Makeing 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);

```python
>>> blankPicture = makePicture( "640x480.jpg" )
>>> blankPicture = makePicture( "7inx95in.jpg" )
>>> blankPicture = makeEmptyPicture( width, height )
```
**Makeing 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

```java
Picture blankPicture = new Picture(width, height);
```

**EXERCISE:** Can you find the constructor method(s) in the Picture class?
Saving a picture

• pictureObject.write("newPictureFileName.jpg");
The “if” Statement

• Use the if statement to conditionally execute another statement or a block of statements

```java
if (boolean test)
    statement

if (boolean test)
{
    // statements to execute if the test is true
}
```
Blocks of Statements

• The if statement will conditionally execute
  – the following statement or
  – A block of statements
  – if the test is true

• To conditionally execute a block of statements
  – Enclose them in ‘{‘ and ‘}’

• Indent the following statement or block of statements
  – To make it easier to read

• It is good practice to always enclose conditional statements in a block
  – Less likely to cause an error if the code is modified

based on slides by Barb Ericson,
Georgia Institute of Technology
public void removeRedEye(int startX, int startY, int endX, int endY, Color newColor)
{
    Pixel pixelObj = null;

    // loop through the pixels in the rectangle defined by the
    // startX, startY, and endX and endY
    for (int x = startX; x < endX; x++)
    {
        for (int y = startY; y < endY; y++)
        {

            // get the current pixel
            pixelObj = getPixel(x,y);

            // process pixel
            // e.g., adjust red value
            if (pixelObj.red > threshold)
            {
                pixelObj.red = newColor.red;
            }
        }
    }
}
Remove Red Eye Method

// if the color is near red then change it
if (pixelObj.colorDistance(Color.red) < 167)
{
    pixelObj.setColor(newColor);
}
}
Testing removeRedEye

• Use the picture explorer to find the values for start x, start y, end x, and end y
• Try the following to test removeRedEye

String file =
    FileChooser.getMediaPath("jenny-red.jpg");
Picture p = new Picture(file);
p.explore();
p.removeRedEye(startX,startY,endX,endY, Color.awt.BLACK);
p.explore();
Edge Detection

• Loop through all the pixels in the picture
  – Calculate the average color for the current pixel and the pixel at the same x but y+1.
  – Get the distance between the two averages
  – If the absolute value of the distance is greater than some value turn the current pixel black
  – Otherwise turn the current pixel white
Edge Detection Algorithm

• To find areas of high contrast
  – Try to loop from row = 0 to row = height – 1
    • Loop from x = 0 to x = width
      – Get the pixel at the x and y (top pixel)
      – Get the pixel at the x and (y + 1) bottom pixel
      – Get the average of the top pixel color values
      – Get the average of the bottom pixel color values
      – If the absolute value of the difference between the averages is over a passed limit
        » Turn the pixel black
        » Otherwise turn the pixel white
Edge Detection Exercise

• Write a method `edgeDetection` that takes an input limit
  – And turns all pixels black where the absolute value of the difference between that pixel and the below pixel is greater than the passed limit
  – And turns all pixels white where the absolute value of the difference between that pixel and the below pixel is less than or equal the passed limit
  – Pixel has a `getAverage()` method that returns the average of the three colors at the pixel
IF, ELSE, AND, OR, exclusive OR

• Use if and else if you have two possibilities to deal with
  if (test)
  {
    // statements to execute when the test is true
  }
  else
  {
    // statements to execute when the test is false
  }

• Complex conditionals
  – Use ‘and’ to test for more than one thing being true
  – Use ‘or’ to test if at least one thing is true
  – Use ‘exclusive or’ to test that one and only one thing is true
  – Use ‘not’ to change the result from true to false and false to true
## Truth Table

<table>
<thead>
<tr>
<th>Conditional</th>
<th>Operand 1</th>
<th>Operand 2</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>And</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>And</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>And</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>And</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>Or</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>Or</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>Or</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>Or</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>Exclusive Or (aka “XOR”)</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>Exclusive Or (aka “XOR”)</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>Exclusive Or (aka “XOR”)</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>Exclusive Or (aka “XOR”)</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>
Conditional Operators

• We can check if several things are true - And
  – Using `&&` (evaluation stops if the first item is false)
  – Using `&` (to always evaluate both operands)

• We can check if at least one of several things are true - Or
  – Using `||` (evaluation stops if the first item is true)
  – Using `|` (to always evaluate both operands)

• We can check if only one and only one of the things is true – Exclusive Or
  – Using `^`
Using \&\& (And) and || (Or)

• Check that a value is in a range
  – Is some value between 0 and 255 (inclusive)
    • for valid pixel color values
  – 0 <= x <= 255 is written as
  – 0 <= x \&\& x <= 255 // in Java or
  – x >= 0 \&\& x <= 255 // is the same

• Check if at least one of several things is true
  – Is this black or white?
  – True if either it is black \textbf{or} it is white
Not Conditional Operator

• Use ! To change the value to the opposite
  – !true = false
  – !false = true

• A not conditional operator applied to a complex conditional changes it
  – !(op1 && op2) = !op1 || !op2
  – !(op1 || op2) = !op1 && !op2

• This is known as De Morgan’s Law
De Morgan’s Law Exercise

• What is equivalent to the following?
  • !(x > 4 && x < 8)
  • !(y > 2 || y < 10)
  • !(x == 2 && y == 4)
  • !(y != 2 && x != 3)
  • !(x == 3 || x == 5)
  • !(y == 2 || y < 5)
Sepia-Toned Pictures

• Have a yellowish tint, used to make things look old and western
Sepia-toned Algorithm

• First make the picture grayscale.
• Loop through the pixels in the picture
  – Change the shadows (darkest grays) to be even darker (0 <= red < 60)
  – Make the middle grays a brown color (60 <= red < 190)
  – Make the highlights (lightest grays) a bit yellow (190 <= red)
    • Increase red and green
    • Or decrease blue

based on slides by Barb Ericson, Georgia Institute of Technology
Conditionals with > 2 Choices

```java
if (0 <= x && x <= 5)
{
}
else if (x <= 10)
{
}
else // what is x?
{
}
```
public void sepiaTint()
{
    Pixel pixelObj = null;
    double redValue = 0;
    double greenValue = 0;
    double blueValue = 0;

    // first change the current picture to grayscale
    this.grayscale();
// loop through the pixels
for (int x = 0; x < this.getWidth(); x++)
{
    for (int y = 0; y < this.getHeight(); y++)
    {
        // get the current pixel and color values
        pixelObj = this.getPixel(x,y);
        redValue = pixelObj.getRed();
        greenValue = pixelObj.getGreen();
        blueValue = pixelObj.getBlue();
// tint the shadows darker
if (redValue < 60)
{
    redValue = redValue * 0.9;
    greenValue = greenValue * 0.9;
    blueValue = blueValue * 0.9;
}

// tint the midtones a light brown by reducing the blue
else if (redValue < 190)
{
    blueValue = blueValue * 0.8;
}
// tint the highlights a light yellow
// by reducing the blue
else
{
    blueValue = blueValue * 0.9;
}

// set the colors
pixelObj.setRed((int) redValue);
pixelObj.setGreen((int) greenValue);
pixelObj.setBlue((int) blueValue);
Testing sepiaTint

String file =
FileChooser.getMediaPath("gorge.jpg");
Picture p = new Picture(file);
p.explore();
p.sepiaTint();
p.explore();
Background Replacement

• If you have a picture of a person in front of some background
• And a picture of the background itself
• Can you replace the pixel colors for the background to be from another image?
Replace Background

• Replace the colors at all the pixels in the source image
  – That are within some range of the background color
  – Use pixels from another background image
Replace Background Algorithm

• Works on the source picture
  – Pass in the original background and the new background pictures

• Loop through all the pixels in the source image
  – Check if the distance from the source pixel color is within 15.0 of the background picture pixel color
    • If so replace it with the color at the new background pixel

Based on slides by Barb Ericson, Georgia Institute of Technology
How we did this in Python ...

- If this pixel is nearly the same as the pixel in a background-only picture, then substitute a pixel from a new background picture

```python
def swapBackground( src, background, newBackground ):
    # src, and background must be the same size
    # newBackground must be at least as big as src and background
    for x in range(1, getWidth( src ) + 1):
        for y in range(1, getHeight( src ) + 1):
            srcPxl = getPixel( src, x, y )
            backgroundPxl = getPixel( background, x, y )
            if (distance(getColor( srcPxl ), getColor( backgroundPxl )) < 15.0):
                setColor( srcPxl, getColor( getPixel( newBackground, x, y ) ) )
    return src
```
public void swapBackground(Picture oldBackground, Picture newBackground) {
    Pixel currPixel = null;
    Pixel oldPixel = null;
    Pixel newPixel = null;

    // loop through the columns
    for (int x=0; x<this.getWidth(); x++) {
        // loop through the rows
        for (int y=0; y<this.getHeight(); y++) {

        }
    }
}
Swap Background - Cont

// get the current pixel and old background pixel
currPixel = this.getPixel(x,y);
oldPixel = oldBackground.getPixel(x,y);

/* if the distance between the current pixel color
 * and the old background pixel color is less than the 15
 * then swap in the new background pixel
 */
if (currPixel.colorDistance(oldPixel.getColor()) < 15.0)
{
    newPixel = newBackground.getPixel(x,y);
    currPixel.setColor(newPixel.getColor());
}

Testing swapBackground

```
Picture p1 = new Picture(FileChooser.getMediaPath("kid-in-frame.jpg"));
Picture oldBack = new Picture(FileChooser.getMediaPath("bgframe.jpg"));
Picture newBack = new Picture(FileChooser.getMediaPath("moon-surface.jpg"));
p1.swapBackground(oldBack,newBack);
p1.show();

>>> src = makePicture( getMediapath("kid-n-frame.jpg") )
>>> background = makePicture( getMediapath("bgframe.jpg") )
>>> newBackground = makePicture( getMediapath("moon-surface.jpg") )
>>> newPic = swapBackground( src, background, newBackground )
>>> show( newPic )
```
Replace Background Result

- The background color was too close to the shirt color
Chroma Key – Blue Screen

• For TV and movie special effects they use a blue or green screen
  – Here just a blue sheet was used
  – Professionally you need an evenly lit, bright, pure blue background
    • With nothing blue in the scene
Chroma Key Exercise

- Write the method `chromakey` that takes a new background picture as an input parameter
  - It will loop through all the pixels
  - If the pixel color is blue (red + green < blue)
  - Replace the pixel color with the color from the new background pixel (at the same location)
HINT: How we did chromakey in Python

```python
def chromaKey( src, background ):
    # src, background, newBackground must be the same size
    for x in range(1, getWidth( src ) + 1 ) :
        for y in range( 1, getHeight( src ) + 1 ) :
            srcPxl = getPixel( src, x, y )
            backgroundPxl = getPixel( background, x, y )
            if (getRed( srcPxl ) + getGreen( srcPxl ) ) < getBlue( srcPxl ) :
                setColor( srcPxl, getColor( getPixel( background, x, y ) ) )
    return src
```
## Python & Java similarities

<table>
<thead>
<tr>
<th>swapBackground &amp; chromakey</th>
<th>Java</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>overall structure</td>
<td>name, init, loop(s)</td>
<td></td>
</tr>
<tr>
<td>loop structure</td>
<td>nested nested</td>
<td></td>
</tr>
<tr>
<td>algorithm</td>
<td>swapBackground: test each pixel to see if same as background only, replace with new background</td>
<td>chromakey: test each pixel to see if green+red &lt; blue, replace with new background</td>
</tr>
</tbody>
</table>
## Python & Java differences

<table>
<thead>
<tr>
<th>swapBackground &amp; chromakey</th>
<th>Java</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>where</td>
<td>methods in class Picture</td>
<td>not defined as method</td>
</tr>
<tr>
<td>arguments</td>
<td>(background, newBackground)</td>
<td>(source, background, newBackground)</td>
</tr>
<tr>
<td>return result?</td>
<td>operate on “this” object</td>
<td>return changed source</td>
</tr>
<tr>
<td>syntax</td>
<td>declare variables for loops blocks in curly braces { }</td>
<td>no type declaration for loops blocks indented</td>
</tr>
</tbody>
</table>
Testing chromakey

```java
Picture markP = new Picture(FileChooser.getMediaPath("blue-mark.jpg");
Picture newBack = new Picture(Chooser.getMediaPath("moon-surface.jpg");
markP.chromakey(newBack);
markP.show();
```
Questions ?
Drawing Shapes

• How would you draw a circle on a picture?
• How would you draw a string of characters?
• You still would need to set the pixel colors of certain pixels
  – Which pixels?
• Java has a way of doing 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 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)

based on slides by Barb Ericson, Georgia Institute of Technology
Working with 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
    • Remember to import java.awt.*; or java.awt.Color;
• Set the current drawing color using
  graphicsObj.setColor(colorObj);
• Get the current drawing color using
  Color currColor = graphicsObj.getColor();
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

Based on slides by Barb Ericson, Georgia Institute of Technology
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
  ```java
gObj.drawString("test", leftX, baselineY);
```

• Use FontMetrics class for drawing information
  ```java
FontMetrics currFontMetrics = g.getFontMetrics();
int baselineY = currFontMetrics.getHeight() -
                currFontMetrics.getDescent();
int width = currFontMetrics.stringWidth("test");
```
Add a String to a Picture Exercise

• Write a method drawString that will add some text to a picture
  – Set the color to draw with
  – Set the font to use when drawing the string
  – Draw a string near the bottom left of the picture
  – If you have time create another method that takes a string and y value and centers the string in x

• Test with

  String file = FileChooser.getMediaPath("kitten2.jpg");
  Picture p = new Picture(file);
  p.drawString("Barbara Ericson");
  p.show();
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);
Summary

• You can draw by changing the color of the pixels
  – Or you can use java.awt.Graphics to do drawing
• Get the Graphics object from a Picture object
  Graphics graphics = pictureObj.getGraphics();
  Set the color
  graphics.setColor(Color.RED);
• Do some simple drawing
  graphics.drawLine(x1,y1,x2,y2);
  graphics.drawOval(x1,y1,width,height);
  graphics.drawString("string to draw",leftX,baseline);
Drawing Arcs

• Arcs
  – Outlined Arc
    
    \texttt{g.drawArc(topLeftX, topLeftY, width, height, startAngle, arcAngle)};

  – Filled Arc
    
    \texttt{g.fillArc((topLeftX, topLeftY, width, height, startAngle, arcAngle)};
Drawing Rectangles

- Rectangle
  - Outlined Rectangle
    \[ g\text{.drawRect}(\text{topLeftX}, \text{topLeftY}, \text{width}, \text{height}); \]
  - Filled Rectangle
    \[ g\text{.fillRect}(\text{topLeftX}, \text{topLeftY}, \text{width}, \text{height}); \]
  - Outlined Rounded Rectangle
    \[ g\text{.drawRoundRect}(\text{topLeftX}, \text{topLeftY}, \text{width}, \text{height}, \text{arcWidth}, \text{arcHeight}); \]
  - Filled Rounded Rectangle
    \[ g\text{.fillRoundRect}(\text{topLeftX}, \text{topLeftY}, \text{width}, \text{height}, \text{arcWidth}, \text{arcHeight}); \]
public void drawFilledRectangles()
{
    Graphics g = this.getGraphics();
    Color color = null;

    // loop 25 times
    for (int i = 25; i > 0; i--)
    {
        color = new Color(i * 10, i * 5, i);
        g.setColor(color);
        g.fillRect(0,0,i*10,i*10);
    }
}
Java 2D Graphics – java.awt

• Newer drawing classes
  – More object-oriented
  – Instead of drawOval() or fillOval() you create a 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 in C:\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
Drawing Lines Exercise

• Create a new method (drawWideX) for adding 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
  \[ \text{g2.setStroke(new BasicStroke(width))}; \]
  – Draw the lines
• You can use redMotorcycle.jpg to test.
Using a Graphics Object to Copy an Image

```java
public void copy(Picture source, int x, int y)
{
    // get the graphics object
    Graphics g = this.getGraphics();

    // copy the image
    g.drawImage(source.getImage(), x, y, null);
}
```
Using Graphics2D to Copy an Image

```java
public void copy2D(Picture source, int x, int y)
{
    // get the graphics object
    Graphics g = this.getGraphics();
    Graphics g2 = (Graphics2D) g;

    // copy the image
    g2.drawImage(source.getImage(), x, y, null);
}
```
Testing the Copy Method

```java
Picture p1 = new Picture(FileChooser.getMediaPath("beach.jpg"));
Picture p2 = new Picture(FileChooser.getMediaPath("turtle.jpg"));
p1.copy2D(p2,194,304);
p1.show();
```
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
General Scaling

• We scaled a picture up or down
  – But what if we want to scale to a specified size?
  – Or what if we want to scale up in x and down in y?
• We can use the class AffineTransform in the java.awt.geom package
  – Create an object of the class AffineTransform
  – Set up the scaling using the method scale
  – Use the AffineTransform object when you draw the image
General Scale Method

```java
class Picture {
    public Picture scale(double xFactor, double yFactor) {
        // set up the scale transform
        AffineTransform scaleTransform = new AffineTransform();
        scaleTransform.scale(xFactor, yFactor);

        // create a new picture object that is the right size
        Picture result = new Picture((int) (getWidth() * xFactor),
                                     (int) (getHeight() * yFactor));

        // get the graphics 2d object to draw on the result
        Graphics graphics = result.getGraphics();
        Graphics2D g2 = (Graphics2D) graphics;

        // draw the current image onto the result image scaled
        g2.drawImage(this.getImage(), scaleTransform, null);

        return result;
    }
}
```

based on slides by Barb Ericson,
Georgia Institute of Technology
Testing General Scale Method

- Notice that the general scale method creates and returns a new picture of the appropriate size
  - So to see the result we need to save a reference to it in a variable

```java
String file = FileChooser.getMediaPath("mattDoor.jpg");
Picture p = new Picture(file);
Picture p1 = p.scale(2.0,0.5);
p1.show();
```
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

based on slides by Barb Ericson, Georgia Institute of Technology
The drawSun Method

```java
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));
}
```

based on slides by Barb Ericson,
Georgia Institute of Technology
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
  ```java
  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
  ```java
  g2.drawImage(this.getImage(),0,0,width,height,null);
  ```
Clipping to an Ellipse Method

```java
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

```java
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
Summary

• Inheritance means that the child class gets all the fields and methods of the parent class
  – See this in the API for Graphics2D

• You can use a class to do general scaling and rotation
  – java.awt.geom.AffineTransform

• You can draw with a gradient paint
  – That changes from one color to another

• Objects of classes that implement an interface can be said to be of that type
  – And can be passed as parameters to methods that take that type

• You can clip a picture to a geometric object
  – Or a path
Coming Attractions

• For Friday:
  – Project 9 due 2:00 PM
  – start of code for group projects due 4:20 PM
  • if you won’t be here for Lab, make arrangements with your team mate!

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Burton</td>
<td>Talley</td>
<td>Davies</td>
<td>Bowers</td>
<td>Demase</td>
<td>Burke</td>
<td>Currin</td>
<td>Thayer</td>
</tr>
<tr>
<td></td>
<td>Highman</td>
<td>D'Augustine</td>
<td>Taylor</td>
<td>Knowles</td>
<td>Ho</td>
<td>Maier</td>
<td>Malhotra</td>
<td>Heitzer</td>
</tr>
<tr>
<td>B</td>
<td>Regione</td>
<td>Zhang</td>
<td>Howell</td>
<td>Ha</td>
<td>Tran</td>
<td>Roithmayr</td>
<td>Pham</td>
<td>Walsh</td>
</tr>
<tr>
<td></td>
<td>Messick</td>
<td>Rhyner</td>
<td>Nassery</td>
<td>Dahiya</td>
<td>Duffy</td>
<td>Merrow</td>
<td>Slack</td>
<td>Hughes</td>
</tr>
</tbody>
</table>

based on slides from Barb Ericson, Georgia Institute of Technology