Graphics & Applets

CSC 1051 – Data Structures and Algorithms I
Dr. Mary-Angela Papalaskari
Department of Computing Sciences
Villanova University

Course website:
www.csc.villanova.edu/~map/1051/

Some slides in this presentation are adapted from the slides accompanying Java Software Solutions by Lewis & Loftus
Applets

• A Java *application* is a stand-alone program with a *main* method (like the ones we've seen so far)

• A Java *applet* is a program that is intended to be transported over the Web and executed using a web browser

• An applet also can be executed using the appletviewer tool of the Java SDK

• An applet doesn't have a *main* method

• Instead, there are several special methods that serve specific purposes
Java Translation

Java source code

Java compiler

Java bytecode

Bytecode interpreter

Bytecode compiler

Machine code

CSC 1051 M.A. Papalaskari, Villanova University
Applets

- We create an applet by extending the `JApplet` class

- The `JApplet` class is part of the `javax.swing` package – so we need to import it to use it.

- See `Einstein.java`
//********************************************************************
//  Einstein.java       Author: Lewis/Loftus
//
//  Demonstrates a basic applet.
//********************************************************************

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

public class Einstein extends JApplet
{
    //---------------------------------------------------------------------------------
    //  Draws a quotation by Albert Einstein among some shapes.
    //---------------------------------------------------------------------------------
    public void paint (Graphics page)
    {
        page.drawRect (50, 50, 40, 40);  // square
        page.drawRect (60, 80, 225, 30);  // rectangle
        page.drawOval (75, 65, 20, 20);  // circle
        page.drawLine (35, 60, 100, 120);  // line

        page.drawString ("Out of clutter, find simplicity.", 110, 70);
        page.drawString ("-- Albert Einstein", 130, 100);
    }
}
import javax.swing.JApplet;
import java.awt.*;

public class Einstein extends JApplet
{
    //---
    // Draws a quotation by Albert Einstein among some shapes.
    //---
    public void paint (Graphics page)
    {
        page.drawRect (50, 50, 40, 40); // square
        page.drawRect (60, 80, 225, 30); // rectangle
        page.drawOval (75, 65, 20, 20); // circle
        page.drawLine (35, 60, 100, 120); // line

        page.drawString ("Out of clutter, find simplicity.", 110, 70);
        page.drawString ("-- Albert Einstein", 130, 100);
    }
}
Drawing a Line

```javascript
page.drawLine (10, 20, 150, 45);
page.drawLine (150, 45, 10, 20);
```

Start:
```
x  y
```
End:
```
x  y
```

CSC 1051 M.A. Papalaskari, Villanova University
Drawing a Rectangle

![Diagram of a rectangle with coordinates and code example]

```javascript
page.drawRect (50, 20, 100, 40);
```

CSC 1051 M.A. Papalaskari, Villanova University
Drawing an Oval

```java
page.drawOval (175, 20, 50, 80);
```

bounding rectangle

CSC 1051 M.A. Papalaskari, Villanova University
Drawing an Arc

- An arc is defined by an oval, a start angle, and an arc angle:
Filled vs unfilled shapes

• Instead of using `drawRect()`, `drawOval()` etc, we can use `fillRect()`, `fillOval()` etc

• We can set the color using `setColor()`

• See `Snowman.java`

• See also `Snowman applet on a webpage`
import javax.swing.JApplet;
import java.awt.*;

public class Snowman extends JApplet
{
  //-------------------------------
  //  Draws a snowman.
  //-------------------------------
  public  void paint (Graphics page)
  {
    final int MID = 150;
    final int TOP = 50;

    setBackground (Color.cyan);
    page.setColor (Color.blue);
    page.fillRect (0, 175, 300, 50);  // ground

    page.setColor (Color.yellow);
    page.fillOval (-40, -40, 80, 80);  // sun

    continued
```java
page.setColor(Color.white);
page.fillOval(MID-20, TOP, 40, 40); // head
page.fillOval(MID-35, TOP+35, 70, 50); // upper torso
page.fillOval(MID-50, TOP+80, 100, 60); // lower torso

page.setColor(Color.black);
page.fillOval(MID-10, TOP+10, 5, 5); // left eye
page.fillOval(MID+5, TOP+10, 5, 5); // right eye

page.drawArc(MID-10, TOP+20, 20, 10, 190, 160); // smile

page.drawLine(MID-25, TOP+60, MID-50, TOP+40); // left arm
page.drawLine(MID+25, TOP+60, MID+55, TOP+60); // right arm

page.drawLine(MID-20, TOP+5, MID+20, TOP+5); // brim of hat
page.fillRect(MID-15, TOP-20, 30, 25); // top of hat
```
continued

```java
page.setColor(Color.white);
page.fillOval(MID-20, TOP, 40, 40); // head
page.fillOval(MID-35, TOP+35, 70, 50); // upper torso
page.fillOval(MID-50, TOP+80, 100, 60); // lower torso
page.setColor(Color.black);
page.fillOval(MID-10, TOP+10, 5, 5); // left eye
page.fillOval(MID+5, TOP+10, 5, 5); // right eye
page.drawArc(MID-10, TOP+20, 20, 10, 190, 160); // smile
page.drawLine(MID-25, TOP+60, MID-50, TOP+40); // left arm
page.drawLine(MID+25, TOP+60, MID+55, TOP+60); // right arm
page.drawLine(MID-20, TOP+5, MID+20, TOP+5); // brim of hat
page.fillRect(MID-15, TOP-20, 30, 25); // top of hat
```

Applet started.
The HTML applet Tag

- An applet is embedded into an HTML file using a tag that references the bytecode file of the applet
- The bytecode version of the program is transported across the web and executed by a Java interpreter that is part of the browser.

```html
<html>
<head>
  <title>The Einstein Applet</title>
</head>
<body>
  <applet code="Einstein.class" width=350 height=175>
  </applet>
</body>
</html>
```
What’s a picture?

- programs represent pictures as grids of picture elements or *pixels*
## Pixel encodings

<table>
<thead>
<tr>
<th>Type</th>
<th>Representation</th>
<th>Colors</th>
<th>Bits/Color</th>
<th>Total Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitmap</td>
<td>Bitmap</td>
<td>3 colors: red, green, blue</td>
<td>8 bits/color</td>
<td>24 bits</td>
</tr>
<tr>
<td>Grayscale</td>
<td>Grayscale</td>
<td></td>
<td>8 bits</td>
<td></td>
</tr>
<tr>
<td>RGB Color</td>
<td>RGB Color</td>
<td></td>
<td>3 colors: red, green, blue</td>
<td>8 bits/color</td>
</tr>
</tbody>
</table>
Additive/Subtractive Color

We choose 3 primary colors that can be combined to produce almost all visible colors:

Additive primaries
- combining light
  Red  Green  Blue

Subtractive primaries
- combining ink, thus subtracting light
  Cyan  Yellow  Magenta
Encoding RGB

- Each component color (red, green, and blue) is encoded as a single byte
- Colors go from (0,0.0) to (255,255,255)
  - (0,0,0) = black
  - (255,255,255) = white
  - If all three components are the same, the color is in greyscale
    - eg: (50,50,50)
The Java Color Class

- A color in a Java program is represented as an object created from the `Color` class

- The `Color` class also contains several predefined colors, including the following:

<table>
<thead>
<tr>
<th>Object</th>
<th>RGB Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color.black</td>
<td>0, 0, 0</td>
</tr>
<tr>
<td>Color.blue</td>
<td>0, 0, 255</td>
</tr>
<tr>
<td>Color.cyan</td>
<td>0, 255, 255</td>
</tr>
<tr>
<td>Color.orange</td>
<td>255, 200, 0</td>
</tr>
<tr>
<td>Color.white</td>
<td>255, 255, 255</td>
</tr>
<tr>
<td>Color.yellow</td>
<td>255, 255, 0</td>
</tr>
</tbody>
</table>
Creating a new color

• You can define a new color by giving its RGB components:

Example:

```java
Color salmon = new Color(255, 140, 128);
page.setColor(salmon);
```