Drawing in Java

Barb Ericson
Georgia Institute of Technology
Sept 2010

Learning Goals

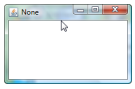
- Understand at a conceptual and practical level
  - Class methods
  - String objects and methods
  - How to use comments in a Java method
  - How to use the Java API
  - How to draw simple shapes in Java
  - How to use the Java 2D API for more complex drawing
  - To introduce inheritance and interface

How to Create a Picture Object

- You can create a picture object by asking the Picture class to create one using the *new* keyword
  > new Picture();
- You will probably want to be able to refer to the picture object again, so you should declare a variable to refer to it
  > Picture picture1 = new Picture();
- You can print out information about the picture
  > System.out.println(picture1);

How to Show a Picture

- You can show a picture using
  > picture1.show();
  Or
  > picture1.setVisible(true);



- If you want to create a picture from a file you will need to specify the full file name as a string.
  - A string is a sequence of characters
  - "C:/intro-prog-java/mediasources/beach.jpg"

Strings

- Java has a String class and you can create objects of this class
- A String object has a sequence of characters inside of a pair of double quotes
  "this is a string"
- You can create string objects using
  new String("the characters");
- You can also declare a variable that refers to a string object without using the new keyword (it is implied)
  > String message = "You won!";

String Methods

- There are many string methods
  >String test = "this is a test";
  > System.out.println(test.toUpperCase());
  THIS IS A TEST
  > System.out.println(test.length());
  14
  > System.out.println(test);
  this is a test
- Strings are immutable
  - They don't change once they have been created
    - Instead a new string is created and returned

A Fully Qualified File Name

- Has a path and base file name

  Examples: "C:/intro-prog-java/mediasources/beach.jpg"

  "/Users/intro-prog-java/mediasources/beach.jpg"

  – The path is everything up to the final path separator
    - "C:/intro-prog-java/mediasources/" – on windows
    - "/Users/intro-prog-java/mediasources/" – on macs
  – The base file name is the name of the file "beach" and the extension ".jpg"
    - The extension tells you the format of the data stored in the file
    - ".jpg" means it is a JPEG image

Special Characters in Strings

- Java characters are stored using Unicode
  – Which uses 16 bits per character
- The '\' character is used as a special character in Java strings
  – '\b' is backspace
  – '\t' is tab
  – '\n' is often used as a new line character
- In order to use it in fully qualified file names you will need to double it

  "c:\\intro-prog-java\\mediasources\\beach.jpg"

Picking a File

- There is a class method (also called a static method) on the FileChooser class that lets you pick a file and returns the fully qualified file name

  > FileChooser.pickAFile();

  This will display a file chooser – navigate to mediasources and select a file that ends in ".jpg"
- But, we didn't save the result of the method, let's try again

  > String filename = FileChooser.pickAFile();

  > System.out.println(filename);

  "C:\intro-prog-java\mediasources\beach.jpg"

Class (Static) Methods

- Class methods can be called using
  – *ClassName.methodName*()
- They do not have to be called on an object of the class
- You already have been using one class method
  – The main method
  – The main method must be a class method since no objects of the class exist when the Java Virtual Machine calls the method
- Class methods are often used for general functionality or for creating objects

The Media Path

- If you store your media (pictures, sounds, movie frames) in mediasources
  – You can set the media path which will remember the path you set
- You can pick the media directory using a file chooser.

  > FileChooser.pickMediaPath();
- You can use the stored media directory path with the base file name. It will return a fully qualified file name

  > FileChooser.getMediaPath("beach.jpg");

Variable Substitution

- You can pick a file, create a picture from the file, and show the resulting picture using
  – new Picture(FileChooser.pickAFile()).show();
- You can also name the result of each command (declare variables)
  – String filename = FileChooser.pickAFile();
  – Picture pictureObj = new Picture(filename);
  – pictureObj.show();

Primitive Variables

- When you declare a variable to be of type int, double, or boolean
  - This is called a primitive variable
  - It sets aside space to store a value of that type
    - int – 32, double – 64, boolean (maybe 1 bit or maybe more)
  - The value of the variable is stored in that space

    > int x = 3;

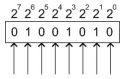
    This sets aside 32 bits and names the space x and sets the value in the space to 3 (using binary numbers in two's complement)

Binary Numbers

- A bit is a binary digit with a value of 0 or 1
  - A group of 8 bits is a byte
- Computer memory is allocated in bytes
- Numbers are stored using the binary number system
  - With digits of 0 or 1 and powers of 2
- Other number systems
  - Decimal- digits of 0 to 9 and powers of 10
  - Octal - digits of 0 to 7 and powers of 8
  - Hexadecimal – digits of 0 to 9 and A, B, C, D, E, F and powers of 16.

Converting from Binary to Decimal

- Multiply the digit value times the place value and add up the results to convert from binary to decimal
- The place values start with 2^0 on the right (which is 1) and increase to the left

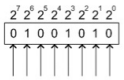


$2^7 = 128 * 0 = 0$
$2^6 = 64 * 1 = 64$
$2^5 = 32 * 0 = 0$
$2^4 = 16 * 0 = 0$
$2^3 = 8 * 1 = 8$
$2^2 = 4 * 0 = 0$
$2^1 = 2 * 1 = 2$
$2^0 = 1 * 0 = 0$
74

Converting from Decimal to Binary

- Subtraction Method

  Keep subtracting out largest power of two until nothing remains



74
-64 1×2^6
10
-8 1×2^3
2
-2 1×2^1
0

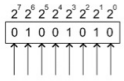
$result : 1001010$

$full byte\ result : 01001010$

Converting from Decimal to Binary

- Division Method

$$
\begin{array}{rl}
0 & R1 \\
2\overline{)1} & R0 \\
2\overline{)2} & R0 \\
2\overline{)4} & R1 \\
2\overline{)9} & R0 \\
2\overline{)18} & R1 \\
2\overline{)37} & R0 \\
2\overline{)74} &
\end{array}
$$



$result : 1001010$

$full byte\ result : 01001010$

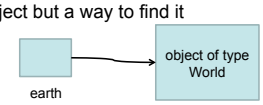
Read result from top to bottom.

Object Variables

- When you declare a variable with the type as a class name

  > World earth = new World();

  It sets aside space for a *reference* to an object
- An object reference is a way to find the memory that is assigned to that object
  - It is like a package tracking number, in that it isn't the object but a way to find it



Drawing on a Picture

- What if we want to draw something on a picture?
- How about drawing a grid of lines on top of a picture?
  - We could use a Turtle object to draw the lines
  - Create the Turtle on a Picture object
    - Picture p = new Picture(FileChooser.pickAFile());
    - Turtle turtle1 = new Turtle(p);
  - Using the methods:
    - moveTo(x,y), penUp(), penDown(),
    - turnRight(), turnLeft(), turn(amount)

Exploring a Picture

- You can create a picture object
  > String beachFile = FileChooser.getMediaPath ("beach.jpg");
  > Picture beachPict = new Picture(beachFile);
- You can explore the picture
  - This makes a copy of the current picture and then displays it
  > beachPict.explore();
- You can get information about the picture
  > int width = beachPict.getWidth();
  > int height = beachPict.getHeight();

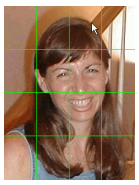
Saving a Modified Picture

- When you draw on a picture you are changing the picture in memory
  - Not changing the original picture file
- You can write out a new picture file with the changed picture data
  - pictureObj.write(*pathWithFile*);
    - pathWithFile is the fully qualified path name to write to including the base file name with the extension
  - You can use FileChooser.getMediaPath(baseFile);
    - pictureObj.write(FileChooser.getMediaPath("barbGrid.jpg");

Drawing Lines Exercise

- Write a method drawGrid in Picture.java to draw horizontal and vertical lines on the current picture, using a Turtle
  - Draw 3 lines in x and 3 in y
- To test it:

```
String file = FileChooser.getMediaPath
    ("barbara.jpg");
Picture p = new Picture(file);
p.drawGrid();
p.show();
```



Drawing Other Shapes

- How would you draw a circle on a picture?
- How would you draw text?
- Java has a class that knows how to do these things
  - Using a Graphics object
    - It knows how to draw and fill simple shapes and images
  - You can draw on a picture object
    - By getting the graphics object from it
      - *pictureObj*.getGraphics();

AWT Graphics Class

- Methods of the Graphics class in the java.awt *package* (group of related classes) let you paint
  - Pick a color to use
  - Draw some shapes
    - Circles, Rectangles, Lines, Polygons, Arcs
  - Shapes drawn on top of other shapes will cover them
  - Set the font to use
    - Draw some letters (strings)

Working with java.awt.Color

- To create a new color object
  - new Color(redValue,greenValue,blueValue)
- There are predefined colors
  - red, green, blue, black, yellow, gray, magenta, cyan, pink, orange
  - To use these do: Color.RED or Color.red
- Set the current drawing color using
  - *graphicsObj*.setColor(colorObj);
- Get the current drawing color using
  - Color currColor = *graphicsObj*.getColor();

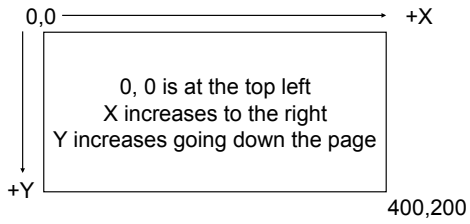
Using Classes in Packages

- Java organizes classes into packages
  - Groups of related classes
- The full name of a class is
  - packageName.ClassName
  - java.awt.Color
- If you want to use just the class name to refer to a class in a package
  - Use the import statement before the class definition in your class

```
import java.awt.Color;   // to allow you to use Color or
import java.awt.*;       // to use any class in this package
```

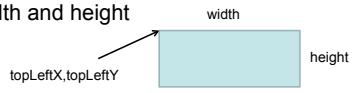

Graphics Environment

- Graphics are often positioned by their top left corner
- Coordinate units are measured in pixels



Drawing Rectangles

- *gObj*.drawRect(*topLeftX,topLeftY,width,height*);
- Will draw a rectangle that has the top left corner at location topLeftX, topLeftY and the specified width and height
- *gObj*.fillRect(*topLeftX,topLeftY,width,height*);
- Will draw a filled rectangle that has the top left corner at location topLeftX, topLeftY and the specified width and height



addBox Method

```
/**
 * Method to add a solid red rectangle to the current picture
 */
public void addBox()
{
  // get the graphics context from the picture
  Graphics g = this.getGraphics();

  // set the color to red
  g.setColor(Color.red);

  // draw the box as a filled rectangle
  g.fillRect(150,200,50,50);
}
```


Testing addBox

```
public static void main(String[] args)
{
   String filename =  FileChooser.getMediaPat
   h("beach-smaller.jpg");
   Picture p = new Picture(filename);
   p.addBox();
   p.show();
}
```

A more general addBox method

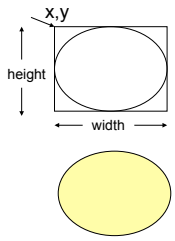
```
public void drawBox(Color color, int topLeftX, int topLeftY,
                    int width, int height)
 {
   // get the graphics context for drawing
   Graphics g = this.getGraphics();

   // set the current color
   g.setColor(color);

   // draw the filled rectangle
   g.fillRect(topLeftX,topLeftY,width,height);
 }
```


Comments

- You should add comments to your methods
  - To help explain what the method is for
  - And to explain what a statement is doing
- There are 3 types of comments in Java
  /** Javadoc type */
  /* multi-line comment */
  // comments out from there to the end of the line
- Javadoc is the Java utility that creates the API documentation from Java source code

Drawing Circles and Ellipses

- *gObj*.drawOval(x,y,width, height)
- *gObj*.fillOval(x,y,width, height)
- Give the x and y of the upper left corner of the enclosing rectangle
  - Not a point on the circle or ellipse
- Give the width and height of the enclosing rectangle
  - To make a circle use the same value for the width and height



Draw Circle Exercise

- Write a method to add a yellow sun to a picture
  - Test with beach.jpg

```
String file = FileChooser.getMediaPath
    ("beach.jpg");
Picture p = new Picture(file);
p.drawSun();
p.show();
```

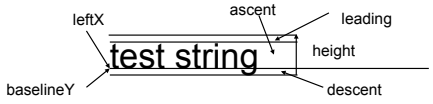
  - Save the new image with *pictureObj*.write(*fileName*);

Working with Fonts

- Create a font object with the font name, style, and point size
  - Font labelFont = new Font("TimesRoman", Font.BOLD, 24);
  - Font normalFont = new Font("Helvetica",Font.PLAIN, 12);
- Set the current font
  - gObj.setFont(labelFont);
- Get font information
  - gObj.getStyle(), g.getSize(), g.getName(), g.getFamily ()

Working with Strings

- To draw a string
  - gObj.drawString("test",leftX,baselineY);
- Use FontMetrics class for drawing information
  - FontMetrics currFontMetrics = g.getFontMetrics();
  - int baselineY = currFontMetrics.getHeight() - currFontMetrics.getDescent();
  - int width = currFontMetrics.stringWidth("test");



Method to draw a string on a picture

```
public void drawString(String text, int x,  int y, Font font, Color color)
 {

   // get the graphics object
   Graphics g = this.getGraphics();

   // set the color
   g.setColor(color);

   // set the font
   g.setFont(font);

   // draw the string
   g.drawString(text,x,y);
}
```

Testing drawString

- Use the picture explorer to find a good x value for left x and y value for the baseline y of the string

```
> Picture p =
      new Picture(FileChooser.getMediaPath("kitten.jpg"));
> p.explore();
> p.drawString("Matt's picture of a kitten in Greece", 67, 283);
> p.explore();
```

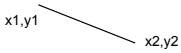
Centering the String in Width

- Using the FontMetrics class we can get the width of the displayed string in pixels
  - In the current font

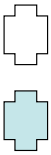
  int strWidth = fontMetrics.stringWidth(text);
- And use this to calculate the left x for the string so that it is centered in width
  - subtract half the string width from half the width of the picture to get the left x

Drawing Lines and Polygons

- Line
  - g.drawLine(x1,y1,x2,y2);



- Polygon
  - Outlined Polygon
    - gObj.drawPolygon(xArray,yArray,numPoints);
    - gObj.drawPolygon(currPolygon);
  - Filled Polygon
    - gObj.fillPolygon(xArray, yArray, numPoints);
    - gObj.fillPolygon(currPolygon);

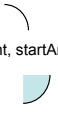


Drawing Lines Exercise

- Write a method (drawX) for adding two crossed lines to a picture
  - Using a passed color
- Start one line at the top left corner of the picture
  - End it at the bottom right corner of the picture
- Start the other line at the bottom left of the picture
  - End it at the top right
- You can test it with barbara.jpg

Drawing Arcs

- Arcs
  - Outlined Arc
    - g.drawArc(topLeftX, topLeftY, width, height, startAngle, arcAngle);
  - Filled Arc
    - g.fillArc((topLeftX, topLeftY, width, height, startAngle, arcAngle);

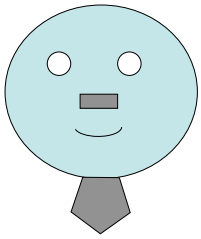


Drawing on a Blank Picture

- You can make pictures from the "blank" files
  - They will have all white pixels
  - 640x480.jpg
  - 7inX95in.jpg
- You can also create a "blank" picture with a width and height
  - They will also have all white pixels
  - Picture blankPicture = new Picture(width,height);

Draw a Picture Exercise

- Create a method that will draw a simple picture
  - Use at least one rectangle
  - Use at least one polygon
  - Use at least one oval
  - Use at least one arc



Bitmapped Versus Vector Graphics

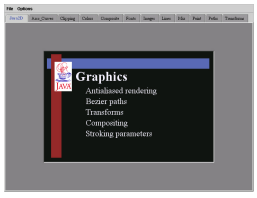
- We have been doing bitmapped graphics
  - Specifying the color of each pixel in the picture
- We just wrote a method that drew a simple picture
  - Which is smaller the program or the picture?
- Some applications use vector graphics which are programs that produce the picture
  - Used in Postscript, Flash, and AutoCAD
  - Advantages: smaller, easy to change, can be scaled

Java 2D Graphics – java.awt

- Newer drawing classes
  - More object-oriented
  - Instead of drawOval() or fillOval() you create an Ellipse2D object and ask a 2d graphics object to draw or fill it
  - Geometric shapes are in the java.awt.geom package
- Advanced Drawing
  - Support for different types of brushes
    - Line thickness, dashed lines, etc
  - Supports cubic curves and general paths
  - Drawing of gradients and textures
  - Move, rotate, scale and shear text and graphics
  - Create composite images

Java 2D Demo

- Open a console window
- Change directory to C :\jdk\demo\jfc\Java2D
- Run the demo
  java –jar Java2Demo.jar
- The source code is inC :\jdk\demo\jfc\Java2D\src

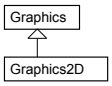


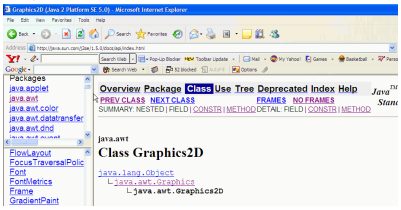
How To Use Java 2D

- Cast the Graphics class to Graphics2D
  - Graphics2D g2 = (Graphics2D) gObj;
- Set up the stroke if desired (type of pen)
  - g2.setStroke(new BasicStroke(widthAsFloat));
- Set up a Color, GradientPaint, or TexturePaint
  - g2.setPaint(Color.blue);
  - g2.setPaint(blueToPurpleGradient);
  - g2.setPaint(texture);
- Create a geometric shape
  - Line2D line2D = new Line2D.Double(0.0,0.0,100.0,100.0);
- Draw the outline of a geometric shape
  - g2.draw(line2d);
- Fill a geometric shape
  - g2.fill(rectangle2d);

Graphics2D inherits from Graphics

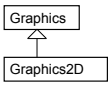
- Inherits basic drawing ability from Graphics
- Adds more advanced drawing ability





Inheritance Terminology

- Graphics is the parent class, base class or superclass
- Graphics2D is the child class, derived class, or subclass
- An object of type Graphics2D can be *upcast* to type Graphics
  - Or *downcast* back to Graphics2D
  - Using (Graphics2D) gObj



Inheritance

- Class Graphics2D inherits from Graphics
- It inherits fields and methods
  - Graphics has a drawImage method
  - Graphics2D inherits this method from Graphics
- The API shows the parent class
  - And the inherited methods
  - Look in package java.awt and then at the class Graphics2D
  - http://java.sun.com/j2se/1.5.0/docs/api/index.html

Drawing Lines Method

- Method drawWideX adds two wide crossed lines to a picture
  - Using a passed color
  - Using a passed line width
- Set up the stroke to make the lines thicker
  - g2.setStroke(new BasicStroke(width));
  - Draw the lines
- You can use redMotorcycle.jpg to test.

drawWideX Method

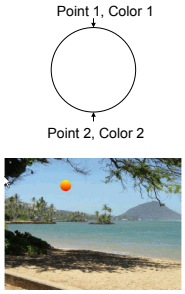
```
public void drawWideX(Color color, float width)
 {
   // get the Graphics2D object
   Graphics graphics = this.getGraphics();
   Graphics2D g2 = (Graphics2D) graphics;

   // set the color and brush width
   g2.setPaint(color);
   g2.setStroke(new BasicStroke(width));

   // get the max x and y values
   int maxX = getWidth() - 1;
   int maxY = getHeight() - 1;
```


drawWideX - continued

```
   // draw the lines
   g2.draw(new Line2D.Double(0,0,maxX,maxY));
   g2.draw(new Line2D.Double(0,maxY, maxX,0));
 }
```

Drawing with a Gradient Paint

- Instead of filling with one color you can fill with a paint that changes from one color to another
  - java.awt.GradientPaint
- Create by specifying a point and the color at that point and then a second point and the color at that point.
  - There will be a change from one color to the other



The drawSun Method

```
public void drawSun(int x, int y, int width, int height)
 {

   // get the graphics2D object for this picture
   Graphics g = this.getGraphics();
   Graphics2D g2 = (Graphics2D) g;

   // create the gradient for painting from yellow to red with
   // yellow at the top of the sun and red at the bottom
   float xMid = (float) (width / 0.5 + x);
   GradientPaint gPaint = new GradientPaint(xMid, y,
                                            Color.yellow,
                                            xMid, y + height,
                                            Color.red);

   // set the gradient and draw the ellipse
   g2.setPaint(gPaint);
   g2.fill(new Ellipse2D.Double(x,y,width,height));
 }
```


Testing drawSun

- String file = FileChooser.getMediaPath ("beach.jpg");
- Picture p = new Picture(file);
- p.drawSun(201,80,40,40);
- p.show();

Paint is an Interface

- Look up the API for Graphics2D
  - Find the setPaint method
    - Notice that it takes a Paint object as a parameter
- How come we can pass this method a java.awt.Color object or a java.awt.GradientPaint object?
  - They both implement the Paint interface
- Objects can be passed to a method that requires an object of an interface type
  - As long as the object is from a class that implements that interface
  - Or inherits from a class that implements the interface

Why Use an Interface?

- A USB interface lets you plug in different devices
  - Camera, disk drive, key drive, etc
- The computer doesn't care what the device is
  - Just that it uses the USB interface
- Java interfaces are the same
  - They let you plug in different classes as long as they implement the interface
    - This means the implementing class must include all the methods defined in the interface

Clipping to a Shape

- You can specify a shape to clip the image to when you draw it
  - Ellipse2D.Double ellipse = new Ellipse2D.Double(0,0,width,height);
  - g2.setClip(ellipse);
- And only the portion of the image that is inside that shape will be drawn
  - g2.drawImage(this.getImage(),0,0,width, height,null);

Clipping to an Ellipse Method

```
public Picture clipToEllipse()
 {
   int width = this.getWidth();
   int height = this.getHeight();
   Picture result = new Picture
    (width,height);

   // get the graphics2D object
   Graphics g = result.getGraphics();
   Graphics2D g2 = (Graphics2D) g;

   // create an ellipse for clipping
   Ellipse2D.Double ellipse =
     new Ellipse2D.Double(0,0,width,height);

   // use the ellipse for clipping
   g2.setClip(ellipse);

   // draw the image
   g2.drawImage(this.getImage(),
                0,0,width,
                height,null);

   // return the result
   return result;
 }
```


Testing clipToEllipse

- This method creates a new picture and returns it so in order to see if we will need to save a reference to it

```
String file = FileChooser.getMediaPath
   ("beach.jpg");
Picture p = new Picture(file);
Picture p2 = p.clipToEllipse();
p2.show();
```


Clipping Exercise

- Write a method that will clip a picture to a triangle
  - You can use the java.awt.geom.GeneralPath class to create the path to clip to
  - You can create a new GeneralPath passing it a
    - Line2D.Double object
  - You can append more
    - Line2D.Double objects

Drawing on Alice Pictures

- Start Alice and set up a scene
- Click the "Play" button
- Pause the action
  - Click the "Take Picture" button to take a picture
  - Alice will display the name and location of the picture
  - You can rename the picture and move it to mediasources

Summary

- Java packages group related classes
  - Package java.awt has classes for Color and Graphics
- You can use an import statement when you want to use short name for classes that are in packages
  - Or use the full name such as java.awt.Color
- Class methods can be called on the class name
- When a class inherits from another class it inherits the object data and behavior
- Objects can be declared to be an interface type