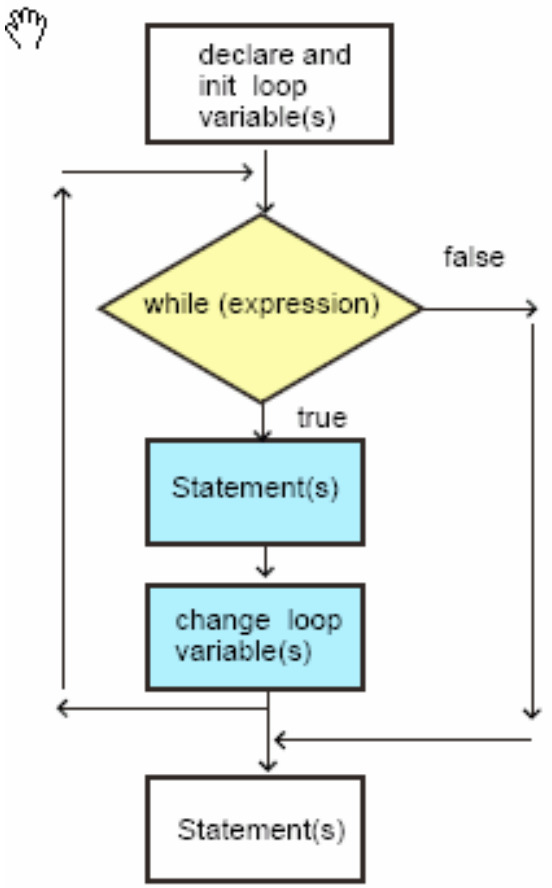
Media Computation

Lecture 13.1, November 17, 2008

Steve Harrison

While Loops

- In Java one way to repeat a block of statements while an expression is true is to use a while loop
- Create a counter and set it to the start value
- Check that the counter is less then the stop value
- If it is less than execute the statements in the loop
- Add one to the counter and go back to check that the counter is less than the stop value



based on slides from Barb Ericson,
Georgia Institute of Technology

While Loop Syntax

- Adding up the numbers from 1 to 100

```
int total = 0;
int num = 1;
while (num <= 100)
{
    total = total + num;
    num = num + 1;
}
System.out.println(total);
```

While Loop Syntax

- Adding up the numbers from 1 to 100

```
int total = 0;  // declare and initialize the total
int num = 1; // declare and init the number

while (num <= 100) // do while num <= 100
{
    total = total + num; // add num to total
    num = num + 1;     // increment the num
}
System.out.println(total);  // print the total
```

based on slides from Barb Ericson,
Georgia Institute of Technology

Parts of a While Loop

- Adding up the numbers from 1 to 100

```
int total = 0;
int num = 1;
while (num <= 100)
{
    total = total + num;
    num = num + 1;
}
System.out.println(total);
```

Declaration and initialization of variables

This test is done each time and when It is true the loop body will be executed

This is the body of the loop. It Starts with a '{' and ends with a '}'. If there is just one statement In a loop body the '{' and '}' aren't needed.

Changing from For-each to While

- In a for-each loop something needs to keep track of the current pixel
  - –And change each time through the loop
  - –To sure that we have gone through all of the pixels
- We can loop through all elements in an array by starting with index 0, then index 1, and so on till index (length – 1)
  - –And get the pixel at the current index value

Reduce Red Algorithm

- Get the array of Pixel objects from the current picture
- Declare a variable to hold the red value
- Declare the index variable and set it to 0
- Declare a variable to refer to the current pixel
- Loop while index is less than the length of the array
  - Get the pixel at the index value
  - Get the current red value from the pixel
  - Divide the current red value by 2
  - Set the red for the current pixel to the changed value
  - Increment the index

based on slides from Barb Ericson,
Georgia Institute of Technology

Decrease Red Method

```
public void decreaseRed()
 {
    Pixel[] pixelArray = this.getPixels();
    int value = 0;
    Pixel pixelObj = null;
    int index = 0;

   // loop through all the pixels
   while(index < pixelArray.length)
   {
     // get the current pixel
     pixelObj = pixelArray[index];
     // get the red value
     value = pixelObj.getRed();

     // decrease the red value
     value = value / 2;

      // set the pixel red
      pixelObj.setRed(value);

      // increment the index
      index = index + 1;
   }
```

Method Names

- If we add this new method to Picture.java and compile will we get an error?
  - We will have two methods with the same name and both take no parameters
  - We can have two methods with the same name
    - As long as the parameter list is different
- We need to name one of them differently
  - Rename the first one to decreaseRedForEach

Comments

- There are several kinds in Java

```
// a comment that lasts to the end of the line
/* a comment that can take up several lines
   until a */
/** a Javadoc comment that is used to create
          html documentation of your code */
```

- They are ignored by the compiler

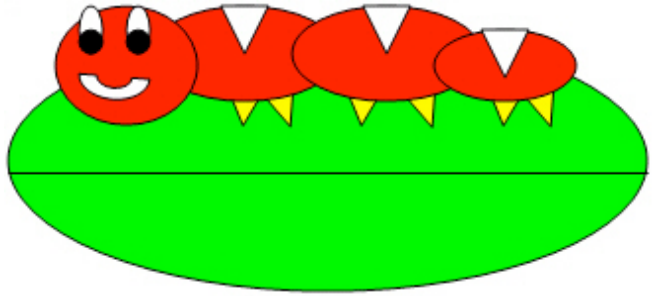
Decrease Red Exercise

- In DrJava
  - Add the method decreaseRed() to Picture.java
    - Before the last } which ends the class definition
  - Compile the method
    - Click the Compile All button
  - Test it by doing the following in the interactions pane

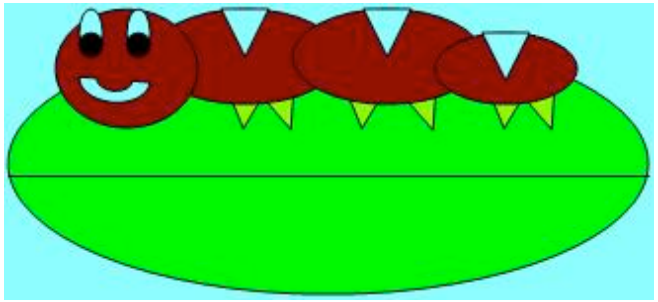
```
> String fileName = "C:/intro-prog-java/mediasources/caterpillar.jpg";
> Picture picture1 = new Picture(fileName);
> picture1.explore();
> picture1.decreaseRed();
> picture1.explore();
```

decreaseRed Method Result

- Before method



- After method



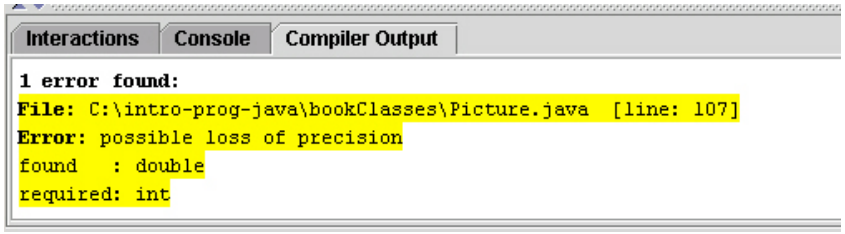
based on slides from Barb Ericson,
Georgia Institute of Technology

Using Multiplication by 0.5

- You could have also multiplied the red value by 0.5 and then cast back to integer
  - value = value * 0.5;
- Change the line that divided by 2 and try to compile this
  - In the decreaseRed method

Loss of Precision

- If you try the code on the previous slide you will get a compiler error
  - Possible loss of precision
- It is complaining about putting a double value into a int variable
  - Loss of fractional part




```
171        // decrease the red value by 50% (1/2)
172        value = value * 0.5;
173
174        // set the red value of the current pixel
175        pixel.setRed(value);
176
```

Casting to Solve Loss of Precision Error

- It will compile if we tell the compiler we know about the possible loss of precision
  - –And that it is intended
    - By using a cast to int

      value = (int) (value * 0.5);
    - Notice that we cast the result of the multiplication back to an integer
- Casting is forcing a value into a type

  (*type*) expression

Put Declarations Outside Loops !

- When you need a variable in a loop it is best to declare it before the loop
  - Otherwise you are declaring a new variable each time through the loop
    - Which is slower than just changing the value associated with the variable
- In some languages you must declare all variables at the beginning of a method (function)
- In Java you can declare variables anywhere in a method
  - As long as you declare them before you use them
  - They are known in the block they are declared in

based on slides from Barb Ericson,
Georgia Institute of Technology

Shortcuts for Common Operations

- In programming you often need to add one to a value

  index = index + 1;

- You may use the shortcut

  index++; or ++index;

- If you wanted to subtract 1 instead

  index = index – 1;

  index--; or -- index;

Faking a Sunset

- If you want to make an outdoor scene look like it happened during sunset
  - – You might want to increase the red
    - But you can't increase past 255
  - – Another idea is to reduce the blue and green
    - To emphasize the red
    - Try to reduce the blue and green by 30%

based on slides from Barb Ericson,
Georgia Institute of Technology

Faking a Sunset Algorithm

- Reduce the blue and green by 30%
  1. Get the array of pixels from the picture
  2. Set up an index to start at 0
  3. Loop while the index is less than the length of the array
     1. Get the pixel at the current index from the array of pixels
     2. Set the blue value at the pixel to 0.7 times the original value
     3. Set the green value at the pixel to 0.7 times the original value
     4. Increment the index and go back to step 3

Faking a Sunset Method

```
/**
 * Method to simulate a sunset by
 * decreasing the green
 * and blue
 */
public void makeSunset()
{
    Pixel[] pixelArray = this.getPixels();
    Pixel pixelObj = null;
    int value = 0;
    int i = 0;

    // loop through all the pixels
    while (i < pixelArray.length)
    {
      // get the current pixel
      pixelObj = pixelArray[i];

      // change the blue value
      value = pixelObj.getBlue();
      pixelObj.setBlue((int) (value * 0.7));

      // change the green value
      value = pixelObj.getGreen();
      pixelObj.setGreen((int) (value * 0.7));

      // increment the index
      i++;
    }
}
```

based on slides from Barb Ericson,
Georgia Institute of Technology

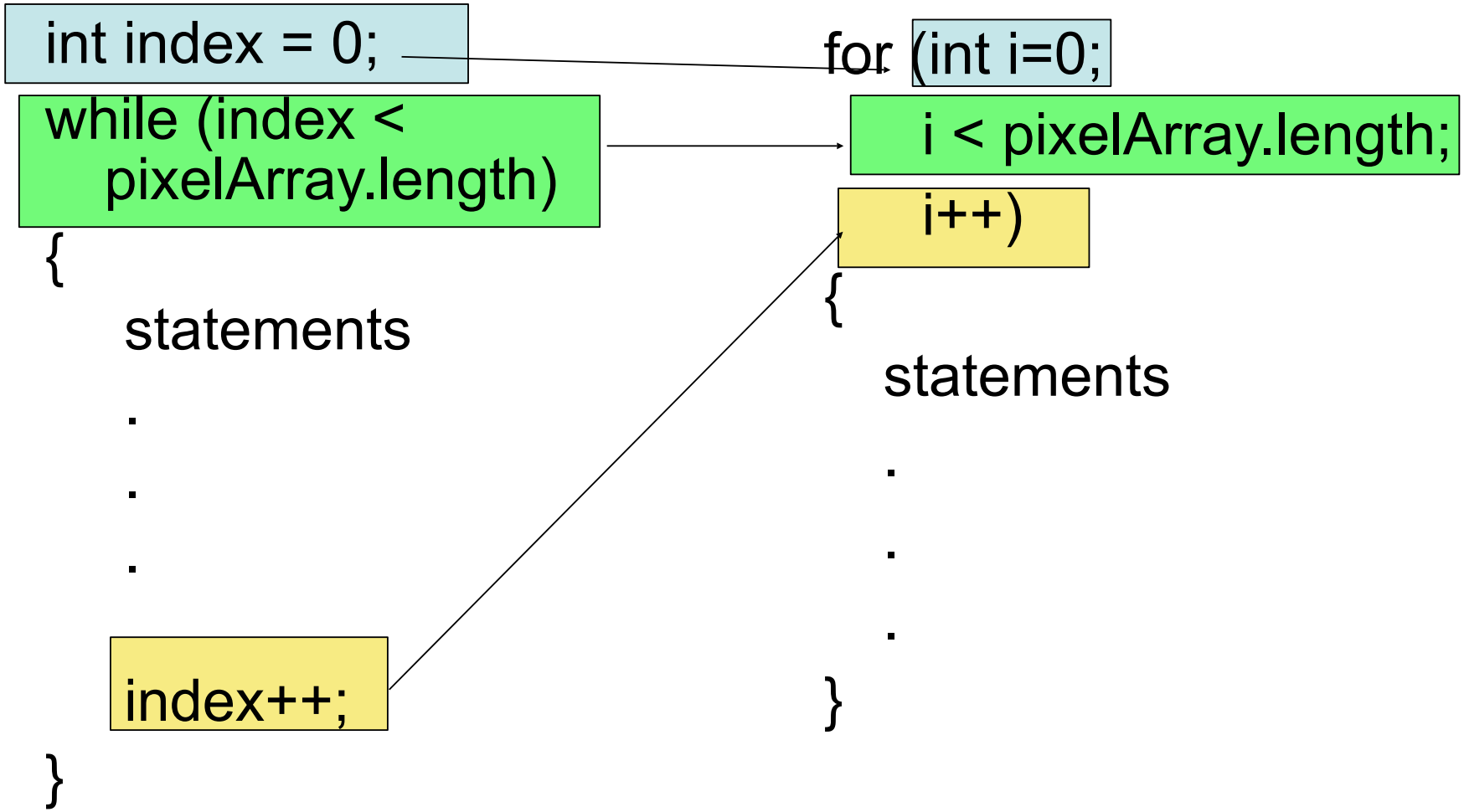
For Loops

- Programmers like shortcuts
  - Especially those that reduce errors
  - And mean less typing
- We have been using a while loop with an index
  - We had to declare the index variable and initialize it before the loop
    - If you forget this there will be a compiler error
  - We had to increment the index in the loop
    - If you forget this it will be an infinite loop
- The shortcut for this is a For Loop

For Loop Syntax

- for (initialization area; continuation test; change area)
  - –Initialization area
    - Declare variables and initialize them
  - –Continuation test
    - If true do body of loop
    - If false jump to next statement after the loop
  - –Change area
    - Change the loop variables
      - –Increment or decrement them

Comparison of While and For Loops



based on slides from Barb Ericson,
Georgia Institute of Technology

Change clearBlue() to use a For Loop

```
/**
 * Method to clear the blue (set
 * the blue to 0 for all pixels)
 */
public void clearBlue()
{
    Pixel pixelObj = null;

    // get the array of pixels
    Pixel[] pixelArray =
     this.getPixels();

    // loop through all the pixels
    for (int i = 0;
          i < pixelArray.length;
          i++)
    {
      // get the current pixel
      pixelObj = pixelArray[i];

      // set the blue on the pixel to 0
      pixelObj.setBlue(0);
    }

}
```

Using System.out.println() in a Loop

- One way to check what is happening in your program is to add

```
System.out.println(expression);
```

- You might add this to the loop to check the value of ‘i’ during it.

–And to verify that the increment happens after the last statement in the loop

Negative Method

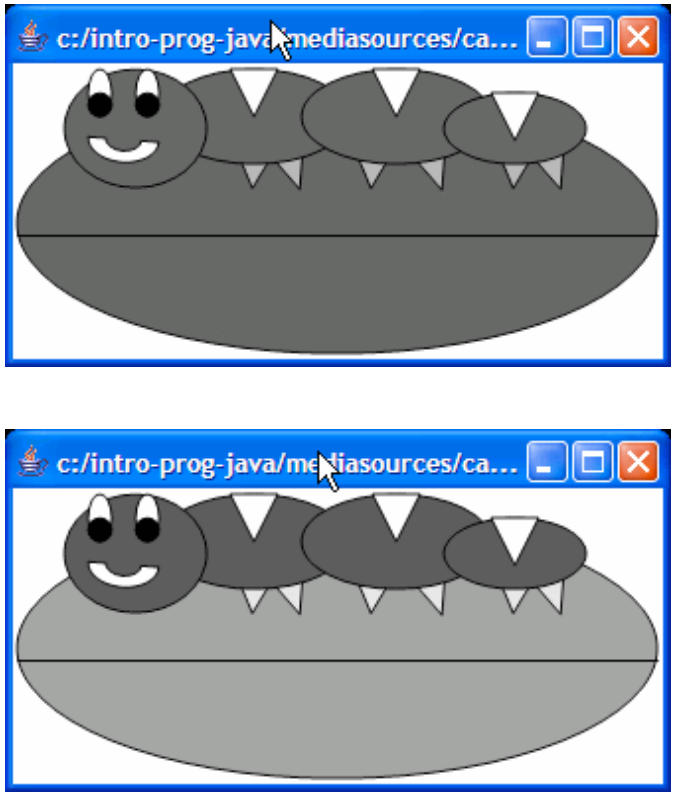
```
/**
 * Method to turn a picture into a negative
     of the picture
 */
public void negate()
{
    Pixel[] pixelArray = this.getPixels();
    Pixel pixelObj = null;
    int redValue, blueValue, greenValue = 0;

    // loop through all the pixels
    for (int i = 0; i < pixelArray.length; i++)
    {
      // get the current pixel
      pixelObj = pixelArray[i];

      // get the values
      redValue = pixelObj.getRed();
      greenValue =
                  pixelObj.getGreen();
      blueValue = pixelObj.getBlue();

      // set the pixel's color
      pixelObj.setColor(
            new Color(255 - redValue,
                      255 - greenValue,
                      255 - blueValue));
    }
}
```

Grayscale Method

```
/**
 * Method to change the picture to gray
   scale
 */
public void grayscale()
{
  Pixel[] pixelArray = this.getPixels();
  Pixel pixelObj = null;
  int intensity = 0;

  // loop through all the pixels
  for (int i = 0; i < pixelArray.length;
    i++)
  {
    // get the current pixel
    pixelObj = pixelArray[i];

    // compute the average intensity
    intensity =
              (pixelObj.getRed() +
                pixelObj.getGreen() +
                pixelObj.getBlue()) / 3;

    // set the pixel color
    pixelObj.setColor(new
    Color(intensity,intensity,intensity));

  }
}
```

Grayscale with Luminance Exercise

- Create a new method grayscaleWithLuminance
- Using the new algorithm for calculating intensity
- intensity = (int) (red * 0.299 + green * 0.587 + blue * 0.114)

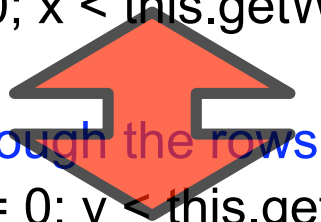


Nested Loop Template

```
// loop through the rows
for (int y = 0; y < this.getHeight(); y++)
{
   // loop through the columns
   for (int x = 0; x < this.getWidth(); x++)
   {
      // get the current pixel
      pixelObj = this.getPixel(x,y);

      // do something to the color

      // set the new color
      pixelObj.setColor(colorObj);
   }
}
```

Alternative Nested Loop Template

```
// loop through the columns (x direction)
for (int x = 0; x < this.getWidth(); x++)
{
   // loop through the rows (y direction)
   for (int y = 0; y < this.getHeight(); y++)
   {
      // get the current pixel
      pixelObj = this.getPixel(x,y);

      // do something to the color

      // set the new color
      pixelObj.setColor(colorObj);
   }
}
```



Lighten the Color Algorithm

- Start x at 0 and loop while x < the picture width (add 1 to x at the end of each loop)
  - Start y at 0 and loop while y < the picture height (add 1 to y at the end of each loop)
    - Get the pixel at this location
    - Get the color at the pixel
    - Lighten (brighten) the color
    - Set the color for the pixel to the lighter color

Lighten the Color with a Nested Loop

```
public void lighten()
  {
    Pixel pixelObj = null;
    Color colorObj = null;

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

      // loop through the rows (y direction)
      for (int y = 0; y < this.getHeight(); y++)
      {
```

based on slides from Barb Ericson,
Georgia Institute of Technology

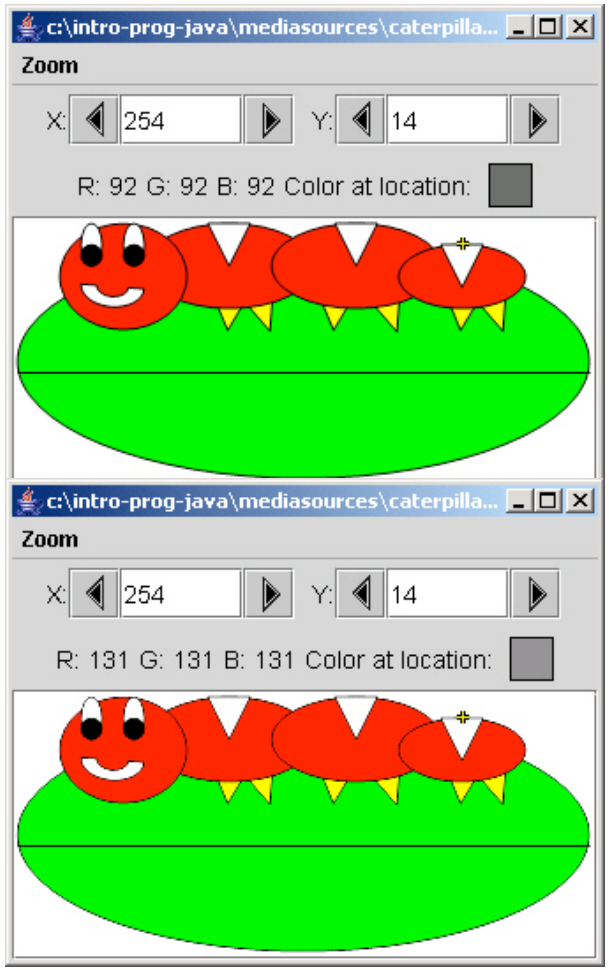
Lighten - Continued

```
      // get pixel at the x and y location
      pixelObj = this.getPixel(x,y);

      // get the current color
      colorObj = pixelObj.getColor();

      // get a lighter color
      colorObj = colorObj.brighter();

      // set the pixel color to the lighter color
      pixelObj.setColor(colorObj);
    }
  }
}
```

Trying the Lighten Method

- In the interactions pane:

```
String file =
"c:/intro-prog-java/mediasources/
    caterpillar.jpg";
Picture p1 = new Picture(file);
p1.explore();
p1.lighten();
p1.explore();
```



Copy Picture Algorithm

- Copy a picture to the 7 by 9.5 inch blank picture
  - Create the target picture object
  - Invoke the method on the target picture
    - Create the source picture object
    - Loop through the source picture pixels
      - Get the source and target pixels
      - Set the color of the target pixel to the color of the source pixel

Copy Algorithm to Code

- Loop through the source pixels

```
// loop through the columns
for (int sourceX = 0, targetX = 0;
     sourceX < sourcePicture.getWidth();
     sourceX++, targetX++)
{

  // loop through the rows
  for (int sourceY = 0, targetY = 0;
       sourceY < sourcePicture.getHeight();
       sourceY++, targetY++)
  {
```

Copy Algorithm to Code – Cont

- Get the source and target pixels

```
sourcePixel =
  sourcePicture.getPixel(sourceX,sourceY);
targetPixel = this.getPixel(targetX,targetY);
```

- Set the color of the target pixel to the color of the source pixel

```
targetPixel.setColor(sourcePixel.getColor());
```

Copy Method

```
public void copyKatie()
 {
   String sourceFile =
     FileChooser.getMediaPath("KatieFancy.jpg");
   Picture sourcePicture = new Picture(sourceFile);
   Pixel sourcePixel = null;
   Pixel targetPixel = null;

   // loop through the columns
   for (int sourceX = 0, targetX = 0;
        sourceX < sourcePicture.getWidth();
        sourceX++, targetX++)
   {
```

Copy Method - Continued

```
    // loop through the rows
    for (int sourceY = 0, targetY = 0;
         sourceY < sourcePicture.getHeight();
         sourceY++, targetY++)
    {

      // set the target pixel color to the source pixel color
      sourcePixel = sourcePicture.getPixel(sourceX,sourceY);
      targetPixel = this.getPixel(targetX,targetY);
      targetPixel.setColor(sourcePixel.getColor());
    }
  }
}
```

Trying the copyKatie Method

- Create a picture object using the 7inX95in.jpg file in the mediasources directory

```
Picture pl = new
Picture(FileChooser.getMediaPath("7inX95in.jpg"));
```

- Show the picture

```
pl.show();
```

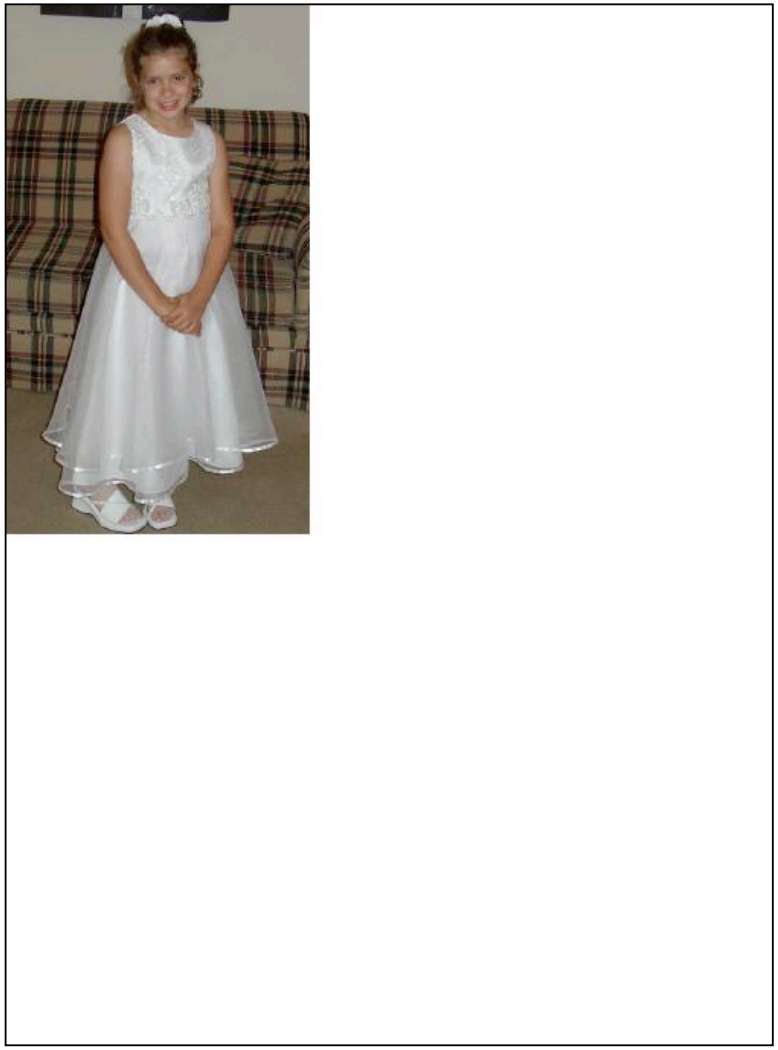
- Invoke the method on this picture object

```
pl.copyKatie();
```

- Repaint the picture

```
pl.repaint();
```

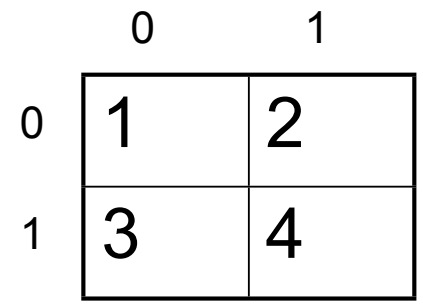
Result of copyKatie Method

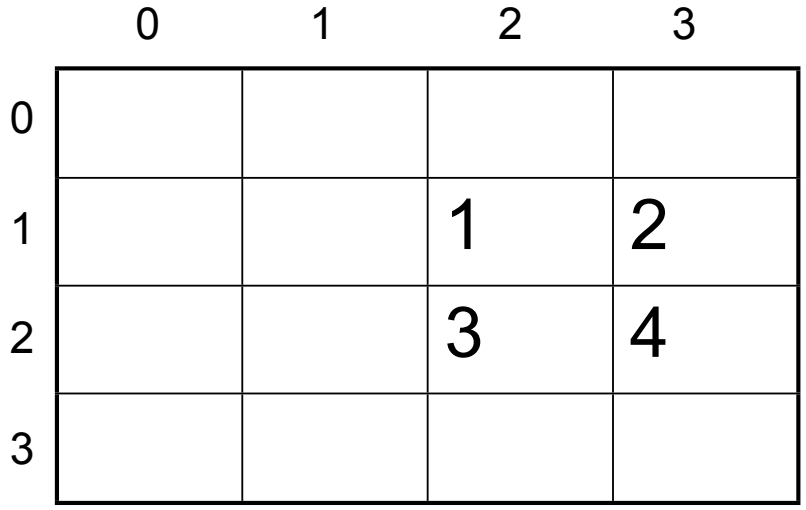


based on slides from Barb Ericson,
Georgia Institute of Technology

Copying Pixels to a New Picture

- What if we want to copy the target to a different location in the source
  - other than 0,0 ?
  - Say startX and startY
- What is an algorithm that will do this?





based on slides from Barb Ericson,
Georgia Institute of Technology

Copy Face Method

```
public void copyKatiesFace()
 {
   String sourceFile =
     FileChooser.getMediaPath("KatieFancy.jpg");
   Picture sourcePicture = new Picture(sourceFile);
   Pixel sourcePixel = null;
   Pixel targetPixel = null;

   // loop through the columns
   for (int sourceX = 70, targetX = 100;
        sourceX < 135; sourceX++, targetX++)
   {
     // loop through the rows
     for (int sourceY = 3, targetY = 100;
          sourceY < 80; sourceY++, targetY++)
     {
```

Copy Face Method - Continued

```
      // set the target pixel color to the source pixel color
      sourcePixel =
          sourcePicture.getPixel(sourceX,sourceY);
      targetPixel = this.getPixel(targetX,targetY);
      targetPixel.setColor(sourcePixel.getColor());
    }
  }
}
```

Testing Copy Katie's Face

- Create a picture object

```
Picture pl = new Picture(FileChooser.getMediaPath(
        "7inX95in.jpg"));
```

- Show the picture

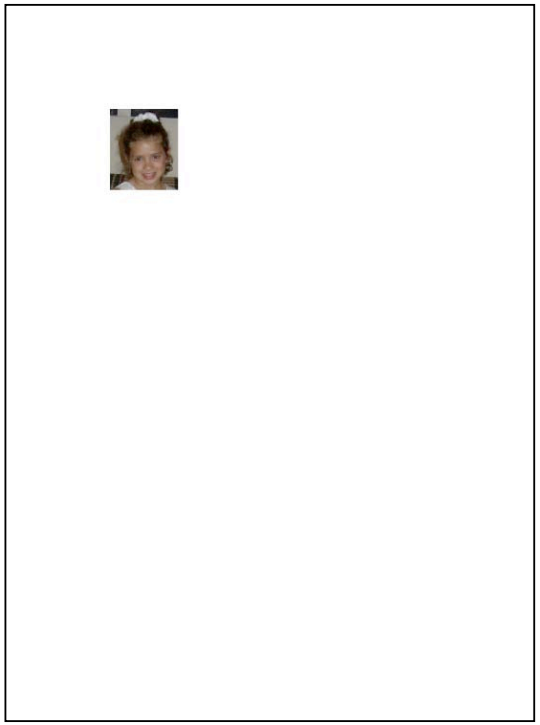
```
pl.show();
```

- Invoke the method

```
pl.copyKatiesFace();
```

- Repaint the picture

```
pl.repaint();
```



based on slides from Barb Ericson,
Georgia Institute of Technology

What makes a Good Method?

- A method should do one and only one thing
  - Accomplish some task
  - The name should tell you what it does
- A method can call other methods to do some of the work
  - Procedural decomposition
- We shouldn't copy code between methods
  - We should make general methods that are reusable
- A method should be in the class that has the data the method is working on

based on slides from Barb Ericson,
Georgia Institute of Technology

Were the last two methods general?

- We specified the file to copy from in the method
  - –Meaning we would need to change the method
  - –or make another method
  - –to copy a different picture

General Copy Algorithm

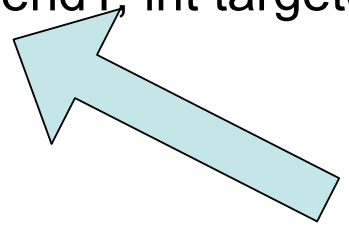
- Create a method that copies pixels from a passed source picture
  - Giving a start x and y and end x and y for the source picture
    - If the start x and y and end x and y cover the entire picture then the whole picture will be copied
    - If the start x and y and end x and y are part of the picture then cropping will occur
  - To the current picture object with a target start x and target start y
    - If the start x and y are 0 then it copies to the upper left corner

General Copy Algorithm

- Loop through the x values between xStart and xEnd
- Loop through the y values between yStart and yEnd
- Get the pixel from the source picture for the current x and y values
  - Get the pixel from the target picture for the targetStartX + x and targetStartY + y values
  - Set the color in the target pixel to the color in the source pixel

General Copy Method

```
public void copy(Picture sourcePicture, int startX, int startY,
                 int endX, int endY, int targetStartX, int targetStartY)
 {
   Pixel sourcePixel = null;
   Pixel targetPixel = null;

   // loop through the x values
   for (int x = startX, tx = targetStartX;
        x < endX;
        x++, tx++)
   {
     // loop through the y values
     for (int y = startY, ty = targetStartY;
          y < endY;
          y++, ty++)
     {
```



What must we assume about the endX and endY ?

General Copy Method - Continued

```
      // copy the source color to the target color
      sourcePixel = sourcePicture.getPixel(x,y);
      targetPixel = this.getPixel(tx,ty);
      targetPixel.setColor(sourcePixel.getColor());
    }
  }
}
```

Scaling

- You can make a picture smaller
  - Faster to download on the web
    - Increment the source x and y by a number larger than 1
      - Don't use all the source pixels in target
- You can make a picture larger
  - Show more detail
    - Copy the same source x and y to more than one target x and y
      - Use source pixels more than once in target

based on slides from Barb Ericson,
Georgia Institute of Technology

Scaling Down a Picture

- passionFlower.jpg is 640pixels wide and 480 pixels high
- If we copy every other pixel we will have a new picture with width (640 / 2 = 320) and height (480 / 2 = 240)

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

0	2
8	10

based on slides from Barb Ericson,
Georgia Institute of Technology

Scaling Down Algorithm

- Create the target picture
- Invoke the method on the target picture
  - Create the source picture
  - Loop with source x starting at 0 and target x starting at 0 as long as < source width
    - Increment the source x by 2 each time through the loop, increment the target x by 1
    - Loop with source y starting at 0 and target y starting at 0 as long as < source height
      - Increment the source y by 2 each time through the loop, increment the target y by 1
        - » Copy the color from the source to target pixel

Scaling Down Method

```
public void copyFlowerSmaller()
 {
   Picture flowerPicture =
     new Picture(
       FileChooser.getMediaPath(“passionFlower.jpg"));
   Pixel sourcePixel = null;
   Pixel targetPixel = null;

   // loop through the columns
   for (int sourceX = 0, targetX=0;
        sourceX < flowerPicture.getWidth();
        sourceX+=2, targetX++)
   {
```

Scaling Down Method - Continued

```
    // loop through the rows
    for (int sourceY=0, targetY=0;
         sourceY < flowerPicture.getHeight();
         sourceY+=2, targetY++)
    {
      sourcePixel =
          flowerPicture.getPixel(sourceX,sourceY);
      targetPixel = this.getPixel(targetX,targetY);
      targetPixel.setColor(sourcePixel.getColor());
    }
  }
}
```

Trying Copy Flower Smaller

- Create a new picture half the size of the original picture (+ 1 if odd size)

```
Picture pl = new Picture(320,240);
```

- Copy the flower to the new picture

```
pl.copyFlowerSmaller();
```

- Show the result

```
pl.show();
```

Scaling Up Exercise

- Write a method copyFlowerBigger to scale up the picture flower1.jpg when you copy it to 640x480.jpg
- Save the result to a file using

```
pictureObj.write("file");
```

Loops Compared

	Java	Python

For each	for (Pixel pxlObj: pixlArray)
	{
	}
		for pxl in getPixel(pic):

For index in range	for (int i=0; i < pixelArray.length; i++)
	{
	}
		for i in range(0, getLength(pic)+1):

For index in range for source & target	for (int sX=0, tX = 0;
	sX < pixelArray.length;
	sX++, tX++)
	{
	}
		tX = 0
		for sX in range(0,getLength(pic)+1):
		....
		tX = tX+1

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 !

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