Manipulating a Picture

- To manipulate a picture we need to manipulate the pixels that make it up.
  - Change the red, green, or blue values at the pixel
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
  
  \[
  \text{pictureObj.getWidth()}
  \]
- It knows the picture height
  
  \[
  \text{pictureObj.getHeight()}
  \]
- It knows how to return an array of pixels
  
  \[
  \text{Pixel[]} \ pixelArray = \text{pictureObj.getPixels()}
  \]

Representing Pixel data in an array.

- We’re used to thinking of the images as a two-dimensional array/grid/matrix of pixels.
- We can also represent it as a one-dimensional array.
Representing Pixel data in an array.

- The grid of pixels represented as a (one-dimensional) array.

---

**Picture Exercise**

- Create a picture in DrJava
  - get the picture’s width, height, and pixels

```java
String fileName = FileChooser.pickAFile();
Picture pictureObj = new Picture(fileName);
pictureObj.show();
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, which we can get or set:

  - `valueR = pixelObj.getRed()`
  - `valueG = pixelObj.getGreen()`
  - `valueB = pixelObj.getBlue()`

  - `pixelObj.setRed(valueR)`
  - `pixelObj.setGreen(valueG)`
  - `pixelObj.setBlue(valueB)`
Pixel Objects

- Each pixel knows the location where it belongs in the picture object:
  
  \[ \text{valueX} = \text{pixelObj.getX()} \]
  
  \[ \text{valueY} = \text{pixelObj.getY()} \]

Pixel Objects

- You can also get and set the color of the pixel:
  
  \[ \text{colorObj} = \text{pixelObj.getColor()} \]
  
  \[ \text{pixelObj.setColor(colorObj)} \]

Pixel Exercise

- In DrJava
  
  - Pick a file and create a picture object.
  
  - Get the array of pixels from the picture object.
  
  - Get the 1st pixel from the array of pixels
    
    ```java
    \text{Pixel} \text{ 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.

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
    
    ```java
    \text{import java.awt.Color;}
    ```
    
    or
    
    ```java
    \text{java.awt.Color;}
    ```
Creating Colors

• You can create a color object by giving the red, green, and blue values:

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

Predefined Colors

• The Color class has defined class constants for many colors:

```java
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

```java
Color.RED, Color.BLUE, Color.BLACK, ...
```

Getting and Setting Pixel Colors

• To get a pixel’s color as a color object

```java
import java.awt.Color;
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

```java
import java.awt.Color;
Color color2 = new Color(red, green, blue);
pixelObj.setColor(color2);
```

Using Classes in Packages

• All classes in the Java language are grouped into packages.

• You can already use any class in java.lang:

  - System.
  - Math.
  - Object

because they are always available in the Java language.

• That is, they are so frequently used, the decision was made to have them automatically available.
Using Classes in Packages

• For classes in other packages you need to import them:
  
  – `import java.awt.Color;`

  or

  – `import java.awt.*;` //import all classes in java.awt package
to use the short name: Color

• Or use the fully qualified name:
  
  – `packageName.ClassName`
  – `java.awt.Color`

Using Classes in Packages

• Some people prefer to import each class by naming each one in an import statement.
  
  – This is helpful for other people who may not know what package each class is in.

• However, if you are using several classes from the same package you can use the '*' to import all classes in the same package.
  
  – This will not extend to sub-packages.

Undefined Class Error

• If you forget to import a class
  – but 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

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 “redraw” it:
  
  – Until you ask it to repaint:
    `pictureObj.repaint();`

  – Or show it again:
    `pictureObj.show();`
Changing a Color

- The Color class has methods for making a color object...
  - Lighter
    - `colorObj.brighter();`
  - Darker
    - `colorObj.darker();`
- Example:
  ```java
  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 make the color darker and then lighter the resulting color is slightly different than 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 necessary to represent some numbers.

Pictures are 2-Dimensional Arrays

- They have columns and rows (\(x\) and \(y\)).
- You can get a pixel at a particular \(x\) and \(y\) location
  ```
  Pixel pixelObj = pictureObj.getPixel(x,y);
  ```
- The columns and rows
  - start with index 0
  - end with \(num-1\), where \(num\) is the size of the row or column.

Changing a Picture Exercise

```java
import java.awt.Color;
String fileName = ".../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();
```
We Need a Loop (Iteration)

- The loop will help us execute a repeated series of statements without having to write each one out.

We Need a Loop (Iteration)

- A loop is a way to execute a series of statements.
- Two key parts of the loop:
  - Something changing each time the statements are executed.
    - For instance, a different index for the pixel to change.
  - And some way to tell when we are done with the repetition.
    - Some test to see if the loop should stop.

Loop Exercise

- Ask a person to clap 12 times
  - How does she know when to stop?
  - What changes each time she claps?
- If you are following a recipe that asks you to stir the ingredients 50 times how would you do this?
- What if you were trying to break a sit-up record
  - How would you know if you did break it?

Loops often need Counters

- If you want to do something $x$ times you often need a counter
  - The counter is also called an *index*.
    - It starts at 0
    - And you add 1 to it each time you do the task you are repeating.
    - When the counter reaches the number you are trying to do you stop the loop.
    - What is the value of the counter the last time the statements of the loop are executed?
Manipulating a Picture

• To manipulate a picture we need to manipulate the pixels that make it up.
  – Change the red, green, or blue values at the pixel

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?

• One way to change a picture is to increase the amount of red in it
  – 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%?

• So, to change the red in an image, we will need to alter the Red value in all of the pixels.

• How do we go about this?

• First, think about how you would change the value of Red for a single pixel.
Changing a Pixel

- What change do we need to reduce the red in a single pixel?
  - Get the pixel
  - Get the red value of the pixel
  - Change the red value to half the original value (value / 2)
  - Put the new red value in the pixel

Changing all the Pixels in a Picture

- Next we need to figure how to do this for all the pixels in an image.
- There are $329 \times 150 = 49,350$ pixels in the caterpillar picture
- Do we really want to write the code to change each one of these individually?
- Clearly, this is a job for a loop.

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 the stop value, 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.

- The picture is a flowchart that shows the order the statements are executed in a program.
- A while loop allows you to repeat a series of statements until an expression is false.
Total the Numbers from 1 to 100

• What if you want to add all the numbers from 1 to 100?
  – You will need something to hold the total.
    • What type should it be?
    • What value should it start out with?
  – You will need something that counts from 1 to 100.
    • And adds each value to the total as it’s counting.
    • Stop when you get to 100.
    • What type should it be?
    • What value should it start with?

While Loop Syntax

• Adding up the numbers from 1 to 100:
  ```java
  int total = 0;
  int counter = 1;
  while (counter <= 100)
  {
    total = total + counter;
    counter = counter + 1;
  }
  System.out.println(total);
  ```

Adding a Main Method

• We have been typing our statements in the interactions pane in DrJava
  – To try out Java code and to try methods
• Most development environments make you write a main method to start execution.
  – DrJava allows this too.
• Having a main() method allows you to run your program on its own, without DrJava.
  – Similar to how other programs like Excel, Firefox, etc. run on their own.

Adding a Main Method

• Each class can have a main method declared as follows:
  ```java
  public static void main(String [] args)
  {
  } // main()
  ```
  with the following properties:
  • It is public so that it can be called by other classes.
  • It is static because no object of the class exists when it is executed.
  • It doesn’t return anything so the return type is void.
  • You can pass several arguments to the main method and these are put in an array of strings.
Main Method

- Make our example a free-standing Java program (WhileExample.java).

```java
public class WhileExample {
    public static void main(String[] args) {
        int total = 0;
        int counter = 1;
        while (counter <= 100) {
            total = total + counter;
            counter = counter + 1;
        } // while
        System.out.println(total);
    } // main
} // class
```

Execute the Main Method

- In DrJava you can run the main method in the class that is displayed in the definitions pane
  - By clicking on Tools then Run Document’s Main Method (or press key F2)
- It will do
  ```java
  Student
  ```
  in the interactions pane
  - Which executes the main in the Student class

Summing from 1 to \( n \)

- Our free-standing Java program (WhileExample.java).

```java
public class WhileExample {
    public static void main(String[] args) {
        int total = 0;
        int counter = 1;
        while (counter <= 100) {
            total = total + counter;
            counter = counter + 1;
        } // while
        System.out.println(total);
    } // main
} // class
```

- By the way, this can be done more efficiently.
- As we saw the loop sums up to 5050.
- The sum of the numbers from 1 to 100 is \([(100)(101)]/2\).
- More generally, the sum of integers from 1 to \( n \) is \([n(n+1)]/2\).
- Try it.