Chapter 3 - Introduction to Java Applets

Outline
3.1 Introduction
3.2 Sample Applets from the Java 2 Software Development Kit
3.3 Simple Java Applet: Drawing a String
3.4 Drawing Strings and Lines
3.3 Simple Java Applet: Drawing a String

• Now, create applets of our own
  – Take a while before we can write applets like in the demos
  – Cover many of same techniques

• Upcoming program
  – Create an applet to display
  "Welcome to Java!!"
  – Show applet and HTML file, then discuss them line by line
public class WelcomeApplet extends Applet {

    public void init() {
    }

    public void paint(Graphics g) {
        g.drawString("Welcome to Java Programming!",
                    25, 25);
    }
}

extends allows us to inherit the capabilities of class Applet.

Method paint is guaranteed to be called in all applets. Its first line must be defined as above.
3.3 Simple Java Applet: Drawing a String

- Import predefined classes grouped into packages
  - When you create applets, import Applet class (in the package java.applet)
  - import the Graphics class (package java.awt) to draw graphics
    - Can draw lines, rectangles ovals, strings of characters
  - import specifies directory structure

```java
import java.awt.*;     // import package with class Graphics
import javax.applet.*;  // import class Applet
```
3.3 Simple Java Applet: Drawing a String

- Applets have at least one class declaration (like applications)
  - Rarely create classes from scratch
  - Use pieces of existing classes
  - Inheritance - create new classes from old ones (will cover soon)

```java
public class WelcomeApplet extends Applet {

- Begins class declaration for class WelcomeApplet
  - Keyword `class` then class name
- `extends` followed by class name
  - Indicates class to extend (Applet)
    - Applet: superclass (base class)
    - WelcomeApplet: subclass (derived class)
  - WelcomeApplet now has methods and data of Applet
```
3.3 Simple Java Applet: Drawing a String

```java
public class WelcomeApplet extends Applet {

  // Class Applet defined for us
  • Someone else defined "what it means to be an applet"
    – Applets require over 200 methods!
  • extends Applet
    – Inherit methods, do not have to declare them all
  • Do not need to know every detail of class Applet
```
3.3 Simple Java Applet: Drawing a String

```java
public class WelcomeApplet extends Applet {

    // Class WelcomeApplet is a blueprint
    // • appletviewer or browser creates an object of class WelcomeApplet
    //   • Keyword public required
    //   • File can only have one public class
    //   • public class name must be file name
```
3.3 Simple Java Applet: Drawing a String

```java
public void paint( Graphics g )
```

- Our class inherits method `paint` from `Applet`
  - By default, `paint` has empty body
  - Override (redefine) `paint` in our class
- Methods `init`, `start` and `paint`.
  - Guaranteed to be called automatically
  - Our applet gets "free" version of these by inheriting from `Applet`
    - Free versions have empty body (do nothing)
    - Every applet does not need all three methods
      - Override the ones you need
- Applet container “draws itself” by calling method `paint`
3.3 Simple Java Applet: Drawing a String

```java
public void paint( Graphics g )
```

- Method `paint`
  - Draws graphics on screen
  - `void` indicates `paint` returns nothing when finishes task
  - Parenthesis define parameter list - where methods receive data to perform tasks
    - Normally, data passed by programmer, as in `JOptionPane.showMessageDialog`
  - `paint` gets parameters automatically
    - `Graphics` object used by `paint`
  - Mimic `paint`'s first line
3.3 Simple Java Applet: Drawing a String

```java
  g.drawString("Welcome to Java Programming!", 25, 25);
```

- **Body of paint**
  - Method `drawString` (of class `Graphics`)
  - Called using `Graphics` object `g` and dot (.)
  - Method name, then parenthesis with arguments
    - First argument: `String` to draw
    - Second: x coordinate (in pixels) location
    - Third: y coordinate (in pixels) location

- **Java coordinate system**
  - Measured in pixels (picture elements)
  - Upper left is (0,0)
3.3 Simple Java Applet: Drawing a String

• Running the applet
  – Compile
    • As you would a regular file in JCreator
    – or
    • `javac WelcomeApplet.java` (from the command line)
    • If no errors, bytecodes stored in `WelcomeApplet.class`
  – Create an HTML file
    • Loads the applet into `appletviewer` or a browser
    • Ends in `.htm` or `.html`
  – To execute an applet
    • Create an HTML file indicating which applet the browser (or `appletviewer`) should load and execute
3.3 Simple Java Applet: Drawing a String

Simple HTML file (WelcomeApplet.html)
- Usually in same directory as .class file
- Remember, .class file created after compilation

HTML codes (tags)
- Usually come in pairs
- Begin with < and end with >

Lines 1 and 4 - begin and end the HTML tags
Line 2 - begins <applet> tag
- Specifies code to use for applet
- Specifies width and height of display area in pixels
Line 3 - ends <applet> tag

```html
<html>
<applet code = "WelcomeApplet.class" width = "300" height = "45">
</applet>
</html>
```
3.3 Simple Java Applet: Drawing a String

```html
<html>
<applet code = "WelcomeApplet.class" width = "300" height = "45">
</applet>
</html>
```

- `appletviewer` only understands `<applet>` tags
  - Ignores everything else
  - Minimal browser
- Executing the applet
  - `appletviewer WelcomeApplet.html`
  - Perform in directory containing `.class` file
3.3 Simple Java Applet: Drawing a String

- Running the applet in a Web browser
3.4 Drawing Strings and Lines

• More applets
  – First example
    • Display two lines of text
    • Use `drawString` to simulate a new line with two `drawString` statements
  – Second example
    • Method `g.drawLine(x1, y1, x2, y2)`
      – Draws a line from \((x1, y1)\) to \((x2, y2)\)
      – Remember that \((0, 0)\) is upper left
    • Use `drawLine` to draw a line beneath and above a string
import java.awt.*;
import java.applet.*;

public class WelcomeApplet2 extends Applet {

    public void init() {
    }

    public void paint(Graphics g) {
        g.drawString( "Welcome to", 25, 25 );
        g.drawString( "Java Programming!", 25, 40 );
    }
}

The two `drawString` statements simulate a newline. In fact, the concept of lines of text does not exist when drawing strings.
<html>
<applet code = "WelcomeApplet2.class" width = "300" height = "60">
</applet>
</html>
import java.awt.*;
import java.applet.*;

public class WelcomeApplet3 extends Applet {

    public void init() {
    }

    public void paint(Graphics g) {
        g.drawString("Welcome to Java Programming!", 25, 25);
        g.drawLine(15, 10, 210, 10);
        g.drawLine(15, 30, 210, 30);
    }
}

2. Class WelcomeLines (extends Applet)
3. paint
3.1 drawLine
3.2 drawLine

Draw horizontal lines with drawLine (endpoints have same y coordinate).

Program Output
<html>
<applet code = "WelcomeApplet3.class" width = "300" height = "40">
</applet>
</html>
3.4 Drawing Strings and Lines

- Method `drawLine` of class `Graphics`
  - Takes as arguments `Graphics` object and line’s end points
  - X and y coordinate of first endpoint
  - X and y coordinate of second endpoint