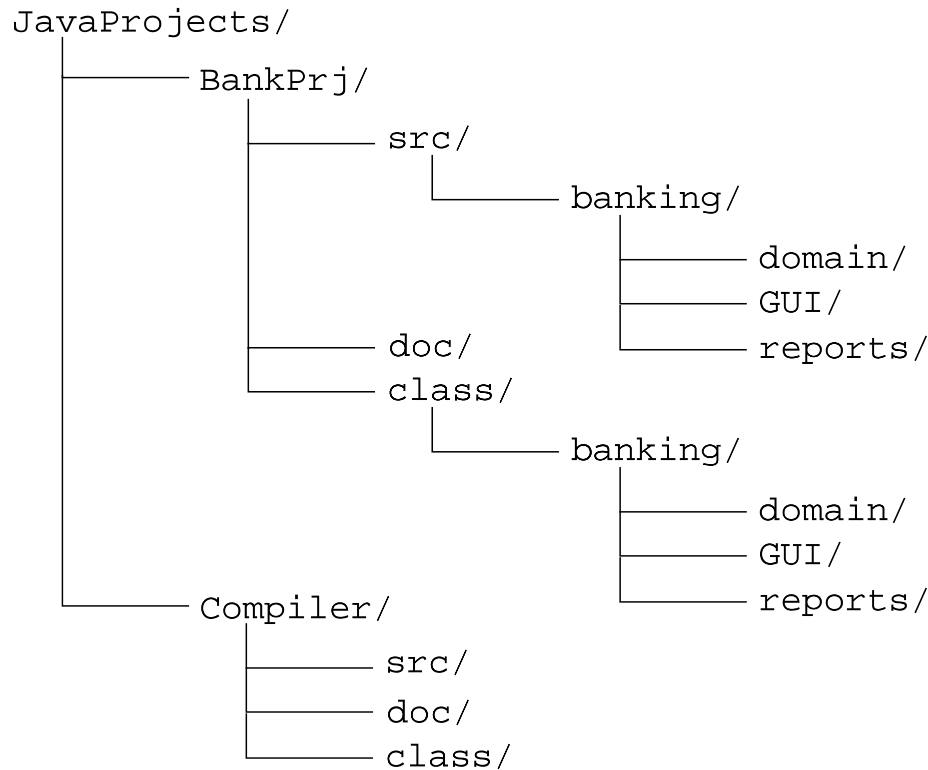
Directory Layout and Packages

- Packages are stored in the directory tree containing the package name
- Example, the "shipping" application packages:



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Directory Layout and Packages



Compiling using `-sourcepath` and `-d`

```
> cd JavaProjects/BankPrj/src/banking/domain
> javac -sourcepath JavaProjects/BankPrj/src
        -d JavaProjects/BankPrj/class *.java
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Terminology Recap

- Class – The source-code blueprint for a run-time object
- Object – An entity of a class
  AKA: instance
- Attribute – A data element of an object
  AKA: data member, instance variable, data field
- Method – A behavioral element of an object
  AKA: algorithm, function, procedure
- Constructor – A "method-like" construct used to initialize a new object
- Package – A grouping of classes and/or sub-packages

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Using the Java API Documentation

- A set of hypertext markup language (HTML) files provides information about the API
- One package contains hyperlinks to information on all of the classes
- A class document includes the class hierarchy, a description of the class, a list of member variables, a list of constructors, and so on

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Example API Documentation Page

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Using Objects and Classes

- Exercise objectives:
  - Become familiar with the Java 2 SDK API
  - Using the correct Java keywords, create a class and an object from the class
  - Compile and run a program
- Tasks:
  - Use a browser to explore the Java 2 SDK API
  - Explore encapsulation
  - Create the basic `Account` class for the Banking project

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Define modeling concepts: *abstraction*, *encapsulation*, and *packages*
- Discuss why Java application code is reusable
- Define *class*, *member*, *attribute*, *method*, *constructor*, and *package*
- Use the access modifiers `private` and `public` as appropriate for the guidelines of encapsulation
- Invoke a method on a particular object

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- In a Java software program, identify the following:
  - The `package` statement
  - The `import` statements
  - Classes, methods, and attributes
  - Constructors
- Use the Java technology application programming interface (API) online documentation

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Think Beyond

- Does your organization spend enough time on analysis and design?
- What domain objects and relationships appear in your existing applications?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Module 3

Identifiers, Keywords, and Types

Objectives

- Use comments in a source program
- Distinguish between valid and invalid identifiers
- Recognize Java technology keywords
- List the eight primitive types
- Define literal values for numeric and textual types
- Define the terms *primitive variable* and *reference variable*

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Objectives

- Declare variables of class type
- Construct an object using `new`
- Describe default initialization
- Describe the significance of a reference variable
- State the consequences of assigning variables of class type

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Relevance

- Do you know the primitive Java types?
- Can you describe the difference between variables holding primitive values as compared with object references?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Comments

- Three permissible styles of comment in a Java technology program are:

```
// comment on one line

/* comment on one
or more lines */

/** documentation comment */
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Semicolons, Blocks, and Whitespace

- A *statement* is one or more lines of code terminated by a semicolon (`;`):

```
totals = a + b + c
         + d + e + f;
```

- A *block* is a collection of statements bound by opening and closing braces:

```
{
  x = y + 1;
  y = x + 1;
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Semicolons, Blocks, and Whitespace

- You can use a *block* in a *class* definition:

```
public class MyDate {
   private int day;
   private int month;
   private int year;
}
```

- You can nest block statements
- Any amount of *whitespace* is allowed in a Java program

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Identifiers

- Are names given to a variable, class, or method
- Can start with a Unicode letter, underscore(_), or dollar sign($)
- Are case sensitive and have no maximum length
- Examples:

```
identifier
userName
user_name
_sys_var1
$change
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Java Keywords

abstract	do	implements	private	this
boolean	double	import	protected	throw
break	else	instanceof	public	throws
byte	extends	int	return	transient
case	false	interface	short	true
catch	final	long	static	try
char	finally	native	strictfp	void
class	float	new	super	volatile
continue	for	null	switch	while
default	if	package	synchronized	

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Primitive Types

- The Java programming language defines eight primitive types:
  - Logical `boolean`
  - Textual `char`
  - Integral `byte`, `short`, `int`, and `long`
  - Floating `double` and `float`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Logical – boolean

- The boolean data type has two literals, true and false.
- For example, the statement:

```
boolean truth = true;
```

declares the variable truth as boolean type and assigns it a value of true.

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Textual – `char` and `String`

`char`

- Represents a 16-bit Unicode character
- Must have its literal enclosed in single quotes(’ ’)
- Uses the following notations:

`'a'`	The letter *a*
`'\t'`	A tab
`'\u????'`	A specific Unicode character, `????`, is replaced with exactly four hexadecimal digits (for example, `’\u03A6’` is the Greek letter phi Φ)

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Textual – char and String

String

- Is not a primitive data type; it is a class
- Has its literal enclosed in double quotes (" ")

```
"The quick brown fox jumps over the lazy dog."
```

- Can be used as follows:

```
String greeting = "Good Morning !! \n";
String errorMessage = "Record Not Found !";
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Integral – `byte`, `short`, `int`, and `long`

- Uses three forms – Decimal, octal, or hexadecimal

`2`	The decimal value is two
`077`	The leading zero indicates an octal value
`0xBAAC`	The leading 0x indicates a hexadecimal value

- Has a default `int`
- Defines `long` by using the letter `L` or `l`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Integral – `byte`, `short`, `int`, and `long`

- Integral data types have the following ranges:

Integer Length	Name or Type	Range
8 bits	`byte`	-2^{7} to $2^{7}-1$
16 bits	`short`	-2^{15} to $2^{15}-1$
32 bits	`int`	-2^{31} to $2^{31}-1$
64 bits	`long`	-2^{63} to $2^{63}-1$

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Floating Point – `float` and `double`

- Default is `double`
- Floating point literal includes either a decimal point or one of the following:
  - `E` or `e` (add exponential value)
  - `F` or `f` (`float`)
  - `D` or `d` (`double`)

`3.14`	A simple floating-point value (a `double`)
`6.02E23`	A large floating-point value
`2.718F`	A simple `float` size value
`123.4E+306D`	A large `double` value with redundant *D*

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Floating Point – `float` and `double`

- Floating point data types have the following ranges:

Float Length	Name or Type
32 bits	`float`
64 bits	`double`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Variables, Declarations, and Assignments

```
1  public class Assign {
2    public static void main(String args []) {
3
4      int x, y; // declare int variables
5      float z = 3.414f;// declare and assign float
6      double w = 3.1415;// declare and assign double
7      boolean truth = true;// declare and assign boolean
8      char c;// declare character variable
9      String str;// declare String
10     String str1 = "bye";// declare and assign String variable
11     c = 'A';// assign value to char variable
12     str = "Hi out there!";// assign value to String variable
13     x = 6;
14     y = 1000;// assign values to int variables
15   ...
16   }
17 }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Java Reference Types

- Beyond primitive types all others are reference types
- A *reference variable* contains a "handle" to an object
- Example:

```
1   public class MyDate {
2     private int day = 1;
3     private int month = 1;
4     private int year = 2000;
5   }
```

```
1   public class TestMyDate {
2     public static void main(String[] args) {
3       MyDate today = new MyDate();
4     }
5   }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Constructing and Initializing Objects

- Calling `new Xxx()` to allocate space for the new object results in:
  - Memory Allocation: Space for the new object is allocated and instance variables are initialized to their default values (e.g. `0`, `false`, `null`, and so on)
  - Explicit attribute initialization is performed
  - A constructor is executed
  - Variable assignment is made to reference the object
- Example:

```
MyDate my_birth = new MyDate(22, 7, 1964);
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Memory Allocation and Layout

- A declaration allocates storage only for a reference:

```
MyDate my_birth = new MyDate(22, 7, 1964);
```

my_birth	????

- Use the `new` operator to allocate space for `MyDate`:

```
MyDate my_birth = new MyDate(22, 7, 1964);
```

my_birth	????
day	0
month	0
year	0

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Explicit Attribute Initialization

- Initialize the attributes:

```
MyDate my_birth = new MyDate(22, 7, 1964);
```

my_birth	????
day	1
month	1
year	2000

- The default values are taken from the attribute declaration in the class

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Executing the Constructor

- Execute the matching constructor:

```
MyDate my_birth = new MyDate(22, 7, 1964);
```

my_birth	????

day	22
month	7
year	1964

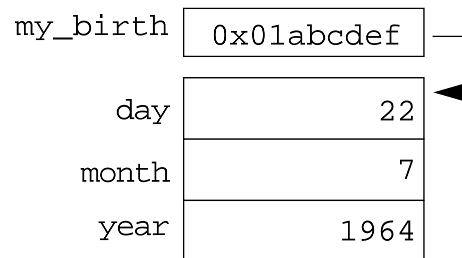
- In the case of an overloaded constructor, the first constructor may call another

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Variable Assignment

- Assign newly created object to reference variable:

```
MyDate my_birth = new MyDate(22, 7, 1964);
```



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Assignment of Reference Variables

- Consider the following code fragment:

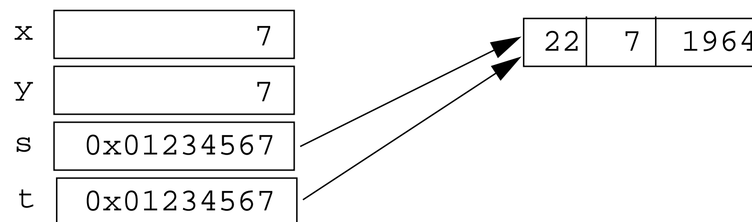
```
int x = 7;
int y = x;
MyDate s = new MyDate(22, 7, 1964);
MyDate t = s;
t = new MyDate(22, 12, 1964);
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Assignment of Reference Variables

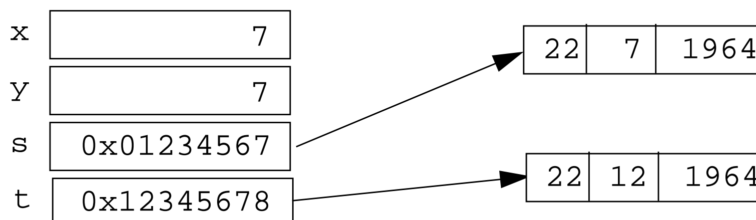
```
int x = 7;
int y = x;
MyDate s = new MyDate(22, 7, 1964);
MyDate t = s;
```

- Two variables refer to a single object




```
t = new MyDate(22, 12, 1964);
```

- Reassignment makes two variables point to two objects



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Pass-by-Value

- The Java programming language only passes arguments by value
- When an object instance is passed as an argument to a method, the value of the argument is a *reference* to the object
- The *contents* of the object can be changed in the called method, but the object reference is never changed

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Pass-by-Value

```
1   public class PassTest {
2
3     // Methods to change the current values
4     public static void changeInt(int value) {
5       value = 55;
6     }
7     public static void changeObjectRef(MyDate ref) {
8       ref = new MyDate(1, 1, 2000);
9     }
10    public static void changeObjectAttr(MyDate ref) {
11      ref.setDay(4);
12    }
13
14    public static void main(String args[]) {
15      MyDate date;
16      int val;
17
18      // Assign the int
19      val = 11;
20      // Try to change it
21      changeInt(val);
22      // What is the current value?
23      System.out.println("Int value is: " + val);
24
25      // Assign the date
26      date = new MyDate(22, 7, 1964);
27      // Try to change it
28      changeObjectRef(date);
29      // What is the current value?
30      date.print();
31
32      // Now change the day attribute
33      // through the object reference
34      changeObjectAttr(date);
35      // What is the current value?
36      date.print();
37    }
38  }
39
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The `this` Reference

Here are a few uses of the `this` keyword:

- To reference local attribute and method members within a local method or constructor
  - This is used to disambiguate a local method or constructor variable from an instance variable
- To pass the current object as a parameter to another method or constructor

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The `this` Reference

```
1   public class MyDate {
2     private int day = 1;
3     private int month = 1;
4     private int year = 2000;
5
6     public MyDate(int day, int month, int year) {
7       this.day   = day;
8       this.month = month;
9       this.year  = year;
10    }
11    public MyDate(MyDate date) {
12      this.day   = date.day;
13      this.month = date.month;
14      this.year  = date.year;
15    }
16
17    public MyDate addDays(int more_days) {
18      MyDate new_date = new MyDate(this);
19
20      new_date.day = new_date.day + more_days;
21      // Not Yet Implemented: wrap around code...
22
23      return new_date;
24    }
25    public void print() {
26      System.out.println("MyDate: " + day + "-" + month +
27                         "-" + year);
28    }
29  }
```

```
1   public class TestMyDate {
2     public static void main(String[] args) {
3       MyDate my_birth = new MyDate(22, 7, 1964);
4       MyDate the_next_week = my_birth.addDays(7);
5
6       the_next_week.print();
7     }
8   }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Java Coding Conventions

- Packages:

```
package banking.domain;
```

- Classes:

```
class SavingsAccount
```

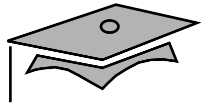
- Interfaces:

```
interface Account
```

- Methods:

```
balanceAccount()
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D



Java Coding Conventions

- Variables:

```
currentCustomer
```

- Constants:

```
HEAD_COUNT
MAXIMUM_SIZE
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Using Identifiers, Keywords, and Types

- Exercise objectives:
  - Verify that the references are assigned and manipulated as described in this module
- Tasks:
  - Investigate reference assignments
  - Extend the Banking project to use object references

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Use comments in a source program
- Distinguish between valid and invalid identifiers
- Recognize Java technology keywords
- List the eight primitive types
- Define literal values for numeric and textual types
- Define the terms *primitive variable* and *reference variable*

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Declare variables of class type
- Construct an object using `new`
- Describe default initialization
- Describe the significance of a reference variable
- State the consequences of assigning variables of class type

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Think Beyond

- Can you think of examples of classes and objects in your existing applications?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Module 4

Expressions and Flow Control

Objectives

- Distinguish between instance and local variables
- Describe how instance variables are initialized
- Identify and correct a `Possible reference before assignment` compiler error
- Recognize, describe, and use Java software operators
- Distinguish between legal and illegal assignments of primitive types

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Objectives

- Identify `boolean` expressions and their requirements in control constructs
- Recognize assignment compatibility and required casts in fundamental types
- Use `if`, `switch`, `for`, `while`, and `do` constructions and the labeled forms of `break` and `continue` as flow control structures in a program

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Relevance

- What types of variables are useful to programmers?
- Can multiple classes have variables with the same name and, if so, what is their scope?
- What types of control structures are used in other languages? What methods do these languages use to control flow?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Variables and Scope

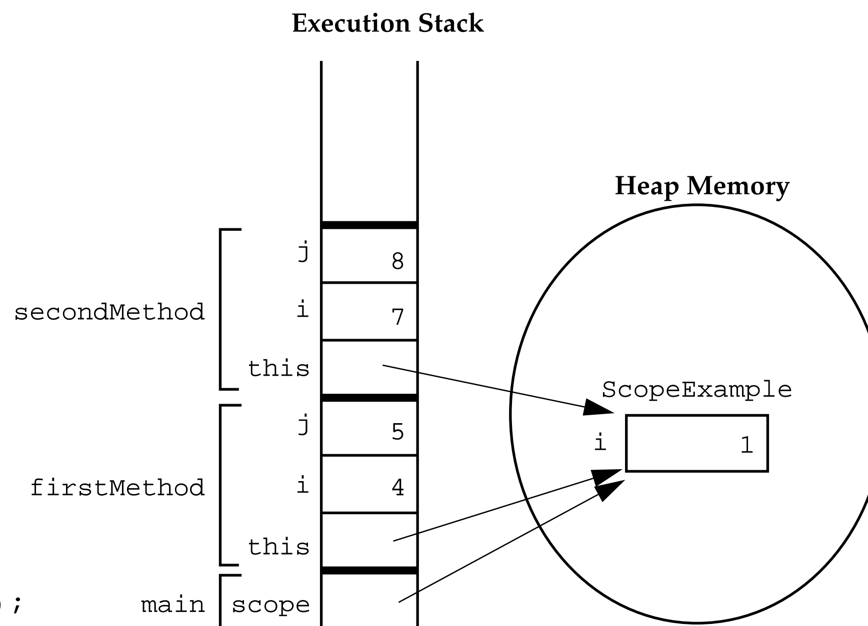
Local variables are:

- Variables that are defined inside a method and are called *local*, *automatic*, *temporary*, or *stack* variables
- Variables that are created when the method is executed are destroyed when the method is exited
- Variables that must be initialized before they are used or compile-time errors will occur

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Variable Scope Example

```
public class ScopeExample {
  private int i=1;

  public void firstMethod() {
    int i=4, j=5;

    this.i = i + j;
    secondMethod(7);
  }
  public void secondMethod(int i) {
    int j=8;
    this.i = i + j;
  }
}

public class TestScoping {
  public static void main(String[] args) {
    ScopeExample scope = new ScopeExample();

    scope.firstMethod();
  }
}
```



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Variable Initialization

Variable	**Value**
`byte`	`0`
`short`	`0`
`int`	`0`
`long`	`0L`
`float`	`0.0f`
`double`	`0.0d`
`char`	`'\u0000'`
`boolean`	`false`
All reference types	`null`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Operators

<table>
<tr><td>Separator</td><td>.   [ ]   ( )   ;   ,</td></tr>
</table>

<table>
<tr><th>Associative</th><th>Operators</th></tr>
<tr><td>R to L</td><td>++ -- + - ~ ! (data type)</td></tr>
<tr><td>L to R</td><td>* / %</td></tr>
<tr><td>L to R</td><td>+ -</td></tr>
<tr><td>L to R</td><td><< >> >>></td></tr>
<tr><td>L to R</td><td>< > <= >= instanceof</td></tr>
<tr><td>L to R</td><td>== !=</td></tr>
<tr><td>L to R</td><td>&</td></tr>
<tr><td>L to R</td><td>^</td></tr>
<tr><td>L to R</td><td>|</td></tr>
<tr><td>L to R</td><td>&&</td></tr>
<tr><td>L to R</td><td>||</td></tr>
<tr><td>R to L</td><td>?:</td></tr>
<tr><td>R to L</td><td>= *= /= %= += -= <<=
>>= >>>= &= ^= |=</td></tr>
</table>

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Logical Operators

- The boolean operators are:

```
! - NOT            & - AND
| - OR             ^ - XOR
```

- The Short-Circuit boolean operators are:

```
&& - AND           || - OR
```

- These operators can be used as follows:

```
MyDate d;
if ((d != null) && (d.day > 31)) {
  // do something with d
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Bitwise Logical Operators

- The Integer *bitwise* operators are:

```
~ - Complement        & - AND
^ - XOR               | - OR
```

- Byte-sized examples:

~	0	1	0	0	1	1	1	1
	1	0	1	1	0	0	0	0

	0	0	1	0	1	1	0	1
&	0	1	0	0	1	1	1	1
	0	0	0	0	1	1	0	1

	0	0	1	0	1	1	0	1
^	0	1	0	0	1	1	1	1
	0	1	1	0	0	0	1	0

	0	0	1	0	1	1	0	1
		0	1	0	0	1	1	1	1
	0	1	1	0	1	1	1	1

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Right-Shift Operators >> and >>>

- *Arithmetic* or *signed* right shift `(>>)` is used as follows:

```
128 >> 1 returns 128/2^1 = 64
256 >> 4 returns 256/2^4 = 16
-256 >> 4 returns -256/2^4 = -16
```

  - The sign bit is copied during the shift
- A *logical* or *unsigned right shift* operator (>>>) is:
  - Used for bit patterns
  - Not copied during the shift

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Left-Shift Operator (<<)

- Left-shift works as follows:

```
128 << 1 returns 128 * 2^1 = 256
16  << 2 returns 16 * 2^2 = 64
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Shift Operator Examples

`1357 =`	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	1	1	0	1
`-1357 =`	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	1	0	0	1	1
`1357 >> 5 =`	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0
`-1357 >> 5 =`	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1
`1357 >>> 5 =`	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0
`-1357 >>> 5 =`	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1
`1357 << 5 =`	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	1	1	0	1	0	0	0	0	0
`-1357 << 5 =`	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	1	0	0	1	1	0	0	0	0	0

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

String Concatenation With +

- The + operator:
  - Performs `String` concatenation
  - Produces a new `String`:

```
String salutation = "Dr.";
String name = "Pete" + " " + "Seymour";
String title = salutation + " " + name;
```

- One argument must be a `String` object
- Non-strings are converted to `String` objects automatically

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Casting

- If information is lost in an assignment, the programmer must confirm the assignment with a typecast.
- The assignment between `long` and `int` requires an explicit cast.

```
long bigValue = 99L;
int squashed = bigValue; // Wrong, needs a cast
int squashed = (int) bigValue; // OK

int squashed = 99L; // Wrong, needs a cast
int squashed = (int) 99L; // OK, but...
int squashed = 99;  // default integer literal
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Promotion and Casting of Expressions

- Variables are automatically promoted to a longer form (such as `int` to `long`).
- Expression is *assignment compatible* if the variable type is at least as large (the same number of bits) as the expression type.

```
long bigval = 6;    // 6 is an int type, OK
int smallval = 99L; // 99L is a long, illegal

double z = 12.414F;  // 12.414F is float, OK
float z1 = 12.414;   // 12.414 is double, illegal
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Branching Statements

The `if, else` statement syntax:

```
if (boolean expression) {
   statement or block;
}

if (boolean expression) {
  statement or block;
} else if (boolean expression) {
  statement or block;
} else {
  statement or block;
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Branching Statements

An `if`, `else` statement example:

```
int count;
count = getCount(); // a method defined in the program
if (count < 0) {
  System.out.println("Error: count value is negative.");
} else if (count > getMaxCount()) {
  System.out.println("Error: count value is too big.");
} else {
  System.out.println("There will be " + count +
                     " people for lunch today.");
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Branching Statements

The `switch` statement syntax:

```
switch (expr1) {
   case constant2:
      statements;
      break;
   case constant3:
      statements;
      break;
   default:
      statements;
      break;
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Branching Statements

A `switch` statement example:

```
switch ( carModel ) {
  case DELUXE:
    addAirConditioning();
    addRadio();
    addWheels();
    addEngine();
    break;
  case STANDARD:
    addRadio();
    addWheels();
    addEngine();
    break;
  default:
    addWheels();
    addEngine();
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Branching Statements

A `switch` statement example:

```
switch ( carModel ) {
  case THE_WORKS:
    addGoldPackage();
    add7WayAdjustableSeats();
  case DELUXE:
    addFloorMats();
    addAirConditioning();
  case STANDARD:
    addRadio();
    addDefroster();
  default:
    addWheels();
    addEngine();
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Looping Statements

The `for` statement:

```
for (init_expr; boolean testexpr; alter_expr) {
  statement or block;
}
```

Example:

```
for (int i = 0; i < 10; i++) {
     System.out.println("Are you finished yet?");
}
System.out.println("Finally!");
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Looping Statements

The `while` loop:

```
while (boolean) {
   statement or block;
}
```

Example:

```
int i = 0;

while (i < 10) {
   System.out.println("Are you finished yet?");
   i++;
}
System.out.println("Done");
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Looping Statements

The `do/while` statement:

```
do {
   statement or block;
} while (boolean test);
```

Example:

```
int i = 0;

do {
   System.out.println("Are you finished yet?");
   i++;
} while (i < 10);
System.out.println("Done");
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Special Loop Flow Control

- `break [label];`
- `continue [label];`
- `label: statement; // Where statement should`
  `// be a loop`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Special Loop Flow Control

The `break` statement:

```
do {
  statement;
  if (condition is true) {
    break;
  }
  statement;
} while (boolean expression);
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Special Loop Flow Control

The `continue` statement:

```
do {
  statement;
  if (condition is true) {
    continue;
  }
  statement;
} while (boolean expression);
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Special Loop Flow Control

Using break with labels:

```
outer:
  do {
    statement;
    do {
      statement;
      if (boolean expression) {
        break outer;
      }
      statement;
    } while (boolean expression);
    statement;
  } while (boolean expression);
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Special Loop Flow Control

Using `continue` with labels:

```
test:
  do {
    statement;
    do {
      statement;
      if (condition is true) {
        continue test;
      }
      statement;
    } while (condition is true);
    statement;
  } while (condition is true);
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Using Expressions

- Exercise objective:
  - Write, compile, and run three programs that use identifiers, expressions, and control structures
- Tasks:
  - Use a loop
  - Modify the `Account` class `withdraw` method
  - Use nested loops and special control flow

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Distinguish between instance and local variables
- Describe how instance variables are initialized
- Identify and correct a `Possible reference before assignment` compiler error
- Recognize, describe, and use Java operators
- Distinguish between legal and illegal assignments of primitive types

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Identify `boolean` expressions and their requirements in control constructs
- Recognize assignment compatibility and required casts in fundamental types
- Use `if`, `switch`, `for`, `while`, and `do` constructions and the labeled forms of `break` and `continue` as flow control structures in a program

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Think Beyond

- What data types do most programming languages use to group similar data elements together?
- How do you perform the same operation on all elements of a group (for example, a matrix)?
- What data types does the Java programming language use?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Module 5

Arrays

Objectives

- Declare and create arrays of primitive, class, or array types
- Explain why elements of an array are initialized
- Explain how to initialize the elements of an array
- Determine the number of elements in an array
- Create a multi-dimensional array
- Write code to copy array values from one array type to another

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Relevance

- What is the purpose of an array?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Declaring Arrays

- Group data objects of the same type
- Declare arrays of primitive or class types

```
char s[];
Point p[];

char [] s;
Point [] p;
```

- Create space for a reference
- An array is an object; it is created with `new`

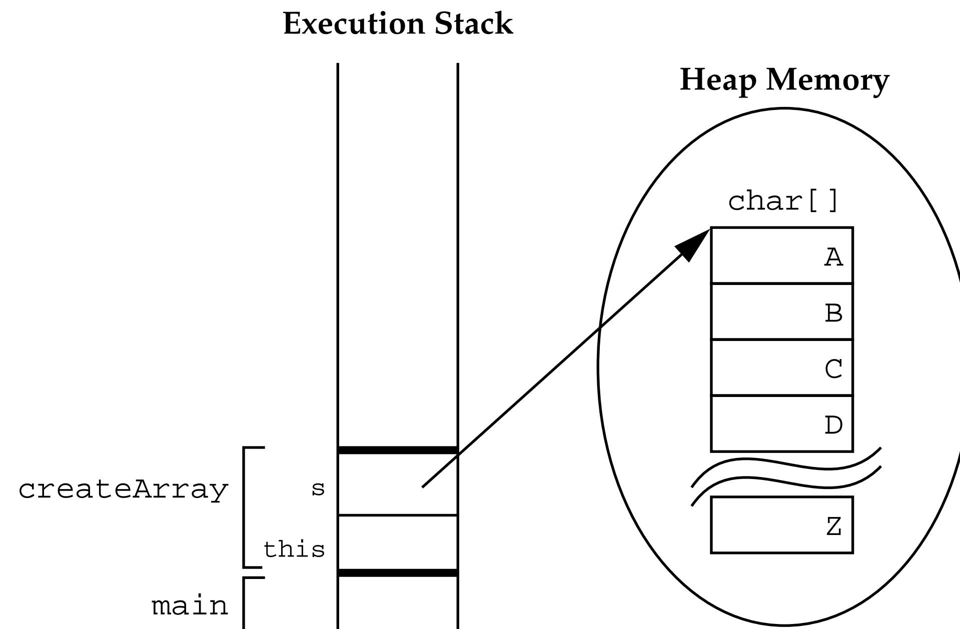
Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Creating Arrays

Use the new keyword to create an array object.

For example, a primitive (char) array:

```
public char[] createArray() {
  char[] s;

  s = new char[26];
  for ( int i=0; i<26; i++ ) {
    s[i] = 'A' + i;
  }

  return p;
}
```

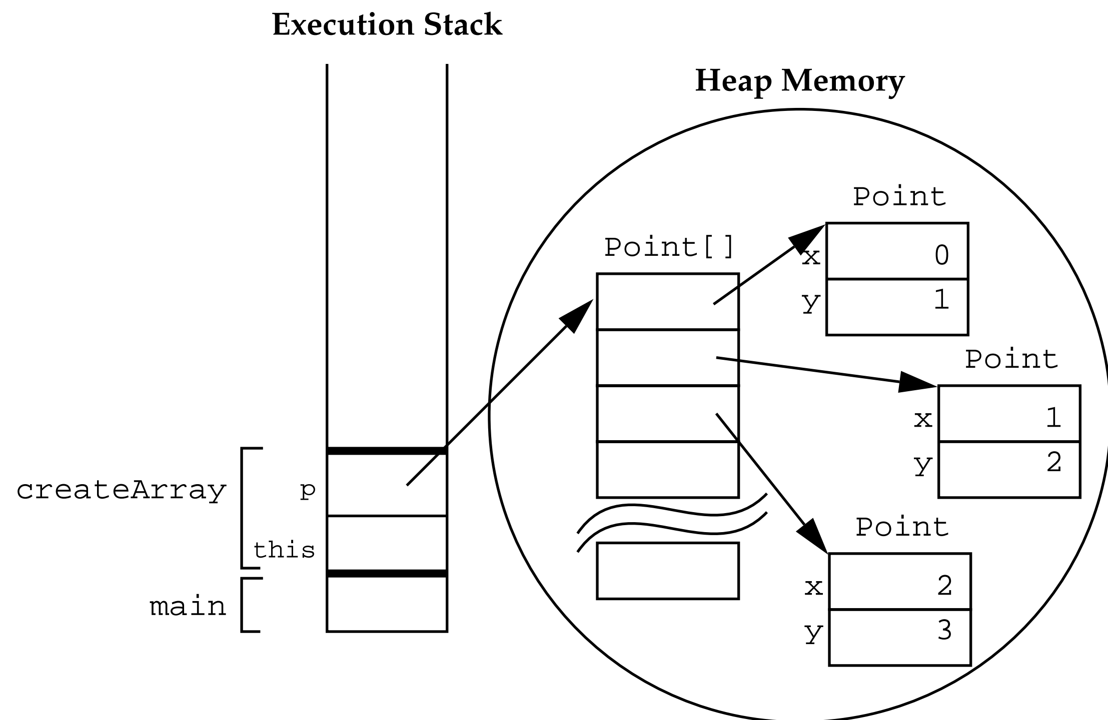


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Creating Arrays

Another example, an object array:

```
public Point[] createArray() {
  Point[] p;

  p = new char[10];
  for ( int i=0; i<10; i++ ) {
    p[i] = new Point(i, i+1);
  }

  return p;
}
```



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Initializing Arrays

- Initialize an array element
- Create an array with initial values:

```
String names[];
names = new String[3];
names[0] = "Georgianna";
names[1] = "Jen";
names[2] = "Simon";
```

```
String names[] = {
    "Georgianna",
    "Jen",
    "Simon"
};
```

```
MyDate dates[];
dates = new MyDate[3];
dates[0] = new MyDate(22, 7, 1964);
dates[1] = new MyDate(1, 1, 2000);
dates[2] = new MyDate(22, 12, 1964);
```

```
MyDate dates[] = {
    new MyDate(22, 7, 1964),
    new MyDate(1, 1, 2000),
    new MyDate(22, 12, 1964)
};
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Multi-Dimensional Arrays

- Arrays of arrays:

```
int twoDim [][] = new int [4][];
twoDim[0] = new int[5];
twoDim[1] = new int[5];

int twoDim [][] = new int [][4]; illegal
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Multi-Dimensional Arrays

- Non-rectangular arrays of arrays:

```
twoDim[0] = new int[2];
twoDim[1] = new int[4];
twoDim[2] = new int[6];
twoDim[3] = new int[8];
```

- Array of four arrays of five integers each:

```
int twoDim[][] = new int[4][5];
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Array Bounds

All array subscripts begin at 0:

```
int list[] = new int [10];
for (int i = 0; i < list.length; i++) {
  System.out.println(list[i]);
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Array Resizing

- Cannot resize an array
- Can use the same reference variable to refer to an entirely new array:

```
int elements[] = new int[6];
elements = new int[10];
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Copying Arrays

The `System.arraycopy()` method:

```
1   //original array
2   int elements[] = { 1, 2, 3, 4, 5, 6 };
3
4   // new larger array
5   int hold[] = { 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 };
6
7   // copy all of the elements array to the hold
8   // array, starting with the 0th index
9   System.arraycopy(elements, 0, hold, 0, elements.length);
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Using Arrays

- Exercise objectives:
  - Define and initialize an array
  - Write a program that defines, initializes, and uses arrays
- Tasks:
  - Declare, create, and manipulate one- and two-dimensional arrays
  - Use arrays to represent the multiplicity of relation-ships between objects in the banking project

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Declare and create arrays of primitive, class, or array types
- Explain why elements of an array are initialized
- Explain how to initialize the elements of an array
- Determine the number of elements in an array
- Create a multi-dimensional array
- Write code to copy array values from one array type to another

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Think Beyond

- How can you create a three-dimensional array?
- What is one disadvantage of using arrays?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Module 6

Inheritance

Objectives

- Define *inheritance*, *polymorphism*, *overloading*, *overriding*, and *virtual method invocation*
- Use the access modifiers `protected` and "package-friendly"
- Describe constructor and method overloading
- Describe the complete object construction and initialization operation

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Objectives

- In a Java program, identify the following:
  - Overloaded methods and constructors
  - The use of `this` to call overloaded constructors
  - Overridden methods
  - The use of `super` to call parent class methods
  - Parent class constructors
  - The use of `super` to call parent class constructors

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Relevance

- How does the Java programming language support object inheritance?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The `is a` Relationship

The `Employee` class:

Employee
+name : String = "" +salary : double +birthDate : Date
+getDetails() : String

```
public class Employee {
  public String name = "";
  public double salary;
  public Date birthDate;

  public String getDetails() {...}
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The `is a` Relationship

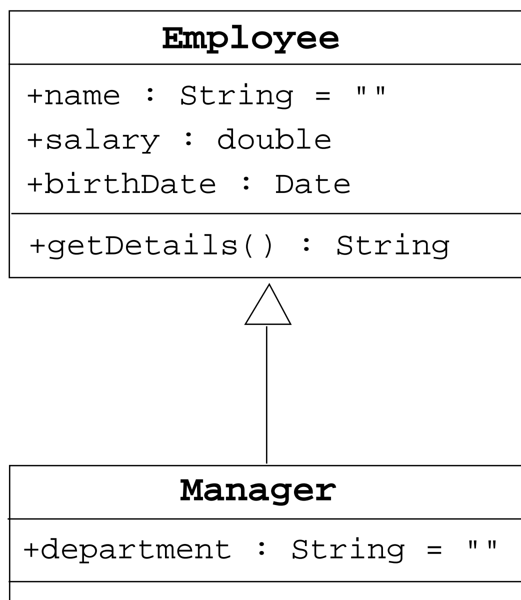
The `Manager` class:

Manager
+name : String = ""   +salary : double   +birthDate : Date   +department : String
+getDetails() : String

```
public class Manager {
  public String name = "";
  public double salary;
  public Date birthDate;
  public String department;

  public String getDetails() {...}
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The is a Relationship




```
public class Employee {
  public String name = "";
  public double salary;
  public Date birthDate;

  public String getDetails() {...}
}
```

```
public class Manager extends Employee {
  public String department = "";
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

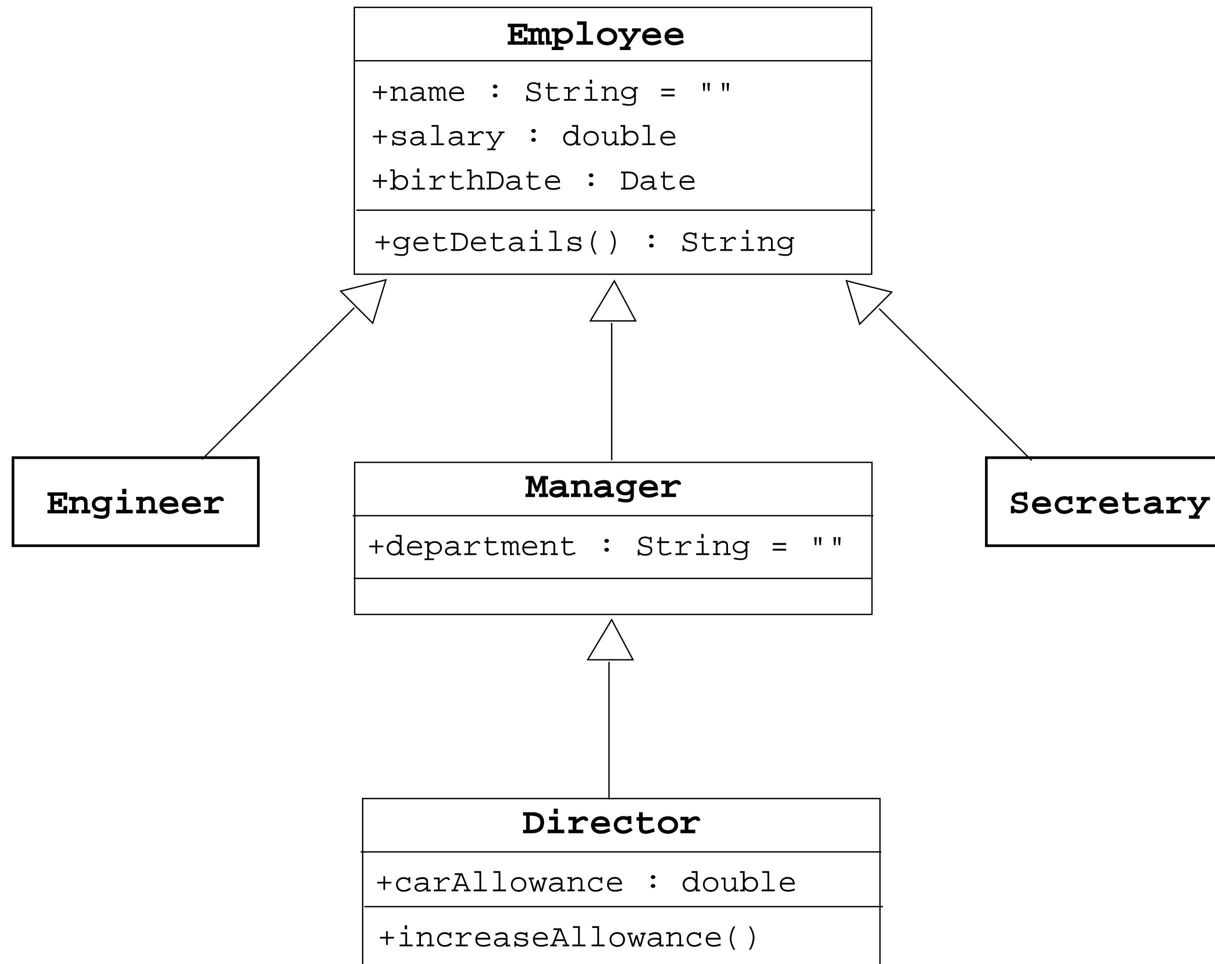
Single Inheritance

- When a class inherits from only one class, it is called *single inheritance*.
- Single inheritance makes code more reliable.
- *Interfaces* provide the benefits of multiple inheritance without drawbacks.
- Syntax of a Java class:

```
<class_declaration> ::=
  <modifier> class <name> [extends <superclass>] {
    <declarations>*
  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Single Inheritance



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Constructors Are Not Inherited

- A subclass inherits all methods and variables from the superclass (parent class)
- A subclass does not inherit the constructor from the superclass
- Two ways to include a constructor are:
  - Use the default constructor
  - Write one or more explicit constructors

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Polymorphism

- *Polymorphism* is the ability to have many different forms; for example, the `Manager` class has access to methods from `Employee` class
- An object has only one form
- A reference variable can refer to objects of different forms

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Polymorphism

```
Employee employee = new Manager() //legal

// Illegal attempt to assign Manager attribute
employee.department = "Sales";
// the variable is declared as a Employee type,
// even though the Manager object has that attribute
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Polymorphic Arguments

- Because a `Manager` is an `Employee`:

```
// In the Employee class
public TaxRate findTaxRate(Employee e) {
}
// Meanwhile, elsewhere in the application class
Manager m = new Manager();
:
TaxRate t = findTaxRate(m);
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Heterogeneous Collections

- Collections of objects with the same class type are called *homogenous* collections.

```
MyDate[]  dates = new MyDate[2];
dates[0] = new MyDate(22, 12, 1964);
dates[1] = new MyDate(22, 7, 1964);
```

- Collections of objects with different class types are called *heterogeneous* collections.

```
Employee [] staff = new Employee[1024];
staff[0] = new Manager();
staff[1] = new Employee();
staff[2] = new Engineer();
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The instanceof Operator

```
public class Employee extends Object
public class Manager extends Employee
public class Engineer extends Employee
----------------------------------------

public void doSomething(Employee e) {
  if (e instanceof Manager) {
    // Process a Manager
  } else if (e instanceof Engineer) {
    // Process an Engineer
  } else {
    // Process any other type of Employee
  }
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

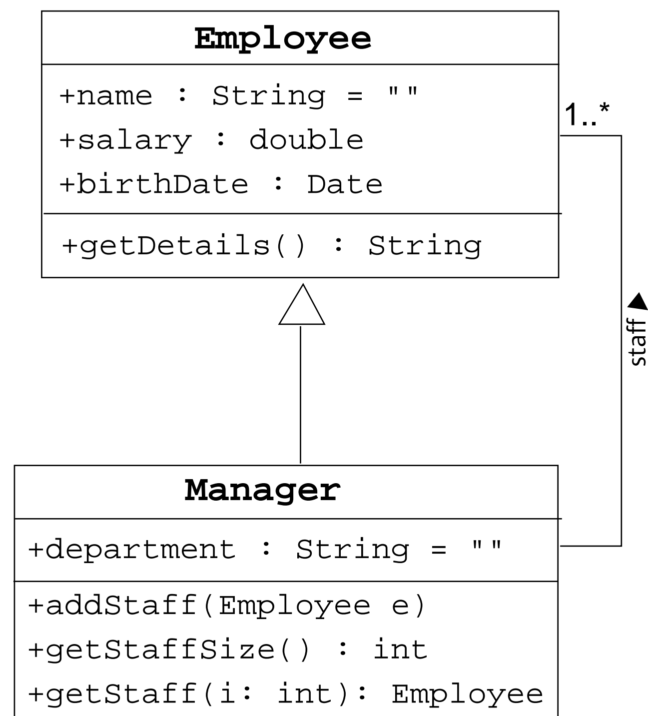
Casting Objects

- Use `instanceof` to test the type of an object
- Restore full functionality of an object by casting
- Check for proper casting using the following guidelines:
  - Casts up hierarchy are done implicitly
  - Downward casts must be to a subclass and checked by the compiler
  - The object type is checked at runtime when runtime errors can occur

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The has a Relationship

```
public class Vehicle {
  private Engine theEngine;
  public Engine getEngine() {
    return theEngine;
  }
}
```




```
public class Manager extends Employee {
  public String department = "";
  public Employee[] staff = new Employee[20];
  public int staffSize = 0;

  public void addStaff(Employee e) {
    staff[staffSize++] = e;
  }
  public int getStaffSize() {
    return staffSize;
  }
  public Employee getStaff(int i) {
    return staff[i];
  }
  ...
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Access Control

Modifier	Same Class	Same Package	Subclass	Universe
`public`	Yes	Yes	Yes	Yes
`protected`	Yes	Yes	Yes	
default	Yes	Yes		
`private`	Yes			

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Overloading Method Names

- It can be used as follows:

```
public void println(int i)
public void println(float f)
public void println(String s)
```

- Argument lists *must* differ
- Return types *can* be different

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Overloading Constructors

- As with methods, constructors can be overloaded
- Example:

```
public Employee(String name, double salary, Date DoB)
public Employee(String name, double salary)
public Employee(String name, Date DoB)
```

- Argument lists *must* differ
- The `this` reference can be used at the first line of a constructor to call another constructor

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Overloading Constructors

```
1   public class Employee {
2     private static final double BASE_SALARY = 15000.00;
3     private String name;
4     private double salary;
5     private Date   birthDate;
6
7     public Employee(String name, double salary, Date DoB) {
8       this.name = name;
9       this.salary = salary;
10      this.birthDate = DoB;
11    }
12    public Employee(String name, double salary) {
13      this(name, salary, null);
14    }
15    public Employee(String name, Date DoB) {
16      this(name, BASE_SALARY, DoB);
17    }
18    public Employee(String name) {
19      this(name, BASE_SALARY);
20    }
21    // more Employee code...
22  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Overriding Methods

- A subclass can modify behavior inherited from a parent class
- A subclass can create a method with different functionality than the parent's method but with the same:
  - Name
  - Return type
  - Argument list

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Overriding Methods

```
public class Employee {
  protected String name;
  protected double salary;
  protected Date birthDate;

  public String getDetails() {
    return "Name: " + name + "\n" +
           "Salary: " + salary;
  }
}


public class Manager extends Employee {
  protected String department;

  public String getDetails() {
    return "Name: " + name + "\n" +
           "Salary: " + salary + "\n" +
           "Manager of: " + department;
  }
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Overriding Methods

- Virtual method invocation:

```
Employee e = new Manager();
e.getDetails();
```

- Compile-time type and runtime type

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Rules About Overridden Methods

- Must have a return type that is identical to the method it overrides
- Cannot be less accessible than the method it overrides

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Rules About Overridden Methods

```
public class Parent {
  public void doSomething() {}
}

public class Child extends Parent {
  private void doSomething() {}
}

public class UseBoth {
  public void doOtherThing() {
    Parent p1 = new Parent();
    Parent p2 = new Child();
    p1.doSomething();
    p2.doSomething();
  }
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The `super` Keyword

- `super` is used in a class to refer to its superclass
- `super` is used to refer to the members of superclass, both data attributes and methods
- Behavior invoked does not have to be in the superclass; it can be further up in the hierarchy

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The super Keyword

```
public class Employee {
   private String name;
   private double salary;
   private Data birthDate;
   public String getDetails() {
      return Name: " + name + "\nSalary: " + salary;
   }
}
public class Manager extends Employee {
   private String department;
   public String getDetails() {
      // call parent method
      return super.getDetails() + "\n" +
        "Manager of: " + department;
   }
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Invoking Parent Class Constructors

- To invoke a parent constructor you must place a call to `super` in the first line of the constructor
- You can call a specific parent constructor by the arguments that you use in the call to `super`
- If no `this` or `super` call is used in a constructor, then the compiler adds an implicit call to `super()` which calls the parent "default" constructor
  - If the parent class does not supply a non-private "default" constructor, then a compiler warning will be issued

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Invoking Parent Class Constructors

```
1   public class Employee {
2     private static final double BASE_SALARY = 15000.00;
3     private String name;
4     private double salary;
5     private Date   birthDate;
6
7     public Employee(String name, double salary, Date DoB) {
8       this.name = name;
9       this.salary = salary;
10      this.birthDate = DoB;
11    }
12    public Employee(String name, double salary) {
13      this(name, salary, null);
14    }
15    public Employee(String name, Date DoB) {
16      this(name, BASE_SALARY, DoB);
17    }
18    public Employee(String name) {
19      this(name, BASE_SALARY);
20    }
21    // more Employee code...
22  }
```

```
1   public class Manager extends Employee {
2     private String department;
3
4     public Manager(String name, double salary, String dept) {
5       super(name, salary);
6       department = dept;
7     }
8     public Manager(String n, String dept) {
9       super(name);
10      department = dept;
11    }
12    public Manager(String dept) { // This code fails: no super()
13      department = d;
14    }
15  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Constructing and Initializing Objects: A Slight Reprise

- Memory is allocated and default initialization occurs
- Instance variable initialization uses these steps recursively:

1. Bind constructor parameters.
2. If explicit `this()`, call recursively and then skip to step 5.
3. Call recursively the implicit or explicit `super` call, except for `Object`.
4. Execute explicit instance variable initializers.
5. Execute body of current constructor.

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

An Example

```
public class Object {
   ...
  public Object() {}
   ...
}

public class Employee extends Object {
  private String name;
  private double salary = 15000.00;
  private Date   birthDate;

  public Employee(String n, Date DoB) {
    // implicit super();
    name = n;
    birthDate = DoB;
  }
  public Employee(String n) {
    this(n, null);
  }
}

public class Manager extends Employee {
  private String department;

  public Manager(String n, String d) {
    super(n);
    department = d;
  }
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

An Example

`0` basic initialization

  `0.1` allocate memory for the complete `Manager` object

  `0.2` initialize all instance variables to their default values

`1` call constructor: `Manager("Joe Smith", "Sales")`

  `1.1` bind constructor parameters: `n="Joe Smith", d="Sales"`

  `1.2` no explicit `this()` call

  `1.3` call `super(n)` for `Employee(String)`

    `1.3.1` bind constructor parameters: `n="Joe Smith"`

    `1.3.2` call `this(n, null)` for `Employee(String, Date)`

      `1.3.2.1` bind constructor parameters: `n="Joe Smith", DoB=null`

      `1.3.2.2` no explicit `this()` call

      `1.3.2.3` call `super()` for `Object()`

        `1.3.2.3.1` no binding necessary

        `1.3.2.3.2` no `this()` call

        `1.3.2.3.3` no `super()` call (`Object` is the root)

        `1.3.2.3.4` no explicit variable initialization for `Object`

        `1.3.2.3.5` no method body to call

      `1.3.2.4` initialize explicit Employee variables: `salary=15000.00;`

      `1.3.2.5` execute body: `name="Joe Smith"; date=null;`

    `1.3.3 - 1.3.4` steps skipped

    `1.3.5` execute body: no body in `Employee(String)`

  `1.4` no explicit initializers for `Manager`

  `1.5` execute body: `department="Sales"`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Implications of the Initialization Process

```
1  public class Employee extends Object {
2    private String name;
3    private double salary = 15000.00;
4    private Date   birthDate;
5    private String summary;
6
7    public Employee(String n, Date DoB) {
8      name = n;
9      birthDate = DoB;
10     summary = getDetails();
11   }
12   public Employee(String n) {
13      this(n, null);
14   }
15
16   public String getDetails() {
17     return "Name: " + name + "\nSalary: " + salary
18            + "\nBirth Date: " + birthDate;
19   }
20 }
```

```
1  public class Manager extends Employee {
2    private String department;
3
4    public Manager(String n, String d) {
5      super(n);
6      department = d;
7    }
8
9    public String getDetails() {
10     return super.getDetails() + "\nDept: " + department;
11   }
12 }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The `Object` Class

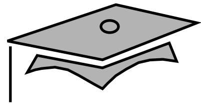
- The `Object` class is the root of all classes in Java
- A class declaration with no extends clause, implicitly uses "extends Object"

```
public class Employee {
  ...
}
```

is equivalent to:

```
public class Employee extends Object {
  ...
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D



The == Operator Compared With the `equals` Method

- The == operator determines if two references are identical to each other (that is, refer to the same object)
- The `equals` method determines if objects are "equal" but not necessarily identical
- The `Object` implementation of the `equals` method uses the == operator
- User classes can override the `equals` method to implement a domain-specific test for equality
- Note: You should override the `hashCode` method, if you override the `equals` method

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

equals Example

```
1   class Employee {
2
3     private String name;
4     private MyDate birthDate;
5     private float  salary;
6
7     // Constructor
8     public Employee(String name, MyDate DoB, float salary) {
9       this.name = name;
10      this.birthDate = DoB;
11      this.salary = salary;
12    }
13
14    public boolean equals(Object o) {
15      boolean result = false;
16      if ( (o != null) && (o instanceof Employee) ) {
17        Employee e = (Employee) o;
18        if (   name.equals(e.name)
19            && birthDate.equals(e.birthDate) ) {
20          result = true;
21        }
22      }
23      return result;
24    }
25
26    public int hashCode() {
27      return ( name.hashCode() ^ birthDate.hashCode() );
28    }
29  }
30
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

`equals` Example

```
1   class TestEquals {
2     public static void main(String[] args) {
3       Employee  emp1 = new Employee("Fred Smith",
4                                     new MyDate(14, 3, 1976),
5                                     25000.0F);
6       Employee  emp2 = new Employee("Fred Smith",
7                                     new MyDate(14, 3, 1976),
8                                     25000.0F);
9
10      if ( emp1 == emp2 ) {
11        System.out.println("emp1 is identical to emp2");
12      } else {
13        System.out.println("emp1 is not identical to emp2");
14      }
15
16      if ( emp1.equals(emp2) ) {
17        System.out.println("emp1 is equal to emp2");
18      } else {
19        System.out.println("emp1 is not equal to emp2");
20      }
21
22      emp2 = emp1;
23      System.out.println("set emp2 = emp1;");
24      if ( emp1 == emp2 ) {
25        System.out.println("emp1 is identical to emp2");
26      } else {
27        System.out.println("emp1 is not identical to emp2");
28      }
29    }
30  }
```

generates the output:

```
emp1 is not identical to emp2
emp1 is equal to emp2
set emp2 = emp1;
emp1 is identical to emp2
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

`toString` Method

- Converts an object to a `String`
- Used during string concatenation
- Override this method to provide information about a user-defined object in readable format
- Primitive types are converted to a `String` using the wrapper class's `toString` static method

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Wrapper Classes

- Look at primitive data elements as objects

Primitive Data Type	Wrapper Class
`boolean`	`Boolean`
`byte`	`Byte`
`char`	`Character`
`short`	`Short`
`int`	`Integer`
`long`	`Long`
`float`	`Float`
`double`	`Double`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Wrapper Classes

```
int pInt = 500;
Integer wInt = new Integer(pInt);

int p2 = wInt.intValue();
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Using Objects and Classes

- Exercise objective:
  - Write, compile, and run two programs that use the object-oriented concepts of inheritance and heterogeneous collections within the Banking project
- Tasks:
  - Create several account types
  - Implement a heterogeneous collection of accounts within the customer

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Define *inheritance*, *polymorphism*, *overloading*, *overriding*, and *virtual method invocation*
- Use the access modifiers `protected` and "package-friendly"
- Describe constructor and method overloading
- Describe the complete object construction and initialization operation

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- In a Java program, identify the following:
  - Overloaded methods and constructors
  - The use of this to all overloaded constructors
  - Overridden methods
  - Invocation of super class methods
  - Parent class constructors
  - Invocation of parent class constructors

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Think Beyond

- Now that you understand inheritance and polymorphism, how can you use this information on a current or future project?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Module 7

Advanced Class Features

Objectives

- Describe `static` variables, methods, and initializers
- Describe `final` classes, methods, and variables
- Explain how and when to use `abstract` classes and methods
- Explain how and when to use inner classes
- Distinguish between static and non-static inner classes
- Explain how and when to use an `interface`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Objectives

- In a Java software program, identify:
  - `static` methods and attributes
  - `final` methods and attributes
  - inner classes
  - `interface` and `abstract` classes
  - `abstract` methods

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Relevance

- How can you create a constant?
- How can you create an instance variable that is set once and can not be reset, even internally?
- How can you declare data that is shared by all instances of a given class?
- How can you keep a class or method from being subclassed or overridden?
- How can you create several classes that implement a common interface yet not be part of a common inheritance tree?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

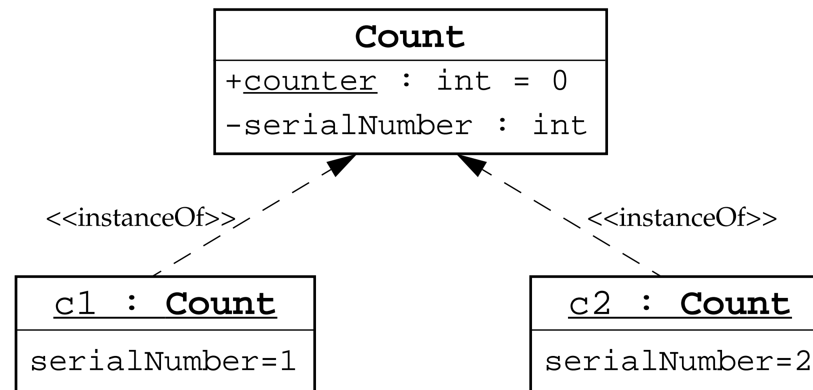
The `static` Keyword

- The `static` keyword is used as a modifier on variables, methods, and inner classes
- The `static` keyword declares the attribute or method is associated with the class as a whole rather than any particular instance of that class
- Thus static members are often called "class members," such as "class attributes" or "class methods"

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Class Attributes

- Are shared among all instances of a class




```
1   public class Count {
2     private int serialNumber;
3     public static int counter = 0;
4
5     public Count() {
6       counter++;
7       serialNumber = counter;
8     }
9   }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Class Attributes

- Can be accessed from outside the class if marked as `public` without an instance of the class

```
1   public class OtherClass {
2     public void incrementNumber() {
3       Count.counter++;
4     }
5   }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Class Methods

- You can invoke `static` method without any instance of the class to which it belongs.

```
1   public class Count {
2     private int serialNumber;
3     private static int counter = 0;
4
5     public static int getTotalCount() {
6       return counter;
7     }
8
9     public Count() {
10      counter++;
11      serialNumber = counter;
12    }
13  }
```

```
1   public class TestCounter {
2     public static void main(String[] args) {
3       System.out.println("Number of counter is "
4                          + Count.getTotalCount());
5       Count count1 = new Count();
6       System.out.println("Number of counter is "
7                          + Count.getTotalCount());
8     }
9   }
```

The output of the `TestCounter` program is:

```
Number of counter is 0
Number of counter is 1
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Static Initializers

- A class can contain code in a *static block* that does not exist within a method body
- Static block code executes only once, when the class is loaded
- A static block is usually used to initialize static (class) attributes

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Static Initializers

```
1   public class Count {
2     public static int counter;
3     static {
4       counter = Integer.getInteger("myApp.Count.counter").intValue();
5     }
6   }
```

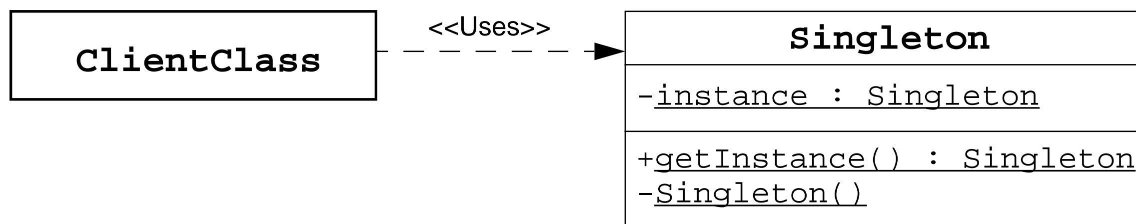
```
1   public class TestStaticInit {
2     public static void main(String args[]) {
3       System.out.println("counter = " + Count.counter);
4     }
5   }
```

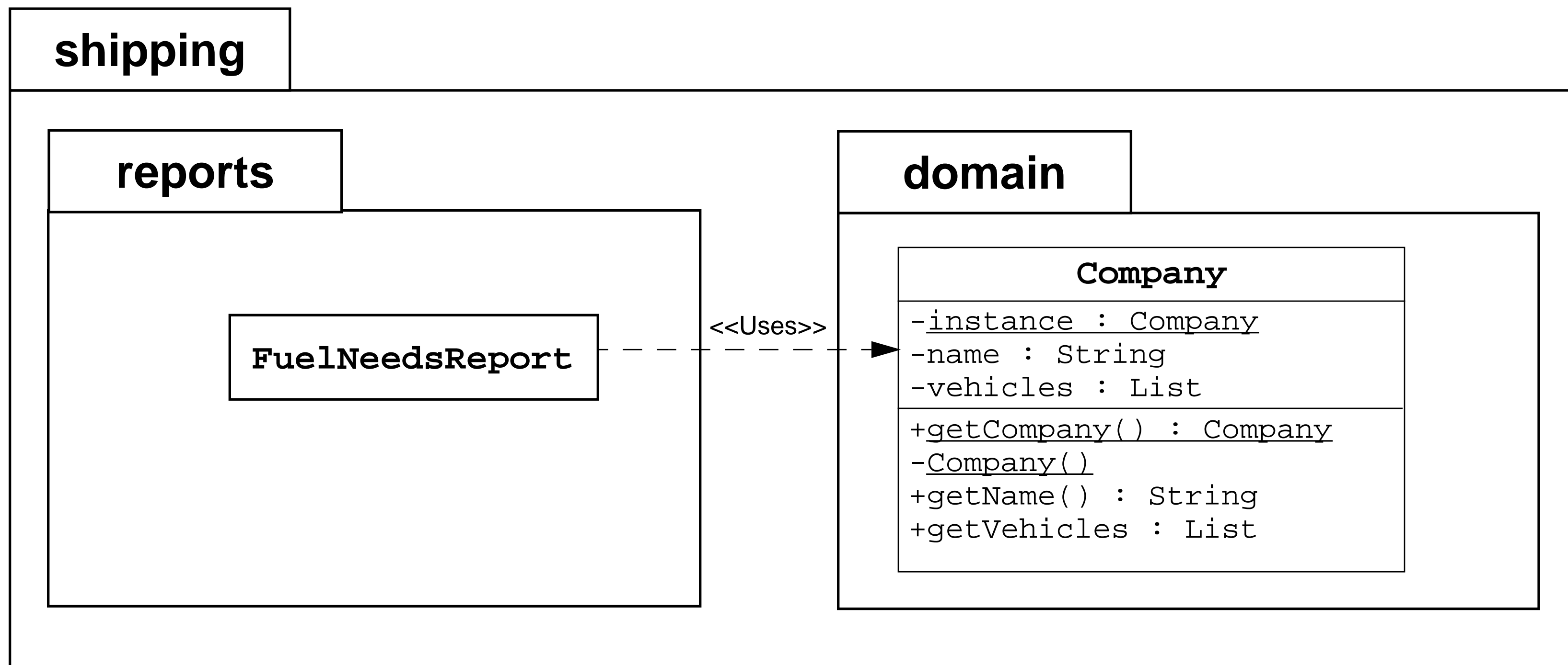
The output of the `TestStaticInit` program is:

```
> java -DmyApplication.counter=47 TestStaticInit
counter = 47
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The Singleton Design Pattern





Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Implementing the Singleton Design Pattern

The Singleton code:

```
1  package shipping.domain;
2
3  public class Company {
4    private static Company instance = new Company();
5    private String name;
6    private Vehicle[] fleet;
7
8    public static Company getCompany() {
9      return instance;
10   }
11
12   private Company() {...}
13
14   // more Company code ...
15 }
```

Usage code:

```
1  package shipping.reports;
2
3  import shipping.domain.*;
4
5  public class FuelNeedsReport {
6    public void generateText(PrintStream output) {
7      Company c = Company.getCompany();
8      // use Company object to retrieve the fleet vehicles
9    }
10 }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The `final` Keyword

- You cannot subclass a `final` class
- You cannot override a `final` method
- A `final` variable is a constant
- A `final` variable can only be set once, but that assignment can occur independently of the declaration; this is called "blank final variable"
  - A blank final instance attribute must be set in every constructor
  - A blank final method variable must be set in the method body before being used

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Final Variables

Constants:

```
public class Bank {
  private static final double  DEFAULT_INTEREST_RATE=3.2; // percent
  ... // more declarations
}
```

Blank Final Instance Attribute:

```
public class Customer {
  private final long  customerID;

  public Customer() {
    customerID = createID();
  }
  public long getID() {
    return customerID;
  }
  private long createID() {
    return ... // generate new ID
  }
  ... // more declarations
}
```

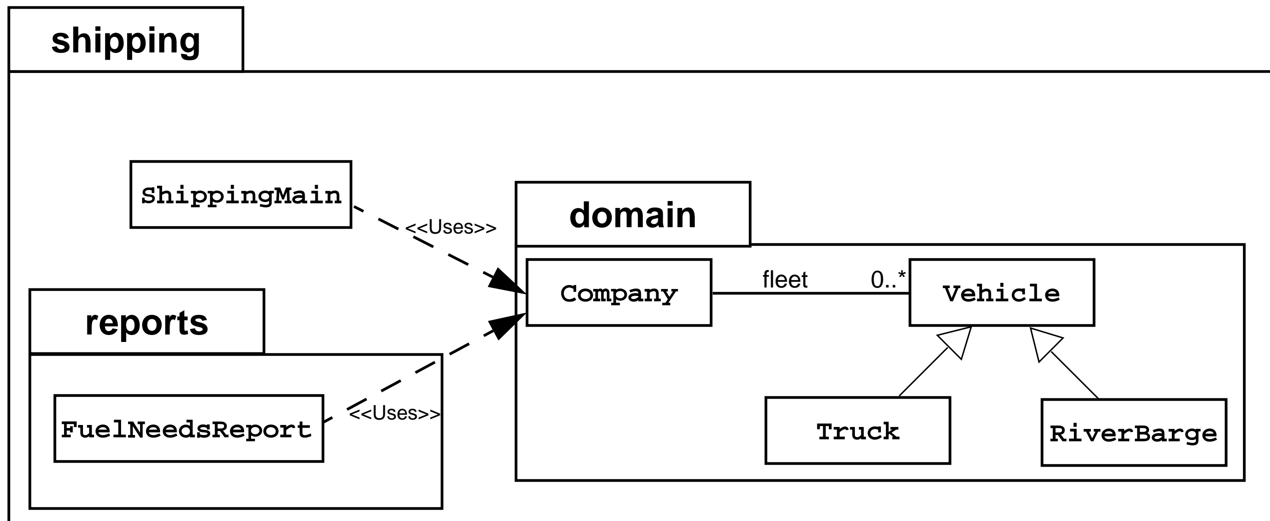

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Working With the `static` and `final` Keywords

- Preparation:
  - You must be familiar with the use of the `static` and `final` keywords
- Task:
  - In this exercise you will modify the `Bank` class to implement the Singleton design pattern

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Abstract Classes: Scenario



Fleet initialization code:

```
1  public class ShippingMain {
2    public static void main(String[] args) {
3      Company c = Company.getCompany();
4
5      // populate the company with a fleet of vehicles
6      c.addVehicle( new Truck(10000.0) );
7      c.addVehicle( new Truck(15000.0) );
8      c.addVehicle( new RiverBarge(500000.0) );
9      c.addVehicle( new Truck(9500.0) );
10     c.addVehicle( new RiverBarge(750000.0) );
11
12     FuelNeedsReport report = new FuelNeedsReport();
13     report.generateText(System.out);
14   }
15 }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Abstract Classes: Scenario

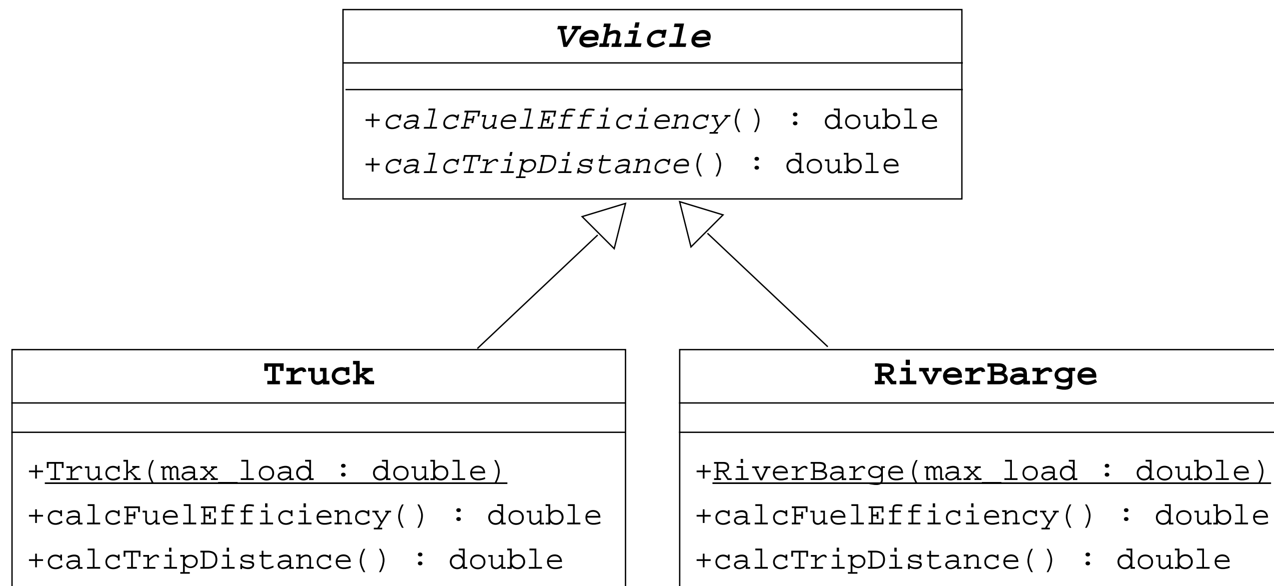
`FuelNeedsReport` code:

```
1  public class FuelNeedsReport {
2    public void generateText(PrintStream output) {
3      Company c = Company.getCompany();
4      Vehicle v;
5      double fuel;
6      double total_fuel = 0.0;
7
8      for ( int i = 0; i < c.getFleetSize(); i++ ) {
9        v = c.getVehicle(i);
10
11       // Calculate the fuel needed for this trip
12       fuel = v.calcTripDistance() / v.calcFuelEfficency();
13
14       output.println("Vehicle " + v.getName() + " needs "
15                      + fuel + " liters of fuel.");
16       total_fuel += fuel;
17     }
18     output.println("Total fuel needs is " + total_fuel + " liters.");
19   }
20 }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Abstract Classes: Solution

- An abstract class is used to model a class of objects where the full implementation is not known, but is supplied by the concrete subclasses



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Abstract Classes: Solution

```
1  public abstract class Vehicle {
2    public abstract double calcFuelEfficiency();
3    public abstract double calcTripDistance();
4  }

1  public class Truck extends Vehicle {
2    public Truck(double max_load) {...}
3
4    public double calcFuelEfficiency() {
5      /* calculate the fuel consumption of a truck at a given load */
6    }
7    public double calcTripDistrance() {
8      /* calculate the distance of this trip on highway */
9    }
10 }

1  public class RiverBarge extends Vehicle {
2    public RiverBarge(double max_load) {...}
3
4    public double calcFuelEfficiency() {
5      /* calculate the fuel efficiency of a river barge */
6    }
7    public double calcTripDistrance() {
8      /* calculate the distance of this trip along the river-ways */
9    }
10 }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Template Method Design Pattern

Vehicle
-load : double = 0 -maxLoad : double = 0
#Vehicle(max_load : double) +getLoad() : double +getMaxLoad() : double +addBox(weight : double) +calcFuelNeeds() : double #*calcFuelEfficiency*() : double #*calcTripDistance*() : double

Note (attached to calcFuelNeeds): This is a Template Method that uses `calcFuelEfficiency` and `calcTripDistance` to determine the fuel needs for the complete shipping trip.

Truck
+Truck(max_load : double) #calcFuelEfficiency() : double #calcTripDistance() : double

RiverBarge
+RiverBarge(max_load : double) #calcFuelEfficiency() : double #calcTripDistance() : double

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Interfaces

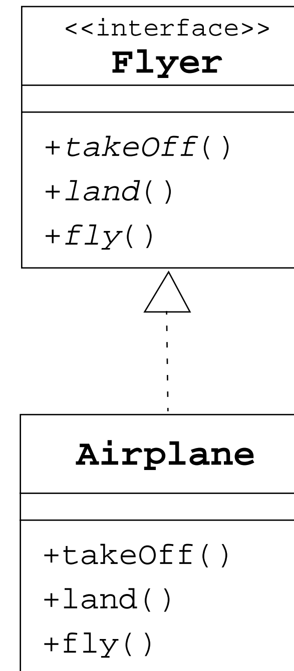
- A "public interface" is a contract between client code and the class that implements that interface
- A Java *interface* is a formal declaration of such a contract in which all methods contain no implementation
- Many, unrelated classes can implement the same interface
- A class can implement many, unrelated interfaces
- Syntax of a Java class:

```
<class_declaration> ::=
  <modifier> class <name> [extends <superclass>]
        [implements <interface> [,<interface>]* ] {
    <declarations>*
  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

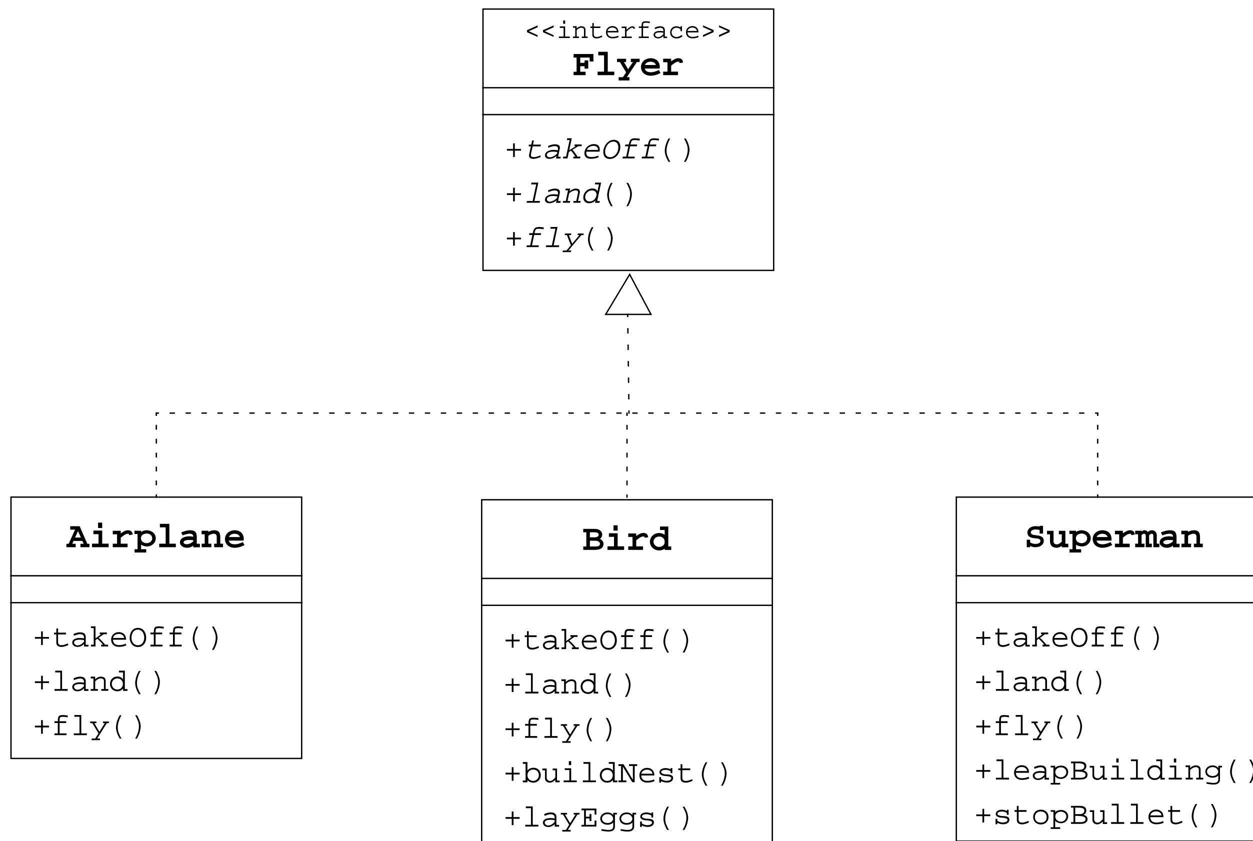
Interface Example

```
public interface Flyer {
  public void takeOff();
  public void land();
  public void fly();
}

public class Airplane implements Flyer {
  public void takeOff() {
    // accelerate until lift-off
    // raise landing gear
  }
  public void land() {
    // lower landing gear
    // deccelerate and lower flaps until touch-down
    // apply breaks
  }
  public void takeOff() {
    // keep those engines running
  }
}
```



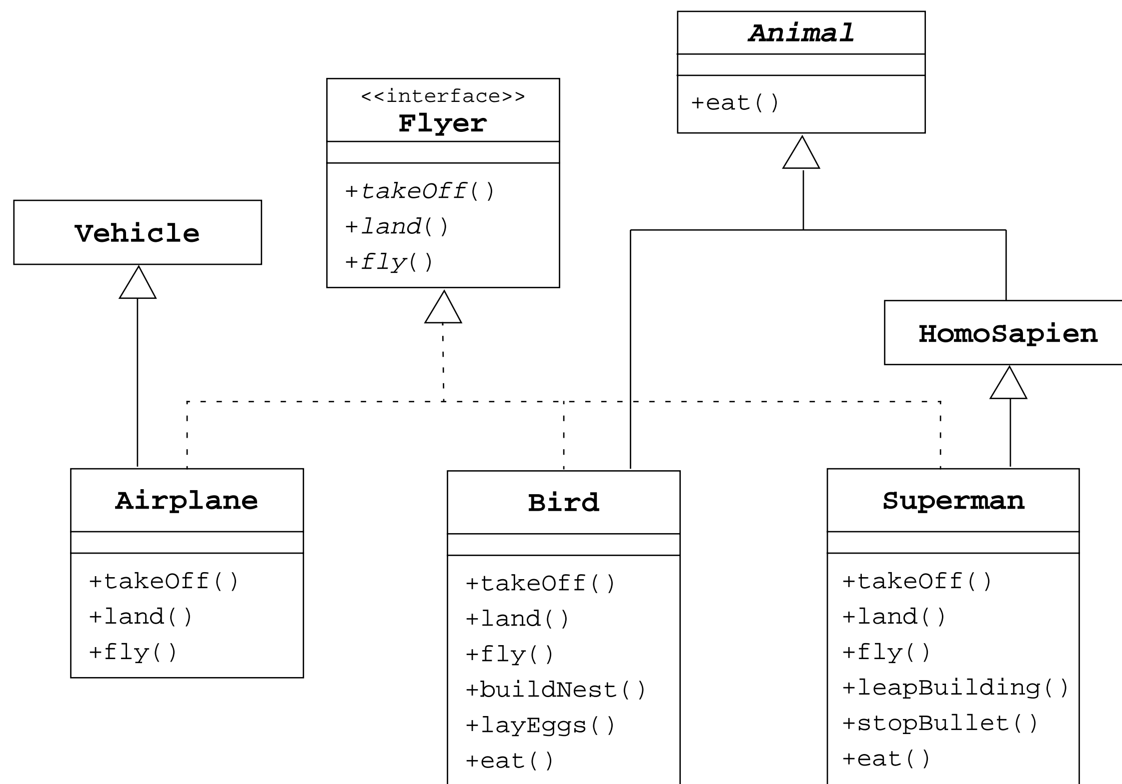
Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Interface Example



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Interface Example



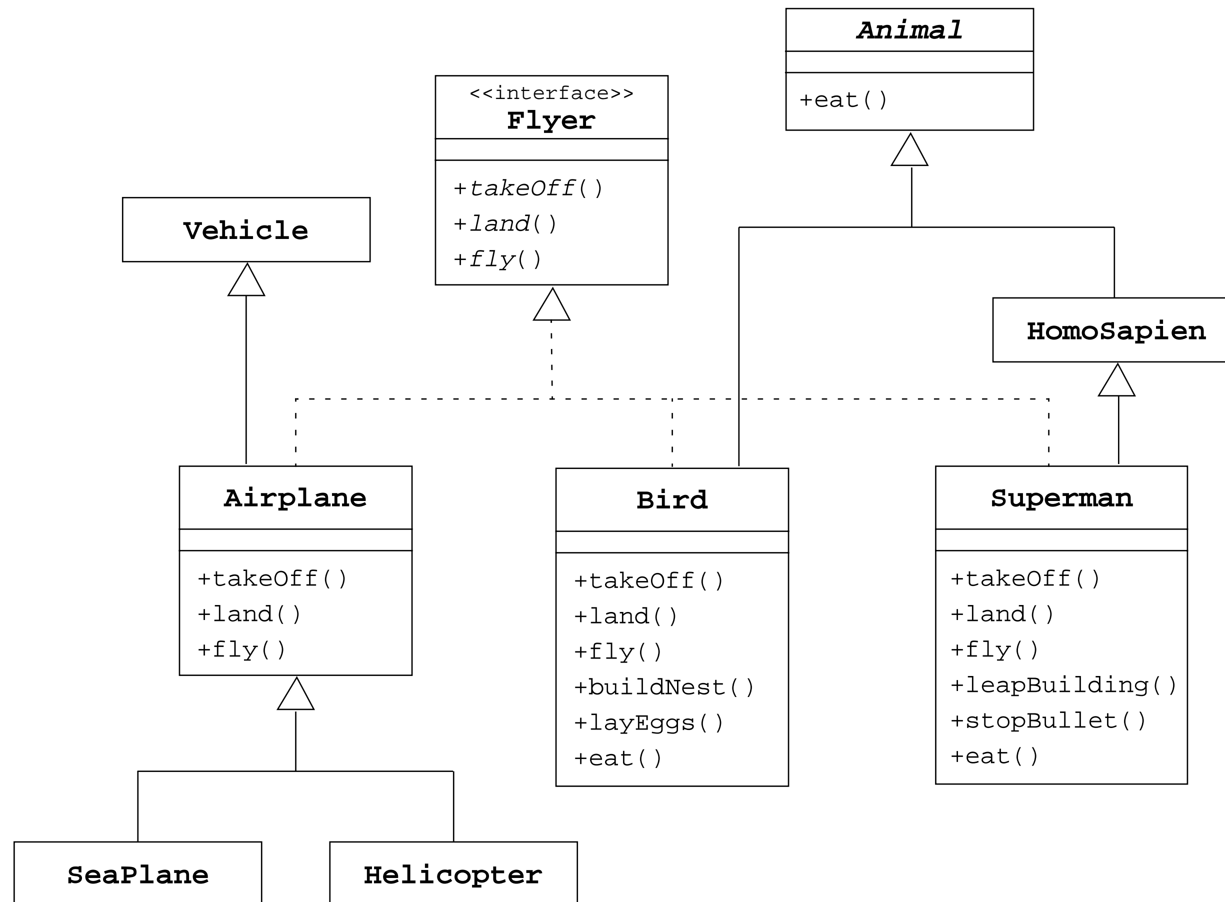
Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Interface Example

```
public class Bird extends Animal implements Flyer {
  public void takeOff()   { /* take-off implementation  */ }
  public void land()      { /* landing implementation   */ }
  public void fly()       { /* fly implementation       */ }
  public void buildNest() { /* nest building behavior   */ }
  public void layEggs()   { /* egg laying behavior      */ }
  public void eat()       { /* override eating behavior */ }
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Interface Example



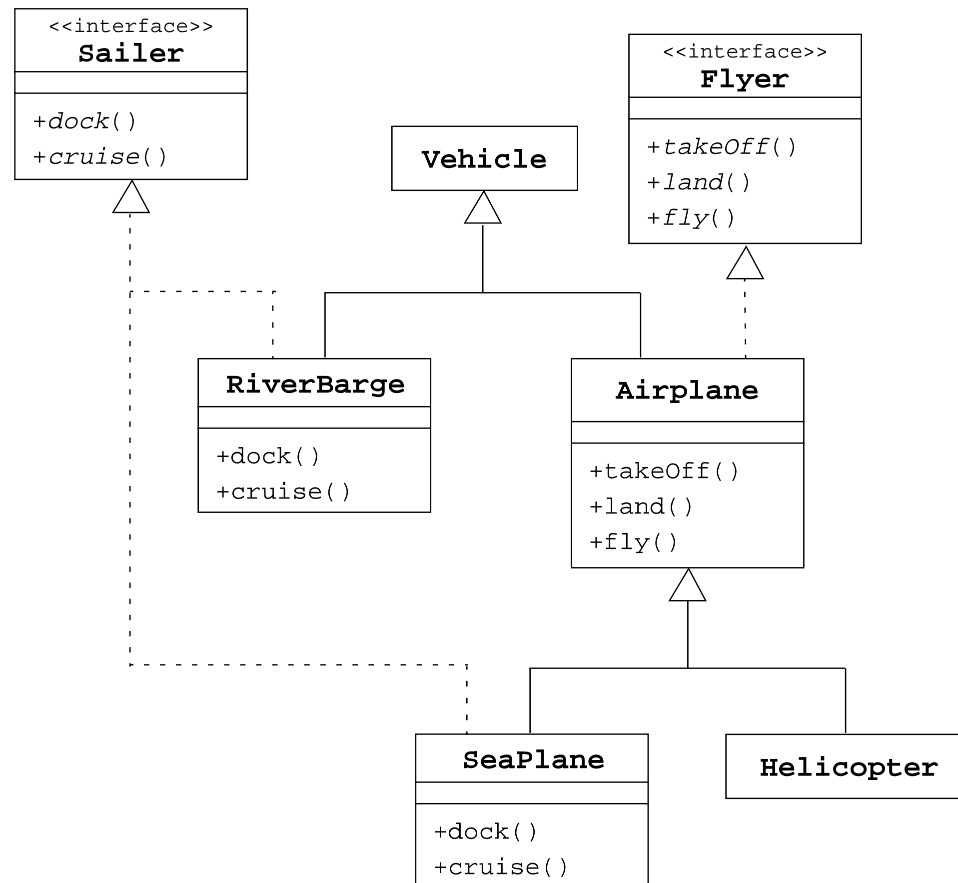
Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Interface Example

```
public class Airport {
  public static void main(String[] args) {
    Airport metropolisAirport = new Airport();
    Helicopter copter = new Helicopter();
    SeaPlane sPlane = new SeaPlane();
    Flyer S = Superman.getSuperman(); // Superman is a Singleton

    metropolisAirport.givePermissionToLand(copter);
    metropolisAirport.givePermissionToLand(sPlane);
    metropolisAirport.givePermissionToLand(S);
  }

  private void givePermissionToLand(Flyer f) {
    f.land();
  }
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Multiple Interface Example



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Multiple Interface Example

```
public class Harbor {
  public static void main(String[] args) {
    Harbor bostonHarbor = new Harbor();
    RiverBarge barge = new RiverBarge();
    SeaPlane sPlane = new SeaPlane();

    bostonHarbor.givePermissionToDock(barge);
    bostonHarbor.givePermissionToDock(sPlane);
  }

  private void givePermissionToDock(Sailer s) {
    s.dock();
  }
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Uses of Interfaces

- Declaring methods that one or more classes are expected to implement
- Determining an object's programming interface without revealing the actual body of the class
- Capturing similarities between unrelated classes without forcing a class relationship
- Simulating multiple inheritance by declaring a class that implements several interfaces

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

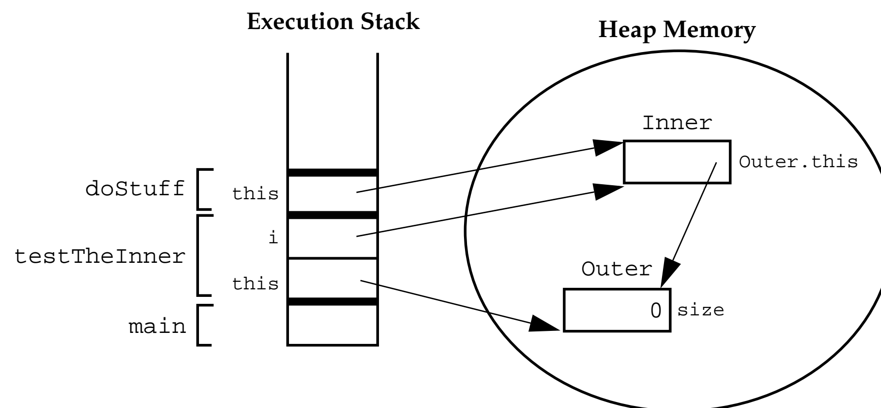
Inner Classes

- Added to JDK 1.1
- Allow a class definition to be placed inside another class definition
- Group classes that logically belong together
- Have access to their enclosing class's scope

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

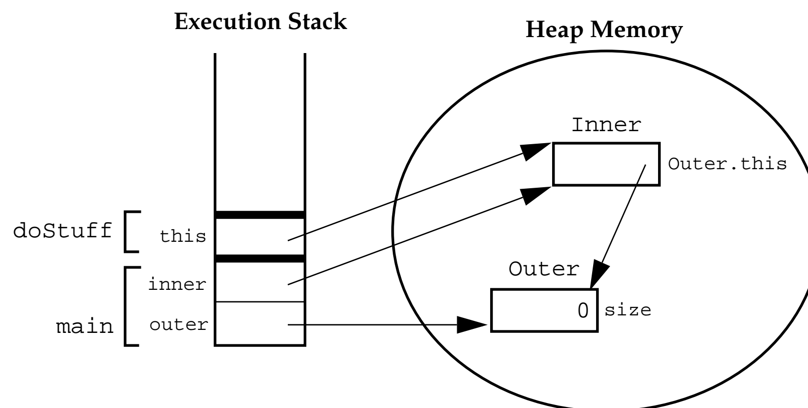
Inner Class Example

```
1   public class Outer1 {
2     private int size;
3
4     /* Declare an inner class called "Inner" */
5     public class Inner {
6       public void doStuff() {
7         // The inner class has access to 'size' from Outer
8         size++;
9       }
10    }
11
12    public void testTheInner() {
13      Inner i = new Inner();
14      i.doStuff();
15    }
16  }
```



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Inner Class Example



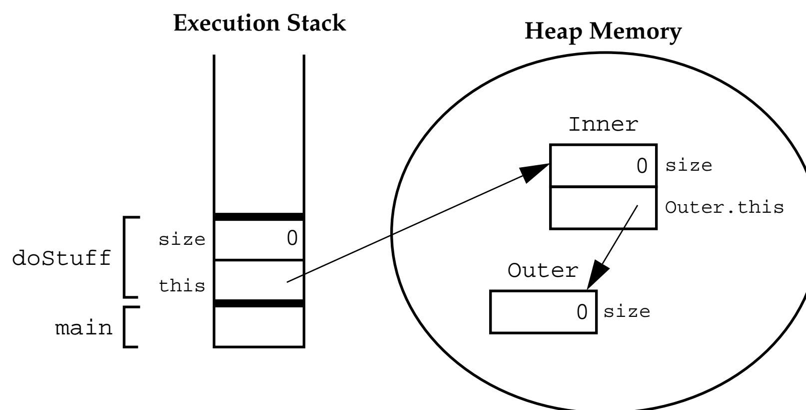

```
1   public class Outer2 {
2     private int size;
3
4     public class Inner {
5       public void doStuff() {
6         size++;
7       }
8     }
9   }
```

```
1   public class TestInner {
2     public static void main(String[] args) {
3       Outer2 outer = new Outer2();
4
5       // Must create an Inner object relative to an Outer
6       Inner inner = outer.new Inner();
7       inner.doStuff();
8     }
9   }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Inner Class Example

```
1   public class Outer3 {
2     private int size;
3
4     public class Inner {
5       private int size;
6
7       public void doStuff(int size) {
8         size++;              // the local parameter
9         this.size++;         // the Inner object attribute
10        Outer.this.size++;   // the Outer object attribute
11      }
12    }
13  }
```



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Inner Class Example

```
1   public class Outer4 {
2     private int size = 5;
3
4     public Object makeTheInner(int localVar) {
5       final int finalLocalVar = 6;
6
7       // Declare a class within a method!?!
8       class Inner {
9         public String toString() {
10          return ("#<Inner size=" + size +
11                  // " localVar=" + localVar + // ERROR: ILLEGAL
12                  "finalLocalVar=" + finalLocalVar + ">");
13        }
14      }
15
16      return new Inner();
17    }
18
19    public static void main(String[] args) {
20      Outer4 outer = new Outer4();
21      Object obj = outer.makeTheInner(47);
22      System.out.println("The object is " + obj);
23    }
24  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Properties of Inner Classes

- You can use the class name only within the defined scope, except when used in a qualified name. The name of the inner class must differ from the enclosing class
- The inner class can be defined inside a method. Only local variables marked as `final`, can be accessed by methods within an inner class.

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Properties of Inner Classes

- The inner class can use both class and instance variables of enclosing classes and local variables of enclosing blocks
- The inner class can be defined as `abstract`
- The inner class can have any access mode
- The inner class can act as an interface implemented by another inner class

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Properties of Inner Classes

- Inner classes that are declared `static` automatically become top-level classes
- Inner classes cannot declare any `static` members; only top-level classes can declare `static` members
- An inner class wanting to use a `static` member must be declared `static`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Working With Interfaces and Abstract Classes

- Exercise objective:
  - Rewrite, compile, and run a program that uses an abstract class and an interface
- Tasks:
  - In this exercise you will create a hierarchy of animals that is rooted in an abstract class `Animal`. Several of the animal classes will implement an interface called `Pet`. You will experiment with variations of these animals, their methods, and polymorphism.

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Describe `static` variables, methods, and initializers
- Describe `final` classes, methods, and variables
- Explain how and when to use `abstract` classes and methods
- Explain how and when to use inner classes
- Distinguish between static and non-static inner classes
- Explain how and when to use an `interface`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- In a Java software program, identify:
  - `static` methods and attributes
  - `final` methods and attributes
  - inner classes
  - `interface` and `abstract` classes
  - `abstract` methods

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Think Beyond

- What features of the Java programming language are used to deal with runtime error conditions?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Module 8

Exceptions

Objectives

- Define exceptions
- Use `try`, `catch`, and `finally` statements
- Describe exception categories
- Identify common exceptions
- Develop programs to handle your own exceptions

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Relevance

- In most programming languages, how are runtime errors resolved?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exceptions

- The `Exception` class defines mild error conditions that your program encounters
- Exceptions can occur when:
  - The file you try to open does not exist
  - The network connection is disrupted
  - Operands being manipulated are out of prescribed ranges
  - The class file you are interested in loading is missing
- An error class defines serious error conditions

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exception Example

```
1   public class HelloWorld {
2     public static void main (String args[]) {
3       int i = 0;
4
5       String greetings [] = {
6         "Hello world!",
7         "No, I mean it!",
8         "HELLO WORLD!!"
9       };
10
11      while (i < 4) {
12        System.out.println (greetings[i]);
13        i++;
14      }
15    }
16  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

`try` and `catch` Statements

```
1   try {
2      // code that might throw a particular exception
3   } catch (MyExceptionType myExcept) {
4      // code to execute if a MyExceptionType exception is thrown
5   } catch (Exception otherExcept) {
6      // code to execute if a general Exception exception is thrown
7   }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Call Stack Mechanism

- If an exception is not handled in the current `try-catch` block, it is thrown to the caller of that method.
- If the exception gets back to the main method and is not handled there, the program is terminated abnormally.

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

finally Statement

```
1      try {
2        startFaucet();
3        waterLawn();
4      } catch (BrokenPipeException e) {
5        logProblem(e);
6      } finally {
7        stopFaucet();
8      }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exception Example Revisited

```
1   public class HelloWorld2 {
2     public static void main (String args[]) {
3       int i = 0;
4
5       String greetings [] = {
6         "Hello world!",
7         "No, I mean it!",
8         "HELLO WORLD!!"
9       };
10
11      while (i < 4) {
12        try {
13          System.out.println (greetings[i]);
14        } catch (ArrayIndexOutOfBoundsException e){
15          System.out.println("Re-setting Index Value");
16          i = -1;
17        } finally {
18          System.out.println("This is always printed");
19        }
20
21        i++;
22      }
23    }
24  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exception Categories

- Throwable
  - Error
    - VirtualMachineError
      - StackOverflowError
      - ...
      - OutOfMemoryError
    - ...
    - AWTError
  - Exception
    - RuntimeException
      - ArithmeticException
      - ...
      - NullPointerException
      - IndexOutOfBoundsException
    - ...
    - IOException
      - EOFException
      - ...
      - FileNotFoundException

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Common Exceptions

- `ArithmeticException`
- `NullPointerException`
- `NegativeArraySizeException`
- `ArrayIndexOutOfBoundsException`
- `SecurityException`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The Handle or Declare Rule

- Handle the exception by using the `try-catch-finally` block
- Declare that the code causes an exception by using the `throws` clause
- A method may declare that it throws more than one exception

```
1   public void readDatabaseFile(String file)
2          throws FileNotFoundException, UTFDataFormatException {
3     // open file stream; may cause FileNotFoundException
4     FileInputStream fis = new FileInputStream(file);
5     // read a string from fis may cause UTFDataFormatException...
6   }
```

- You do not need to handle or declare runtime exceptions or errors

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Method Overriding and Exceptions

- Must throw exceptions that are the same class as the exceptions being thrown by the overridden method
- May throw exceptions that are subclasses of the exceptions being thrown by the overridden method
- If a superclass method throws multiple exceptions, the overriding method must throw a proper subset of exceptions thrown by the overridden method

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Method Overriding Examples

```
1   public class TestA {
2     public void methodA() throws RuntimeException {
3       // do some number crunching
4     }
5   }
```

```
1   public class TestB1 extends TestA {
2     public void methodA() throws ArithmeticException {
3       // do some number crunching
4     }
5   }
```

```
1   public class TestB2 extends TestA {
2     public void methodA() throws Exception {
3       // do some number crunching
4     }
5   }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Method Overriding Examples

```
1   import java.io.*;
2
3   public class TestMultiA {
4     public void methodA()
5          throws IOException, RuntimeException {
6       // do some IO stuff
7     }
8   }
```

```
1   import java.io.*;
2
3   public class TestMultiB1 extends TestMultiA {
4     public void methodA()
5          throws FileNotFoundException, UTFDataFormatException,
6            ArithmeticException {
7       // do some IO and number crunching stuff
8     }
9   }
```

```
1   import java.io.*;
2   import java.sql.*;
3
4   public class TestMultiB2 extends TestMultiA {
5     public void methodA()
6          throws FileNotFoundException, UTFDataFormatException,
7            ArithmeticException, SQLException {
8       // do some IO, number crunching, and SQL stuff
9     }
10  }
```

```
1   public class TestMultiB3 extends TestMultiA {
2     public void methodA() throws java.io.FileNotFoundException {
3       // do some file IO
4     }
5   }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Creating Your Own Exceptions

```
1   public class ServerTimedOutException extends Exception {
2     private int port;
3
4     public ServerTimedOutException(String message, int port) {
5       super(message);
6       this.port = port;
7     }
8
9   // Use getMessage method to get the reason the exception was made
10
11    public int getPort() {
12      return port;
13    }
14  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Handling User-Defined Exceptions

```
1   public void connectMe(String serverName)
2       throws ServerTimedOutException {
3     int success;
4     int portToConnect = 80;
5
6     success = open(serverName, portToConnect);
7
8     if (success == -1) {
9       throw new ServerTimedOutException("Could not connect",
10                                        portToConnect);
11    }
12  }
```

```
1   public void findServer() {
2       try {
3         connectMe(defaultServer);
4       } catch (ServerTimedOutException e) {
5         System.out.println("Server timed out, trying alternative");
6         try {
7           connectMe(alternativeServer);
8         } catch (ServerTimedOutException e1) {
9           System.out.println("Error: " + e1.getMessage() +
10                              " connecting to port " + e1.getPort());
11        }
12      }
13    }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Working With Exceptions

- Exercise objective:
  - Write, compile, and run a program that catches an exception. Write, compile, and run a program that uses a user-defined exception.
- Tasks:
  - In this exercise you will use the try-catch block to handle a simple run-time exception.
  - In this exercise you will create an `OverdraftException` that is thrown by the `withdraw` method in the `Account` class.

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Define exceptions
- Use `try`, `catch`, and `finally` statements
- Describe exception categories
- Identify common exceptions
- Develop programs to handle your own exceptions

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Think Beyond

- How many situations can you think of that would require you to create new classes of exceptions?
- Can you think of situations where a constructor would throw an exception?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

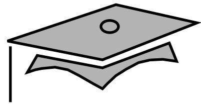
Module 9

Text-Based Applications

Objectives

- Write a program that uses command-line arguments and system properties
- Write a program that reads from *standard input*
- Write a program that can create, read, and write files
- Describe the basic hierarchy of collections in Java 2 SDK
- Write a program that uses sets and lists
- Write a program to iterate over a collection

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D



Objectives

- Write a program to sort an array or a list
- Describe the collection classes that existed before Java 2 SDK
- Describe and use the `javadoc` and `jar` tools
- Identify deprecated classes and explain how to migrate from JDK 1.0 to JDK 1.1 to Java 2 JDK

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Relevance

- It is often the case that certain elements of a program should not be hardcoded, such as file names or the name of a database. How can a program be coded to supply these elements at runtime?
- Simple arrays are far too static for most collections (that is, a fixed number of elements). What Java technology features exist to support more flexible collections?
- Besides computation, what are key elements of any text-based application?
- Documentation is a key source of technology transfer. What Java technology tools support package and class API documentation?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Command-Line Arguments

- Any Java technology application can use command-line arguments
- These string arguments are placed on the command line to launch the Java interpreter, after the class name:

```
java TestArgs arg1 arg2 "another arg"
```

- Each command-line argument is placed in the `args` array that is passed to the static `main` method:

```
public static main(String[] args)
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Command-Line Args

```
1   public class TestArgs {
2     public static void main(String[] args) {
3       for ( int i = 0; i < args.length; i++ ) {
4         System.out.println("args[" + i + "] is '" + args[i] + "'");
5       }
6     }
7   }
```

```
> java TestArgs arg1 arg2 "another arg"
```

Here is an excerpt of the output:

```
args[0] is 'arg1'
args[1] is 'arg2'
args[2] is 'another arg'
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

System Properties

- System properties is a feature that replaces the concept of *environment variables* (which is platform-specific)
- The `System.getProperties` method returns a `Properties` object
- The `getProperty` method returns a `String` representing the value of the named property
- Use the `-D` option to include a new property

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The `Properties` Class

- The `Properties` class implements a mapping of names to values (a `String` to `String` map)
- The `propertyNames` method returns an `Enumeration` of all property names
- The `getProperty` method returns a `String` representing the value of the named property
- You can also read and write a properties collection into a file using `load` and `store`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

System Properties

```
1   import java.util.Properties;
2   import java.util.Enumeration;
3
4   public class TestProperties {
5     public static void main(String[] args) {
6       Properties props = System.getProperties();
7       Enumeration prop_names = props.propertyNames();
8
9       while ( prop_names.hasMoreElements() ) {
10        String prop_name = (String) prop_names.nextElement();
11        String property = props.getProperty(prop_name);
12        System.out.println("property '" + prop_name
13           + "' is '" + property + "'");
14      }
15    }
16  }
```

```
> java -DmyProp=theValue TestProperties
```

Here is an excerpt of the output:

```
property 'java.vm.version' is '1.2.2'
property 'java.compiler' is 'NONE'
property 'path.separator' is ':'
property 'file.separator' is '/'
property 'user.home' is '/home/basham'
property 'java.specification.vendor' is 'Sun Microsystems Inc.'
property 'user.language' is 'en'
property 'user.name' is 'basham'
property 'myProp' is 'theValue'
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Console I/O

- `System.out` allows you to write to "standard output"
  - It is an object of type `PrintStream`
- `System.in` allows you to read from "standard input"
  - It is an object of type `InputStream`
- `System.err` allows you to write to "standard error"
  - It is an object of type `PrintStream`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Writing to Standard Output

- The `println` methods print the argument and a newline (`\n`)
- The `print` methods print the argument without a newline
- The `print` and `println` methods are overloaded for most primitive types (`boolean`, `char`, `int`, `long`, `float`, and `double`) and for `char[]`, `Object`, and `String`
- The `print(Object)` and `println(Object)` methods call the `toString` method on the argument

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Reading From Standard Input

```
1   import java.io.*;
2
3   public class KeyboardInput {
4     public static void main (String args[]) {
5       String s;
6       // Create a buffered reader to read
7       // each line from the keyboard.
8       InputStreamReader ir = new InputStreamReader(System.in);
9       BufferedReader in = new BufferedReader(ir);
10
11      System.out.println("Type ctrl-d or ctrl-c to exit.");
12
13      try {
14        // Read each input line and echo it to the screen.
15        s = in.readLine();
16        while ( s != null ) {
17          System.out.println("Read: " + s);
18          s = in.readLine();
19        }
20
21        // Close the buffered reader.
22        in.close();
23      } catch (IOException e) { // Catch any IO exceptions.
24        e.printStackTrace();
25      }
26    }
27  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Files and File I/O

- The `java.io` package
- Creating `File` objects
- Manipulating `File` objects
- Reading and writing to file streams

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Creating a New `File` Object

- `File myFile;`
- `myFile = new File("myfile.txt");`
- `myFile = new File("MyDocs", "myfile.txt");`
- Directories are treated just like files in Java; the `File` class supports methods for retrieving an array of files in the directory
- `File myDir = new File("MyDocs");`
  `myFile = new File(myDir, "myfile.txt");`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

`File` Tests and Utilities

- `File` names:

```
String getName()
String getPath()
String getAbsolutePath()
String getParent()
boolean renameTo(File newName)
```

- `File` tests:

```
boolean exists()
boolean canWrite()
boolean canRead()
boolean isFile()
boolean isDirectory()
boolean isAbsolute();
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

`File` Tests and Utilities

- General file information and utilities:

```
long lastModified()
long length()
boolean delete()
```

- Directory utilities:

```
boolean mkdir()
String[] list()
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

File Stream I/O

- File Input:
  - Use the `FileReader` class to read characters
  - Use the `BufferedReader` class to use the `readLine` method
- File Output:
  - Use the `FileWriter` class to write characters
  - Use the `PrintWriter` class to use the `print` and `println` methods

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

File Input Example

```
1   import java.io.*;
2   public class ReadFile {
3     public static void main (String args[]) {
4       // Create file
5       File file = new File(args[0]);
6
7       try {
8         // Create a buffered reader to read each line from a file.
9         BufferedReader in = new BufferedReader(new FileReader(file));
10        String s;
11
12        // Read each line from the file and echo it to the screen.
13        s = in.readLine();
14        while ( s != null ) {
15          System.out.println("Read: " + s);
16          s = in.readLine();
17        }
18        // Close the buffered reader, which also closes the file reader.
19        in.close();
20
21      } catch (FileNotFoundException e1) {
22      // If this file does not exist
23        System.err.println("File not found: " + file);
24
25      } catch (IOException e2) {
26      // Catch any other IO exceptions.
27        e2.printStackTrace();
28      }
29    }
30  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

File Output Example

```
1   import java.io.*;
2
3   public class WriteFile {
4     public static void main (String args[]) {
5       // Create file
6       File file = new File(args[0]);
7
8       try {
9        // Create a buffered reader to read each line from standard
in.
10        BufferedReader in
11         = new BufferedReader(new InputStreamReader(System.in));
12        // Create a print writer on this file.
13        PrintWriter out
14          = new PrintWriter(new FileWriter(file));
15        String s;
16
17        System.out.print("Enter file text.  ");
18        System.out.println("[Type ctrl-d to stop.]");
19
20        // Read each input line and echo it to the screen.
21        while ((s = in.readLine()) != null) {
22          out.println(s);
23        }
24
25        // Close the buffered reader and the file print writer.
26        in.close();
27        out.close();
28
29      } catch (IOException e) {
30      // Catch any IO exceptions.
31        e.printStackTrace();
32      }
33    }
34  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Writing User Input to a File

- Exercise objectives:
  - Become familiar with using command-line arguments
  - Become familiar with reading text from standard input and writing text to a file
- Tasks:
  - Create a program to read text from standard input and write it to a file
  - Create a program to perform a directory listing

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The `Math` Class

The `Math` class contains a group of static math functions:

- truncation: `ceil`, `floor`, and `round`
- variations on `max`, `min`, and `abs` (absolute value)
- trigonometry: `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `toDegrees`, and `toRadians`
- logarithms: `log` and `exp`
- others: `sqrt`, `pow`, and `random`
- constants: `PI` and `E`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The `String` Class

- `String` objects are *immutable* sequences of Unicode characters
- Operations that create new strings: `concat`, `replace`, `substring`, `toLowerCase`, `toUpperCase`, and `trim`
- Search operations: `endsWith`, `startsWith`, `indexOf`, and `lastIndexOf`
- Comparisons: `equals`, `equalsIgnoreCase`, and `compareTo`
- Others: `charAt` and `length`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The `StringBuffer` Class

- `StringBuffer` objects are mutable sequences of Unicode characters
- Constructors:
  - `StringBuffer()` – Creates an empty buffer
  - `StringBuffer(int capacity)` – Creates an empty buffer with a specified initial capacity
  - `StringBuffer(String initialString)` – Creates a buffer that initially contains the specified string
- Modification operations: `append`, `insert`, `reverse`, `setCharAt`, and `setLength`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The Collections API

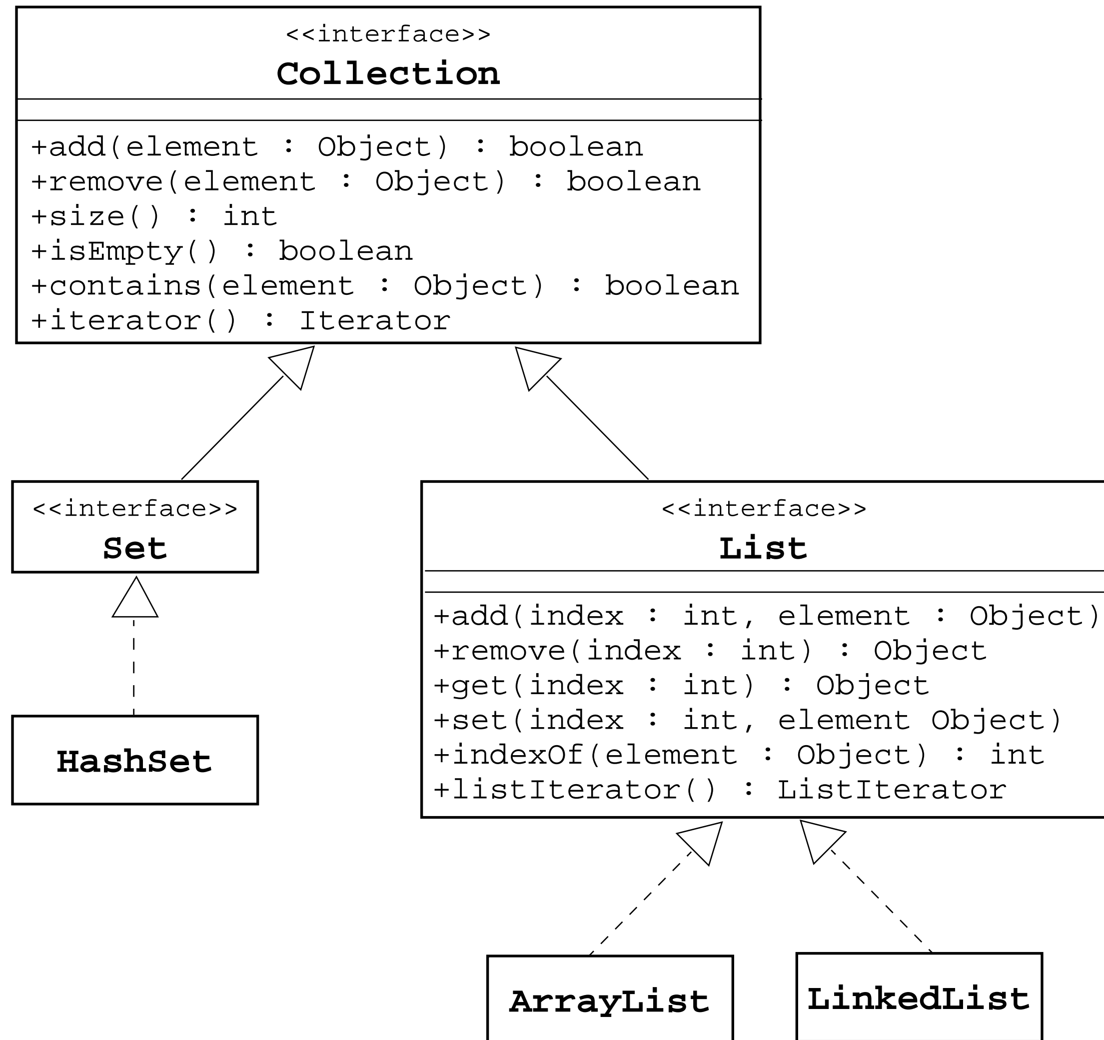
- What is a Collection, Set, and List?
- What are Iterators?
- What are Maps?
- How to sort arrays and collections
- Collections in JDK 1.1

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Collections

- A *collection* is a single object representing a group of objects known as its elements
- The Collection API contains interfaces that group objects as a:
  - `Collection` – A group of objects with no specific ordering; duplicates are permitted
  - `Set` – An unordered collection; no duplicates are permitted
  - `List` – An ordered collection; duplicates are permitted

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

A Set Example

```
1   import java.util.*
2
3   public class SetExample {
4     public static void main(String[] args) {
5       Set set = new HashSet();
6       set.add("one");
7       set.add("second");
8       set.add("3rd");
9       set.add(new Integer(4));
10      set.add(new Float(5.0F));
11      set.add("second");        // duplicate, not added
12      set.add(new Integer(4));  // duplicate, not added
13      System.out.println(set);
14    }
15  }
```

The output generated from this program is:

```
[one, second, 5.0, 3rd, 4]
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

A List Example

```
1   import java.util.*
2
3   public class ListExample {
4     public static void main(String[] args) {
5       List list = new ArrayList();
6       list.add("one");
7       list.add("second");
8       list.add("3rd");
9       list.add(new Integer(4));
10      list.add(new Float(5.0F));
11      list.add("second");        // duplicate, is added
12      list.add(new Integer(4));  // duplicate, is added
13      System.out.println(list);
14    }
15  }
```

The output generated from this program is:

```
[one, second, 3rd, 4, 5.0, second, 4]
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

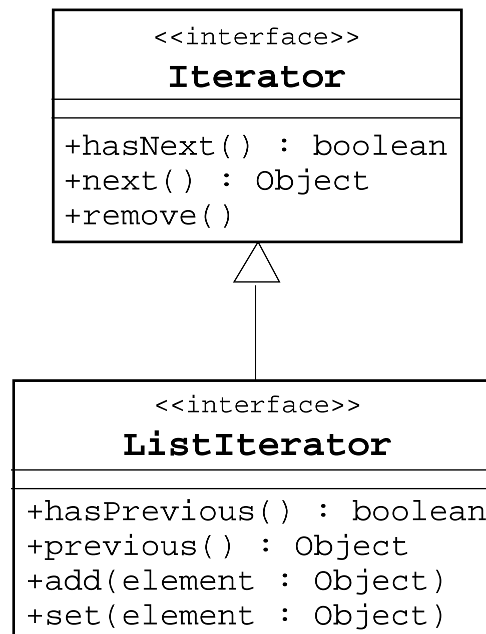
Iterators

- Iteration is the process of retrieving every element in a collection
- An `Iterator` of a `Set` is unordered
- A `ListIterator` of a `List` can be scanned forwards (using the `next` method) or backwards (using the `previous` method)

```
List list = new ArrayList();
// add some elements
Iterator elements = list.iterator();
while ( elements.hasNext() ) {
  System.out.println(elements.next());
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

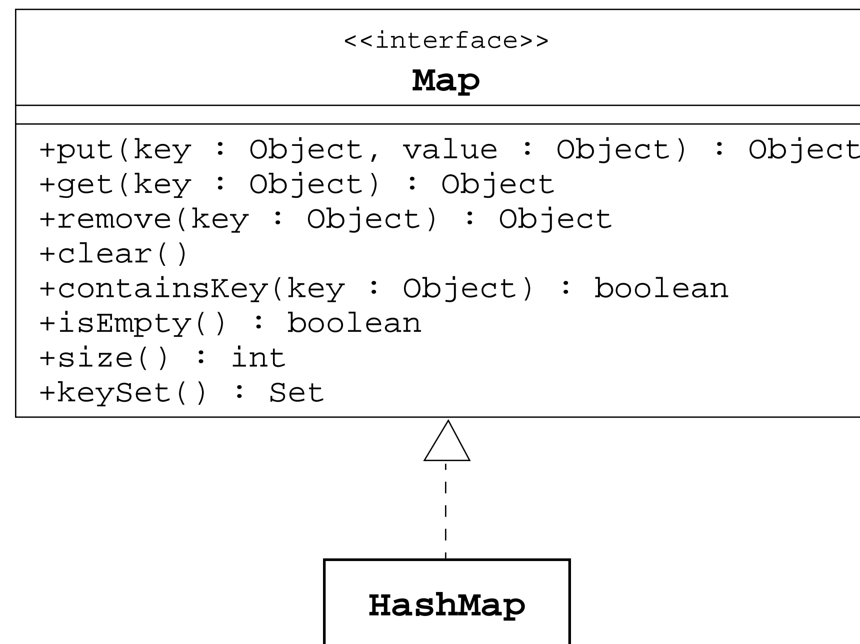
The Iterator Interface Hierarchy



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Maps

- A map is a collection of *key* and *value* pairs, where the keys and values can be any arbitrary `Object`; thus, a heterogeneous mapping



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Map: Word Counter

```
1    import java.util.Map
2    import java.util.HashMap;
3    import java.util.Iterator;
4    import java.io.FileReader;
5
6    public class MapExample {
7      public static void main(String[] args)
8            throws java.io.FileNotFoundException {
9        Map        word_count_map = new HashMap();
10       FileReader reader = new FileReader(args[0]);
11       Iterator   words = new WordStreamIterator(reader);
12
13       while ( words.hasNext() ) {
14         String word = (String) words.next();
15         String word_lowercase = word.toLowerCase();  // this is the key
16         Integer frequency = (Integer)word_count_map.get(word_lowercase);
17
18         if ( frequency == null ) {
19           frequency = new Integer(1);
20         } else {
21           int value = frequency.intValue();
22           frequency = new Integer(value + 1);
23         }
24         word_count_map.put(word_lowercase, frequency);
25       }
26       System.out.println(word_count_map);
27     }
28   }
```

This program might produce the following output:

```
{unclean=1, with=2, scene=1, passage=1, our=3, ancient=1, two=3, these=1,
mark'd=1, patient=1, do=1, cross'd=1, where=2, lovers=1, fatal=1, stage=1,
verona=1, new=1, bury=1, forth=1, strife=1, lay=1, fair=1, we=1, alike=1,
could=1, piteous=1, is=1, hands=1, mend=1, in=2, nought=1, both=1,
continuance=1, life=1, if=1, shall=2, the=5, traffic=1, and=1, a=1, toil=1,
take=1, which=2, loins=1, of=5, here=1, end=1, what=1, civil=2, their=6, love=1,
but=1, makes=1, miss=1, rage=1, foes=1, you=1, ears=1, whose=1, now=1, to=2,
dignity=1, fearful=1, pair=1, star=1, strive=1, households=1, hours'=1,
grudge=1, break=1, misadventured=1, mutiny=1, attend=1, overthrows=1,
parents'=2, blood=1, from=2, children's=1, remove=1, death=2}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Sorting Arrays and Collections

- Sorting arrays using `Arrays.sort` methods:
  - `void sort(`<*type*>` array[])`
  - `void sort(`<*type*>` array[], int fromIndex,`
    `int toIndex)`
    where <*type*> is any primitive type (except `boolean`)
- Sorting lists using `Collections.sort` methods:
  - `void sort(List)`
  - `void sort(List, Comparator)`
- The `Comparable` and `Comparator` interfaces
- Sorting a `Set` using a `SortedSet` implementation

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Sorting Arrays

```
1   import java.util.Arrays;
2   import java.text.DecimalFormat;
3
4   public class SortingExample1 {
5     public static void main(String[] args) {
6       double[]  random_values = new double[10];
7
8       // populate the array with random numbers
9       for ( int i = 0; i < random_values.length; i++ ) {
10        random_values[i] = Math.random();
11      }
12
13      // print out unsorted array
14      System.out.println("Unsorted Array:");
15      printArray(random_values);
16
17      // print out sorted array
18      Arrays.sort(random_values);
19      System.out.println("Sorted Array:");
20      printArray(random_values);
21    }
22
23    private static void printArray(double array[]) {
24      System.out.print('[');
25      for ( int i = 0; i < array.length; i++ ) {
26        System.out.print(FORMAT.format(array[i]));
27        if ( (i + 1) < array.length ) {
28          System.out.print(", ");
29        }
30      }
31      System.out.println(']');
32    }
33    private static DecimalFormat FORMAT
34      = new DecimalFormat("0.000");
35  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Sorting Lists

```
1   import java.util.*;
2
3   public class SortingExample2 {
4     public static void main(String[] args) {
5       Map   word_count_map = new WordCountMap(System.in);
6       Set   entry_set = word_count_map.entrySet();
7
8       System.out.println("Unsorted Entry Set:\n" + entry_set);
9
10      // Create a list of the entries and sort it alphabetically
11      List      entry_list = new ArrayList(entry_set);
12      Collections.sort(entry_list, new AlphaComparator());
13      System.out.println("\nEntry Set (sorted alpha):\n" + entry_list);
14
15      // Sort the list by frequency
16      Collections.sort(entry_list, new FreqComparator());
17      System.out.println("\nEntry Set (sorted by freq):\n" + entry_list);
18    }
19
20    private static class AlphaComparator implements Comparator {
21      public int compare(Object e1, Object e2) {
22        String word1 = (String) ((Map.Entry) e1).getKey();
23        String word2 = (String) ((Map.Entry) e2).getKey();
24        return word1.compareTo(word2);
25      }
26    }
27    private static class FreqComparator implements Comparator {
28      public int compare(Object e1, Object e2) {
29        Integer freq1 = (Integer) ((Map.Entry) e1).getValue();
30        Integer freq2 = (Integer) ((Map.Entry) e2).getValue();
31        return freq2.compareTo(freq1);
32      }
33    }
34  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Collections in JDK 1.1

- `Vector` implements the `List` interface
- `Stack` is a subclass of `Vector` and supports the `push`, `pop`, and `peek` methods
- `Hashtable` implements the `Map` interface
- `Enumeration` is a variation on the `Iterator` interface
  - An enumeration is returned by the `elements` method in `Vector`, `Stack`, and `Hashtable`
- These classes are thread-safe, and therefore, "heavy-weight"

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Using Collections to Represent Aggregation

- Exercise objectives:
  - Become familiar with collections and iterators by rewriting the bank project to use the Java 2 SDK Collections API instead of arrays
- Tasks:
  - Use an `ArrayList` to implement the multiplicity of the association between the `Bank` object and set of customers
  - Sort the bank's list of customers, in lexicographical order by name, to produce an ordered summary report

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Using the javadoc Tool

- This Java 2 SDK tool generates HTML documentation pages
- Usage: `javadoc [options] [packages|files]`

  This example generates the API documentation for the complete Banking project:

```
javadoc -d ../doc/api banking banking.domain /
   banking.reports
```

Option	Value	Description
`-d`	output path	The directory where the generated HTML files should be placed.
`-sourcepath`	directory path	The root directory where the source file package tree.
`-public`		Specifies that only public declarations be included. (*default*)
`-private`		Specifies that all declarations be included.

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Documentation Tags

- Comments starting with `/**` are parsed by the `javadoc` tool
- These comments should immediately precede the declaration they reflect

Tag	Purpose	class/ interface	constructor	method	attribute
`@see`	To create a link to another declaration (or any other HTML page)	✓	✓	✓	✓
`@deprecated`	Documents that the declaration has been deprecated in this release	✓	✓	✓	✓
`@author`	The author of the class or interface	✓			
`@param`	Documents a parameter		✓	✓	
`@throws` `@exception`	Documents why an exception might be thrown		✓	✓	
`@return`	Documents the return value/type			✓	

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Example Java File

```
1    /*
2     * This is an example using javadoc tags.
3     */
4
5    package mypack;
6
7    import java.util.List;
8
9    /**
10    * This class contains a bunch of documentation tags.
11    * @author Bryan Basham
12    * @version 0.5(beta)
13    */
14   public class DocExample {
15
16     /** A simple attribute tag. */
17     private int x;
18
19     /**
20      * This variable a list of stuff.
21      * @see #getStuff()
22      */
23     private List stuff;
24
25     /**
26      * This constructor initializes the x attribute.
27      * @param x_value  the value of x
28      */
29     public DocExample(int x_value) {
30       this.x = x_value;
31     }
32
33     /**
34      * This method return some stuff.
35      * @throws IllegalStateException if no stuff is found
36      * @return List the list of stuff
37      */
38     public List getStuff()
39          throws IllegalStateException {
40       if ( stuff == null ) {
41         throw new java.lang.IllegalStateException("ugh, no stuff");
42       }
43       return stuff;
44     }
45   }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Public Documentation

```
> javadoc -d doc/api/public DocExample.java
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Private Documentation

```
> javadoc -private -d doc/api/private DocExample.java
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Deprecation

- Deprecation makes classes, attributes, methods, constructors, and so on, obsolete
- Obsolete declarations are replaced by methods with a more standardized naming convention
- When migrating code, compile the code with the `-deprecation` flag:

```
javac -deprecation MyFile.java
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Deprecation

JDK 1.1 code, before deprecation is as follows:

```
1   package myutilities;
2
3   import java.util.*;
4   import java.text.*;
5
6   public final class DateConverter {
7     private static final String DAY_OF_THE_WEEK [] =
8       {"Sunday", "Monday", "Tuesday", "Wednesday",
9       "Thursday", "Friday", "Saturday"};
10
11    public static String getDayOfWeek (String theDate){
12      int month, day, year;
13
14      StringTokenizer st = new StringTokenizer (theDate, "/");
15
16      month = Integer.parseInt(st.nextToken ());
17      day = Integer.parseInt(st.nextToken());
18      year = Integer.parseInt(st.nextToken());
19      Date d = new Date (year, month, day);
20
21      return (DAY_OF_THE_WEEK[d.getDay()]);
22    }
23  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Deprecation

Compiling previous code with the `-deprecation` flag yields:

```
% javac -deprecation DateConverter.java
DateConverter.java:19: Note: The constructor java.util.Date(int,int,int) has been
deprecated.
                Date d = new Date (year, month, day);
                         ^
DateConverter.java:21: Note: The method int getDay() in class java.util.Date has
been deprecated.
                return (day_of_the_week[d.getDay()]);
                                               ^
Note: DateConverter.java uses a deprecated API.Please consult the documentation for
a better alternative.
3 warnings
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Deprecation

A Java 2 SDK version rewritten is:

```
1   package myutilities;
2
3   import java.util.*;
4   import java.text.*;
5
6   public final class DateConverter2 {
7     private static String DAY_OF_THE_WEEK[] =
8         {"Sunday", "Monday", "Tuesday", "Wednesday",
9           "Thursday", "Friday", "Saturday"};
10
11    public static String getDayOfWeek (String theDate) {
12      Date d = null;
13      SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yy");
14
15      try {
16        d = sdf.parse (theDate);
17      } catch (ParseException e) {
18        System.out.println (e);
19        e.printStackTrace();
20      }
21
22      // Create a GregorianCalendar object
23      Calendar c =
24          new GregorianCalendar(
25              TimeZone.getTimeZone("EST"),Locale.US);
26      c.setTime (d);
27
28      return(
29          DAY_OF_THE_WEEK[(c.get(Calendar.DAY_OF_WEEK)-1)]);
30    }
31  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Using the jar Tool

- This Java 2 SDK tool generates a compressed archive of `.class` and media files
- Usage: `jar [options] [archive_file] [files]`

This generates an archive for the Banking project:

```
jar cvf banking.jar banking/domain/*.class banking/reports/*.class
```

This extracts an archive for the Banking project:

```
jar xvf banking.jar
```

Option	Value	Description
c		This option creates a new archive.
f	filepath	This option specifies the filepath of the JAR (Java archive) file.
x		This option extracts an archive to the current directory.
v		This option specifies verbose output from the `jar` tool.

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Building a System

- Exercise objectives:
  - Become familiar with the `javadoc` tool
  - Become familiar with the `jar` tool
- Tasks:
  - Use the `javadoc` tool to build a set of HTML pages that document the banking system
  - Use the `jar` tool to build a JAR file that can be used to deploy the banking system

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Write a program that uses command-line arguments and system properties
- Write a program that reads from *standard input*
- Write a program that can create, read, and write files
- Describe the basic hierarchy of collections in Java 2 SDK
- Write a program that uses sets and lists
- Write a program to iterate over a collection

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Write a program to sort an array or a list
- Describe the collection classes that existed before Java 2 SDK
- Describe and use the `javadoc` and `jar` tools
- Identify deprecated classes and explain how to migrate from JDK 1.0 to JDK 1.1 to Java 2 JDK

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Think Beyond

- Most applications are text-based. What other styles of programs exist?
- What features does the Java application environment have that support user interface development?
- How were interfaces used in this module? Could they have been replaced by some other mechanism, such as abstract classes?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Module 10

Building Java GUIs

Objectives

- Describe the AWT package and its components
- Define the terms *containers*, *components*, and *layout managers*, and how they work together to build a graphical user interface (GUI)
- Use layout managers
- Use the `FlowLayout`, `BorderLayout`, `GridLayout`, and `CardLayout` managers to achieve a desired dynamic layout
- Add components to a container
- Use the `Frame` and `Panel` containers appropriately

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Objectives

- Describe how complex layouts with nested containers work
- In a Java program, identify the following:
    - Containers
    - The associated layout managers
    - The layout hierarchy of all components

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Relevance

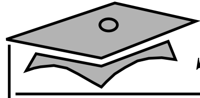
- As a platform-independent programming language, how is Java technology used to make the GUI platform independent?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Abstract Window Toolkit (AWT)

- Provides graphical user interface (GUI) components that are used in all Java applets and applications
- Contains classes that can be extended and their properties inherited; classes can also be abstract
- Ensures that every GUI component that is displayed on the screen is a subclass of the abstract class `Component` or `MenuComponent`
- Has `Container`, which is an abstract subclass of `Component` and includes two subclasses:
  - `Panel`
  - `Window`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D



The `java.awt` Package

java.lang.**Object**

- **BorderLayout**
- CardLayout
- **CheckboxGroup**
- Color
- Dimension
- **Event**
- Font
- **FlowLayout**
- FontMetrics
- Graphics
- GridBagConstraints
- GridBagLayout
- **GridLayout**
- Image
- Insets
- Point
- Polygon
- Rectangle
- Toolkit
- MenuComponent
  - MenuBar
  - MenuItem
    - Menu – PopupMenu
    - CheckboxMenuItem
- **Component**
  - **Button**
  - Canvas
  - Checkbox
  - Choice
  - **Container**
    - **Panel**
      - **Applet** (java.applet package)
    - **Window**
      - Dialog – FileDialog
      - **Frame**
    - ScrollPane
  - Label
  - List
  - Scrollbar
  - TextComponent
    - TextArea
    - TextField

Exceptions – AWTException

Errors – AWTError

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Containers

- Add components with the `add()` method
- The two main types of containers are `Window` and `Panel`
- A `Window` is a free floating window on the display
- A `Panel` is a container of GUI components that must exist in the context of some other container, such as a window or applet

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Building Graphical User Interfaces

- The position and size of a component in a container is determined by a layout manager.
- You can control the size or position of components by disabling the layout manager.

  You must then use `setLocation()`, `setSize()`, or `setBounds()` on components to locate them in the container.

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Frames

- Are a subclass of `Window`
- Have title and resizing corners
- Are initially invisible, use `setVisible(true)` to expose the frame
- Have `BorderLayout` as the default layout manager
- Use the `setLayout` method to change the default layout manager

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

FrameExample.java

```
1   import java.awt.*;
2
3   public class FrameExample {
4     private Frame f;
5
6     public FrameExample() {
7       f = new Frame("Hello Out There!");
8     }
9
10    public void launchFrame() {
11      f.setSize(170,170);
12      f.setBackground(Color.blue);
13      f.setVisible(true);
14    }
15
16    public static void main(String args[]) {
17      FrameExample guiWindow = new FrameExample();
18      guiWindow.launchFrame();
19    }
20  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

FrameExample.java

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Panels

- Provide a space for components
- Allow subpanels to have their own layout manager

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

FrameWithPanel.java

```
1   import java.awt.*;
2
3   public class FrameWithPanel {
4     private Frame f;
5     Panel pan;
6
7     public FrameWithPanel(String title) {
8       f = new Frame(title);
9       pan = new Panel();
10  }
11
12    public void launchFrame() {
13      f.setSize(200,200);
14      f.setBackground(Color.blue);
15      f.setLayout(null);  // Override default layout mgr
16
17      pan.setSize(100,100);
18      pan.setBackground(Color.yellow);
19      f.add(pan);
20      f.setVisible(true);
21    }
22
23    public static void main(String args[]) {
24      FrameWithPanel guiWindow =
25          new FrameWithPanel("Frame with Panel");
26      guiWindow.launchFrame();
27    }
28  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

FrameWithPanel.java

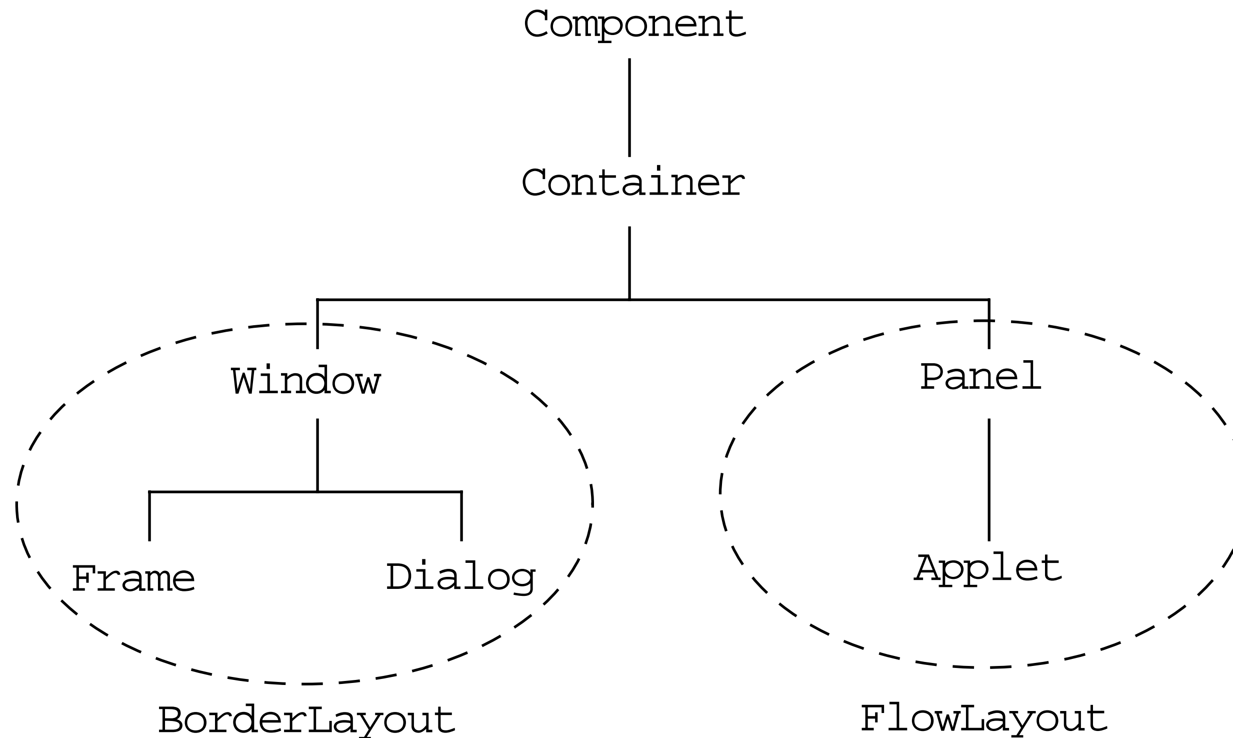
Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Container Layouts

- `FlowLayout`
- `BorderLayout`
- `GridLayout`
- `CardLayout`
- `GridBagLayout`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Default Layout Managers



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

A Simple FlowLayout Example

```
1   import java.awt.*;
2
3   public class LayoutExample {
4     private Frame f;
5     private Button b1;
6     private Button b2;
7
8     public LayoutExample() {
9       f = new Frame("GUI example");
10      b1 = new Button("Press Me");
11      b2 = new Button("Don't press Me");
12    }
13
14    public void launchFrame() {
15      f.setLayout(new FlowLayout());
16      f.add(b1);
17      f.add(b2);
18      f.pack();
19      f.setVisible(true);
20    }
21
22    public static void main(String args[]) {
23      LayoutExample guiWindow = new LayoutExample();
24      guiWindow.launchFrame();
25    }
26  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

`FlowLayout` Manager

- Default layout for the `Panel` class
- Components added from left to right
- Default alignment is centered
- Uses components' preferred sizes
- Uses the constructor to tune behavior

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

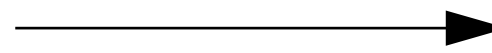
FlowExample.java

```
1   import java.awt.*;
2
3   public class FlowExample {
4     private Frame f;
5     private Button button1;
6     private Button button2;
7     private Button button3;
8
9     public FlowExample() {
10      f = new Frame("Flow Layout");
11      button1 = new Button("Ok");
12      button2 = new Button("Open");
13      button3 = new Button("Close");
14    }
15
16    public void launchFrame() {
17      f.setLayout(new FlowLayout());
18      f.add(button1);
19      f.add(button2);
20      f.add(button3);
21      f.setSize(100,100);
22      f.setVisible(true);
23    }
24
25    public static void main(String args[]) {
26      FlowExample guiWindow = new FlowExample();
27      guiWindow.launchFrame();
28    }
29  }
```

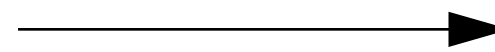

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

FlowExample.java

After user or
program resizes



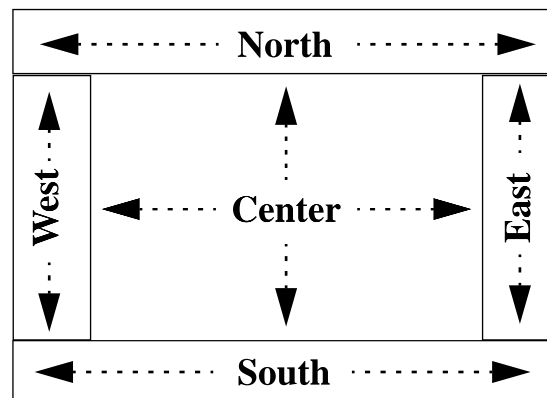
After user or
program resizes



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

`BorderLayout` Manager

- Default layout for the `Frame` class
- Components added to specific regions
- The resizing behavior:
  - North, South, and Center regions adjust horizontally
  - East, West, and Center regions adjust vertically



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

BorderExample.java

```
1   import java.awt.*;
2
3   public class BorderExample {
4     private Frame f;
5     private Button bn, bs, bw, be, bc;
6
7     public BorderExample() {
8       f = new Frame("Border Layout");
9       bn = new Button("B1");
10      bs = new Button("B2");
11      bw = new Button("B3");
12      be = new Button("B4");
13      bc = new Button("B5");
14    }
15
16    public void launchFrame() {
17      f.add(bn, BorderLayout.NORTH);
18      f.add(bs, BorderLayout.SOUTH);
19      f.add(bw, BorderLayout.WEST);
20      f.add(be, BorderLayout.EAST);
21      f.add(bc, BorderLayout.CENTER);
22      f.setSize(200,200);
23      f.setVisible(true);
24    }
25
26    public static void main(String args[]) {
27      BorderExample guiWindow2 = new BorderExample();
28      guiWindow2.launchFrame();
29    }
30  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

BorderExample.java

After window is resized

After window is resized

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

`GridLayout` Manager

- Components are added left to right, top to bottom
- All regions are equally sized
- The constructor specifies the rows and columns

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

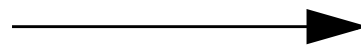
GridExample.java

```
1   import java.awt.*;
2
3   public class GridExample {
4     private Frame f;
5     private Button b1, b2, b3, b4, b5, b6;
6
7     public GridExample() {
8       f = new Frame("Grid Example");
9       b1 = new Button("1");
10      b2 = new Button("2");
11      b3 = new Button("3");
12      b4 = new Button("4");
13      b5 = new Button("5");
14      b6 = new Button("6");
15    }
16
17    public void launchFrame() {
18      f.setLayout (new GridLayout(3,2));
19
20      f.add(b1);
21      f.add(b2);
22      f.add(b3);
23      f.add(b4);
24      f.add(b5);
25      f.add(b6);
26
27      f.pack();
28      f.setVisible(true);
29    }
30
31    public static void main(String args[]) {
32      GridExample grid = new GridExample();
33      grid.launchFrame();
34    }
35  }
```

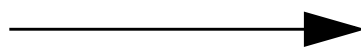

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

GridEx.java

After window is resized

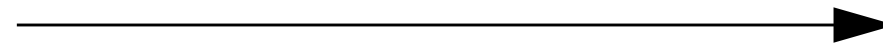


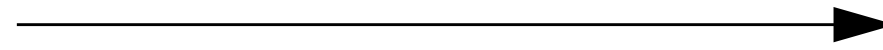
After window is resized



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

CardLayout Manager





Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

CardExample.java

```
1   import java.awt.*;
2   import java.awt.event.*;
3
4   public class CardExample implements MouseListener {
5     private Panel p1, p2, p3, p4, p5;
6     private Label lb1, lb2, lb3, lb4, lb5;
7
8     // Declare a CardLayout object to call its methods.
9     private CardLayout myCard;
10    private Frame f;
11
12    public CardExample() {
13      f = new Frame ("Card Test");
14      myCard = new CardLayout();
15
16      // Create the panels that I want to use as cards.
17      p1 = new Panel();
18      p2 = new Panel();
19      p3 = new Panel();
20      p4 = new Panel();
21      p5 = new Panel();
22
23      // Create a label to attach to each panel
24      lb1 = new Label("This is the first Panel");
25      lb2 = new Label("This is the second Panel");
26      lb3 = new Label("This is the third Panel");
27      lb4 = new Label("This is the fourth Panel");
28      lb5 = new Label("This is the fifth Panel");
29    }
30
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

CardLayout Manager

```
31    public void launchFrame() {
32      f.setLayout(myCard);
33
34      // change the color of each panel, so they are
35      // easily distinguishable
36      p1.setBackground(Color.yellow);
37      p1.add(lb1);
38      p2.setBackground(Color.green);
39      p2.add(lb2);
40      p3.setBackground(Color.magenta);
41      p3.add(lb3);
42      p4.setBackground(Color.white);
43      p4.add(lb4);
44      p5.setBackground(Color.cyan);
45      p5.add(lb5);
46
47      // Set up the event handling here.
48      p1.addMouseListener(this);
49      p2.addMouseListener(this);
50      p3.addMouseListener(this);
51      p4.addMouseListener(this);
52      p5.addMouseListener(this);
53
54      // Add each panel to my CardLayout
55      f.add(p1, "First");
56      f.add(p2, "Second");
57      f.add(p3, "Third");
58      f.add(p4, "Fourth");
59      f.add(p5, "Fifth");
60
61      // Display the first panel.
62      myCard.show(f, "First");
63
64      f.setSize(200,200);
65      f.setVisible(true);
66    }
67
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

CardLayout Manager

```
68    public void mousePressed(MouseEvent e) {
69      myCard.next(f);
70    }
71
72    public void mouseReleased(MouseEvent e) { }
73    public void mouseClicked(MouseEvent e) { }
74    public void mouseEntered(MouseEvent e) { }
75    public void mouseExited(MouseEvent e) { }
76
77    public static void main (String args[]) {
78      CardExample ct = new CardExample();
79      ct.launchFrame();
80    }
81  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

`GridBagLayout` Manager

- Complex layout facilities can be placed in a grid
- A single component can take its preferred size
- A component can extend over more than one cell

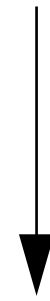
Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

ComplexLayoutExample.java

```
1   import java.awt.*;
2
3   public class ComplexLayoutExample {
4     private Frame f;
5     private Panel p;
6     private Button bw, bc;
7     private Button bfile, bhelp;
8
9     public ComplexLayoutExample() {
10      f = new Frame("GUI example 3");
11      bw = new Button("West");
12      bc = new Button("Work space region");
13      bfile = new Button("File");
14      bhelp = new Button("Help");
15    }
16
17    public void launchFrame() {
18      // Add bw and bc buttons in the frame border
19      f.add(bw, BorderLayout.WEST);
20      f.add(bc, BorderLayout.CENTER);
21      // Create panel for the buttons in the north border
22      p = new Panel();
23      p.add(bfile);
24      p.add(bhelp);
25      f.add(p, BorderLayout.NORTH);
26      // Pack the frame and make it visible
27      f.pack();
28      f.setVisible(true);
29    }
30
31    public static void main(String args[]) {
32      ComplexLayoutExample gui = new ComplexLayoutExample();
33      gui.launchFrame();
34    }
35  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Output of `ComplexLayoutExample.java`



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Drawing in AWT

- You can draw in any `Component` (although AWT provides the `Canvas` class just for this purpose)
- Typically, you would create a subclass of `Canvas` and override the `paint` method
- The `paint` method is called every time the component is shown (for example, if another window was overlapping the component and then removed)
- Every component has a `Graphics` object
- The `Graphics` class implements many drawing methods

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Drawing With the Graphics Object

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Building Java GUIs

- Exercise objective:
  - Develop two graphical user interfaces using the AWT
- Tasks:
  - Create a chat room GUI
  - Create a calculator GUI

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Describe the AWT package and its components
- Define the terms *containers*, *components*, and *layout managers*, and how they work together to build a graphical user interface (GUI)
- Use layout managers
- Use the `FlowLayout`, `BorderLayout`, `GridLayout`, and `CardLayout` managers to achieve a desired dynamic layout
- Add components to a container
- Use the `Frame` and `Panel` containers appropriately

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Describe how complex layouts with nested containers work
- In a Java program, identify the following:
  - Containers
  - The associated layout managers
  - The layout hierarchy of all components

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Think Beyond

- You now know how to display a GUI on the computer screen. What do you need to make the GUI useful?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Module 11

GUI Event Handling

Objectives

- Define events and event handling
- Write code to handle events that occur in a GUI
- Describe the concept of adapter classes, including how and when to use them
- Determine the user action that originated the event from the event object details

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Objectives

- Identify the appropriate interface for a variety of event types
- Create the appropriate event handler methods for a variety of event types
- Understand the use of inner classes and anonymous classes in event handling

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

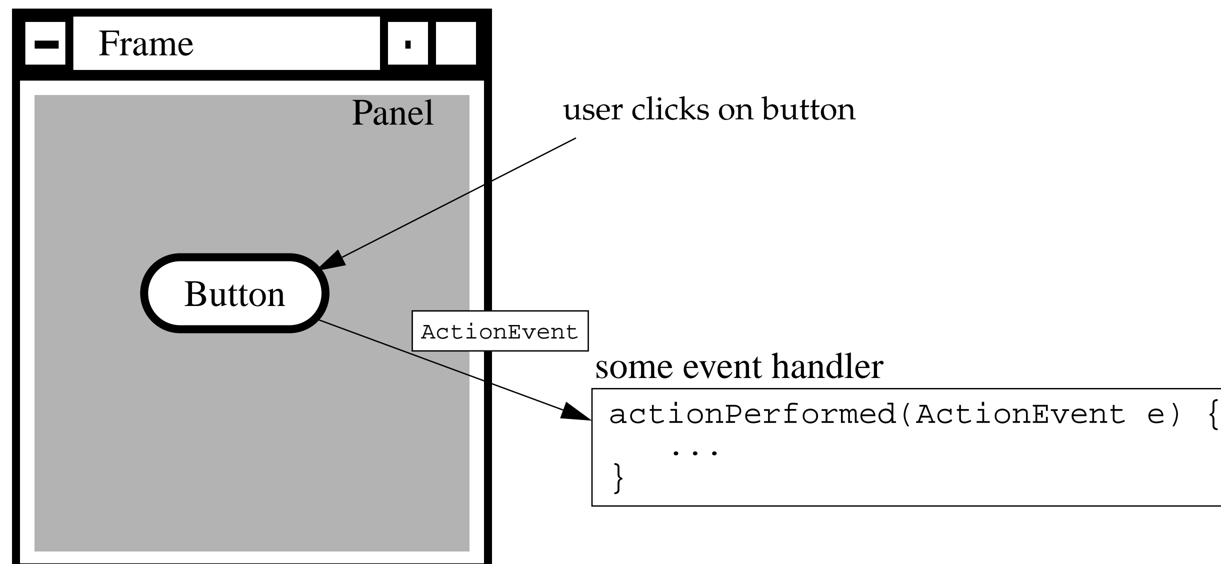
Relevance

- What parts of a GUI are required to make it useful?
- How does a graphical program handle a mouse click or any other type of user interaction?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

What Is an Event?

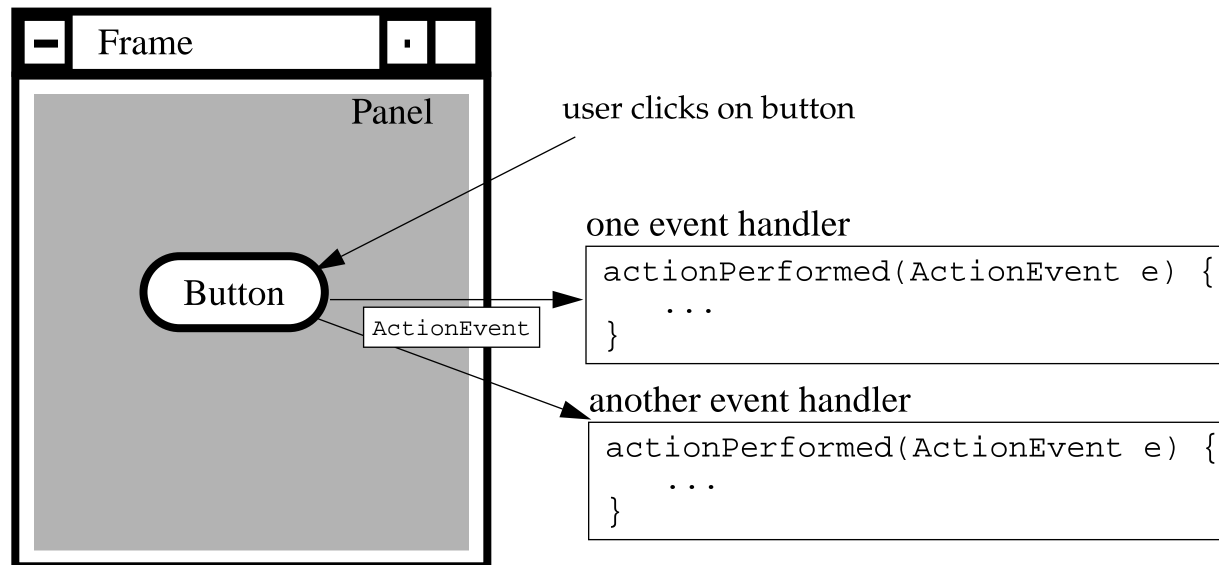
- Events – Objects that describe what happened
- Event sources – The generator of an event
- Event handlers – A method that receives an event object, deciphers it, and processes the user's interaction



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Delegation Model

- An event can be sent to many event handlers



- Event handlers register with components when they are interested in events generated by that component

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Delegation Model

```
1   import java.awt.*;
2
3   public class TestButton {
4     private Frame f;
5     private Button b;
6
7     public TestButton() {
8       f = new Frame("Test");
9       b = new Button("Press Me!");
10      b.setActionCommand("ButtonPressed");
11    }
12
13    public void launchFrame() {
14      b.addActionListener(new ButtonHandler());
15      f.add(b,BorderLayout.CENTER);
16      f.pack();
17      f.setVisible(true);
18    }
19
20    public static void main(String args[]) {
21      TestButton guiApp = new TestButton();
22      guiApp.launchFrame();
23    }
24  }
```

```
1   import java.awt.event.*;
2
3   public class ButtonHandler implements ActionListener {
4     public void actionPerformed(ActionEvent e) {
5       System.out.println("Action occurred");
6       System.out.println("Button’s command is: "
7                          + e.getActionCommand());
8     }
9   }
```

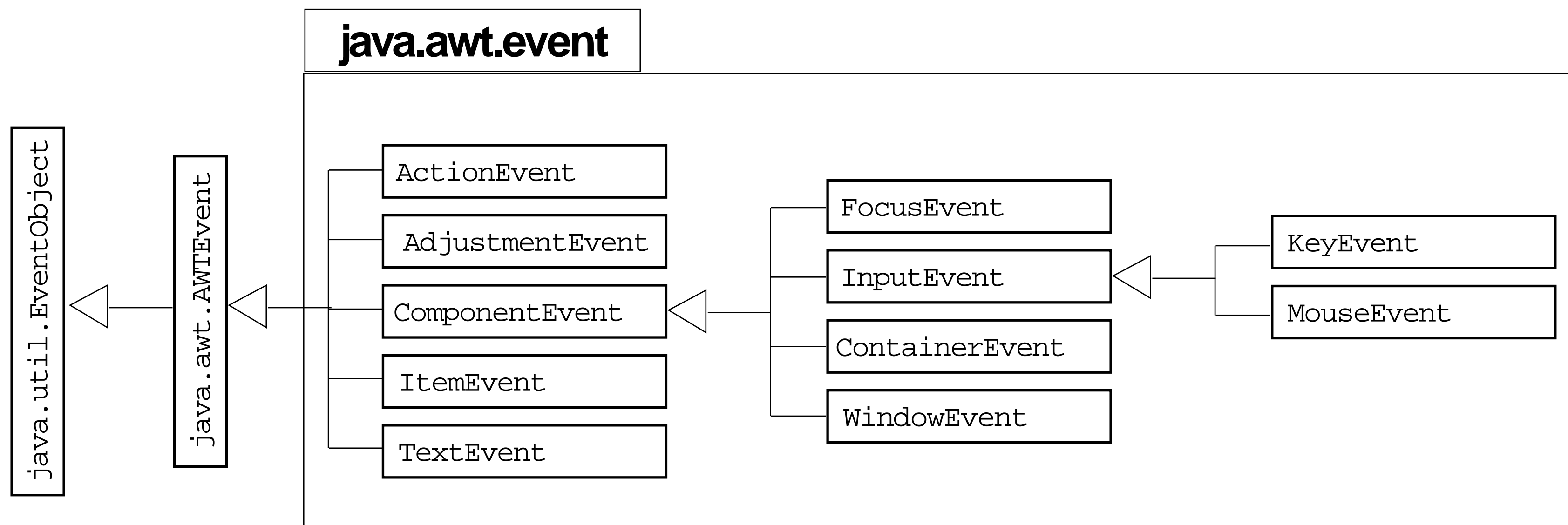

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Delegation Model

- Client objects (handlers) register with a GUI component they wish to observe
- GUI components only trigger the handlers for the type of event that has occurred
  - Most components can trigger more than one type of event
- Distributes the work among multiple classes

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Event Categories



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Java GUI Behavior

Category	Interface Name	Methods
Action	`ActionListener`	`actionPerformed(ActionEvent)`
Item	`ItemListener`	`itemStateChanged(ItemEvent)`

Mouse	`MouseListener`	`mousePressed(MouseEvent)`
		`mouseReleased(MouseEvent)`
		`mouseEntered(MouseEvent)`
		`mouseExited(MouseEvent)`
		`mouseClicked(MouseEvent)`

Mouse Motion	`MouseMotionListener`	`mouseDragged(MouseEvent)` `mouseMoved(MouseEvent)`

Key	`KeyListener`	`keyPressed(KeyEvent)`
		`keyReleased(KeyEvent)`
		`keyTyped(KeyEvent)`

Focus	`FocusListener`	`focusGained(FocusEvent)` `focusLost(FocusEvent)`
Adjustment	`AdjustmentListener`	`adjustmentValueChanged` `(AdjustmentEvent)`

Component	`ComponentListener`	`componentMoved(ComponentEvent)`
		`componentHidden(ComponentEvent)`
		`componentResized(ComponentEvent)`
		`componentShown(ComponentEvent)`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Java GUI Behavior

Category	Interface Name	Methods

Window	`WindowListener`	`windowClosing(WindowEvent)`
		`windowOpened(WindowEvent)`
		`windowIconified(WindowEvent)`
		`windowDeiconified(WindowEvent)`
		`windowClosed(WindowEvent)`
		`windowActivated(WindowEvent)`
		`windowDeactivated(WindowEvent)`

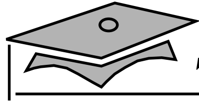
Container	`ContainerListener`	`componentAdded(ContainerEvent)` `componentRemoved(ContainerEvent)`
Text	`TextListener`	`textValueChanged(TextEvent)`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Complex Example

```
1   import java.awt.*;
2   import java.awt.event.*;
3
4   public class TwoListener
5        implements MouseMotionListener,
6                   MouseListener {
7     private Frame f;
8     private TextField tf;
9
10    public TwoListener() {
11      f = new Frame("Two listeners example");
12      tf = new TextField(30);
13    }
14
15    public void launchFrame() {
16      Label label = new Label("Click and drag the mouse");
17      // Add components to the frame
18      f.add(label, BorderLayout.NORTH);
19      f.add(tf, BorderLayout.SOUTH);
20      // Add this object as a listener
21      f.addMouseMotionListener(this);
22      f.addMouseListener(this);
23      // Size the frame and make it visible
24      f.setSize(300, 200);
25      f.setVisible(true);
26    }
27
28    // These are MouseMotionListener events
29    public void mouseDragged(MouseEvent e) {
30      String s =  "Mouse dragging:  X = " + e.getX()
31                  + " Y = " + e.getY();
32      tf.setText(s);
33    }
34
35    public void mouseEntered(MouseEvent e) {
36      String s = "The mouse entered";
37      tf.setText(s);
38    }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D



Complex Example

```
39
40    public void mouseExited(MouseEvent e) {
41      String s = "The mouse has left the building";
42      tf.setText(s);
43    }
44
45    // Unused MouseMotionListener method.
46    // All methods of a listener must be present in the
47    // class even if they are not used.
48    public void mouseMoved(MouseEvent e) { }
49
50    // Unused MouseListener methods.
51    public void mousePressed(MouseEvent e) { }
52    public void mouseClicked(MouseEvent e) { }
53    public void mouseReleased(MouseEvent e) { }
54
55    public static void main(String args[]) {
56      TwoListener two = new TwoListener();
57      two.launchFrame();
58    }
59  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Multiple Listeners

- Multiple listeners cause unrelated parts of a program to react to the same event
- The handlers of all registered listeners are called when the event occurs

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Event Adapters

- The listener classes that you define can extend adapter classes and override only the methods that you need
- Example:

```
1   import java.awt.*;
2   import java.awt.event.*;
3
4   public class MouseClickHandler extends MouseAdapter {
5
6     // We just need the mouseClick handler, so we use
7     // the an adapter to avoid having to write all the
8     // event handler methods
9
10    public void mouseClicked(MouseEvent e) {
11      // Do stuff with the mouse click...
12    }
13  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Inner Classes

```
1   import java.awt.*;
2   import java.awt.event.*;
3
4   public class TestInner {
5     private Frame f;
6     private TextField tf;
7
8     public TestInner() {
9       f = new Frame("Inner classes example");
10      tf = new TextField(30);
11    }
12
13    public void launchFrame() {
14      Label label = new Label("Click and drag the mouse");
15      // Add components to the frame
16      f.add(label, BorderLayout.NORTH);
17      f.add(tf, BorderLayout.SOUTH);
18      // Add a listener that uses an Inner class
19      f.addMouseMotionListener(new MyMouseMotionListener());
20      f.addMouseListener(new MouseClickHandler());
21      // Size the frame and make it visible
22      f.setSize(300, 200);
23      f.setVisible(true);
24    }
25
26    class MyMouseMotionListener extends MouseMotionAdapter {
27        public void mouseDragged(MouseEvent e) {
28          String s = "Mouse dragging:  X = "+ e.getX()
29                      + " Y = " + e.getY();
30          tf.setText(s);
31        }
32      }
33
34    public static void main(String args[]) {
35      TestInner obj = new TestInner();
36      obj.launchFrame();
37    }
38  }
39
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Anonymous Classes

```
1   import java.awt.*;
2   import java.awt.event.*;
3
4   public class TestAnonymous {
5     private Frame f;
6     private TextField tf;
7
8     public TestAnonymous() {
9       f = new Frame("Anonymous classes example");
10      tf = new TextField(30);
11    }
12
13    public void launchFrame() {
14      Label label = new Label("Click and drag the mouse");
15      // Add components to the frame
16      f.add(label, BorderLayout.NORTH);
17      f.add(tf, BorderLayout.SOUTH);
18      // Add a listener that uses an anonymous class
19      f.addMouseMotionListener(new MouseMotionAdapter() {
20        public void mouseDragged(MouseEvent e) {
21          String s = "Mouse dragging:  X = "+ e.getX()
22                      + " Y = " + e.getY();
23          tf.setText(s);
24        }
25      }); // <- note the closing parenthesis
26      f.addMouseListener(new MouseClickHandler());
27      // Size the frame and make it visible
28      f.setSize(300, 200);
29      f.setVisible(true);
30    }
31
32    public static void main(String args[]) {
33      TestAnonymous obj = new TestAnonymous();
34      obj.launchFrame();
35    }
36  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Working With Events

- Exercise objective:
  - You will write, compile, and run the revised ChatClient GUI and Calculator GUI codes to include event handlers
- Tasks:
  - Add event handlers to the ChatClient GUI
  - Add event handlers to the Calculator GUI

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Define events and event handling
- Write code to handle events that occur in a GUI
- Describe the concept of adapter classes, including how and when to use them
- Determine the user action that originated the event from the event object details

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Identify the appropriate interface for a variety of event types
- Create the appropriate event handler methods for a variety of event types
- Understand the use of inner classes and anonymous classes in event handling

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Think Beyond

- You now know how to set up a Java GUI for both graphic output and interactive user input. However, only a few of the components from which GUIs can be built have been described. What other components would be useful in a GUI?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Module 12

Introduction to Java Applets

Objectives

- Differentiate between a standalone application and an applet
- Write an HTML tag to call a Java applet
- Describe the class hierarchy of the applet and AWT classes
- Create the `HelloWorld.Java` applet
- List the major methods of an applet

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Objectives

- Describe and use the painting model of AWT
- Use applet methods to read images and files from URLs
- Handle various mouse events within the applet
- Pass parameters to an applet from an HTML file using the `<param>` tag

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Relevance

- What advantages do applets provide over standalone applications?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

What Is an Applet?

A Java class that can be:

- Embedded within an HTML page and downloaded and executed by a Web browser
- Loaded using the browser as follows:

  1. Browser loads URL.
  2. Browser loads HTML document.
  3. Browser loads applet classes.
  4. Browser runs applet.

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

What Is An Applet?

http://someLocation/file.html

HTML file:

```
<HTML>
:
<APPLET CODE = ...>
</APPLET>
```

Applet Classes

Browser

Location: http://someLocation/file.html

Loading...

1. Browser loads URL.
2. Browser loads HTML document.
3. Browser loads applet classes.
4. Browser runs applet.

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Applet Security Restrictions

- Most browsers prevent the following:
  - Runtime execution of another program
  - File I/O
  - Calls to any native methods
  - Attempts to open a socket to any system except the host that provided the applet

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Applet Class Hierarchy

```
java.lang.Object
        |
java.awt.Component
        |
java.awt.Container
   /             \
java.awt.Window   java.awt.Panel
   |                  |
java.awt.Frame    java.applet.Applet
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Key Applet Methods

- `init()`
- `start()`
- `stop()`
- `destroy()`
- `paint()`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The Applet Life Cycle

- `init()`
  - Called when the applet is created
  - Can be used to initialize data values
- `start()`
  - Called when the applet becomes visible
- `stop()`
  - Called when the applet becomes invisible

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Applet Display

- Applets are graphical in nature
- The browser environment calls the `paint()` method

```
1   import java.awt.*;
2   import java.applet.*;
3
4   public class HelloWorld extends Applet {
5     private paintCount;
6     public void init() {
7       paintCount = 0;
8     }
9     public void paint(Graphics g){
10      g.drawString("Hello World!", 25, 25);
11      paintCount++;
12      g.drawString("Number of times paint called: "
13                    + paintCount, 25, 50);
14    }
15  }
```

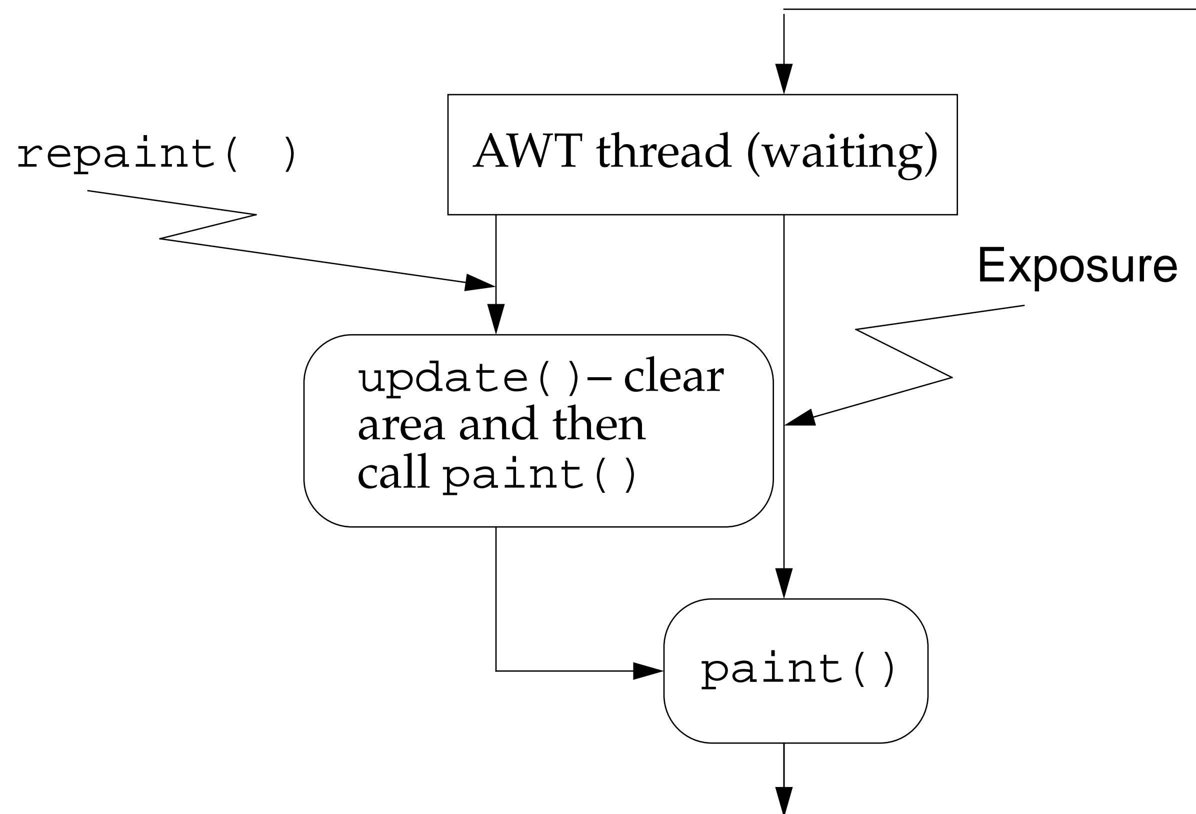

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

AWT Painting

- `paint(Graphics g)` – Called when the component is "exposed". This is where the programmer implements the painting algorithm.
- `repaint()` – You call this method to ask the AWT thread to repaint the component.
- `update(Graphics g)` – This method is called by the AWT thread when a repaint has been requested.

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

AWT Painting

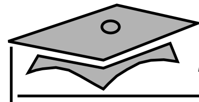


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Applet Display Strategies

- Maintain a model of the display
- Use `paint()` to render the display based only on the model
- Update the model and call `repaint()` to change the display

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D



An Example Paint Model

- Requirement: Paint "`Hello World!`" wherever the user clicks.
- The model is the last user click point.

```
1   // <APPLET CODE="PaintModel1.class" WIDTH=200 HEIGHT=200></APPLET>
2
3   import java.applet.*;
4   import java.awt.event.*;
5   import java.awt.*;
6
7   public class PaintModel1 extends Applet {
8     // The paint model: the last click Point
9     private Point lastClick = null;
10
11    public void init() {
12      addMouseListener(new MyModelRecorder());
13    }
14
15    public void paint(Graphics g) {
16      if ( lastClick != null ) {
17        g.drawString("Hello World!", lastClick.x, lastClick.y);
18      }
19    }
20
21    private class MyModelRecorder extends MouseAdapter {
22      public void mousePressed(MouseEvent e) {
23        lastClick = e.getPoint();
24        repaint();
25      }
26    }
27  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

An Example Paint Model

- The model is the last user click point.
- The update method is overridden to avoid clearing the screen.

```
1   // <APPLET CODE="PaintModel2.class" WIDTH=200 HEIGHT=200></APPLET>
2
3   import java.applet.*;
4   import java.awt.event.*;
5   import java.awt.*;
6
7   public class PaintModel2 extends Applet {
8     // The paint model: the last click Point
9     private Point lastClick = null;
10
11    public void init() {
12      addMouseListener(new MyModelRecorder());
13    }
14
15    public void update(Graphics g) {
16      paint(g);
17    }
18
19    public void paint(Graphics g) {
20      if ( lastClick != null ) {
21        g.drawString("Hello World!", lastClick.x, lastClick.y);
22      }
23    }
24
25    private class MyModelRecorder extends MouseAdapter {
26      public void mousePressed(MouseEvent e) {
27        lastClick = e.getPoint();
28        repaint();
29      }
30    }
31  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

An Example Paint Model

- The model is a list of all user click points.

```
1   //<APPLET CODE="PaintModel3.class" WIDTH=200 HEIGHT=200></APPLET>
2
3   import java.applet.*;
4   import java.awt.event.*;
5   import java.awt.*;
6   import java.util.List;
7   import java.util.ArrayList;
8
9   public class PaintModel3 extends Applet {
10    // The paint model: a list of click Points
11    private List mouseClicks = new ArrayList(5);
12
13    public void init() {
14      addMouseListener(new MyModelRecorder());
15    }
16
17    public void update(Graphics g) {
18      paint(g);
19    }
20
21    public void paint(Graphics g) {
22      for(int x = 0; x < mouseClicks.size(); x++) {
23        Point p = (Point) mouseClicks.get(x);
24        g.drawString("Hello World!", p.x, p.y);
25      }
26    }
27
28    private class MyModelRecorder extends MouseAdapter {
29      public void mousePressed(MouseEvent e) {
30        mouseClicks.add(e.getPoint());
31        repaint();
32      }
33    }
34  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

What Is the `appletviewer`?

A Java application that:

- Enables you to run applets without using a Web browser
- Loads the HTML file supplied as an argument
- `appletviewer HelloWorld.html`
- Needs at least the following HTML code:

```
1   <applet code="HelloWorld.class" width=300 height=300>
2   </applet>
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Running the `appletviewer`

Synopsis:

```
appletviewer [-debug] URLs...
```

Example:

```
appletviewer HelloWorld.html
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The applet Tag

```
<applet
   [archive=archiveList]
   code=appletFile.class
   width=pixels
   height=pixels
   [codebase=codebaseURL]
   [alt=alternateText]
   [name=appletInstanceName]
   [align=alignment]
   [vspace=pixels] [hspace=pixels]
>
   [<param name=appletAttribute1 value=value>]
   [<param name=appletAttribute2 value=value>]
    . . .
   [alternateHTML]
</applet>
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Additional Applet Features

- `getDocumentBase()` – Returns a `URL` object that describes the directory of the current browser page
- `getCodeBase()` – Returns a `URL` object that describes the source directory of the applet class
- `getImage(URL base, String target)` and `getAudioClip(URL base, String target)` – Use the `URL` object as a starting point

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

A Simple Image Test

```
1   // Applet which shows an image of Duke in surfing mode
2
3   import java.awt.*;
4   import java.applet.Applet;
5
6   public class HwImage extends Applet {
7     Image duke;
8
9     public void init() {
10      duke = getImage(getDocumentBase(),
11          "graphics/surferDuke.gif");
12    }
13
14    public void paint(Graphics g) {
15      g.drawImage(duke, 25, 25, this);
16    }
17  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Audio Clips

Playing a clip:

```
play(URL soundDirectory, String soundFile);
play(URL soundURL);
```

Example:

```
play(getDocumentBase(), "bark.au");
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

A Simple Audio Test

```
1   // Applet which plays a sound on every mouse click
2
3   import java.awt.Graphics;
4   import java.awt.event.MouseAdapter;
5   import java.awt.event.MouseEvent;
6   import java.applet.Applet;
7
8   public class HwAudio extends Applet {
9     public void init() {
10      addMouseListener(new MouseAdapter() {
11        public void mouseClicked(MouseEvent event) {
12          play(getCodeBase(), "sounds/cuckoo.au");
13        }
14      });
15    }
16    public void paint(Graphics g) {
17      g.drawString("Audio Test", 25, 25);
18    }
19  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Looping an Audio Clip

- Loading an Audio Clip
- Playing an Audio Clip
- Stopping an Audio Clip

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

A Simple Looping Test

```
1   // Applet which continuously repeats a sound
2
3   import java.awt.Graphics;
4   import java.applet.*;
5
6   public class HwLoop extends Applet {
7     AudioClip sound;
8
9     public void init() {
10      sound = getAudioClip(getCodeBase(), "sounds/cuckoo.au");
11    }
12
13    public void paint(Graphics g) {
14      g.drawString("Audio Test", 25, 25);
15    }
16
17    public void start() {
18      sound.loop();
19    }
20
21    public void stop() {
22      sound.stop();
23    }
24  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Mouse Input

- `mouseClicked` – The mouse has been clicked (mouse button pressed and then released in one motion)
- `mouseEntered` – The mouse cursor enters a component
- `mouseExited` – The mouse cursor leaves a component
- `mousePressed` – The mouse button is pressed down
- `mouseReleased` – The mouse button is later released

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

A Simple Mouse Test

```
1   // This applet is HelloWorld extended to watch for mouse
2   // input. "Hello World!" is reprinted at the location of
3   // the mouse press.
4
5   import java.awt.Graphics;
6   import java.awt.event.*;
7   import java.applet.Applet;
8
9   public class HwMouse extends Applet {
10    // "paint model data"
11    private int mouseX = 25;
12    private int mouseY = 25;
13
14    // Register an anonymous mouse events handler.
15    public void init() {
16      addMouseListener(new MouseHandler());
17    }
18
19    public void paint(Graphics g) {
20      g.drawString("Hello World!", mouseX, mouseY);
21    }
22
23    private class MouseHandler extends MouseAdapter {
24      public void mousePressed(MouseEvent evt) {
25        // record the position of the mouse
26        // in the "paint model data"
27        mouseX = evt.getX();
28        mouseY = evt.getY();
29        // inform AWT to repaint the applet
30        repaint();
31      }
32    }
33  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Reading Parameters

- Applet code

```
1   <html>
2   <applet code="Parameters.class" width=200 height=200>
3     <param name=speed value="12">
4     <param name=distance value="500m">
5   </applet>
6   </html>
```

- Program code

```
1   // Parameter test applet. To see a change in "speed",
2   // you must supply it as a <param> tag in the HTML file
3   // which calls this applet.
4
5   import java.applet.Applet;
6   import java.awt.Graphics;
7
8   public class Parameters extends Applet {
9     private String toDisplay;
10    private int speed;
11
12    public void init() {
13      String pv;
14      pv = getParameter("speed");
15      if (pv == null){
16        speed = 10;
17      } else {
18        speed = Integer.parseInt (pv);
19      }
20      toDisplay = "Speed given: " + speed;
21    }
22
23    public void paint(Graphics g) {
24      g.drawString(toDisplay, 25, 25);
25    }
26  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Creating Applets

- Exercise objective:
  - Become familiar with programming Java applets
- Tasks:
  - Write an applet
  - Create concentric squares
  - Create a rollover applet

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Differentiate between a standalone application and an applet
- Write an HTML tag to call a Java applet
- Describe the class hierarchy of the applet and AWT classes
- Create the `HelloWorld.Java` applet
- List the major methods of an applet

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Describe and use the painting model of AWT
- Use applet methods to read images and files from URLs
- Handle various mouse events within the applet
- Pass parameters to an applet from an HTML file using the `<param>` tag

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Think Beyond

- How can you use applets on your company’s Web page to improve the overall presentation?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Module 13

GUI-Based Applications

Objectives

- Identify the key AWT components and the events that they trigger
- Describe how to construct a menu bar, menu, and menu items in a Java GUI
- Understand how to change the color and font of a component
- Use the Java printing mechanism
- Understand how to construct a GUI class that can be used within a Frame or within an Applet

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Relevance

- You now know how to set up a Java GUI for both graphic output and interactive user input. However, only a few of the components from which GUIs can be built have been described. What other components would be useful in a GUI?
- How can you create a menu for your GUI frame?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

AWT Components

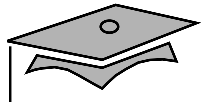
Component Type	Description
Button	A named rectangular box used for receiving mouse clicks
Canvas	A panel used for drawing
Checkbox	A component allowing the user to select an item
CheckboxMenuItem	A checkbox within a menu
Choice	A pull-down static list of items
Component	The parent of all AWT components, except menu components
Container	The parent of all AWT containers
Dialog	The base class of all modal dialog boxes
Frame	The base class of all GUI windows with window manager controls
Label	A text string component
List	A component that contains a dynamic set of items
Menu	An element under the menu bar, which contains a set of menu items
MenuItem	An item within a menu
Panel	A basic container class used most often to create complex layouts
Scrollbar	A component which allows a user to "select from a range of values"
ScrollPane	A container class which implements automatic horizontal and/or vertical scrolling for a single child component
TextArea	A component that allows the user to enter a block of text
TextField	A component that allows the user to enter a single line of text
Window	The base class of all GUI windows, with no window manager controls

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Component Events

Component Type	Act	Adj	Cmp	Cnt	Foc	Itm	Key	Mou	MM	Text	Win
Button	✓		✓		✓		✓	✓	✓		
Canvas			✓		✓		✓	✓	✓		
Checkbox			✓		✓	✓	✓	✓	✓		
CheckboxMenuItem						✓					
Choice			✓		✓	✓	✓	✓	✓		
Component			✓		✓		✓	✓	✓		
Container			✓	✓	✓		✓	✓	✓		
Dialog			✓	✓	✓		✓	✓	✓		✓
Frame			✓	✓	✓		✓	✓	✓		✓
Label			✓		✓		✓	✓	✓		
List	✓		✓		✓	✓	✓	✓	✓		
MenuItem	✓										
Panel			✓	✓	✓		✓	✓	✓		
Scrollbar		✓	✓		✓		✓	✓	✓		
ScrollPane			✓	✓	✓		✓	✓	✓		
TextArea			✓		✓		✓	✓	✓	✓	
TextField	✓		✓		✓		✓	✓	✓	✓	
Window			✓	✓	✓		✓	✓	✓		✓

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D



How to Create a Menu

1. Create a `MenuBar` object and set it into a menu container such as a `Frame`.
2. Create one or more `Menu` objects and add them to the menu bar object.
3. Create one or more `MenuItem` objects and add them to the menu object.

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Creating a MenuBar

```
1  Frame f = new Frame("MenuBar");
2  MenuBar mb = new MenuBar();
3  f.setMenuBar(mb);
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Creating a Menu

```
1      Frame f = new Frame("Menu");
2      MenuBar mb = new MenuBar();
3      Menu m1 = new Menu("File");
4      Menu m2 = new Menu("Edit");
5      Menu m3 = new Menu("Help");
6      mb.add(m1);
7      mb.add(m2);
8      mb.setHelpMenu(m3);
9      f.setMenuBar(mb);
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Creating a `MenuItem`

```
1       MenuItem mi1 = new MenuItem("New");
2       MenuItem mi2 = new MenuItem("Save");
3       MenuItem mi3 = new MenuItem("Load");
4       MenuItem mi4 = new MenuItem("Quit");
5       mi1.addActionListener(this);
6       mi2.addActionListener(this);
7       mi3.addActionListener(this);
8       mi4.addActionListener(this);
9       m1.add(mi1);
10      m1.add(mi2);
11      m1.add(mi3);
12      m1.addSeparator();
13      m1.add(mi4);
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Creating a CheckBoxMenuItem

```
1       MenuBar mb = new MenuBar();
2       Menu m1 = new Menu("File");
3       Menu m2 = new Menu("Edit");
4       Menu m3 = new Menu("Help");
5       mb.add(m1);
6       mb.add(m2);
7       mb.setHelpMenu(m3);
8       f.setMenuBar(mb);
9       .....
10      MenuItem mi2 = new MenuItem("Save");
11      mi2.addActionListener(this);
12      m1.add(mi2);
13      ......
14      CheckboxMenuItem mi5 = new CheckboxMenuItem("Persistent");
15      mi5.addItemListener(this);
16      m1.add(mi5);
```

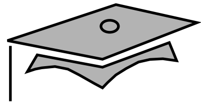

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Controlling Visual Aspects

- Colors:
  - `setForeground()`
  - `setBackground()`
- Example:

```
int r = 255;
Color c = new Color(r, 0, 0);
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D



Controlling Visual Aspects

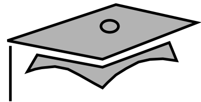
- Fonts:
  - You can use the `setFont()` method to specify the font used for displaying text
  - `Dialog`, `DialogInput`, `Serif`, and `SansSerif` are valid font names
- Example:

```
Font font = new Font("TimesRoman", Font.PLAIN, 14);
```

- Use the `GraphicsEnvironment` class to retrieve the set of all available fonts:

```
GraphicsEnvironment ge =
  GraphicsEnvironment.getLocalGraphicsEnvironment();
Font[] fonts = ge.getAllFonts();
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D



Controlling Visual Aspects

- The `Toolkit` class is an abstract superclass of all actual implementations of the Abstract Window Toolkit
- Subclasses of `Toolkit` are used to bind the various components to particular native toolkit implementations
- Useful methods:

```
getDefaultToolkit
getImage(String filename)
getScreenResolution
getScreenSize
getPrintJob
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Printing

- The follow code fragment prints a Frame:

```
1   Frame f = new Frame("Print test");
2   Toolkit toolkit = frame.getToolkit();
3   PrintJob job = toolkit.getPrintJob(frame, "Test Printing", null);
4   Graphics g = job.getGraphics();
5   frame.printComponents(g);
6   g.dispose();
7   job.end();
```

1. Obtain graphics object (line 4).
2. Draw on the graphics object (line 5).
3. Send the graphics object to printer (line 6).
4. End the print job (line 7).

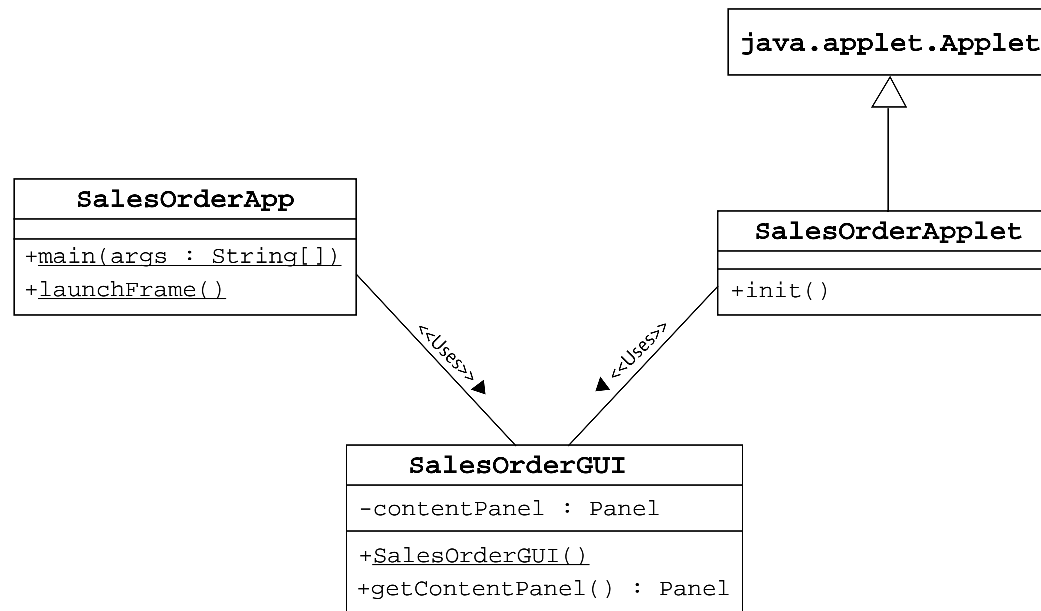
Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Dual-Purpose Code

- You can write GUI code that can be used as a stand-alone application or an applet
- GUI code can be written as a panel independent of being embedded in a frame or applet

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

UML Model of the SalesOrderGUI



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The GUI Mediator Code

Fragments of the `SalesOrderGUI` code:

```
import java.awt.*;
import java.awt.event.*;

public class SalesOrderGUI {
  // declaration of GUI components
    ...
  private Panel contentPanel = null;

  public SalesOrderGUI() {
    // initialize GUI components
    ...
  }

  public Panel getContentPanel() {
    // return the panel if it has already been created
    if ( contentPanel != null ) {
      return contentPanel;
    }
    contentPanel = new Panel();

    // construction and layout of GUI components
    ...

    // Set up event handling
    ...

    return contentPanel;
  }
  // Event handler inner class declarations
  ...
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The App and Applet Code

```
1  import java.awt.Frame;
2  import java.awt.BorderLayout;
3  import java.awt.event.WindowAdapter;
4  import java.awt.event.WindowEvent;
5
6  public class SalesOrderApp {
7
8    private static void launchFrame() {
9      SalesOrderGUI salesGUI = new SalesOrderGUI();
10     Frame f = new Frame("Sales Order Entry");
11
12     f.addWindowListener(new WindowAdapter() {
13       public void windowClosing(WindowEvent event) {
14         System.exit(0);
15       }
16     });
17     f.setSize(200, 200);
18     f.add(salesGUI.getContentPanel(), BorderLayout.CENTER);
19     f.setVisible (true);
20   }
21
22   public static void main(String args[]) {
23     launchFrame();
24   }
25 }
```

```
1  import java.applet.Applet;
2  import java.awt.BorderLayout;
3
4  public class SalesOrderApplet extends Applet {
5    public void init() {
6      SalesOrderGUI salesGUI = new SalesOrderGUI();
7      setLayout(new BorderLayout());
8      add(salesGUI.getContentPanel(), BorderLayout.CENTER);
9    }
10 }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Discussion of Dual-Purpose Code

- Does your business wish to present the same application presentation (the GUI) within the company's intranet as well as over the internet?
- Security issues hinder applet functionality
- Firewall security hinders applet <-> server communication mechanisms (such as raw sockets)
- There are different methods for loading images
- There is some redundant code in the `XxxApp` and `XxxApplet` classes for initializing the GUI and the model

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Swing

- Swing is a second-generation GUI toolkit
- It builds on top of AWT, but supplants the components with "light-weight" versions
- There are several more components: `JTable`, `JTree`, and `JComboBox`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise:

- Exercise objective:
  - Add menus to a Frame
  - Create dual-purpose code
- Tasks:
  - Add menus to the ChatClient GUI
  - Convert the Calculator to dual-purpose code

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Identify the key AWT components and the events that they trigger
- Describe how to construct a menu bar, menu, and menu items in a Java GUI
- Understand how to change the color and font of a component
- Use the Java printing mechanism
- Understand how to construct a GUI class that can be used within a Frame or within an Applet

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Think Beyond

- What problems occur when your GUI code must wait for the application logic to perform its job?
- What are the limitation of AWT?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Module 14

Threads

Objectives

- Define a thread
- Create separate threads in a Java program, controlling the code and data that are used by that thread
- Control the execution of a thread and write platform-independent code with threads
- Describe the difficulties that might arise when multiple threads share data

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Objectives

- Use `wait` and `notify` to communicate between threads
- Use `synchronized` to protect data from corruption
- Explain why `suspend`, `resume`, and `stop` methods have been deprecated in JDK 1.2

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Relevance

- How do you get programs to perform multiple tasks concurrently?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Threads

- What are threads?
  - Virtual CPU

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Three Parts of a Thread

- CPU
- Code
- Data

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Creating the Thread

```
1   public class ThreadTester {
2     public static void main(String args[]) {
3       HelloRunner r = new HelloRunner();
4       Thread t = new Thread(r);
5       t.start();
6     }
7   }
8
9   class HelloRunner implements Runnable {
10    int i;
11
12    public void run() {
13      i = 0;
14
15      while (true) {
16        System.out.println("Hello " + i++);
17        if ( i == 50 ) {
18          break;
19        }
20      }
21    }
22  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Creating the Thread

- Multithreaded programming:
  - Multiple threads from the same `Runnable` instance
  - Threads share the same data and code
- Example:

```
Thread t1 = new Thread(r);
Thread t2 = new Thread(r);
```

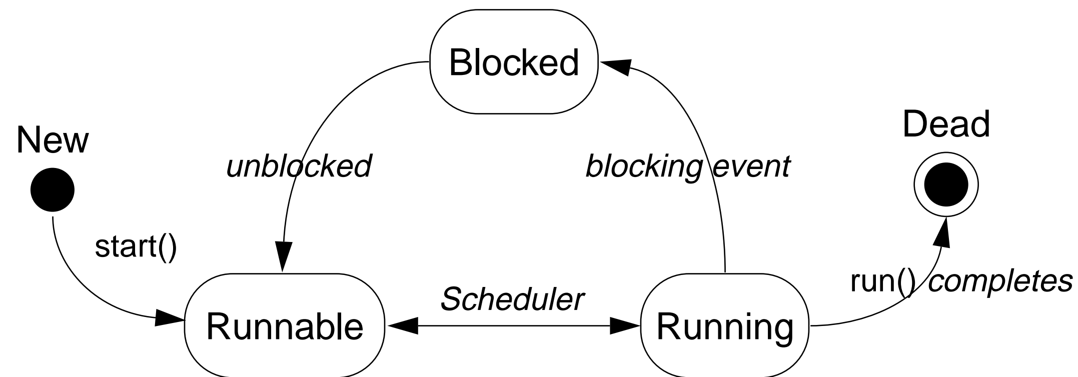

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Starting the Thread

- Using the `start` method
- Placing the thread in runnable state

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Thread Scheduling



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Thread Scheduling

```
1   public class Runner implements Runnable {
2     public void run() {
3       while (true) {
4         // do lots of interesting stuff
5         :
6         // Give other threads a chance
7         try {
8           Thread.sleep(10);
9         } catch (InterruptedException e) {
10          // This thread's sleep was interrupted
11          // by another thread
12        }
13      }
14    }
15  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Terminating a Thread

```
1   public class Runner implements Runnable {
2     private boolean timeToQuit=false;
3
4     public void run() {
5       while ( ! timeToQuit ) {
6         ...
7       }
8       // clean up before run() ends
9     }
10
11    public void stopRunning() {
12      timeToQuit=true;
13    }
14  }
15
16  public class ThreadController {
17    private Runner r = new Runner();
18    private Thread t = new Thread(r);
19
20    public void startThread() {
21      t.start();
22    }
23
24    public void stopThread() {
25      // use specific instance of Runner
26      r.stopRunning();
27    }
28  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Basic Control of Threads

- Testing threads:
  - `isAlive()`
- Thread priority:
  - `getPriority()`
  - `setPriority()`
- Putting threads on hold:
  - `Thread.sleep()`
  - `join()`
  - `Thread.yield()`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Putting Threads on Hold

```
1   public class Runner implements Runnable {
2   ...
3     public void run() {
4       while (running) {
5         // do your task
6         try {
7           Thread.sleep((int)(Math.random() * 100));
8         } catch (InterruptedException e) {
9           // somebody woke me up
10        }
11        ...
12      }
13    }
14  }
15
16  public class ThreadTest {
17    public static void main(String args[]) {
18      Runnable r = new Runner();
19      Thread t1 = new Thread(r);
20      t1.start();
21    }
22  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Putting Threads on Hold

```
1  public static void main(String[] args) {
2    Thread t = new Thread(new Runner());
3    t.start();
4    ...
5    // Do stuff in parallel with the other thread for a while
6    ...
7    // Wait here for the timer thread to finish
8    try {
9      t.join();
10   } catch (InterruptedException e) {
11     // t came back early
12   }
13   ...
14   // Now continue in this thread
15   ...
16 }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Extending the Thread Class

```
1   public class MyThread extends Thread {
2     public void run() {
3       while (running) {
4         // do lots of interesting stuff
5         try {
6           sleep(100);
7         } catch (InterruptedException e) {
8           // sleep interrupted
9         }
10      }
11    }
12
13    public static void main(String args[]) {
14      Thread t = new MyThread();
15      t.start();
16    }
17  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Selecting a Way to Create Threads

- Implementing `Runnable`:
  - Better object-oriented design
  - Single inheritance
  - Consistency
- Extending `Thread`:
  - Simpler code

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Using Basic Threads

- Exercise objectives:
  - Become familiar with the basic thread concepts
- Tasks:
  - Create three threads

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Using the `synchronized` Keyword

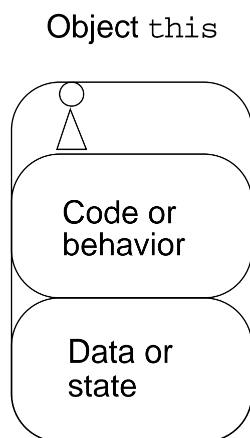
```
1   public class MyStack {
2     int idx = 0;
3     char [] data = new char[6];
4
5     public void push(char c) {
6       data[idx] = c;
7       idx++;
8     }
9
10    public char pop() {
11      idx--;
12      return data[idx];
13    }
14  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The Object Lock Flag

- Every object has a flag that can be thought of as a "lock flag"
- `synchronized` allows interaction with the lock flag

Object `this`

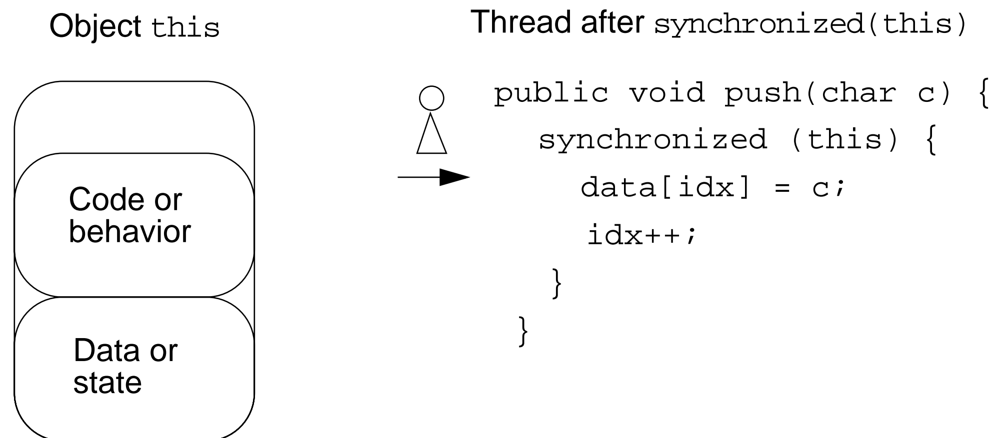


Thread before `synchronized(this)`

```
public void push(char c) {
  synchronized (this) {
    data[idx] = c;
    idx++;
  }
}
```

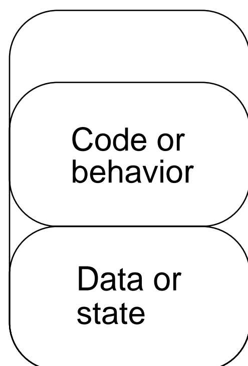

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The Object Lock Flag



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The Object Lock Flag



Thread, trying to execute `synchronized(this)`

Waiting for object lock

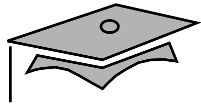
```
public char pop() {
  synchronized (this) {
    idx--;
    return data[idx];
  }
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Releasing the Lock Flag

- Released when the thread passes the end of the `synchronized` code block
- Automatically released when a break or exception is thrown by the `synchronized` code block

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D



`synchronized` – Putting It Together

- *All* access to delicate data should be `synchronized`
- Delicate data protected by `synchronized` should be `private`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

synchronized – Putting It Together

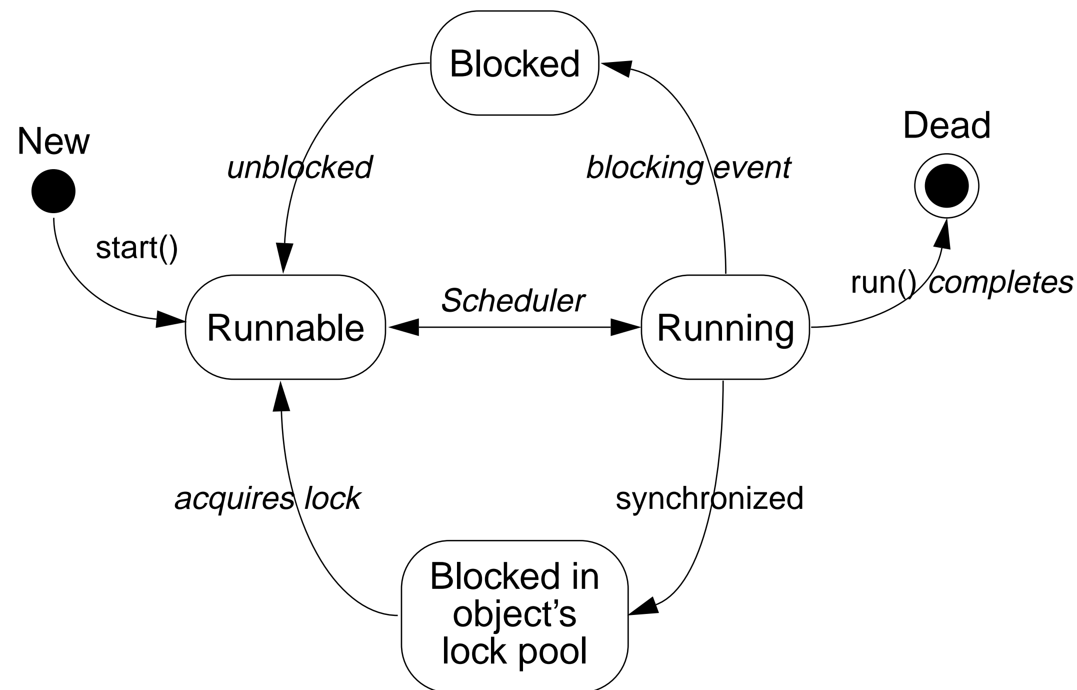
- The following two code segments are equivalent:

```
public void push(char c) {
  synchronized(this) {
    :
    :
  }
}
```

```
public synchronized void push(char c) {
  :
  :
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Threads State Diagram With Synchronization



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Deadlock

- Is two threads, each waiting for a lock from the other
- Is not detected or avoided
- Can be avoided by:
  - Deciding on the order to obtain locks
  - Adhering to this order throughout
  - Releasing locks in reverse order

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Thread Interaction – `wait` and `notify`

- Scenario:
  - Consider yourself and a cab driver as two threads
- The problem:
  - How to determine when you are at your destination
- The solution:
  - You notify the cabbie of your destination and relax
  - Cabbie drives and notifies you upon arrival at your destination

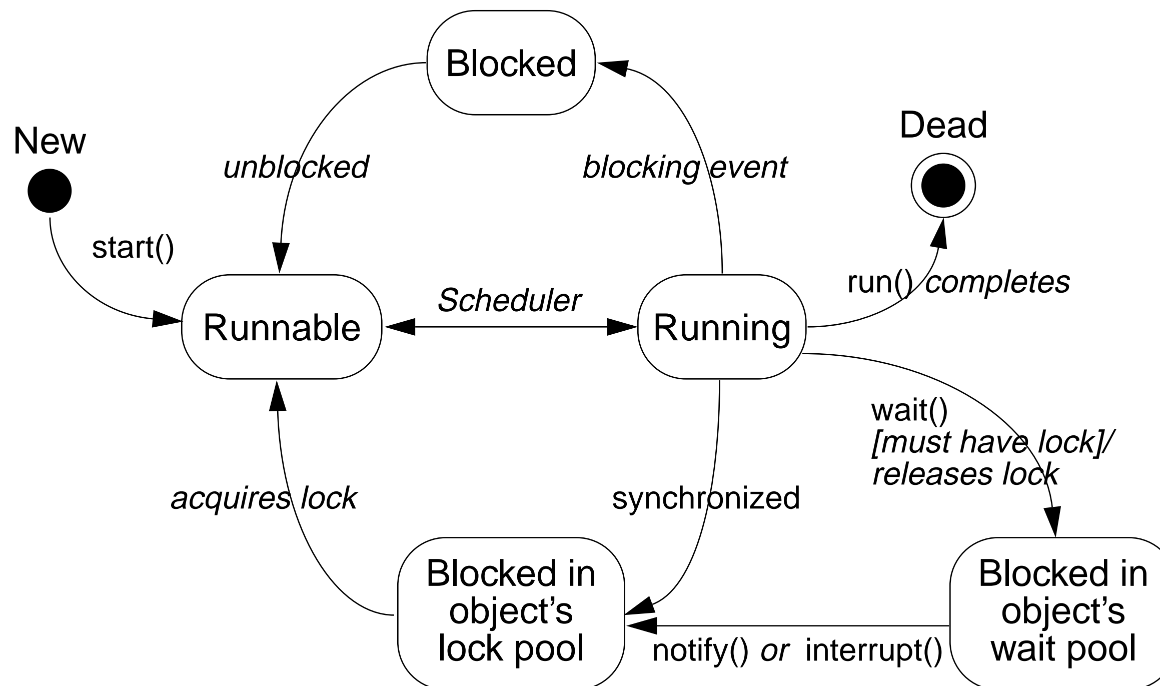
Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Thread Interaction

- `wait` and `notify`
- The pools:
  - Wait pool
  - Lock pool

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Threads State Diagram With `wait` and `notify`



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Monitor Model for Synchronization

- Leave shared data in a consistent state
- Ensure programs cannot deadlock
- Do not put threads expecting different notifications in the same wait pool

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Producer

```
1     public void run() {
2       char c;
3
4       for (int i = 0; i < 200; i++) {
5         c = (char)(Math.random() * 26 +'A');
6         theStack.push(c);
7         System.out.println("Producer" + num + ": " + c);
8         try {
9           Thread.sleep((int)(Math.random() * 300));
10        } catch (InterruptedException e) {
11          // ignore it
12        }
13      }
14    }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Consumer

```
1      public void run() {
2        char c;
3        for (int i = 0; i < 200; i++) {
4          c = theStack.pop();
5          System.out.println("Consumer" + num + ": " + c);
6
7          try {
8            Thread.sleep((int)(Math.random() * 300));
9          } catch (InterruptedException e) { }
10
11       }
12     }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

`SyncStack` Class

```
public class SyncStack {
  private List buffer = new ArrayList(400);

  public synchronized char pop() {
  }

  public synchronized void push(char c) {
  }
}
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

pop Method

```
1     public synchronized char pop() {
2       char c;
3       while (buffer.size() == 0) {
4         try {
5           this.wait();
6         } catch (InterruptedException e) {
7           // ignore it...
8         }
9       }
10      c = ((Character)buffer.remove(buffer.size()-1)).charValue();
11      return c;
12    }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

push Method

```
1    public synchronized void push(char c) {
2       this.notify();
3       Character charObj = new Character(c);
4       buffer.addElement(charObj);
5    }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

SyncTest.java

```
1   package threads;
2
3   public class SyncTest {
4
5     public static void main(String[] args) {
6
7       SyncStack stack = new SyncStack();
8
9       Producer p1 = new Producer(stack);
10      Thread prodT1 = new Thread (p1);
11      prodT1.start();
12
13      Producer p2 = new Producer(stack);
14      Thread prodT2 = new Thread (p2);
15      prodT2.start();
16
17      Consumer c1 = new Consumer(stack);
18      Thread consT1 = new Thread (c1);
19      consT1.start();
20
21      Consumer c2 = new Consumer(stack);
22      Thread consT2 = new Thread (c2);
23      consT2.start();
24    }
25  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Producer.java

```
1   package threads;
2
3   public class Producer implements Runnable {
4     private SyncStack theStack;
5     private int num;
6     private static int counter = 1;
7
8     public Producer (SyncStack s) {
9       theStack = s;
10      num = counter++;
11    }
12
13    public void run() {
14      char c;
15      for (int i = 0; i < 200; i++) {
16        c = (char)(Math.random() * 26 +'A');
17        theStack.push(c);
18        System.out.println("Producer" + num + ": " + c);
19        try {
20          Thread.sleep((int)(Math.random() * 300));
21        } catch (InterruptedException e) {
22          // ignore it
23        }
24      }
25    }
26  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Consumer.java

```
1   package threads;
2
3   public class Consumer implements Runnable {
4     private SyncStack theStack;
5     private int num;
6     private static int counter = 1;
7
8     public Consumer (SyncStack s) {
9       theStack = s;
10      num = counter++;
11    }
12
13    public void run() {
14      char c;
15      for (int i = 0; i < 200; i++) {
16        c = theStack.pop();
17        System.out.println("Consumer" + num + ": " + c);
18
19        try {
20          Thread.sleep((int)(Math.random() * 300));
21        } catch (InterruptedException e) { }
22
23      }
24    }
25  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

SyncStack.java

```
1   package threads;
2
3   import java.util.*;
4
5   public class SyncStack {
6     private List buffer = new ArrayList(400);
7
8     public synchronized char pop() {
9       char c;
10      while (buffer.size() == 0) {
11        try {
12          this.wait();
13        } catch (InterruptedException e) {
14          // ignore it...
15        }
16      }
17      c = ((Character)buffer.remove(buffer.size()-1)).
18          charValue();
19      return c;
20    }
21
22    public synchronized void push(char c) {
23      this.notify();
24      Character charObj = new Character(c);
25      buffer.addElement(charObj);
26    }
27  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

SyncStack Example

```
Producer2: F
Consumer1: F
Producer2: K
Consumer2: K
Producer2: T
Producer1: N
Producer1: V
Consumer2: V
Consumer1: N
Producer2: V
Producer2: U
Consumer2: U
Consumer2: V
Producer1: F
Consumer1: F
Producer2: M
Consumer2: M
Consumer2: T
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The `suspend` and `resume` Methods

- Have been deprecated in JDK 1.2
- Should be replaced with `wait` and `notify`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

The `stop` Method

- Releases the lock before it terminates
- Can leave shared data in an inconsistent state
- Should be replaced with `wait` and `notify`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Proper Thread Control

```
1   public class ControlledThread extends Thread {
2     static final int SUSP = 1;
3     static final int STOP = 2;
4     static final int RUN = 0;
5     private int state = RUN;
6
7     public synchronized void setState(int s) {
8       state = s;
9       if ( s == RUN ) {
10        notify();
11      }
12    }
13
14    public synchronized boolean checkState() {
15      while ( state == SUSP ) {
16        try {
17          wait();
18        } catch (InterruptedException e) {
19          // ignore
20        }
21      }
22      if ( state == STOP ) {
23        return false;
24      }
25      return true;
26    }
27
28    public void run() {
29      while ( true ) {
30        // doSomething();
31
32        // Be sure shared data is in consistent state in
33        // case the thread is waited or marked for exiting
34        // from run()
35        if ( !checkState() ) {
36          break;
37        }
38      }
39    }} // just to fit it on this page
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Using Multithreaded Programming

- Exercise objectives:
  - Become familiar with the concepts of multithreading by writing some multithreaded programs
  - Create a multithreaded applet
- Tasks:
  - Using threads to create animation
  - Using advanced thread control

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Define a thread
- Create separate threads in a Java program, controlling the code and data that are used by that thread
- Control the execution of a thread and write platform-independent code with threads
- Describe the difficulties that might arise when multiple threads share data

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Use `wait` and `notify` to communicate between threads
- Use `synchronized` to protect data from corruption
- Explain why `suspend`, `resume`, and `stop` methods have been deprecated in JDK 1.2

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Think Beyond

- Do you have applications that could benefit from being multithreaded?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Module 15

Advanced I/O Streams

Objectives

- Describe the main features of the `java.io` package
- Construct node and processing streams, and use them appropriately
- Distinguish readers and writers from streams, and select appropriately between them
- Use the `Serialization` interface to encode the state of an object

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Relevance

- What mechanisms are in place within the Java programming language to read and write from sources (or sinks) other than files?
- How are international character sets supported in I/O operations?
- What are the possible sources and sinks of character and byte streams?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

I/O Fundamentals

- A *stream* can be thought of as a flow of data from a source to a sink
- A *source* stream initiates the flow of data, also called an input stream
- A *sink* stream terminates the flow of data, also called an output stream
- Sources and sinks are both *node streams*
- Types of node streams are: files, memory, and pipes between threads or processes

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Fundamental Stream Classes

	byte streams	**character streams**
source streams	`InputStream`	`Reader`
sink streams	`OutputStream`	`Writer`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Data Within Streams

- Java technology supports two types of streams: character and byte
- Input and output of character data is handled by readers and writers
- Input and output of byte data is handled by input streams and output streams
  - Normally, the term stream refers to a byte stream
  - The terms reader and writer refer to character streams

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

`InputStream` Methods

- The three basic `read` methods:

```
int read()
int read(byte[] buffer)
int read(byte[] buffer, int offset, int length)
```

- The other methods:

```
void close()
int available()
skip(long n)
boolean markSupported()
void mark(int readlimit)
void reset()
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

OutputStream Methods

- The three basic write methods:

```
void write(int c)
void write(byte[] buffer)
void write(byte[] buffer, int offset, int length)
```

- The other methods:

```
void close()
void flush()
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

`Reader` Methods

- The three basic `read` methods:

```
int read()
int read(char[] cbuf)
int read(char[] cbuf, int offset, int length)
```

- The other methods:

```
void close()
boolean ready()
skip(long n)
boolean markSupported()
void mark(int readAheadLimit)
void reset()
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

`Writer` Methods

- The three basic `write` methods:

```
void write(int c)
void write(char[] cbuf)
void write(char[] cbuf, int offset, int length)
void write(String string)
void write(String string, int offset, int length)
```

- The other methods:

```
void close()
void flush()
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Node Streams

Type	Character Streams	Byte Streams
File	`FileReader` `FileWriter`	`FileInputStream` `FileOutputStream`
Memory: Array	`CharArrayReader` `CharArrayWriter`	`ByteArrayInputStream` `ByteArrayOutputStream`
Memory: String	`StringReader` `StringWriter`	
Pipe	`PipedReader` `PipedWriter`	`PipedInputStream` `PipedOutputStream`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

A Simple Example

- This program performs a copy file operation:

```
> java TestNodeStreams file1 file2
```

```
1   import java.io.*;
2
3   public class TestNodeStreams {
4     public static void main(String[] args) {
5       try {
6         FileReader input = new FileReader(args[0]);
7         FileWriter output = new FileWriter(args[1]);
8         char[]     buffer = new char[128];
9         int        charsRead;
10
11        // read the first buffer
12        charsRead = input.read(buffer);
13
14        while ( charsRead != -1 ) {
15          // write the buffer out to the output file
16          output.write(buffer, 0, charsRead);
17
18          // read the next buffer
19          charsRead = input.read(buffer);
20        }
21
22        input.close();
23        output.close();
24      } catch (IOException e) {
25        e.printStackTrace();
26      }
27    }
28  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Buffered Streams

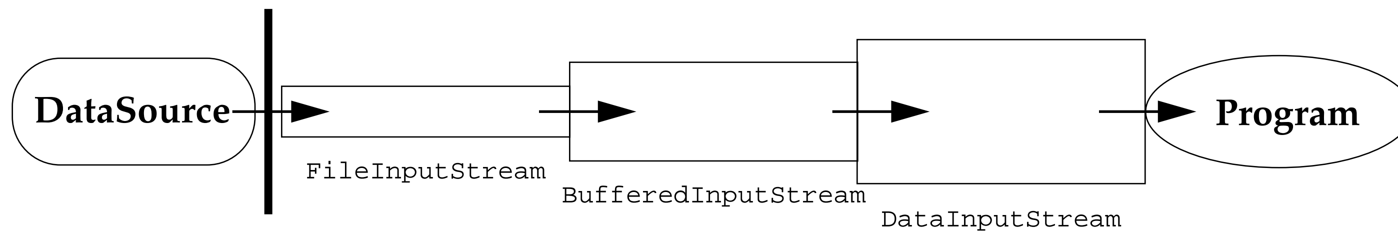
> **java TestBufferedStreams file1 file2**

```
1   import java.io.*;
2
3   public class TestBufferedStreams {
4     public static void main(String[] args) {
5       try {
6         FileReader     input     = new FileReader(args[0]);
7         BufferedReader bufInput  = new BufferedReader(input);
8         FileWriter     output    = new FileWriter(args[1]);
9         BufferedWriter bufOutput = new BufferedWriter(output);
10        String line;
11
12        // read the first line
13        line = bufInput.readLine();
14
15        while ( line != null ) {
16          // write the line out to the output file
17          bufOutput.write(line, 0, line.length());
18          bufOutput.newLine();
19
20          // read the next line
21          line = bufInput.readLine();
22        }
23
24        bufInput.close();
25        bufOutput.close();
26      } catch (IOException e) {
27        e.printStackTrace();
28      }
29    }
30  }
```

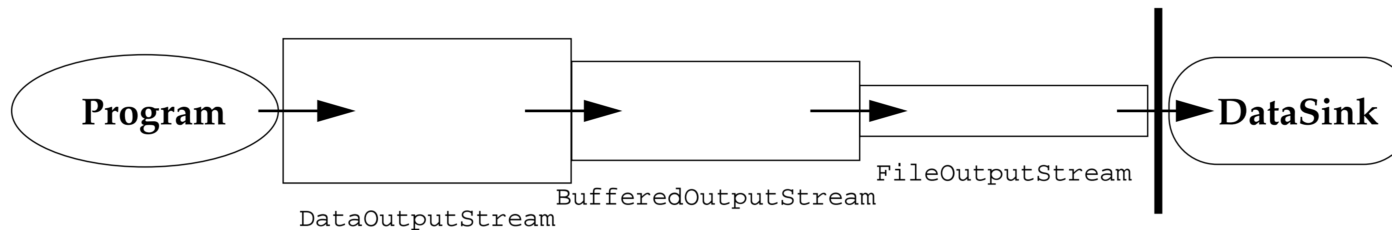

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

I/O Stream Chaining

Input Stream Chain



Output Stream Chain



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Processing Streams

Type	Character Streams	Byte Streams
Buffering	`BufferedReader` `BufferedWriter`	`BufferedInputStream` `BufferedOutputStream`
Filtering	*`FilterReader`* *`FilterWriter`*	*`FilterInputStream`* *`FilterOutputStream`*
Converting between bytes and character	`InputStreamReader` `OuptutStreamWriter`	
Object serialization		`ObjectInputStream` `ObjectOutputStream`
Data conversion		`DataInputStream` `DataOutputStream`
Counting	`LineNumberReader`	`LineNumberInputStream`
Peeking ahead	`PushbackReader`	`PushbackInputStream`
Printing	`PrintWriter`	`PrintStream`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Processing Streams as Decorators

- A Decorator is a design pattern that allows one object (the decorator) to wrap around another object
- The `FilterXxx` classes provide a base class for you to extend to provide your own processing of an input or output stream
- For example, you could write a pair of classes, `RecordInputStream` and `RecordOutputStream`, that reads and writes database records to a stream
- A program could then decorate a file input stream with a record input stream to read database records

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Record Streams Classes

InputStream

FilterInputStream

#in : InputStream

#FilterInputStream(InputStream)

RecordInputStream

+RecordInputStream(DataInputStream)

+readRecord() : Record

OutputStream

FilterOutputStream

#out : OutputStream

#FilterOutputStream(OutputStream)

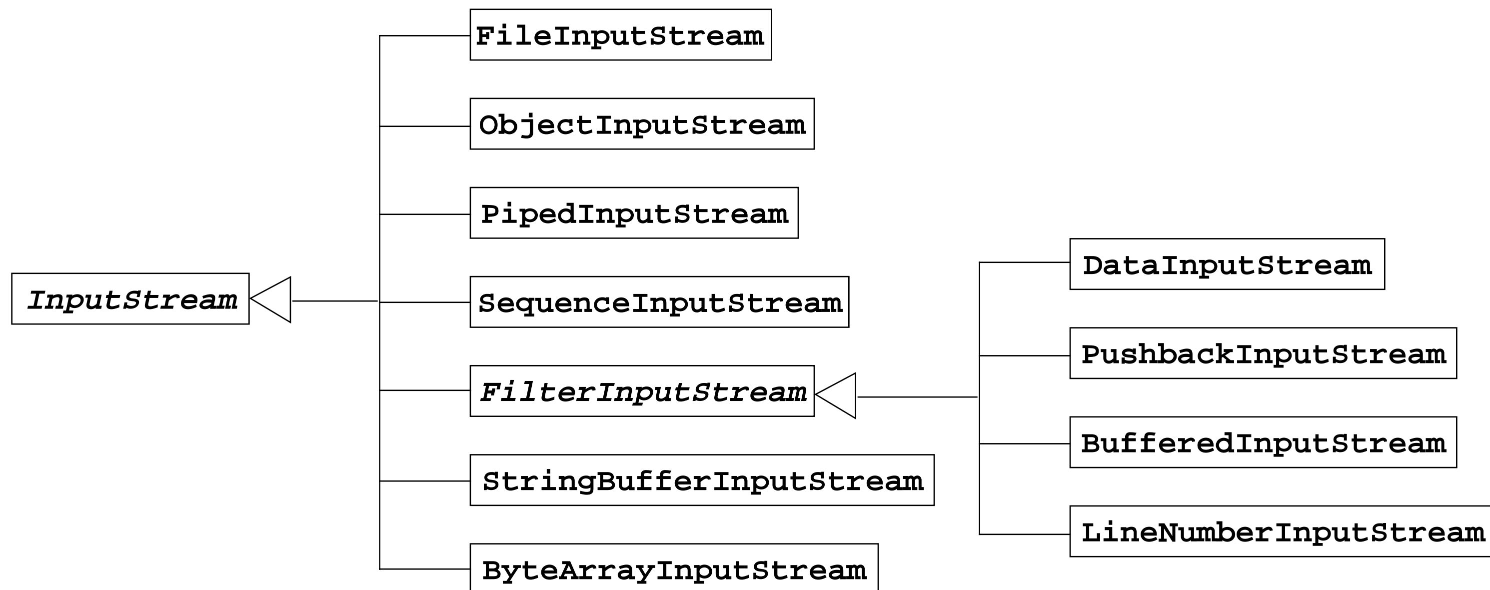
RecordOutputStream

+RecordOutputStream(DataOutputStream)

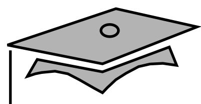
+writeRecord(record : Record)

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

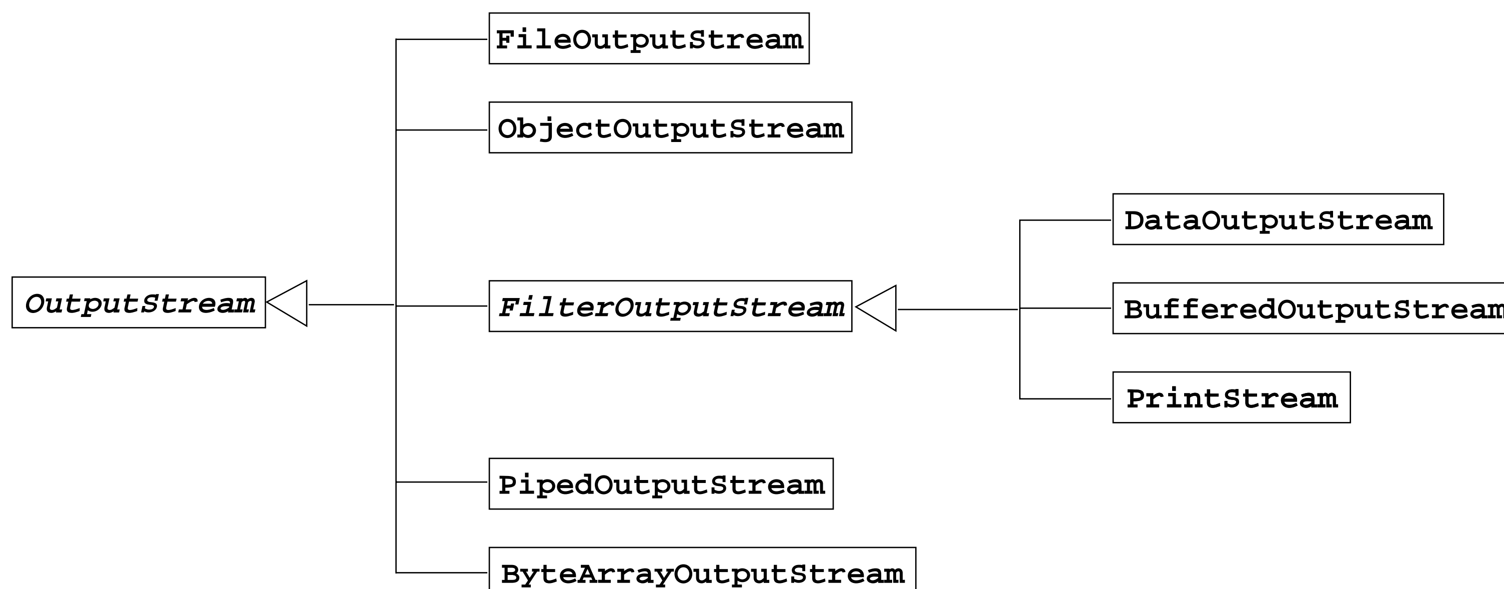
Input Stream Class Hierarchy



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

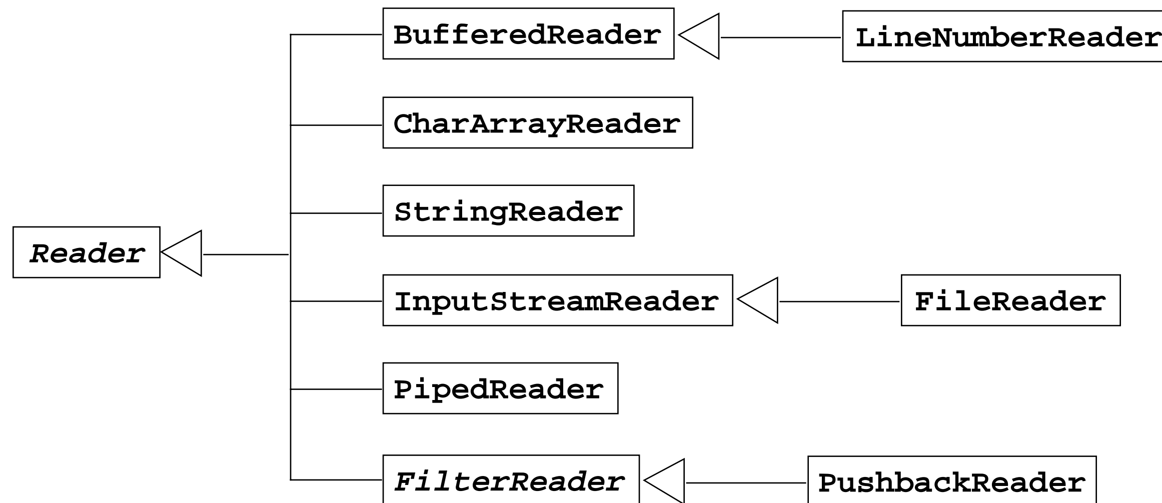


Output Stream Class Hierarchy



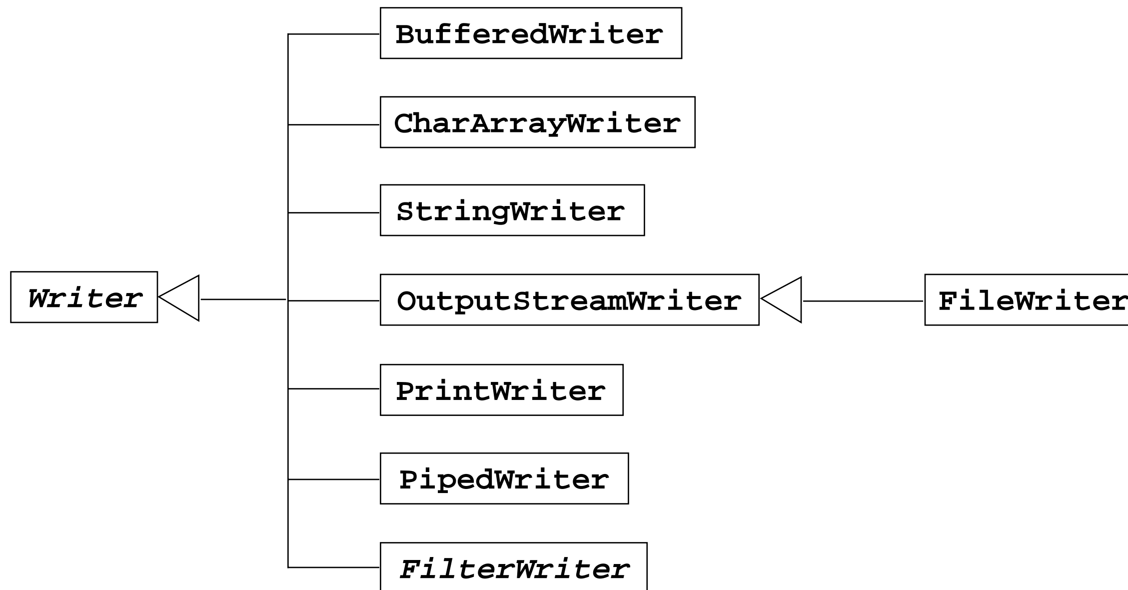
Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Reader Class Hierarchy



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Writer Class Hierarchy



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

URL Input Streams

```
1   java.net.URL imageSource;
2
3   try {
4     imageSource = new URL("http://mysite.com/~info");
5   } catch (MalformedURLException e) {
6     // ignore
7   }
8
9   images[0] = getImage(imageSource, "Duke/T1.gif");
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Opening a URL Input Stream

```
1  InputStream is = null;
2  String fileName = new String("Data/data.1-96");
3  byte buffer[] = new byte[24];
4
5  try {
6    // new URL throws a MalformedURLException
7    URL fileLocation = new URL(getDocumentBase(), fileName);
8
9    // URL.openStream() throws an IOException
10   is = fileLocation.openStream();
11 } catch (Exception e) {
12   // ignore
13 }
```

Now you can use the variable `is` to read information, just as with a `FileInputStream` object:

```
14 try {
15   is.read(buffer, 0, buffer.length);
16 } catch (IOException e1) {
17   // ignore
18 }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Creating a Random Access File

- With the file name:

```
myRAFile = new RandomAccessFile(
                  String name, String mode);
```

- With a `File` object:

```
myRAFile = new RandomAccessFile(
                  File file, String mode);
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Random Access Files

- `long getFilePointer()`
- `void seek(long pos)`
- `long length()`

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Serialization

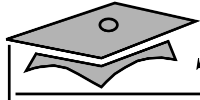
- Only the object's data are serialized
- Data marked with the `transient` keyword are not serialized

```
1   public class MyClass implements Serializable {
2     public transient Thread myThread;
3     private String customerID;
4     private int total;
5   }
```

```
1   public class MyClass implements Serializable {
2     public transient Thread myThread;
3     private transient String customerID;
4     private int total;
5   }
```

- Serialization is used to store the state of an object to a file; storing the state of an object is called *persistence*

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D



Writing an Object to a File Stream

```
1   import java.io.*;
2   import java.util.Date;
3
4   public class SerializeDate {
5
6     SerializeDate() {
7       Date d = new Date ();
8
9       try {
10        FileOutputStream f =
11            new FileOutputStream ("date.ser");
12        ObjectOutputStream s =
13            new ObjectOutputStream (f);
14        s.writeObject (d);
15        s.close ();
16      } catch (IOException e) {
17        e.printStackTrace ();
18      }
19    }
20
21    public static void main (String args[]) {
22      new SerializeDate();
23    }
24  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Reading an Object From a File Stream

```
1   import java.io.*;
2   import java.util.Date;
3
4   public class UnSerializeDate {
5
6     UnSerializeDate () {
7       Date d = null;
8
9       try {
10        FileInputStream f =
11            new FileInputStream ("date.ser");
12        ObjectInputStream s =
13            new ObjectInputStream (f);
14        d = (Date) s.readObject ();
15        s.close ();
16      } catch (Exception e) {
17        e.printStackTrace ();
18      }
19
20      System.out.println(
21        "Unserialized Date object from date.ser");
22      System.out.println("Date: "+d);
23    }
24
25    public static void main (String args[]) {
26      new UnSerializeDate();
27    }
28  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Getting Acquainted With I/O

- Exercise objective:
  - Become familiar with stream I/O by writing programs that perform I/O operations
- Tasks:
  - Object serialization
  - Implementing a processing stream
  - Create a simple database program

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Describe the main features of the `java.io` package
- Construct node and processing streams, and use them appropriately
- Distinguish readers and writers from streams, and select appropriately between them
- Use the `Serialization` interface to encode the state of an object

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Think Beyond

- Do you have applications that could benefit from creating specialized stream or character filters?

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Module 16

Networking

Objectives

- Develop code to set up the network connection
- Understand the TCP/IP protocol
- Use `ServerSocket` and `Socket` classes for implementing TCP/IP clients and servers

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Relevance

- How can a communication link between a client machine and a server on the network be established?

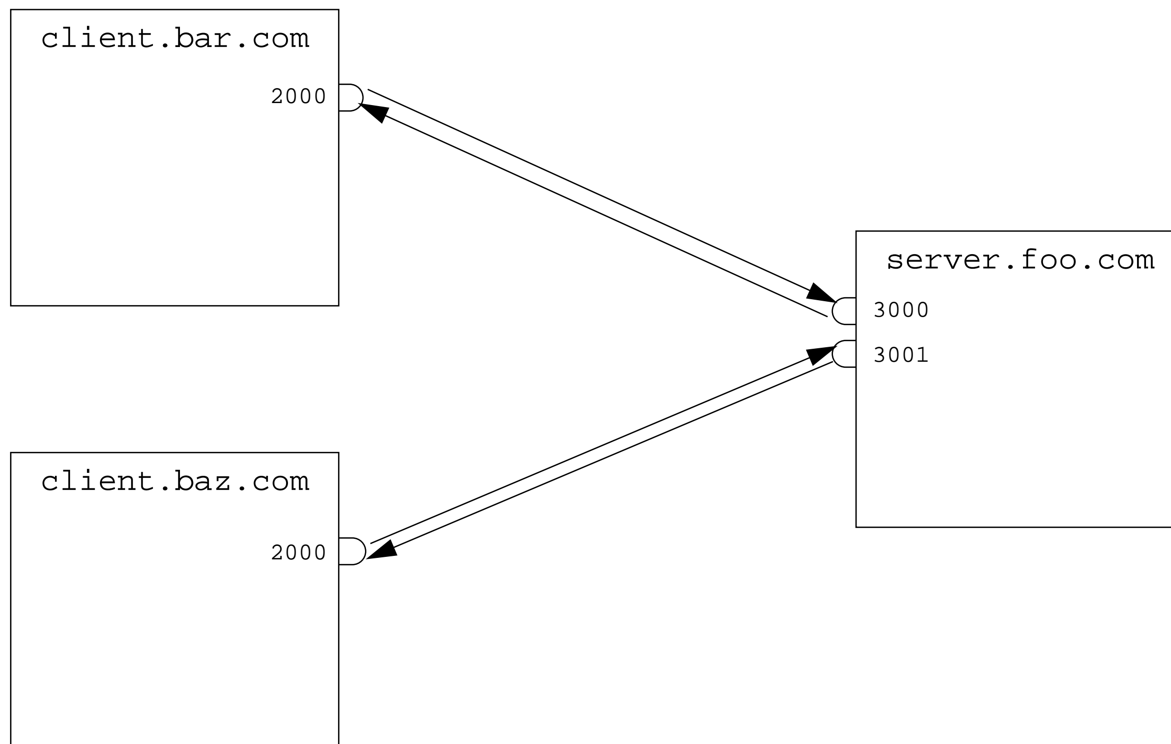
Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Networking

- Sockets:
  - Sockets hold two streams
- Setting up the connection:
  - Set up is similar to a telephone system

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Networking



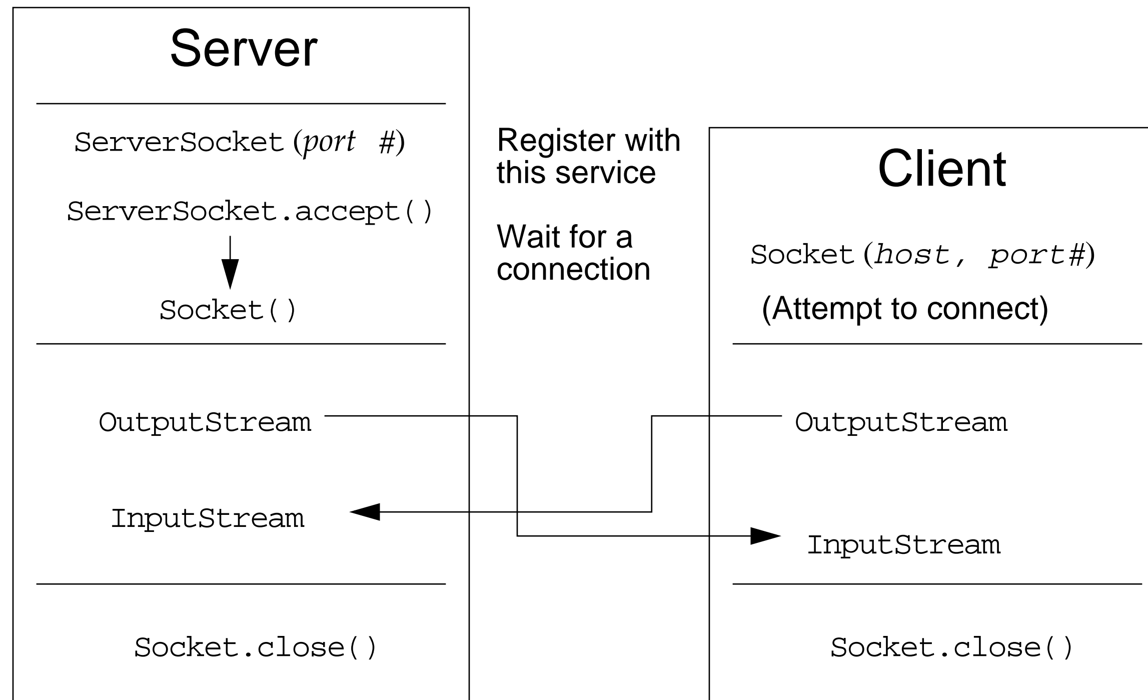
Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Networking With Java Technology

- Addressing the connection:
  - Address or name of remote machine
  - Port number to identify purpose
- Port numbers:
  - Range from 0 to 65535

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Java Networking Model



Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Minimal TCP/IP Server

```
1   import java.net.*;
2   import java.io.*;
3
4   public class SimpleServer {
5     public static void main(String args[]) {
6       ServerSocket s;
7
8       // Register your service on port 5432
9       try {
10        s = new ServerSocket(5432);
11      } catch (IOException e) {
12        // ignore
13      }
14
15     // Run the listen/accept loop forever
16      while (true) {
17        try {
18          // Wait here and listen for a connection
19          Socket s1 = s.accept();
20
21          // Get output stream associated with the socket
22          OutputStream s1out = s1.getOutputStream();
23          DataOutputStream dos = new DataOutputStream(s1out);
24
25          // Send your string!
26          dos.writeUTF("Hello Net World!");
27
28          // Close the connection, but not the server socket
29          dos.close();
30          s1.close();
31        } catch (IOException e) {
32          // ignore
33        }
34      }
35    }
36  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Minimal TCP/IP Client

```
1   import java.net.*;
2   import java.io.*;
3
4   public class SimpleClient {
5     public static void main(String args[]) {
6       try {
7         // Open your connection to a server, at port 5432
8         // localhost used here
9         Socket s1 = new Socket("127.0.0.1", 5432);
10
11        // Get an input stream from the socket
12        InputStream is = s1.getInputStream();
13        // Decorate it with a "data" input stream
14        DataInputStream dis = new DataInputStream(is);
15
16        // Read the input and print it to the screen
17        System.out.println(dis.readUTF());
18
19        // When done, just close the steam and connection
20        dis.close();
21        s1.close();
22      } catch (ConnectException connExc) {
23        System.err.println("Could not connect to the server.");
24      } catch (IOException e) {
25        // ignore
26      }
27    }
28  }
```


Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Exercise: Using Socket Programming

- Exercise objective:
  - Gain experience using sockets by implementing a client which communicates to a server using sockets
- Tasks:
  - Complete the ChatClient program using a TCP/IP server
  - Create a simple file transfer server and client

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Check Your Progress

- Develop code to set up the network connection
- Understand the TCP/IP protocol
- Use `ServerSocket` and `Socket` classes for implementing TCP/IP clients and servers

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Think Beyond

- How can you create a distributed object system using object serialization and these network protocols? Have you heard of Remote Method Invocation (RMI)?
- There are several advanced Java platform topics, many of which are addressed in other Sun Educational Services courses. Be sure and check out the JavaSoft™ web site (`www.javasoft.com`) as well.

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000 Revision D

Copyright 2000 Sun Microsystems Inc., 901 San Antonio Road, Palo Alto, California 94303, Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a.

Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.
Des parties de ce produit pourront être dérivées du systèmes Berkeley 4.3 BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company Ltd.

Sun, Sun Microsystems, le logo Sun, Java, JavaOS, JVM et Solaris sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

Toutes les marques SPARC sont utilisées sous licence sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays.

Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

UNIX est une marques déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

L'interfaces d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour larecherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

L'accord du gouvernement américain est requis avant l'exportation du produit.

Le système X Window est un produit de X Consortium, Inc.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.

About This Course ....About This Course-1
Course Goals ....About This Course-2
Course Overview ....About This Course-3
Course Map ....About This Course-4
Module-by-Module Overview ....About This Course-5
Course Objectives ....About This Course-7
Skills Gained by Module ....About This Course-9
Guidelines for Module Pacing ....About This Course-10
Topics Not Covered ....About This Course-11
How Prepared Are You? ....About This Course-12
Introductions ....About This Course-13
How to Use Course Materials ....About This Course-14
Course Icons ....About This Course-15
Typographical Conventions ....About This Course-16

Getting Started 1-1
Objectives 1-2
Relevance 1-3
What Is the
Java Technology? 1-4
Primary Goals of the Java Technology 1-5
A Basic Java Application 1-8
Compiling and Running `TestGreeting` 1-9
Compile-Time Errors 1-10
Runtime Errors 1-11
Java Runtime Environment 1-12
The Java Virtual Machine 1-13
Garbage Collection 1-16
Code Security 1-17
Just-In-Time Code Generator 1-18

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Java Runtime Environment 1-19
Class Loader 1-20
Bytecode Verifier 1-21
Exercise: Performing Basic Java Tasks 1-22
Check Your Progress 1-23
Think Beyond 1-24

Object-Oriented Programming 2-1
Objectives 2-2
Relevance 2-4
Software Engineering 2-5
Analysis and Design 2-6
Abstraction 2-7
Classes as Blueprints for Objects 2-8
Declaring Java Classes 2-9
Declaring Attributes 2-10
Declaring Methods 2-11
Accessing Object Members 2-12
Information Hiding 2-13
Encapsulation 2-15
Declaring Constructors 2-16
The Default Constructor 2-18
Source File Layout 2-19
Software Packages 2-20
The `package` Statement 2-21
The import Statement 2-22
Directory Layout and Packages 2-23
Terminology Recap 2-25
Using the Java API Documentation 2-26
Example API Documentation Page 2-27

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Exercise: Using Objects and Classes 2-28
Check Your Progress 2-29
Think Beyond 2-31

Identifiers, Keywords, and Types 3-1
Objectives 3-2
Relevance 3-4
Comments 3-5
Semicolons, Blocks, and Whitespace 3-6
Identifiers 3-8
Java Keywords 3-9
Primitive Types 3-10
Logical – boolean 3-11
Textual – char and String 3-12
Integral – byte, short, int, and long 3-14
Floating Point – float and double 3-16
Variables, Declarations, and Assignments 3-18
Java Reference Types 3-19
Constructing and Initializing Objects 3-20
Memory Allocation and Layout 3-21
Explicit Attribute Initialization 3-22
Executing the Constructor 3-23
Variable Assignment 3-24
Assignment of Reference Variables 3-25
Pass-by-Value 3-27
The `this` Reference 3-29
Java Coding Conventions 3-31
Exercise: Using Identifiers, Keywords, and Types 3-33
Check Your Progress 3-34
Think Beyond 3-36

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Expressions and Flow Control .... _4-1_
Objectives 4-2
Relevance 4-4
Variables and Scope 4-5
Variable Scope Example 4-6
Variable Initialization 4-7
Operators 4-8
Logical Operators 4-9
Bitwise Logical Operators 4-10
Right-Shift Operators >> and >>> 4-11
Left-Shift Operator (<<) 4-12
Shift Operator Examples 4-13
String Concatenation With + ... 4-14
Casting 4-15
Promotion and Casting of Expressions 4-16
Branching Statements 4-17
Looping Statements 4-22
Special Loop Flow Control 4-25
Exercise: Using Expressions 4-30
Check Your Progress 4-31
Think Beyond 4-33

Arrays .... _5-1_
Objectives 5-2
Relevance 5-3
Declaring Arrays 5-4
Creating Arrays 5-5
Initializing Arrays 5-7
Multi-Dimensional Arrays 5-8
Array Bounds 5-10

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Array Resizing 5-11
Exercise: Using Arrays 5-13
Check Your Progress 5-14
Think Beyond 5-15

Inheritance* *6-1
Objectives 6-2
Relevance 6-4
The `is a` Relationship 6-5
Single Inheritance 6-8
Constructors Are Not Inherited 6-10
Polymorphism 6-11
Polymorphic Arguments 6-13
Heterogeneous Collections 6-14
The `instanceof` Operator 6-15
Casting Objects 6-16
The `has a` Relationship 6-17
Access Control 6-18
Overloading Method Names 6-19
Overloading Constructors 6-20
Overriding Methods 6-22
Rules About Overridden Methods 6-25
The `super` Keyword 6-27
Invoking Parent Class Constructors 6-29
Constructing and Initializing Objects:
A Slight Reprise 6-31
An Example 6-32
Implications of the Initialization Process 6-34
The `Object` Class 6-35
The == Operator Compared With the equals Method 6-36

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

`equals` Example 6-37
`toString` Method 6-39
Wrapper Classes 6-40
Exercise: Using Objects and Classes 6-42
Check Your Progress 6-43
Think Beyond 6-45

Advanced Class Features 7-1
Objectives 7-2
Relevance 7-4
The `static` Keyword 7-5
Class Attributes 7-6
Class Methods 7-8
Static Initializers 7-9
The Singleton Design Pattern 7-11
Implementing the Singleton Design Pattern 7-12
The final Keyword 7-13
Final Variables 7-14
Exercise: Working With the `static` and `final` Keywords 7-15
Abstract Classes: Scenario 7-16
Abstract Classes: Solution 7-18
Template Method Design Pattern 7-20
Interfaces 7-21
Interface Example 7-22
Multiple Interface Example 7-28
Uses of Interfaces 7-30
Inner Classes 7-31
Inner Class Example 7-32
Properties of Inner Classes 7-36
Exercise: Working With Interfaces and Abstract Classes 7-39

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Check Your Progress 7-40
Think Beyond 7-42

Exceptions* *8-1
Objectives 8-2
Relevance 8-3
Exceptions 8-4
Exception Example 8-5
`try` and `catch` Statements 8-6
Call Stack Mechanism 8-7
`finally` Statement 8-8
Exception Example Revisited 8-9
Exception Categories 8-10
Common Exceptions 8-11
The Handle or Declare Rule 8-12
Method Overriding and Exceptions 8-13
Method Overriding Examples 8-15
Creating Your Own Exceptions 8-16
Handling User-Defined Exceptions 8-17
Exercise: Working With Exceptions 8-18
Check Your Progress 8-19
Think Beyond 8-20

Text-Based Applications* *9-1
Objectives 9-2
Relevance 9-4
Command-Line Arguments 9-5
Command-Line Args 9-6
System Properties 9-7
The `Properties` Class 9-8

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

System Properties 9-9
Console I/O 9-10
Writing to Standard Output 9-11
Reading From Standard Input 9-12
Files and File I/O 9-13
Creating a New `File` Object 9-14
`File` Tests and Utilities 9-15
File Stream I/O 9-17
File Input Example 9-18
File Output Example 9-19
Exercise: Writing User Input to a File 9-20
The `Math` Class 9-21
The `String` Class 9-22
The `StringBuffer` Class 9-23
The Collections API 9-24
Collections 9-25
A Set Example 9-27
A List Example 9-28
Iterators 9-29
The Iterator Interface Hierarchy 9-30
Maps 9-31
Map: Word Counter 9-32
Sorting Arrays and Collections 9-33
Sorting Arrays 9-34
Sorting Lists 9-35
Collections in JDK 1.1 9-36
Exercise: Using Collections to Represent Aggregation 9-37
Using the `javadoc` Tool 9-38
Documentation Tags 9-39
Example Java File 9-40
Public Documentation 9-41

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Private Documentation 9-42
Deprecation 9-43
Using the jar Tool 9-47
Exercise: Building a System 9-48
Check Your Progress 9-49
Think Beyond 9-51

Building Java GUIs 10-1
Objectives 10-2
Relevance 10-4
Abstract Window Toolkit (AWT) 10-5
The java.awt Package 10-6
Containers 10-7
Building Graphical User Interfaces 10-8
Frames 10-9
FrameExample.java 10-10
Panels 10-12
FrameWithPanel.java 10-13
Container Layouts 10-15
Default Layout Managers 10-16
A Simple FlowLayout Example 10-17
FlowLayout Manager 10-18
FlowExample.java 10-19
BorderLayout Manager 10-21
BorderExample.java 10-22
GridLayout Manager 10-24
GridExample.java 10-25
CardLayout Manager 10-27
CardExample.java 10-28
CardLayout Manager 10-29

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

`GridBagLayout` Manager 10-31
`ComplexLayoutExample.java` 10-32
Output of ComplexLayoutExample.java 10-33
Drawing in AWT 10-34
Drawing With the Graphics Object 10-35
Exercise: Building Java GUIs 10-36
Check Your Progress 10-37
Think Beyond 10-39

GUI Event Handling 11-1
Objectives 11-2
Relevance 11-4
What Is an Event? 11-5
Delegation Model 11-6
Event Categories 11-9
Java GUI Behavior 11-11
Complex Example 11-12
Multiple Listeners 11-14
Event Adapters 11-15
Inner Classes 11-16
Anonymous Classes 11-17
Exercise: Working With Events 11-18
Check Your Progress 11-19
Think Beyond 11-21

Introduction to Java Applets 12-1
Objectives 12-2
Relevance 12-4
What Is an Applet? 12-5
Applet Security Restrictions 12-7

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Applet Class Hierarchy 12-8
Key Applet Methods 12-9
The Applet Life Cycle 12-10
Applet Display 12-11
AWT Painting 12-12
Applet Display Strategies 12-14
An Example Paint Model 12-15
What Is the `appletviewer`? 12-18
Running the `appletviewer` 12-19
The `applet` Tag 12-20
Additional Applet Features 12-21
A Simple Image Test 12-22
Audio Clips 12-23
A Simple Audio Test 12-24
Looping an Audio Clip 12-25
A Simple Looping Test 12-26
Mouse Input 12-27
A Simple Mouse Test 12-28
Reading Parameters 12-29
Exercise: Creating Applets 12-30
Check Your Progress 12-31
Think Beyond 12-33

GUI-Based Applications 13-1
Objectives 13-2
Relevance 13-3
AWT Components 13-4
Component Events 13-5
How to Create a Menu 13-6
Creating a `MenuBar` 13-7

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Creating a `Menu` 13-8
Creating a `MenuItem` 13-9
Creating a `CheckBoxMenuItem` 13-10
Controlling Visual Aspects 13-11
Printing 13-14
Dual-Purpose Code 13-15
UML Model of the `SalesOrderGUI` 13-16
The GUI Mediator Code 13-17
The App and Applet Code 13-18
Discussion of Dual-Purpose Code 13-19
Swing 13-20
Exercise: 13-21
Check Your Progress 13-22
Think Beyond 13-23

Threads* *14-1
Objectives 14-2
Relevance 14-4
Threads 14-5
Three Parts of a Thread 14-6
Creating the Thread 14-7
Starting the Thread 14-9
Thread Scheduling 14-10
Terminating a Thread 14-12
Basic Control of Threads 14-13
Putting Threads on Hold 14-14
Extending the Thread Class 14-16
Selecting a Way to Create Threads 14-17
Exercise: Using Basic Threads 14-18
Using the synchronized Keyword 14-19

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

The Object Lock Flag 14-20
Releasing the Lock Flag 14-23
`synchronized` – Putting It Together 14-24
Threads State Diagram
With Synchronization 14-26
Deadlock 14-27
Thread Interaction – `wait` and `notify` 14-28
Thread Interaction 14-29
Threads State Diagram
With `wait` and `notify` 14-30
Monitor Model for Synchronization 14-31
`Producer` 14-32
`Consumer` 14-33
`SyncStack` Class 14-34
`pop` Method 14-35
`push` Method 14-36
`SyncTest.java` 14-37
`Producer.java` 14-38
`Consumer.java` 14-39
`SyncStack.java` 14-40
`SyncStack` Example 14-41
The `suspend` and `resume` Methods 14-42
The `stop` Method 14-43
Proper Thread Control 14-44
Exercise: Using Multithreaded Programming 14-45
Check Your Progress 14-46
Think Beyond 14-48

Advanced I/O Streams 15-1
Objectives 15-2

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Relevance 15-3
I/O Fundamentals 15-4
Fundamental Stream Classes 15-5
Data Within Streams 15-6
InputStream Methods 15-7
OutputStream Methods 15-8
Reader Methods 15-9
Writer Methods 15-10
Node Streams 15-11
A Simple Example 15-12
Buffered Streams 15-13
I/O Stream Chaining 15-14
Processing Streams 15-15
Processing Streams as Decorators 15-16
Record Streams Classes 15-17
Input Stream Class Hierarchy 15-18
Output Stream Class Hierarchy 15-19
Reader Class Hierarchy 15-20
Writer Class Hierarchy 15-21
URL Input Streams 15-22
Opening a URL Input Stream 15-23
Creating a Random Access File 15-24
Random Access Files 15-25
Serialization 15-26
Writing an Object to a File Stream 15-27
Reading an Object From a File Stream 15-28
Exercise: Getting Acquainted With I/O 15-29
Check Your Progress 15-30
Think Beyond 15-31

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D

Networking* *16-1

Objectives 16-2
Relevance 16-3
Networking 16-4
Networking With Java Technology 16-6
Java Networking Model 16-7
Minimal TCP/IP Server 16-8
Minimal TCP/IP Client 16-9
Exercise: Using Socket Programming 16-10
Check Your Progress 16-11
Think Beyond 16-12

Copyright 2000 Sun Microsystems, Inc. All Rights Reserved. Enterprise Services April 2000, Revision D