

Introduction to Java Applet Programs

Section 1.3 Intro. to Java Applet Programs:
a Greeter Applet

Section 3.7 Graphical/Internet Java:
Applet: An Einstein Calculator

Sections 4.5 & 5.4 Graphical/Internet Java:
Old MacDonald ... Applet

1

Swing Containers

<http://www.cs.huji.ac.il/support/docs/java/jdk1.3/demo/jfc/SwingSet2/SwingSet2Plugin.html>

Top-Level: _____

- They create an area of the screen called a _____ or _____ where the graphical output is to take place.

```
import javax.swing.*;

class FrameExample {

    public static void main(String [] args) {

        JFrame frame = new JFrame("A JFrame");
        frame.show();
    }
}
```

2

Panes:

- Each of these frames contains layers of containers called _____ (or _____). The most important of these is the _____.
- The content pane is what is used to group and position the widgets that actually do the _____ (i.e., drawing) on the screen, or perform some other action.

3

- The content pane can be accessed via the _____ method. For example:

```
frame.getContentPane().add(JButton("Press"));
```

JButton creates a button.

- Constructors can create an unlabeled button, one with text on it, or an icon on it, or both.
- **addActionListener()** is used to add a *listener* to the button that listens for events (e.g., mouse press)

4

Action Listeners:

- They are used for certain widgets that _____ e.g., buttons.
- `java.awt.event.ActionListener` is an interface that demands only one method be implemented:

```
public void  
    actionPerformed(ActionEvent event);
```
- The event passed to this method can be safely ignored.
- For a button, this method will be called when the button is pressed.

5

Layouts:

- How the components are organized on the content pane _ size and how they are arranged _ is determined by the container's _____.
- Containers inherit a `setLayout()` method from the `Container` class for setting the layout. But several of them have a default layout.
- _____ provides several standard ones; each of them is a class in the `java.awt` package.

6

Layout:

- The simplest AWT layout manager. It's the default layout manager for `JPanel`.
- `FlowLayout` lays out components side-by-side starting near the top of the content pane, working downward as each row is filled.
- Constructor: Just the default one `FlowLayout()`

7

Layout:

- `GridLayout` lays out components in a grid.
- Constructor:

```
GridLayout(int rows, int cols)
```

 - Parameters are the number of rows and columns in the grid.
 - Either (but not both) may be 0 to represent "infinity."
 - Components are added to the content pane row-by-row in the specified grid pattern.

8

Layout:

- It's the *default layout* for **JApplet** and **JFrame**.
- **BorderLayout** divides the content pane into sections:
 - north, south, east, west, center
 - Center is the largest; extra space goes here
 - Some sections may be ignored.
- Constructor: Default: **BorderLayout()**
- Add a component to the a specific region of the content pane by adding a second argument:

**BorderLayout.NORTH, BorderLayout.SOUTH,
BorderLayout.WEST, BorderLayout.EAST,
BorderLayout.CENTER**

9

Applets vs. Applications

- Applications are _____ programs
 - executed with Java interpreter
- An applet is a small program that:
 - can be placed on a _____
 - can be executed by a _____ or with Java's _____
 - gives web pages "dynamic content"

See Greeter2.java and
Greeter2.html in §1.3

10

```
import javax.swing.*; // JApplet, JLabel
import java.util.*;   // Date
public class Greeter extends JApplet {

    public void init() {
        Date currentDate = new Date();
        String today = currentDate.toString();

        JLabel greeting = new JLabel(
            "Today, " + today +
            ", we begin our study of applets!" );
        getContentPane().add(greeting);
    }
}
```

11

Notes about Applets:

- Syntax
 - public class** **ClassName** **extends** **JApplet**
 - Extends **JApplet** (or **Applet**)
 - Must be public
- Body
 - There is no **main()** method in an applet
 - **JApplet** class provides other methods instead
 - First method executed is the **init()** method that is not declared **static**

12

- Statements

- Declaration statements for `Date` are no different than in applications.
- Label declaration

```
JLabel greeting = new JLabel ( ... )
```

for freestanding strings of text (not part of a button or menu)

- Content pane: portion of window where label is added:

```
getContentPane().add(greeting)
```

13

Example: Building a Temperature-Conversion Applet

(similar to Einstein Applet in §3.7)

Problem Scenario:

Write a program to read a temperature in Celsius, compute and display the equivalent Fahrenheit temperature. However, instead of the text-based solution or GUI-application from before, use an *applet*.

14

```
/* TemperatureApplet.java converts Celsius
 * temperatures to Fahrenheit.
 * Author: L. Nyhoff
 * Date: Nov. 29, 2002
 */

import java.awt.*; // BorderLayout
import javax.swing.*; // JApplet, JTextArea, JOptionPane

public class TemperatureApplet extends JApplet {

    JTextArea introArea,
               inputArea,
               outputArea;

    public void init() {

        introArea = new JTextArea("Celsius-to-Fahrenheit Converter");
        inputArea = new JTextArea("\n\nCelsius\nTemperature:\n");
        outputArea = new
            JTextArea("\n\nFahrenheit\nTemperature:\n");

        getContentPane().add(introArea, BorderLayout.NORTH);
        getContentPane().add(inputArea, BorderLayout.CENTER);
        getContentPane().add(outputArea, BorderLayout.EAST);
    }
}
```

15

```
String TITLE = "Celsius-to-Fahrenheit Conversion";
String celsiusString = JOptionPane.showInputDialog(
    null,
    "Please enter the temperature in Celsius: ",
    TITLE,
    JOptionPane.QUESTION_MESSAGE);

inputArea.append(celsiusString);

double celsius = Double.parseDouble(celsiusString);
double fahrenheit = ((9.0/5.0)*celsius) + 32;

outputArea.append(Double.toString(fahrenheit));

}
}
```

16

Applet can be loaded as part of a web page and executed using [Java's appletviewer](#) (or by a browser):

```
<!-- TemperatureApplet.html -->

<title>TemperatureApplet</title>
<hr>
<applet CODEBASE="" CODE="TemperatureApplet.class"
        WIDTH = 300 HEIGHT = 200>
</applet>
<hr>
<a href="TemperatureApplet.java">The source.</a>
```

To run the applet:

`appletviewer TemperatureApplet.html`

17

"Just Messing Around..."

```
/* TemperatureApplet1.java converts Celsius
 * temperatures to Fahrenheit.
 * Author: L. Nyhoff
 * Date: Nov. 29, 2002
 */
import java.awt.*; // BorderLayout
import javax.swing.*; // JApplet, JLabel, JTextArea, JOptionPane

public class TemperatureApplet1 extends JApplet
{
    JLabel thermLabel,
        coldLabel,
        hotLabel,
        comfortLabel;
    JTextArea introArea,
        inputArea,
        outputArea;

    public void init() {

        thermLabel = new JLabel(new ImageIcon("thermometer.jpg"));
        hotLabel = new JLabel(new ImageIcon("tempHot.gif"));
        coldLabel = new JLabel(new ImageIcon("tempCold.gif"));
        comfortLabel = new JLabel(new ImageIcon("tempNice.gif"));
    }
}
```

18

```
introArea = new JTextArea("Celsius-to-Fahrenheit Converter");
Font boldFont = new Font("Arial", Font.BOLD, 16);
introArea.setFont(boldFont);
inputArea = new JTextArea("\n\nCelsius\nTemperature:\n");
outputArea = new JTextArea("\n\nFahrenheit\nTemperature:\n");

getContentPane().add(introArea, BorderLayout.NORTH);
getContentPane().add(thermLabel, BorderLayout.WEST);
getContentPane().add(inputArea, BorderLayout.CENTER);
getContentPane().add(outputArea, BorderLayout.EAST);

String TITLE = "Celsius-to-Fahrenheit Conversion";
String celsiusString = JOptionPane.showInputDialog(
    null,
    "Please enter the temperature in Celsius: ",
    TITLE,
    JOptionPane.QUESTION_MESSAGE);
inputArea.append(celsiusString);

double celsius = Double.parseDouble(celsiusString);
double fahrenheit = (9.0/5.0)*celsius + 32;
outputArea.append(Double.toString(fahrenheit));

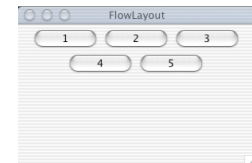
if (fahrenheit <= 32)
    getContentPane().add(coldLabel, BorderLayout.SOUTH);
else if (fahrenheit >= 90)
    getContentPane().add(hotLabel, BorderLayout.SOUTH);
else
    getContentPane().add(comfortLabel, BorderLayout.SOUTH);
}
```

19

```
import javax.swing.*;
import java.awt.*;

public class FlowLayoutExample {
    public static void main(String[] args) {
        JFrame f = new JFrame("FlowLayout");

        f.getContentPane().setLayout(new FlowLayout());
        for (int i = 1; i <= 5; i++) {
            JButton b = new JButton(" " + i);
            f.getContentPane().add(b);
        }
        f.setVisible(true);
    }
}
```



20

```
import javax.swing.*;
import java.awt.*;

public class GridLayoutExample {
    public static void main(String[] args) {
        JFrame f = new JFrame("GridLayout");

        f.getContentPane().setLayout(new GridLayout(2, 3));
        for (int i = 1; i <= 6; i++) {
            JButton b = new JButton("" + i);
            f.getContentPane().add(b);
        }
        f.setVisible(true);
    }
}
```



21

```
import javax.swing.*;
import java.awt.*;
import ann.easyio.*;

public class BorderLayoutExample {
    public static void main(String[] args) {
        Keyboard k = new Keyboard();

        JFrame f = new JFrame("BorderLayout");
        f.getContentPane().setLayout(new BorderLayout());

        JButton b = new JButton("NORTH");
        f.getContentPane().add(b, BorderLayout.NORTH);
        f.show();
        k.getChar();

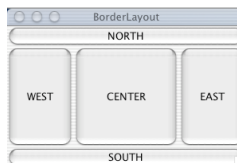
        b = new JButton("SOUTH");
        f.getContentPane().add(b, BorderLayout.SOUTH);
        f.show();
        k.getChar();
    }
}
```

22

```
b = new JButton("CENTER");
f.getContentPane().add(b, BorderLayout.CENTER);
f.show();
k.getChar();

b = new JButton("WEST");
f.getContentPane().add(b, BorderLayout.WEST);
f.show();
k.getChar();

b = new JButton("EAST");
f.getContentPane().add(b, BorderLayout.EAST);
f.show();
}
}
```



23

Graphics on GUIs & Applets (§ 6.5)

We must, of course, first create a frame
(window) on which to paint (draw):

- _____ for applets
- _____ (an extension of _____) from the `ann.gui` package is convenient for GUI apps, because it handles window-closing events

24

The recommended practice is to design a "canvas" on which to paint that is a subclass of _____,

```
class Canvas extends JPanel {  
    ... }
```

and then make this the content pane of the basic frame:

```
_____ panel = new Canvas(...);  
_____;
```

25

Our canvas typically has two attributes — height and width — and the constructor sets their values and usually the background color (to something other than the default gray)

```
class Canvas extends JPanel {  
    public Canvas(int width, int height) {  
        setSize(width, height);  
        myHeight = height; myWidth = width;  
        setBackground(Color.white);  
    }  
  
    // at least one other method  
    ...  
}
```

26

One other method that must be added is:

```
public void _____{  
    // statements to paint the component  
}
```

The object **pen** of type **Graphics** (from AWT) is what does the actual painting with methods like (see p. 309):

```
drawArc(), drawLine(), drawOval(),  
drawRect(), drawString(),  
fillArc(), fillOval(), fillRect(),  
translate()
```

27

Example: Dutch Flag Applet (§ 6.5)

Problem: Build an applet to paint an image of the Dutch flag:



28

```

/** DutchFlag2.java is an applet that draws a Dutch flag
 * Output: A graphic representing the flag of the Netherlands.
 */

import javax.swing.*;      // JApplet, JPanel
import java.awt.*;         // Graphics, Color, ...

public class DutchFlag2 extends JApplet {

    private static final int FLAG_WIDTH  = 350;
    private static final int FLAG_HEIGHT = 200;
    private static final int TITLE_ROWS = 30;

    public DutchFlag2() {

        JPanel flagPanel = new FlagOfHolland(FLAG_WIDTH,
                                              FLAG_HEIGHT);
        setContentPane(flagPanel);
    }
}

```

29

```

public void init() {

    DutchFlag2 myPicture = new DutchFlag2();
    myPicture.setSize(FLAG_WIDTH, FLAG_HEIGHT + TITLE_ROWS);
    myPicture.setVisible(true);
}

}

class FlagOfHolland extends JPanel {

    public FlagOfHolland(int width, int height)
    {
        setSize(width, height);
        myHeight = height;
        myWidth = width;

        // draw white field as background
        setBackground(Color.white);
    }
}

```

30

```

public void paintComponent(Graphics pen) {

    super.paintComponent(pen);

    myHeight = getHeight();
    myWidth = getWidth();

    int stripeHeight = myHeight / 3;

    // draw red stripe
    pen.setColor(Color.red);
    pen.fillRect(0, 0, myWidth, stripeHeight);

    // draw blue stripe
    pen.setColor(Color.blue);
    pen.fillRect(0, stripeHeight * 2, myWidth, myHeight);
}

int myHeight, myWidth;
}

```

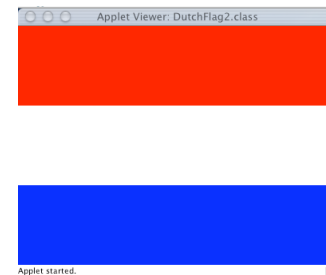
31

```

<!-- DutchFlag2.html -->

<HTML>
<HEAD>
</HEAD>
<BODY>
<APPLET CODE="DutchFlag2.class" WIDTH=400 HEIGHT=300 >
</APPLET>
</BODY>
</HTML>

```



32