Interactive Programs and Graphics in Java

Alark Joshi
Announcements

• Lab 1 is due today
• Questions/concerns?
SVN - Subversion

• Versioning and revision control system
  1. allows you to maintain a copy of your code on a server
  2. keeps track of all previous versions
  3. allows multiple developers to work on the same code
     – though, we won't use that this semester

• I can retrieve the code that was submitted before the deadline rather than the latest code

• Popular options for version control are CVS, SVN, Git
Submitting via SVN

• A hard drive crash the hour before a deadline is NOT an excuse for not submitting anything
• Submit a working version of your code every time you make changes

• Accessing SVN via Eclipse
Interactive Programs

• Programs generally need input on which to operate

• The `Scanner` class provides convenient methods for reading input values of various types

• A `Scanner` object can be set up to read input from various sources, including the user typing values on the keyboard

• Keyboard input is represented by the `System.in` object
Reading Input

• The following line creates a `Scanner` object that reads from the keyboard:

```
Scanner scan = new Scanner (System.in);
```

• The `new` operator creates the `Scanner` object

• Once created, the `Scanner` object can be used to invoke various input methods, such as:

```
answer = scan.nextLine();
```
Reading Input

- The `Scanner` class is part of the `java.util` class library, and must be imported into a program to be used.
- The `nextLine` method reads all of the input until the end of the line is found.
- See `Echo.java`.
- The details of object creation and class libraries are discussed further in Chapter 3.
import java.util.Scanner;

public class Echo {
    //---
    // Reads a character string from the user and prints it.
    //---
    public static void main (String[] args) {
        String message;
        Scanner scan = new Scanner (System.in);

        System.out.println ("Enter a line of text:");

        message = scan.nextLine();

        System.out.println ("You entered: \
" + message + \\
"\n");
    }
}
import java.util.Scanner;

public class Echo
{
    // -------------------------------
    // Reads a character string from the user and prints it.
    // -------------------------------
    public static void main (String[] args)
    {
        String message;
        Scanner scan = new Scanner (System.in);

        System.out.println ("Enter a line of text:");

        message = scan.nextLine();

        System.out.println ("You entered: \"" + message + "\"");
    }
}

Sample Run
Enter a line of text:
You want fries with that?
You entered: "You want fries with that?"
Input Tokens

• Unless specified otherwise, *white space* is used to separate the elements (called *tokens*) of the input

• White space includes space characters, tabs, new line characters
Input Tokens

• The `next` method of the `Scanner` class reads the next input token and returns it as a string

• Methods such as `nextInt` and `nextDouble` read data of particular types

• See `GasMileage.java`
import java.util.Scanner;

public class GasMileage{
    public static void main (String[] args) {
        int miles;
        double gallons, mpg;

        Scanner scan = new Scanner(System.in);
        continue
System.out.print ("Enter the number of miles: ");
miles = scan.nextInt();

System.out.print ("Enter the gallons of fuel used: ");
gallons = scan.nextDouble();

mpg = miles / gallons;

System.out.println ("Miles Per Gallon: " + mpg);
System.out.print("Enter the number of miles: ");
miles = scan.nextInt();

System.out.print("Enter the gallons of fuel used: ");
gallons = scan.nextDouble();

mpg = miles / gallons;

System.out.println("Miles Per Gallon: "+mpg);
Miles to Kilometers conversion

• Write a program to take input from the user in miles and display the corresponding kilometers
• 1 mile = 1.60934 kilometers
Introduction to Graphics

• A picture or drawing must be digitized for storage on a computer

• A picture is made up of *pixels* (picture elements), and each pixel is stored separately

• The number of pixels used to represent a picture is called the *picture resolution*

• The number of pixels that can be displayed by a monitor is called the *monitor resolution*
Representing Images

• A digitized picture with a small portion magnified:
Coordinate Systems

• Each pixel can be identified using a two-dimensional coordinate system

• When referring to a pixel in a Java program, we use a coordinate system with the origin in the top-left corner
Representing Color

• A black and white picture could be stored using one bit per pixel (0 = white and 1 = black)

• A colored picture requires more information; there are several techniques for representing colors

• Every color can be represented as a mixture of the three additive primary colors Red, Green, and Blue

• Each color is represented by three numbers between 0 and 255 that collectively are called an RGB value
The 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>
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
Applets

• The `paint` method is executed automatically whenever the applet’s contents are drawn

• The `paint` method accepts a parameter that is an object of the `Graphics` class
Graphics class

• A Graphics object defines a graphics context on which we can draw shapes and text

• The Graphics class has several methods for drawing shapes
  – drawRect()
  – drawOval()
  – drawLine() ...
Applets

• We create an applet by extending the JApplet class

• The JApplet class is part of the javax.swing package

• This makes use of inheritance, which is explored in more detail in Chapter 8

• See Einstein.java
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
    //-------
    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);
    }
}

The HTML applet Tag

• An applet is embedded into an HTML file (applet 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
Drawing Shapes

• Let's explore some of the methods of the Graphics class that draw shapes in more detail

• A shape can be filled or unfilled, depending on which method is invoked

• The method parameters specify coordinates and sizes

• Shapes with curves, like an oval, are usually drawn by specifying the shape’s bounding rectangle

• An arc can be thought of as a section of an oval
page.drawLine (10, 20, 150, 45);

or

page.drawLine (150, 45, 10, 20);
Drawing a Rectangle

page.drawRect (50, 20, 100, 40);
Drawing an Oval

page.drawOval (175, 20, 50, 80);
Drawing an Arc

• An arc is defined by an oval, a start angle, and an arc angle:
Drawing Shapes

- Every drawing surface has a background color
- Every graphics context has a current foreground color
- Both can be set explicitly
- See Snowman.java
/***********************
// Snowman.java       Author: Lewis/Loftus
//
// Demonstrates basic drawing methods and the use of color.
***********************

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
continued

    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
}
```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
```