Chapter 2
First Java Programs

Fundamentals of Java

Objectives

- Discuss why Java is an important programming language.
- Explain the Java virtual machine and byte code.
- Choose a user interface style.
- Describe the structure of a simple Java program.

Objectives (cont.)

- Write a simple program.
- Edit, compile, and run a program using a Java development environment.
- Format a program to give a pleasing, consistent appearance.
- Understand compile-time errors.
- Write a simple graphics program.

Vocabulary

- Applet
- Assignment operator
- Byte code
- DOS development environment
- Graphical user interface (GUI)
- Hacking
- Integrated development environment (IDE)

Vocabulary (cont.)

- Java virtual machine (JVM)
- Just-in-time compilation (JIT)
- Parameter
- Source code
- Statement
- Terminal I/O user interface
- Variable

Why Java?

- Java is:
  - The world's fastest growing programming language
  - **Secure:** Enables construction of virus-free, tamper-free systems
  - **Robust:** Supports development of programs that do not overwrite memory
  - **Portable:** Programs can be run on different types of computers without change

Why Java? (cont.)

- Java is well-suited for distributed, network-based applications.
- Java supports **threads**.
  - A process that can run concurrently with other processes
- Java resembles C++.
  - Easy for C++ programmers to learn Java
- Java runs more slowly than other languages.

The Java Virtual Machine and Byte Code

- The Java compiler translates Java source code into Java **byte code**.
  - A pseudo-machine language
- Byte code is executed by the **Java Virtual Machine (JVM)**.
  - **Interpreter:** A program that behaves like a computer
  - Any computer with the JVM installed can execute Java byte code (portability).

The Java Virtual Machine and Byte Code (cont.)

- Some JVMs support **just-in-time compilation (JIT)**.
  - Translate byte code instructions into machine language when they are first encountered
- **Java applets:** Java programs that run in a Web browser
  - A JVM is incorporated into the browser.

Choosing a User Interface Style

- Two choices:
  - **Graphical user interface (GUI):** Program interacts with users via "windows" with graphical components
  - **Terminal I/O user interface:** Programs interact with users via a command terminal
    - MS-DOS console or Unix terminal

Choosing a User Interface Style (cont.)

Figure 2-1: Two user interfaces for a temperature conversion program

Hello World

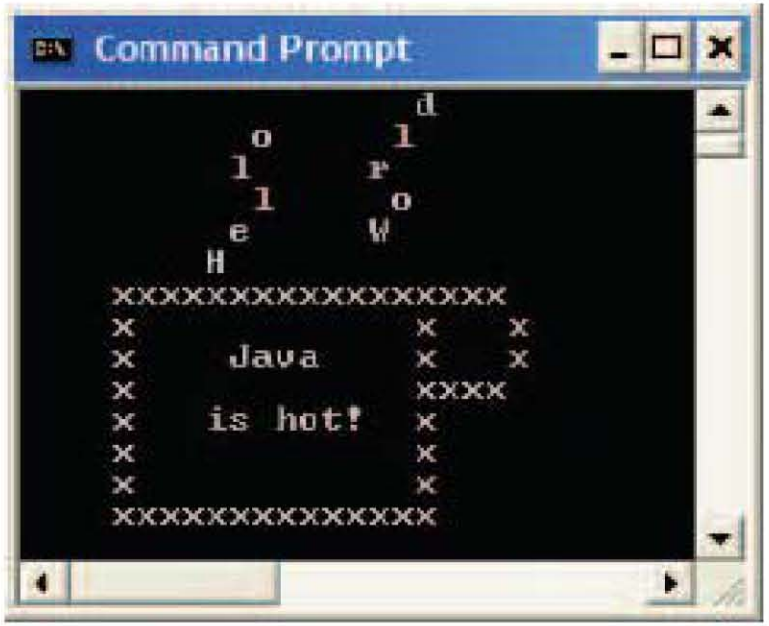


Figure 2-2: Hello World

Hello World (cont.)

- A program is a sequence of instructions to perform a task.
- **Source code:** The programming language instructions for a program

Hello World (cont.)

```
public class HelloWorld{

   public static void main(String [] args) {
      System.out.println("                  d          ");
      System.out.println("           o      l          ");
      System.out.println("         l      r            ");
      System.out.println("           l      o          ");
      System.out.println("         e      W            ");
      System.out.println("        H                    ");
      System.out.println("    xxxxxxxxxxxxxxxx         ");
      System.out.println("    x             x   x      ");
      System.out.println("    x    Java     x   x      ");
      System.out.println("    x             xxxx       ");
      System.out.println("    x   is hot!   x          ");
      System.out.println("    x             x          ");
      System.out.println("    x             x          ");
      System.out.println("    xxxxxxxxxxxxxx           ");
   }
}
```

Example 2-1: Our first program

Hello World (cont.)

- **`System.out`:** An object that knows how to display or print characters in a terminal window
  - **`println`:** The message being sent to the `System.out` object
  - Strings in quotation marks contain characters to be printed.
    - **Parameters**

Hello World (cont.)

- Semicolons (`;`) mark the end of each **statement**.
  - A "sentence" in a program
- A message may require zero, one, or multiple parameters.
- Format for sending messages to objects:
  - `<name of object>.<name of message>(<parameters>)`

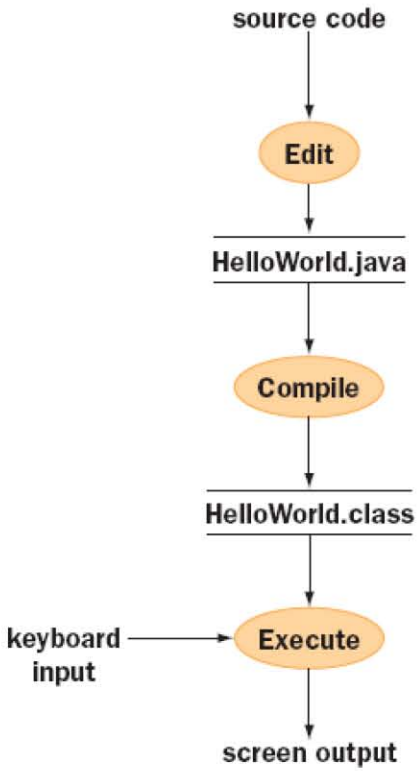
Hello World (cont.)

- **Method selector operator:** The period (.) between the object's name and the message's name
- Framework for any Java program:

```
public class <name of program> {
   public static void main(String [] args) {
      . . . put the source code here . . .
   }
}
```

Edit, Compile, and Execute

Figure 2-3: Editing, compiling, and running a program



Edit, Compile, and Execute (cont.)

- The file extension for a Java source code file is `.java`.
- The file extension for a Java class file is `.class`.

Development Environments

- Use UNIX or Linux using a standard text editor with command-line activation of the compiler and the JVM.
- **DOS development environment:**
  - Use a text editor to write source code.
  - Activate compiler and JVM from the command line in a command or DOS Window.

Development Environments (cont.)

- **Integrated development environment (IDE):** Combines an editor, Java compiler, debugger, and JVM in one program
  - Increases programmer productivity
  - Usually costs money
  - Examples: Metrowerks Code Warrior, Borland JBuilder, BlueJ, and JGrasp

Preparing Your Development Environment

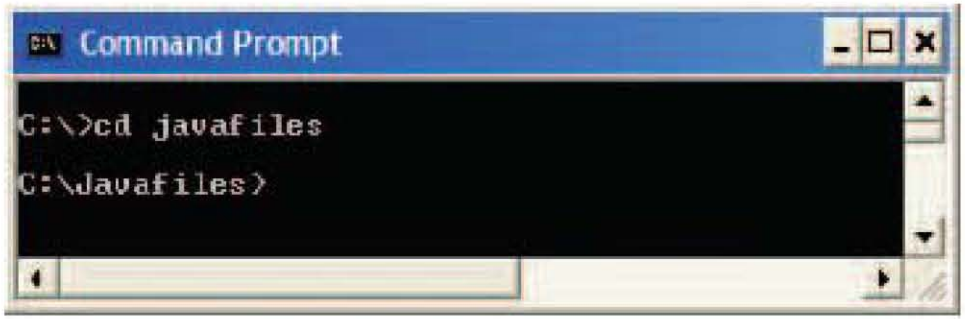


Figure 2-4: Using the cd command to move to the working directory

Preparing Your Development Environment (cont.)

Figure 2-5: Activating Notepad to edit the program

Preparing Your Development Environment (cont.)

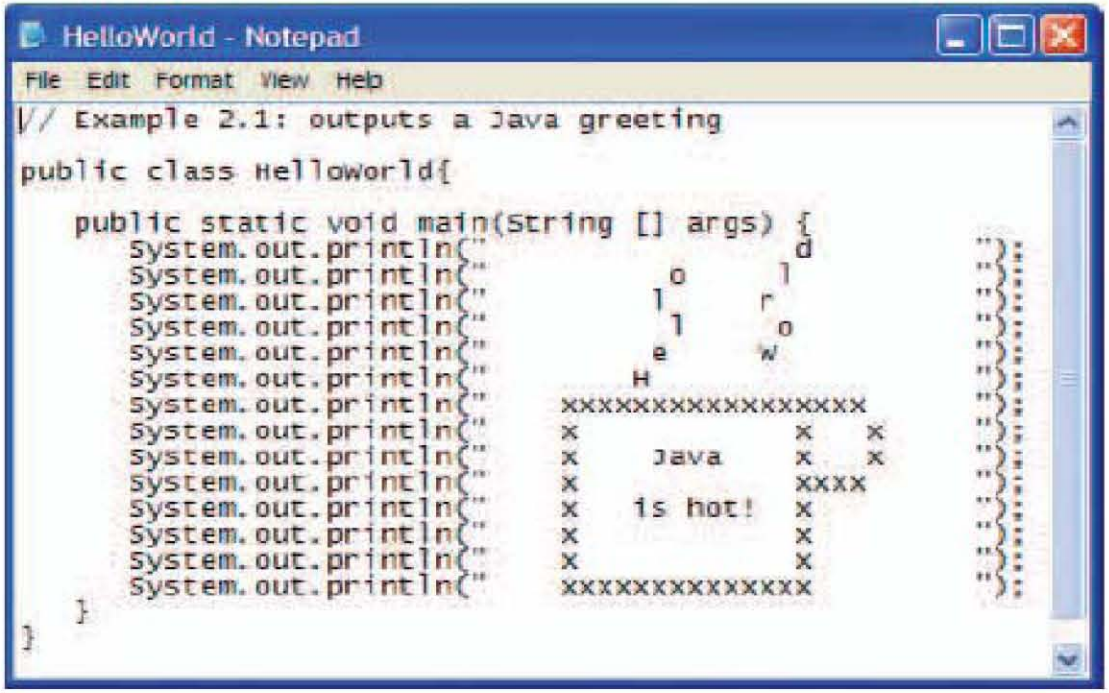


Figure 2-6: Program as typed into Notepad

Preparing Your Development Environment (cont.)

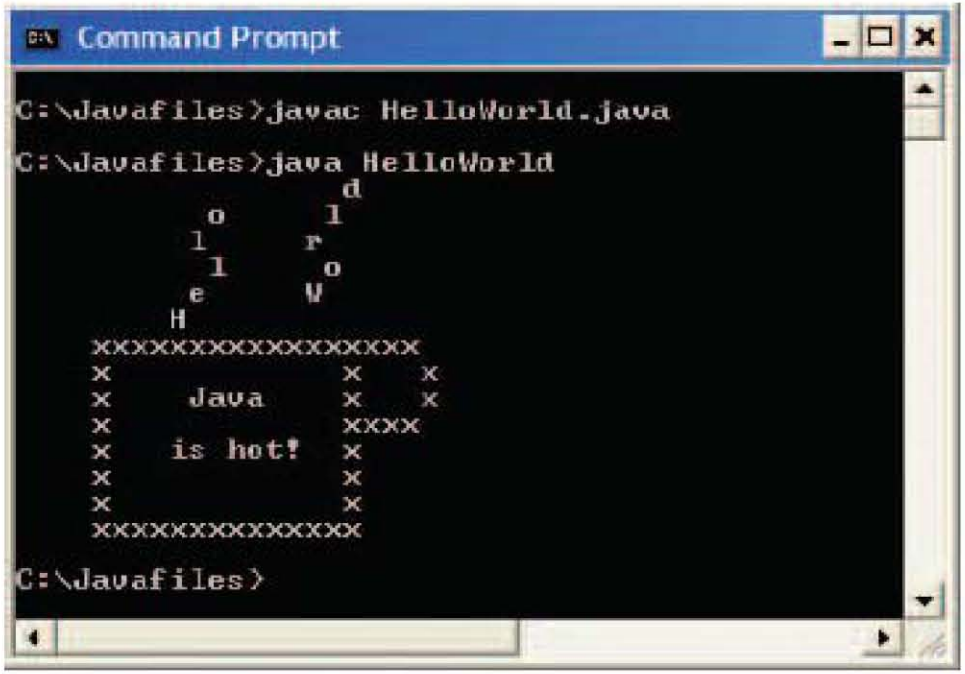


Figure 2-7: Compiling and running the program

Compile-Time Errors

- Mistakes detected by the compiler
  - Also called **syntax errors**
- Compiler displays errors in terminal window
  - Indicates file and line number where error was detected
  - Indicates type of error detected
  - May suggest solutions to error

Compile-Time Errors (cont.)

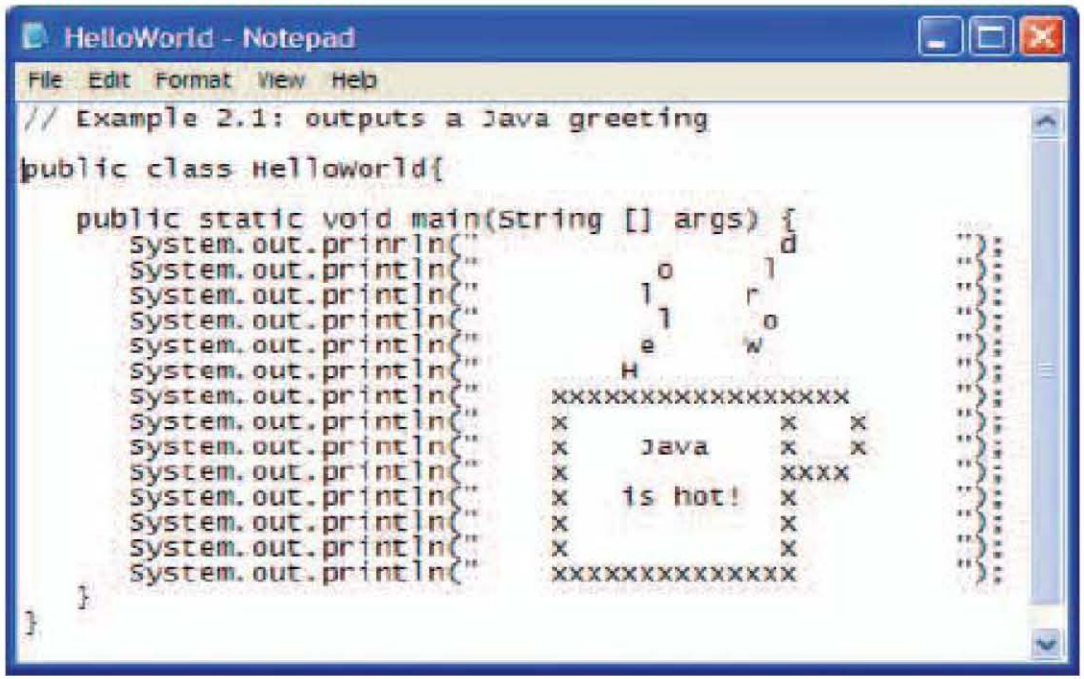


Figure 2-8: Program with a compile-time error on line 6

Compile-Time Errors (cont.)

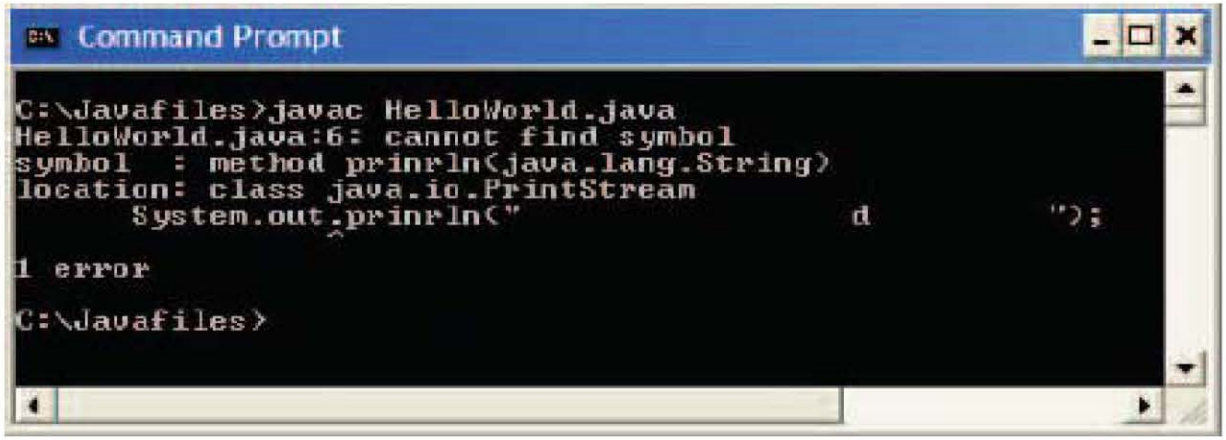


Figure 2-9: Compiler's error message

Readability

- The main factor affecting readability is layout.
  - Spacing
  - Indentation
- The compiler ignores layout.
- It is important for your code to be readable by others.
  - Your programs may be maintained by others.

Temperature Conversion

Figure 2-10: User interface for the temperature conversion program

Temperature Conversion: The Source Code

```
import java.util.Scanner;

public class Convert{

   public static void main(String [] args){
      Scanner reader = new Scanner(System.in);
      double fahrenheit;
      double celsius;

      System.out.print("Enter degrees Fahrenheit: ");
      fahrenheit = reader.nextDouble();

      celsius = (fahrenheit - 32.0) * 5.0 / 9.0;

     System.out.print("The equivalent in Celsius is ");
     System.out.println(celsius);
   }
}
```

Temperature Conversion: The Explanation

- First line of code is an **import statement**.
  - Tells compiler where to find a class that will be used during the program
- **`Scanner` object:** Provides functionality for reading input entered at the keyboard
  - Scanner object **instantiated** by the code:
    - `Scanner reader = new Scanner(System.in);`

Temperature Conversion: The Explanation (cont.)

- In general, instantiation takes the form:
  - `SomeClass someObject = new SomeClass(some parameters);`
- A numeric **variable** names a location in RAM in which a number can be stored.
  - e.g., `double celsius;`
  - A variable's **value** may change during a program.
  - A variable's **type** defines what kind of values it can have.

Temperature Conversion: The Explanation (cont.)

- **Assignment operator:** The symbol =
  - Used to assign a value to a variable
  - Examples:
    - `fahrenheit = reader.nextDouble();`
    - `celsius = (fahrenheit - 32.0) * 5.0/9.0;`
- **Assignment statements:** Statements using the assignment operator

Temperature Conversion: The Explanation (cont.)

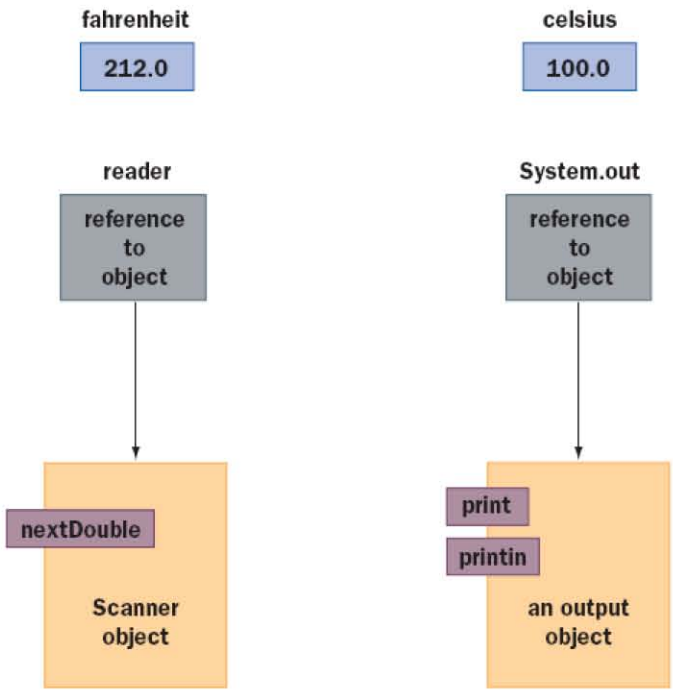
- Mathematical operators:
  - `+`: Addition
  - `-`: Subtraction
  - `*`: Multiplication
  - `/`: Division
- `System.out.println()` can be used to print values of variables.
  - `System.out.println(fahrenheit);`

Temperature Conversion: Variables and Objects

- Variables `fahrenheit` and `celsius` each hold a single floating-point number.
- Variables `reader` and `System.out` hold **references** to objects.
  - References used to send messages to the objects

Temperature Conversion: Variables and Objects (cont.)

Figure 2-11: Variables and objects used in the conversion program



Graphics and GUIs: Windows and Panels

- Standalone GUI applications run in windows.
  - Containers for graphical components to be displayed to the user
- Windows have numerous properties.
  - Width and height
  - Title bar
  - Ability to be dragged or resized by the user

Graphics and GUIs: Windows and Panels (cont.)

```
import javax.swing.*;      // Access JFrame

public class GUIWindow{

   public static void main(String[] args){
      JFrame theGUI = new JFrame();
      theGUI.setTitle("First GUI Program");
      theGUI.setSize(300, 200);
      theGUI.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      theGUI.setVisible(true);
   }
}
```

Example 2-3: Empty frame

Graphics and GUIs: Windows and Panels (cont.)

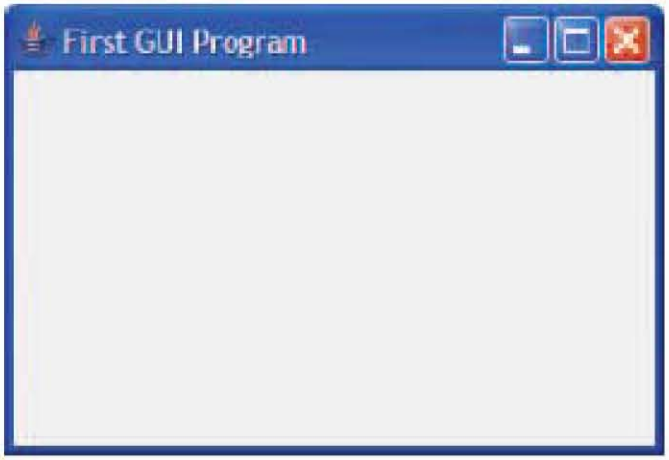


Figure 2-12: GUI program with an empty window

Graphics and GUIs: Windows and Panels (cont.)

JFRAME METHOD	WHAT IT DOES
`Container getContentPane()`	Returns the frame's container to which components can be added
`void setResizable(boolean b)`	If `b` is `false`, the user cannot resize the window; if `b` is `true`, the user can resize the window. The default is that the window is resizable.
`void setDefaultCloseIndicator(int i)`	Sets the operation to be performed when the user closes the frame
`void setSize(int width, int height)`	Sets the size of the frame to the width and height in pixels
`void setTitle(String title)`	Displays the title in the frame's title bar
`void setVisible(boolean b)`	Displays the frame if `b` is `true` or hides it if `b` is `false`

Table 2-1: Some commonly used `JFrame` methods

Panels and Colors

- **Panel:** A flat, rectangular area suitable for displaying other objects
  - Geometric shapes and images
  - `JPanel` class in `javax.swing` package
- **`Color` class:** Can be used to create any RGB color value
  - `Color aColor = new Color(redValue, greenValue, blueValue)`

Panels and Colors (cont.)

COLOR CONSTANT	RGB VALUE
Color.red	new Color(255, 0, 0)
Color.green	new Color(0, 255, 0)
Color.blue	new Color(0, 0, 255)
Color.yellow	new Color(255, 255, 0)
Color.cyan	new Color(0, 255, 255)
Color.magenta	new Color(255, 0, 255)
Color.orange	new Color(255, 200, 0)
Color.pink	new Color(255, 175, 175)
Color.black	new Color(0, 0, 0)
Color.white	new Color(255, 255, 255)
Color.gray	new Color(128, 128, 128)
Color.lightGray	new Color(192, 192, 192)
Color.darkGray	new Color(64, 64, 64)

Table 2-2: Some `Color` constants

Panels and Colors (cont.)

```
import javax.swing.*;     // For JFrame and JPanel
import java.awt.*;        // For Color and Container

public class GUIWindow{

   public static void main(String[] args){
      JFrame theGUI = new JFrame();
      theGUI.setTitle("Second GUI Program");
      theGUI.setSize(300, 200);
      theGUI.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      JPanel panel = new JPanel();
      panel.setBackground(Color.pink);
      Container pane = theGUI.getContentPane();
      pane.add(panel);
      theGUI.setVisible(true);
   }
}
```

Example 2-4: Frame with an empty, pink panel

Layout Managers and Multiple Panels

- Every container object (frame or panel) uses a **layout manager** object to organize and lay out contained graphical components.
  - Default layout manager for frames is an instance of the class `BorderLayout`
    - Can arrange up to five graphical objects in a container
      - NORTH, SOUTH, EAST, WEST, and CENTER
  - `GridLayout` **class:** Divides a container into rows and columns

Layout Managers and Multiple Panels (cont.)

Example 2-6: Frame with a 2-by-2 grid of colored panels

```
import javax.swing.*;     // For JFrame and JPanel
import java.awt.*;        // For Color, Container, and GridLayout

public class GUIWindow{

   public static void main(String[] args){
      JFrame theGUI = new JFrame();
      theGUI.setTitle("Fourth GUI Program");
      theGUI.setSize(300, 200);
      theGUI.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      JPanel panel1 = new JPanel();
      panel1.setBackground(Color.white);
      JPanel panel2 = new JPanel();
      panel2.setBackground(Color.black);
      JPanel panel3 = new JPanel();
      panel3.setBackground(Color.gray);
      JPanel panel4 = new JPanel();
      panel4.setBackground(Color.white);
      Container pane = theGUI.getContentPane();
      pane.setLayout(new GridLayout(2, 2));
      pane.add(panel1);
      pane.add(panel2);
      pane.add(panel3);
      pane.add(panel4);
      theGUI.setVisible(true);
   }
}
```

Layout Managers and Multiple Panels (cont.)

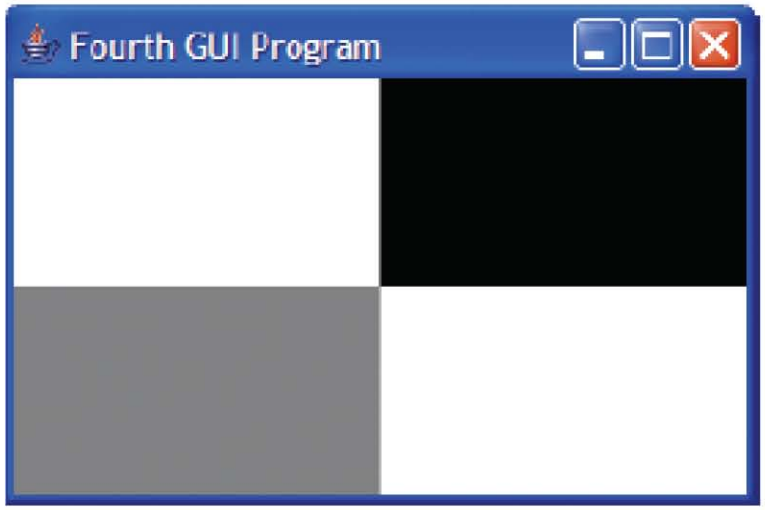


Figure 2-14: 2-by-2 grid layout with four panels

Summary

- Java is the fastest growing programming language in the world. It is secure, robust, and portable.
- The Java compiler translates Java into a pseudomachine language called Java byte code. Byte code can be run on any computer that has a Java virtual machine installed. The Java virtual machine (JVM) is a program that behaves like a computer—an interpreter.

Summary (cont.)

- Java programs include variables, arithmetic expressions, statements, objects, messages, and methods.
- Three basic steps in the coding process are editing, compiling, and running a program using a Java development environment. Programmers should pay attention to a program's format to ensure readability.

Summary (cont.)

- Java programs accomplish many tasks by sending messages to objects. Examples are sending text to the terminal window for output and receiving input data from the keyboard.
- There are several user interface styles, among them terminal based and graphical based.