

# CS 1124 MEDIA COMPUTATION

Lecture 12.2 November 12, 2008

Steve Harrison

# Use a loop!

## Our first picture recipe

```
def decreaseRed(picture):  
    for p in getPixels(picture):  
        value=getRed(p)  
        setRed(p,value*0.5)
```

### Used like this:

```
>>> file=pickAFile() <--- barbara.jpg  
>>> picture=makePicture(file)  
>>> show(picture)  
>>> decreaseRed(picture)  
>>> repaint(picture)
```



# The Picture Class

- To make doing media manipulation easier
  - We have created a set of classes for you to use
    - Picture, ColorChooser, FileChooser, Pixel, etc
- These are not part of the Java language
  - But were created at Georgia Tech
- You should have added the directory that has these classes to your classpath
  - Already done to work with Turtles
  - This tells Java where to find the classes

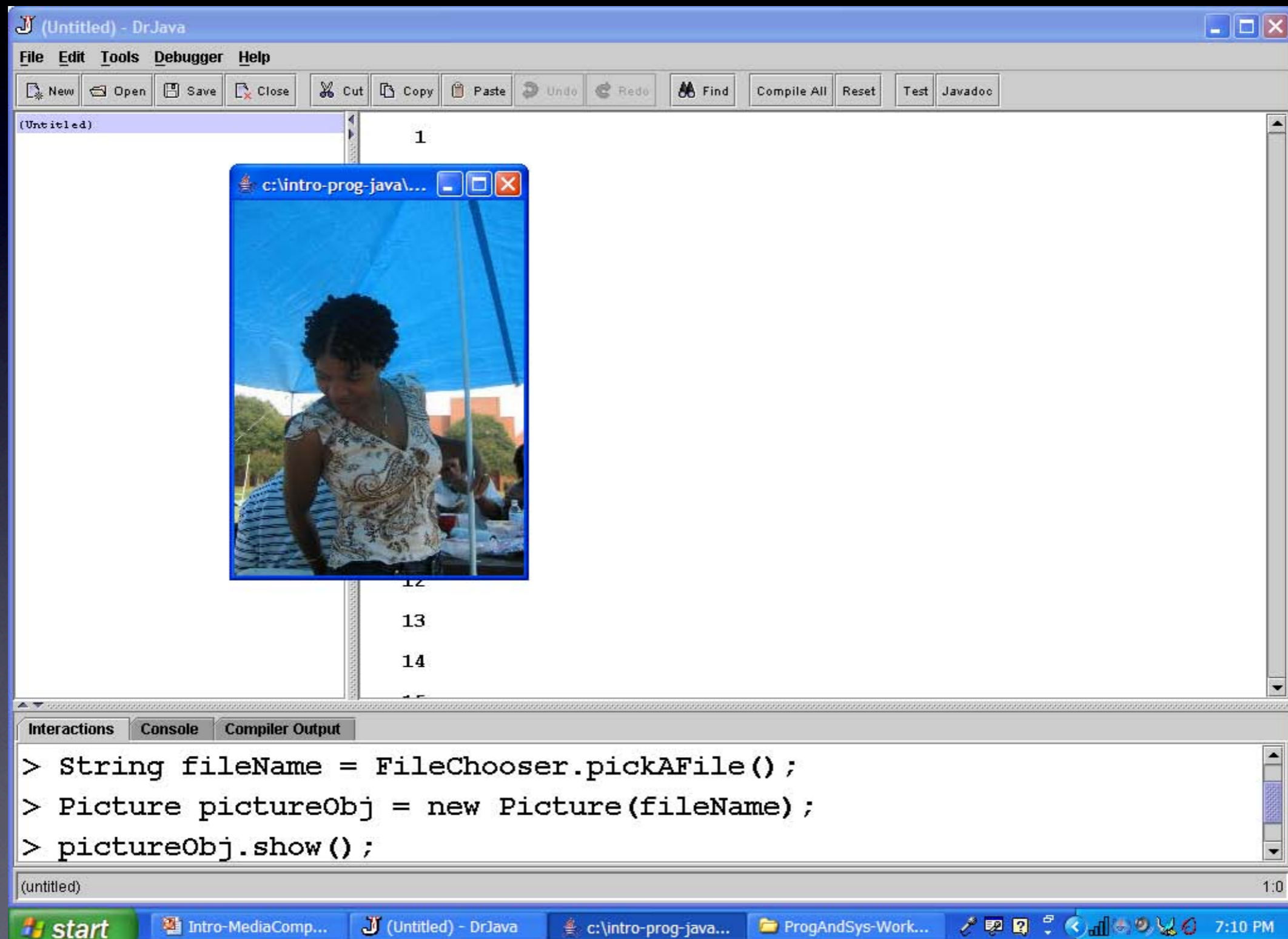
# Creating a Picture Object

- To create a picture object from a file
  - We need the full name of the file
    - We can use `FilePicker.pickAFile()` to get that
      - Class method that returns the full file name as a String
  - We need to ask the `Picture` class to create the picture object
    - Using the data from the specified file
      - `new Picture(fileName)`
  - If we want to see the picture we have created
    - We will ask the picture object to show itself

# Naming each Piece

- First let's pick a file name and save a reference to the resulting String object in a variable called fileName
  - *String fileName = FileChooser.pickAFile();*
- Next, let's create a Picture object and save a reference to it in a variable called pictureObj
  - *Picture pictureObj = new Picture(fileName);*
- Now send the show() message to the picture object
  - *pictureObj.show();*

# Naming Each Part



The screenshot shows the DrJava IDE interface. The main window displays a Java program with line numbers 1 through 15. A small window titled "c:\intro-prog-java\..." is open, showing a photograph of a woman with dark curly hair wearing a patterned top, standing under a blue tent. The IDE's console window at the bottom shows the following code being executed:

```
> String fileName = FileChooser.pickAFile();  
> Picture pictureObj = new Picture(fileName);  
> pictureObj.show();
```

The Windows taskbar at the bottom shows the Start button, several open applications including "Intro-MediaComp...", "DrJava", and "c:\intro-prog-java...", and the system tray with the time 7:10 PM.

# Doing it all at Once

- You can create a picture object
  - by passing it the result of using the FileChooser to pickAFile()
  - and then tell that picture object to show itself
  - All in one line

```
new Picture(FileChooser.pickAFile()).show();
```

But then you don't have a way to refer to the file or picture again.

# Show Picture Exercise

- Try both ways of creating a picture object and showing it.
  - *new Picture(FileChooser.pickAFile()).show();*
  - And do each piece one step at a time naming the result of each method
    - *String fileName = FileChooser.pickAFile();*
    - *System.out.println(fileName);*
    - *Picture picture = new Picture(fileName);*
    - *System.out.println(picture);*
    - *picture.show();*



# Show Picture Exercise

- Try both ways of creating a picture object and showing it.
  - *`new Picture(FileChooser.pickAFile()).show();`*
  - And do each piece one step at a time naming the result of each method
    - *`String fileName = FileChooser.pickAFile();`*
      - `>>> file=pickAFile()`
    - *`System.out.println(fileName);`*
    - *`Picture picture = new Picture(fileName);`*
    - *`System.out.println(picture);`*
    - *`picture.show();`*

# Show Picture Exercise

- Try both ways of creating a picture object and showing it.
  - *`new Picture(FileChooser.pickAFile()).show();`*
  - And do each piece one step at a time naming the result of each method
    - *`String fileName = FileChooser.pickAFile();`*  
`>>> file=pickAFile()`
    - *`System.out.println(fileName);`*  
`>>> print file`
    - *`Picture picture = new Picture(fileName);`*
    - *`System.out.println(picture);`*
    - *`picture.show();`*

# Show Picture Exercise

- Try both ways of creating a picture object and showing it.
  - *`new Picture(FileChooser.pickAFile()).show();`*
  - And do each piece one step at a time naming the result of each method
    - *`String fileName = FileChooser.pickAFile();`*
      - `>>> file=pickAFile()`
    - *`System.out.println(fileName);`*
      - `>>> print file`
    - *`Picture picture = new Picture(fileName);`*
      - `>>> picture=makePicture(file)`
    - *`System.out.println(picture);`*
    - *`picture.show();`*

# Show Picture Exercise

- Try both ways of creating a picture object and showing it.
  - *new Picture(FileChooser.pickAFile()).show();*
  - And do each piece one step at a time naming the result of each method
    - *String fileName = FileChooser.pickAFile();*  
`>>> file=pickAFile()`
    - *System.out.println(fileName);*  
`>>> print file`
    - *Picture picture = new Picture(fileName);*  
`>>> picture=makePicture(file)`
    - *System.out.println(picture);*  
`>>> print picture`
    - *picture.show();*

# Show Picture Exercise

- Try both ways of creating a picture object and showing it.
  - *new Picture(FileChooser.pickAFile()).show();*
  - And do each piece one step at a time naming the result of each method
    - *String fileName = FileChooser.pickAFile();*  
`>>> file=pickAFile()`
    - *System.out.println(fileName);*  
`>>> print file`
    - *Picture picture = new Picture(fileName);*  
`>>> picture=makePicture(file)`
    - *System.out.println(picture);*  
`>>> print picture`
    - *picture.show();*  
`>>> show(picture)`

# What Data does a Picture Object Have?

- A picture object has an array of pixel objects
  - That it read from the JPEG file

- It knows the picture width

*pictureObj.getWidth()*

- It knows the picture height

*pictureObj.getHeight()*

- It knows how to return an array of pixels

*Pixel[] pixelArray = pictureObj.getPixels()*

# Picture Exercise

- Create a picture in DrJava
- get the pictures width, height, and number of pixels

```
String fileName = FileChooser.pickAFile();  
Picture pictureObj = new Picture(fileName);  
int width = pictureObj.getWidth();  
System.out.println("The picture width is " + width);  
int height = pictureObj.getHeight();  
System.out.println("The picture height is " +  
height);  
Pixel[] pixelArray = pictureObj.getPixels();  
System.out.println(pixelArray.length + " pixels");
```

# Pixel Objects

- Each pixel has a red, green, and blue value
  - `getRed()`, `getGreen()`, `getBlue()`
  - `setRed(v)`, `setGreen(v)`, `setBlue(v)`
- Each pixel knows the location it was in the picture object
  - `getX()`, `getY()`
- You can also get and set the color at the pixel
  - `getColor()`, `setColor(color)`



# Color Objects

- There is a class defined in Java that represents color
- The Color class in the package java.awt
- To use the class you must either
  - `import java.awt.Color;`
  - Use the full name `java.awt.Color`
- You can create a color object by giving the red, green, and blue values for it

```
Color colorObj = new Color(255, 10, 125);
```

# Predefined Colors

- The Color class has defined class constants for many colors
- Color.red, Color.green, Color.blue, Color.black, Color.white, Color.yellow, Color.gray, Color.orange, Color.pink, Color.cyan, Color.magenta
- Or you can use all uppercase names
  - Color.RED, Color.BLUE, Color.BLACK, ...



# Getting and Setting Pixel Colors

- To get a pixel's color as a color object

```
Color color1 = pixelObj.getColor();
```

```
int red = color1.getRed();
```

```
int green = color1.getGreen();
```

```
int blue = color1.getBlue();
```

- To set a pixel's color using a new color object

```
red = 20;
```

```
green = 30;
```

```
blue = 100;
```

```
Color color2 = new Color(red,green,blue);
```

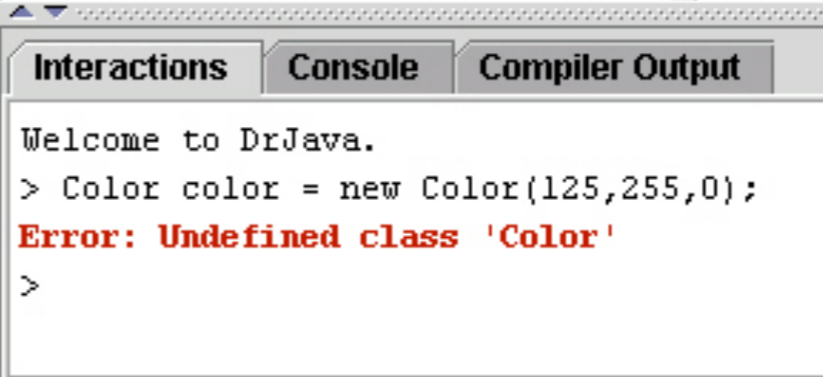
```
pixelObj.setColor(color2);
```

# Using Classes in Packages

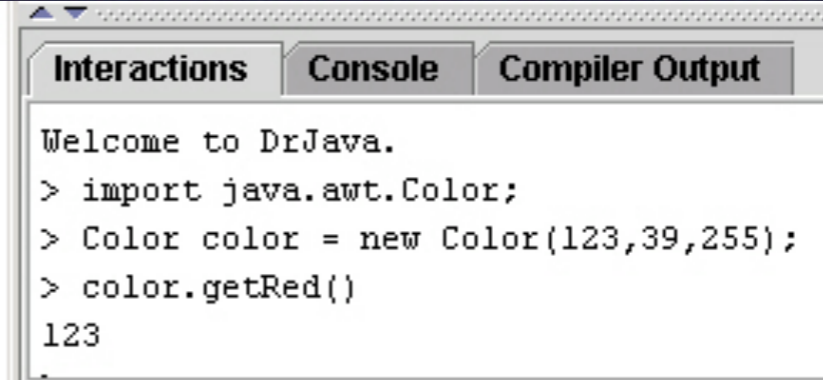
- All classes in the Java language are in packages
  - You can use any class in java.lang
    - System, Math, Object
- For classes in other packages you need to import them
  - `import java.awt.Color;`
  - `Import java.awt.*; //import all classes in this package`
    - To use the short name: Color
- Or use the fully qualified name *packageName.ClassName*
  - `java.awt.Color`

# Undefined Class Error

- If you forget to import a class
  - Yet, you use the short name for the class
  - It won't compile
    - Undefined class error
- Undefined class errors mean
  - You need to import the class
  - Or you misspelled the class
  - Or used the wrong case



```
Interactions Console Compiler Output
Welcome to DrJava.
> Color color = new Color(125,255,0);
Error: Undefined class 'Color'
>
```



```
Interactions Console Compiler Output
Welcome to DrJava.
> import java.awt.Color;
> Color color = new Color(123,39,255);
> color.getRed()
123
```

# Pixel Exercise

- In DrJava
  - Pick a file and create a picture object
  - Get the array of pixels from the picture object
  - Get the 1<sup>st</sup> pixel from the array of pixels
    - Pixel pixelObj = pixelArray[0]; // 0 is first one
  - Get the red, green, and blue value for this pixel
  - Get the x and y location of this pixel
  - Get the color of this pixel
    - Get the red, green, and blue values of the color

# Changing Pixel Colors

- There are two ways to change the color of a pixel in a picture
  - Set the red, green, and blue values individually
    - `pixelObj.setRed(value),`
    - `pixelObj.setGreen(value),`
    - `pixelObj.setBlue(value),`
  - Or set the color
    - `pixelObj.setColor(colorObj)`
- But, you won't **see** any change in the picture
  - Until you ask it to repaint: `pictureObj.repaint();`

# Changing a Color

- The Color class has methods for making a color object
  - Lighter
    - `colorObj.brighter();`
  - Darker
    - `colorObj.darker();`

- Example

```
> import java.awt.Color;
> Color testColor = new Color(168,131,105);
> System.out.println(testColor);
java.awt.Color[r=168,g=131,b=105]
> testColor = testColor.darker();
> System.out.println(testColor);
java.awt.Color[r=117,g=91,b=73]
> testColor = testColor.brighter();
> System.out.println(testColor);
java.awt.Color[r=167,g=130,b=104]
```

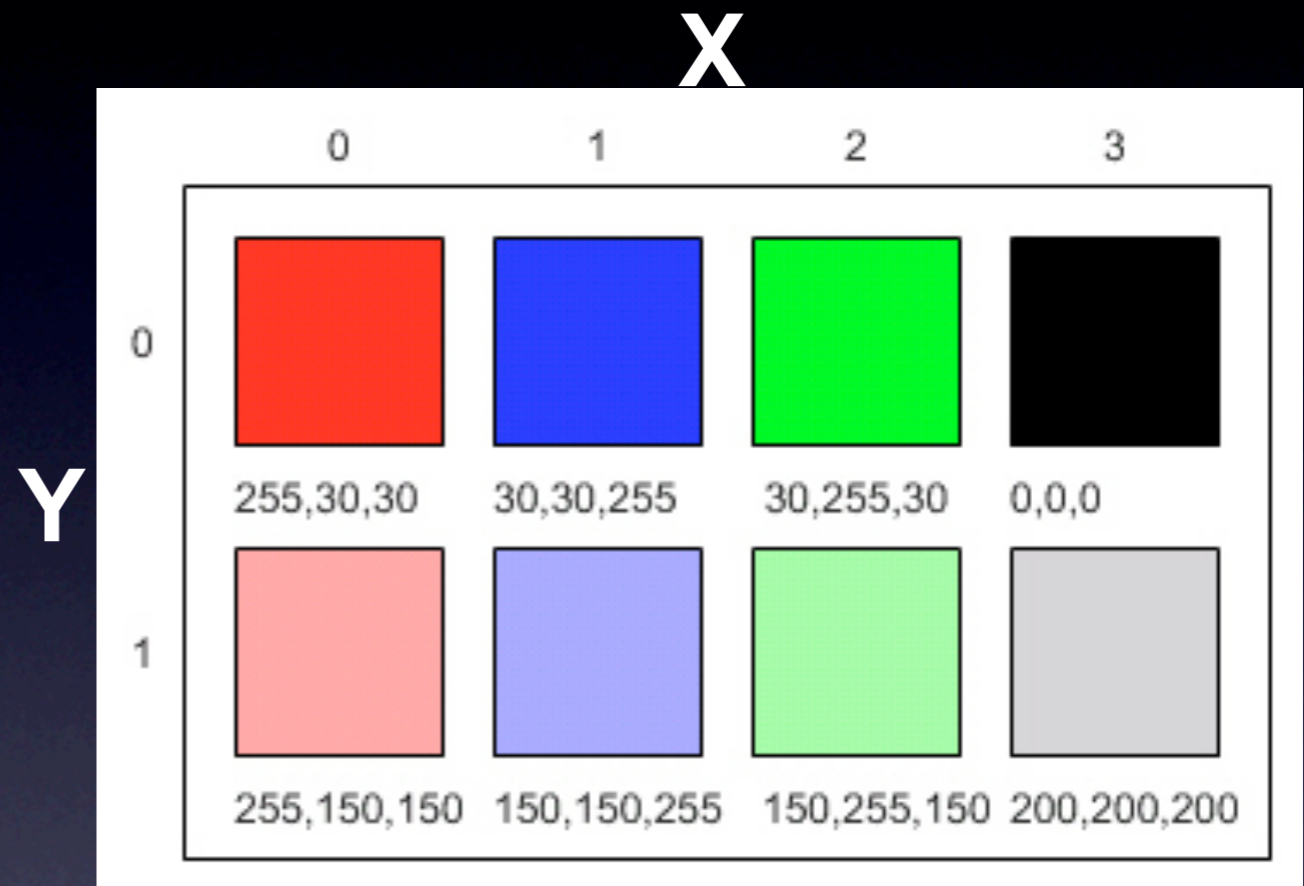


# Rounding Errors

- Notice that when you made the color darker and then lighter the resulting color was slightly off of the original
  - The change is calculated in floating point
  - The result is stored in integer form
  - The decimal part is lost
- Rounding errors also occur because of the limited storage for floating point numbers
  - We can't store all the digits in some numbers

# Pictures are 2-D Arrays

- They have columns and rows (x and y)
- You can get a pixel at a particular x and y location
- Pixel pixelObj = picture.getPixel(x,y);
- The columns and rows
  - start with index 0
  - end with num - 1



# Changing a Picture

## Exercise

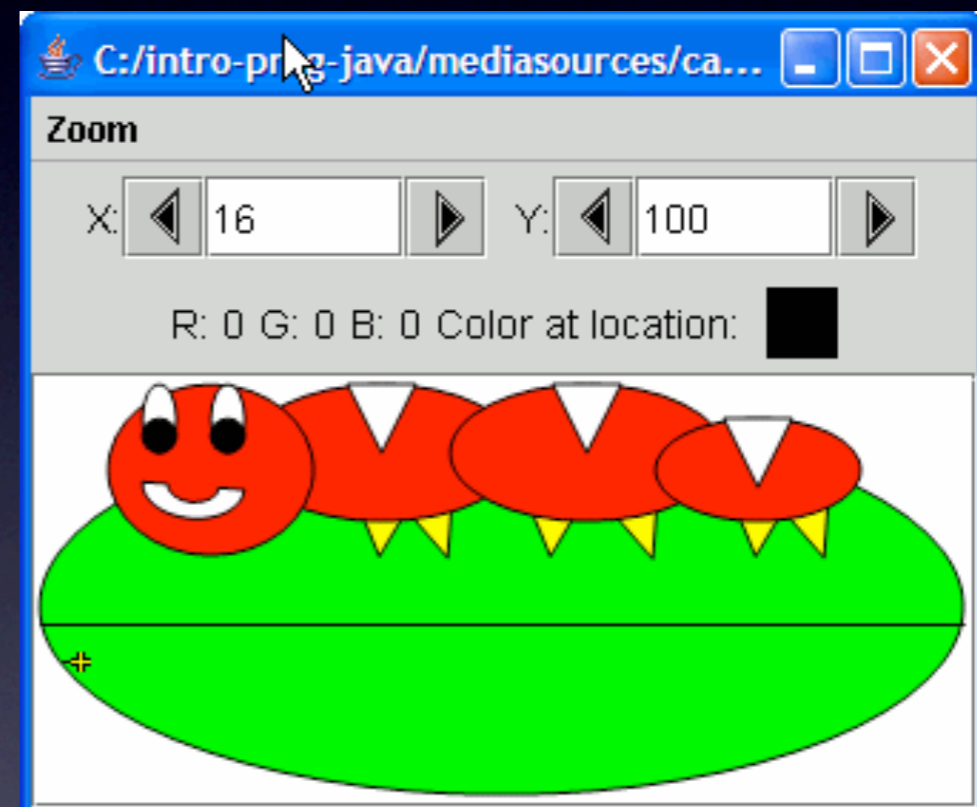
```
> import java.awt.Color;
> String fileName = "C:/intro-prog-java/mediasources/caterpillar.jpg";
> Picture pictureObj = new Picture(fileName);
> pictureObj.show();
> pictureObj.getPixel(10,100).setColor(Color.black);
> pictureObj.getPixel(11,100).setColor(Color.black);
> pictureObj.getPixel(12,100).setColor(Color.black);
> pictureObj.getPixel(13,100).setColor(Color.black);
> pictureObj.getPixel(14,100).setColor(Color.black);
> pictureObj.getPixel(15,100).setColor(Color.black);
> pictureObj.getPixel(16,100).setColor(Color.black);
> pictureObj.getPixel(17,100).setColor(Color.black);
> pictureObj.getPixel(18,100).setColor(Color.black);
> pictureObj.getPixel(19,100).setColor(Color.black);
> pictureObj.repaint();
```

# How do we Know if it Worked?

- A very important part of programming is testing the result
- Just because code compiles and runs without error doesn't mean it is correct
- There could be an error in the logic
- It could fail under certain conditions
- It could even return the correct answer but for the wrong reason

# The Picture Explorer

- Tool that creates a copy of the current picture and lets you explore it
  - See the color, x, and y values at the cursor
- To use the tool on a picture object
  - `pictureObj.explore();`
- Use it to see if the colors have changed



# Changing the Red in a Picture

- One way to change a picture is to reduce the amount of red in it
  - What if we want to decrease it by half?
    - If we have a value of 200 what should the new value be?
    - How do we reduce any value by half?
  - What if we want to increase it by 25%?
    - If we have a value of 100 what should the new value be?
    - How do we increase any value by 25%?

# We Need a Loop (Iteration)

- A way to execute a series of statements in the body of the loop
- With something changing each time the statements are executed
  - Different pixel to change
- And some way to tell when we are done with the repetition
  - Some test to see if the loop should stop
    - Done with all of the items in an array

# For-each Loop

- In Java if we want to do something to each item in an array

- We can use the for-each loop

- `for (Type : variableName arrayName)`

```
{
```

```
    // body of the loop
```

```
}
```

- Which means for each element in the array do the statements in the body of the loop



# Method to Decrease Red

- Loop through all the pixels in an array of Pixel objects and change the red in each

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

    // loop through all the pixels in the array
    for (Pixel pixelObj : pixelArray)
    {
        // get the red value
        value = pixelObj.getRed();

        // decrease the red value by 50% (1/2)
        value = value / 2;

        // set the red value of the current pixel to the new value
        pixelObj.setRed(value);
    }
}
```

The body of the loop is in { }

```
def decreaseRed(picture):
    for p in getPixels(picture):
        value=getRed(p)
        setRed(p,value*0.5)
```

# Objects send Messages

- Objects don't "tell" each other what to do
  - They "ask" each other to do things
- Objects can refuse to do what they are asked
  - The object must protect it's data
    - Not let it get into an incorrect state
    - A bank account object shouldn't let you withdraw more money that you have in the account

# Creating a Method

- We can name a block of Java statements and then execute them again
  - By declaring a method in a class
- The syntax for declaring a method is
  - *visibility returnType name(parameterList)*
    - Visibility determines access
      - Usually public or private
      - The return type is the type of thing returned
      - If nothing is returned use the keyword *void*
    - Name the method starting with a lowercase word and uppercasing the first letter of each additional word

# Example Method

```
public void drawSquare()
{
    this.turnRight();
    this.forward(30);
    this.turnRight();
    this.forward(30);
    this.turnRight();
    this.forward(30);
    this.turnRight();
    this.forward(30);
}
```

- The visibility is **public**
- The keyword **void** means this method doesn't return a value
- The method name is **drawSquare**
- There are no parameters
  - Notice that the parentheses are still required
- The keyword **this** means the object this method was invoked on

# Adding a Method to a Class

**1. Open file Turtle.java**

```
57 ////////////////////////////////////////////////// methods //////////////////////////////////////
58 public void drawSquare(int width)
59 {
60     this.turnRight ();
61     this.forward (width) ;
62     this.turnRight ();
63     this.forward (width) ;
64     this.turnRight ();
65     this.forward (width) ;
66     this.turnRight ();
67     this.forward (width) ;
68 }
69
70 } // end of class Turtle, put all new methods before this line
```

**2. Type the method before the last } // end**

**3. Compile**

Interactions Console Compiler Output

```
> 3 + 7
10
>
```

C:\book\cdrom\intro-prog-java\bookClasses\Turtle.java 69:2

# Compile Errors

Clicking on the error takes you to the code and highlights it.

Case matters in Java!  
turnright isn't the same as turnRight

```
53 // let the parent constructor handle it
54 super (p) ;
55 }
56
57 ////////////////////////////////////////////////// methods //////////////////////////////////////
58 public void drawSquare (int width)
59 {
60     this.turnright () ;
61     this.Forward (width) ;
62     this.turnright () ;
63     this.Forward (width) ;
64     this.turnright () ;
65     this.Forward (width) ;
66     this.turnright () ;
67     this.Forward (width) ;
68 }
```

Interactions Console **Compiler Output**

```
Error: cannot resolve symbol
symbol : method turnright ()
location: class Turtle
```

Compiler  
javac 1.4.1+ (user)

C:\bookcdrom\intro-prog-java\bookClasses\Turtle.java 60:4

start 2 Microsof... Turtle - DrJ... Preferences World cdrom 3:04 PM

# Try the New Method

- Compiling resets the interactions pane
  - Clearing all variables
    - But you can still use the up arrow to pull up previous statements
  - You will need to create a world and turtle again

```
World world1 = new World();
```

```
Turtle turtle1 = new Turtle(world1);
```

```
turtle1.forward(50);
```

```
turtle1.drawSquare();
```

```
turtle1.turn(30);
```

## Saving the Interactions History in DrJava

- You can save the interactions history into a script
  - And optionally edit it first before you save it
  - Click on Tools then on Save Interactions History
- And then later load and execute the statements in the script
  - Click on Tools and Load Interactions History as Script
    - Use the next button to see the next statement and click on the execute button to execute it



# Better Method to Draw a Square

- A method to draw a square

```
public void drawSquare()
{
    int width = 30;
    this.turnRight();
    this.forward(width);
    this.turnRight();
    this.forward(width);
    this.turnRight();
    this.forward(width);
    this.turnRight();
    this.forward(width);
}
```

- We added a local variable for the width
  - Only known inside the method
- This makes it easier to change the width of the square
- But, we still have to recompile to draw a different size square

# Testing the Better Method

- Type the following in the interactions pane

```
World world1 = new World();
```

```
Turtle turtle1 = new Turtle(world1);
```

```
turtle1.forward(50);
```

```
turtle1.drawSquare();
```

```
turtle1.turn(30);
```

```
turtle1.drawSquare();
```

- Or use the saved script if you saved the last interactions history

# Passing a Parameter

```
public void drawSquare(int width)
```

```
{
```

```
    this.turnRight();
```

```
    this.forward(width);
```

```
    this.turnRight();
```

```
    this.forward(width);
```

```
    this.turnRight();
```

```
    this.forward(width);
```

```
    this.turnRight();
```

```
    this.forward(width);
```

thing passed and a name to use to refer to the value in the method

- The type of this parameter is **int**
- The name is **width**
- Values are passed by making a copy of the passed value

# Testing with a Parameter

- Type the following in the interactions pane

```
World world1 = new World();
```

```
Turtle turtle1 = new Turtle(world1);
```

```
turtle1.forward(50);
```

```
turtle1.drawSquare(30);
```

```
turtle1.turn(30);
```

# How Does That Work?

- When you ask turtle1 to drawSquare(30)  
turtle1.drawSquare(30);
  - It will ask the Turtle Class if it has a method drawSquare that takes an int value
    - And start executing that method
    - The parameter width will have the value of 30 during the executing of the method
    - The *this* keyword refers to turtle1
- When you ask turtle1 to drawSquare(50)

# Challenges

- Create a method for drawing a rectangle
  - Pass the width and height
- Create a method for drawing an equilateral triangle
  - all sides have the same length
  - Pass in the length
- Create a method for drawing a diamond

# Testing decreaseRed

- Add decreaseRed to Picture.java
  - Before the last }
- Compile Picture.java
  - Click on Compile All Button
- Type the following in the interactions pane
  - > `String fileName = FileChooser.pickAFile();`
  - > `Picture pictObj = new Picture(fileName);`
  - > `pictObj.explore();`
  - > `pictObj.increaseRed();`
  - > `pictObj.explore();`

# How This Works

- First we have the method declaration

```
public void decreaseRed()
```

- This is a public method that doesn't return anything and the name is decreaseRed. This method does not take any parameters since there is nothing in the ()
- Next we start the body of the method
  - Inside of the body of the method are the statements to be executed when the method is called (executed)
  - Next we declare an array of pixels and get them from the current Picture object
- Next we declare a primitive variable to hold the current red value at the pixel

```
Pixel[] pixelArray = this.getPixels();
```

```
int value = 0;
```



# How This Works - Cont

- Next start the for-each loop

```
for (Pixel pixelObj : pixelArray)
```

```
{
```

- Each time through the loop set the variable pixelObj to refer to the next Pixel object in the array of Pixel objects called pixelArray. Execute the statements in the body of the loop.

- Get the red value from the pixelObj

```
value = pixelObj.getRed();
```

- Decrease the red by 1/2

```
value = value / 2;
```

- Set the red value at the pixelObj to the new value

```
pixelObj.setRed(value);
```

- End the for-each loop body and the body of the method

```
}
```

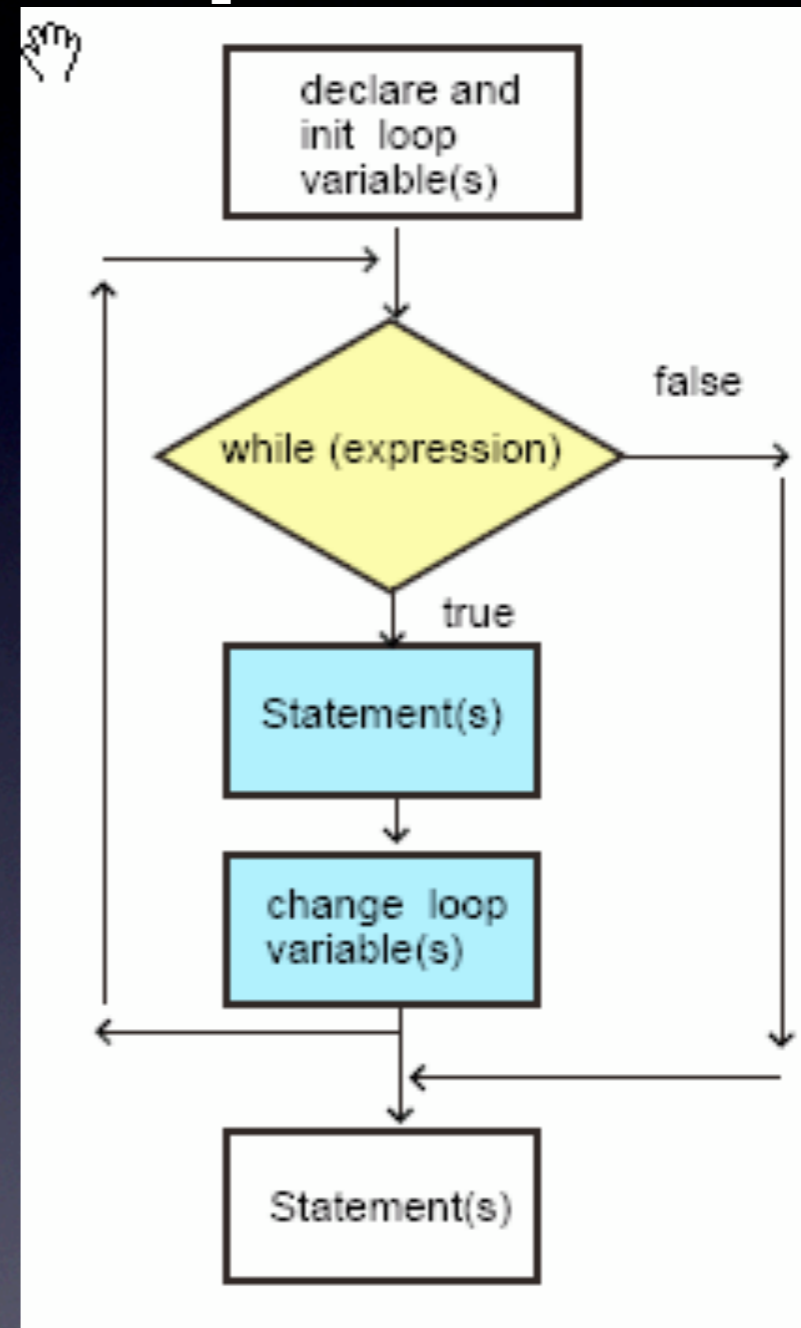
```
}
```

# Exercise

- Write a method `increaseRed()` to loop through all the pixels in a picture and double the red values
  - Multiply by 2
- To try this method do the following:
  - `String fileName = FileChooser.pickAFile();`
  - `Picture pictObj = new Picture(fileName);`
  - `pictObj.explore();`
  - `pictObj.increaseRed();`
  - `pictObj.explore();`

# 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 than 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



# 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
```

# 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);
```

# Project

- Project 3 - Signs in Pictures -- in Java

# Coming Attractions

- Monday:
  - quiz (on [www.javabat.com](http://www.javabat.com))
- next Friday:
  - Project 8 - Java picture due