AP® Computer Science A
Picture Lab

The AP Program wishes to acknowledge and thank Barbara Ericson of the Georgia Institute of Technology, who developed this lab and the accompanying documentation.

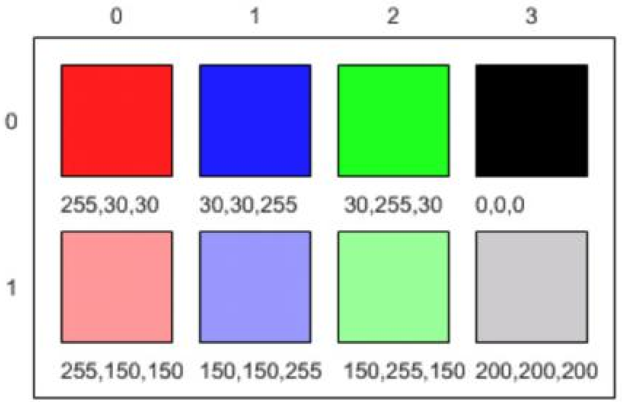
CollegeBoard

Introduction

- In this lab you will be writing methods that modify digital pictures.
- In writing these methods you will learn how to traverse a two-dimensional array of integers or objects. You will also be introduced to nested loops, binary numbers, interfaces, and inheritance.
- Activities:
- You will be working through a set of activities. These activities will help you learn about how:
  - ✓ digital pictures are represented on a computer;
  - ✓ the binary number system is used to represent values;
  - ✓ to create colors using light;
  - ✓ Java handles two-dimensional arrays;
  - ✓ data from a picture is stored; and
  - ✓ to modify a digital picture.

A1: Introduction to digital pictures and color

- If you look at an advertisement for a digital camera, it will tell you how many megapixels the camera can record. What is a megapixel?
- A digital camera has sensors that record color at millions of points arranged in rows and columns.Each point is a pixel or picture (abbreviated pix) element.A megapixel is one million pixels
- How is the color of a pixel recorded?
    - ✓ It can be represented using the RGB (Red, Green, Blue) color model, which stores values for red, green, and blue, each ranging from 0 to 255



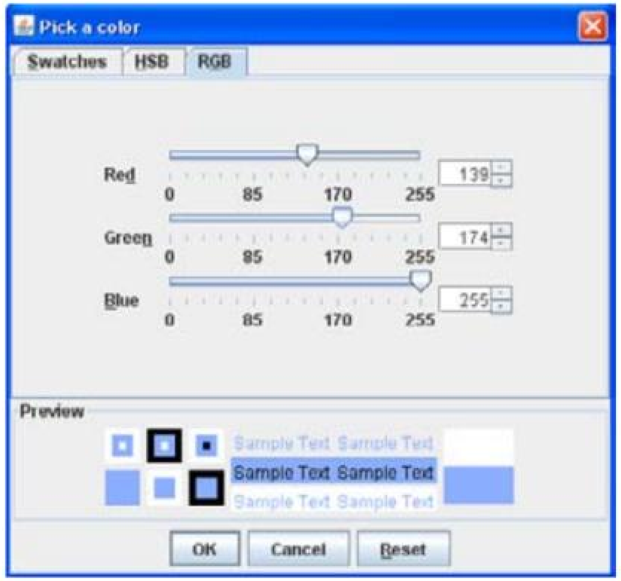
	0	1	2	3
0	255,30,30	30,30,255	30,255,30	0,0,0
1	255,150,150	150,150,255	150,255,150	200,200,200

A1: Introduction to digital pictures and color

- How does the computer represent the values from 0 to 255?
- Computers use binary numbers, which usethe digits 0 and 1 and powers of 2 to represent values using groups of bits.
- A bit is a binary digit, which can be either 0 or 1. **A group of 8 bits is called a byte.**
- The binary number 110 means 0 ones (20) plus 1 two (21) plus 1 four (22), for a total of 6.
- Questions:
- 1. How many bits does it take to represent the values from 0 to 255?
- 2. How many bytes does it take to represent a color in the RGB color model?
- 3. How many pixels are in a picture that is 640 pixels wide and 480 pixels high?

A2: Picking a color

- Run the `main` method in `ColorChooser.java`. This will pop up a window asking you to pick a color. Click on the RGB tab and move the sliders to make different colors.



Red 139

Green 174

Blue 255

- When you click the OK button, the red, green, and blue values for the color you picked will be displayed as shown below. The Color class has a `toString` method that displays the class name followed by the red, green, and blue values. The `toString` method is automatically called when you print an object.

```
java.awt.Color[r=139,g=174,b=255]
```

A2: Picking a color

- Java represents color using the `java.awt.Color` class. Java groups related classes into packages. The `awt` stands for Abstract Windowing Toolkit,which is the package that contains the original Graphical User Interface (GUI) classes developed for Java.
- The `Picture` class contains the following import statement.

```
import java.awt.Color;
```

- Questions
- 1. How can you make pink?
- 2. How can you make yellow?
- 3. How can you make purple?
- 4. How can you make white?
- 5. How can you make dark gray?

A3 Exploring a picture

- Run the `main` method in `PictureExplorer.java`. This will load a picture of a beach from a file, make a copy of that picture in memory, and show it in the explorer tool .

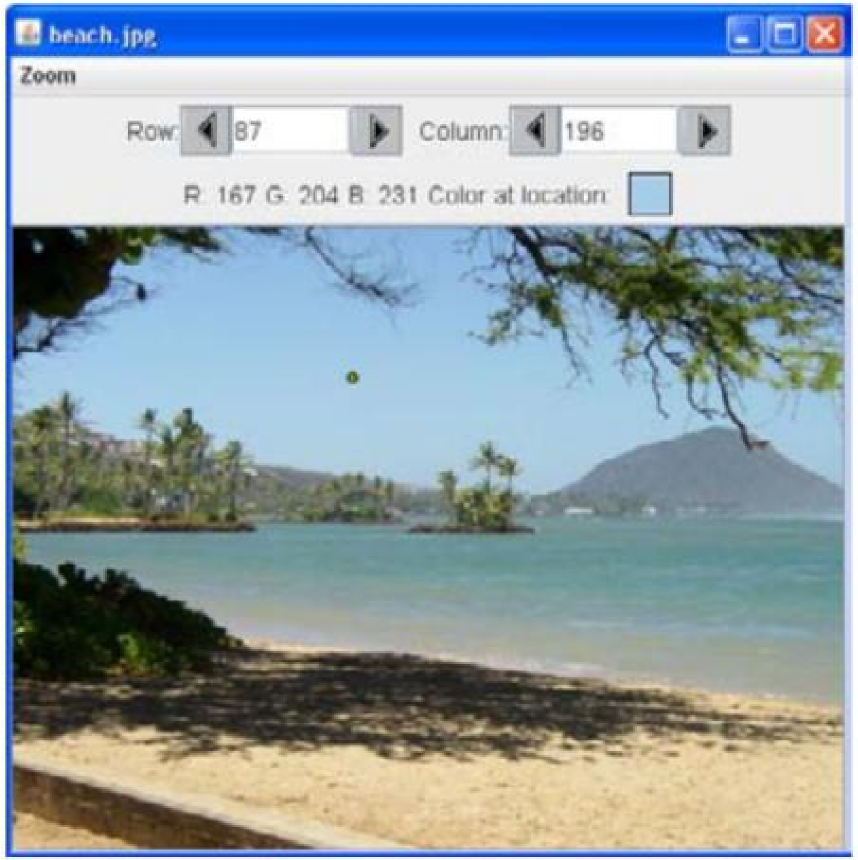
You can use the explorer tool to explore the pixels in a picture.

✓ **Click any location** (pixel) in the picture and it will display the row index, column index, and red, green, and blue values for that location.

✓ The location will be **highlighted with yellow crosshairs**.

✓ You can click on the **arrow keys or even type in values** and hit the enter button to update the display.

✓ You can also use the menu to change the **zoom** level.



Row: 87 Column: 196

R: 167 G: 204 B: 231 Color at location:

A3 Exploring a picture

- Questions
- 1. What is the row index for the top left corner of the picture?
- 2. What is the column index for the top left corner of the picture?
- 3. The width of this picture is 640. What is the right most column index?
- 4. The height of this picture is 480. What is the bottom most row index?
- 5. Does the row index increase from left to right or top to bottom?
- 6. Does the column index increase from left to right or top to bottom?
- 7. Set the zoom to 500%. Can you see squares of color? This is called pixelation. Pixelation means displaying a picture so magnified that the individual pixels look like small squares.

A3 Exploring a picture

- Creating and exploring other pictures: Here is the `main` method in the class `PictureExplorer`. Every class in Java can have a main method, and it is where execution starts when you execute the command java ClassName.

```
public static void main( String args[])

{

Picture pix = new Picture("beach.jpg");

pix.explore();

}
```

- Exercises
- 1. Modify the `main` method in the `PictureExplorer` class to create and explore a different picture from the `images` folder.
- 2. Add a picture to the `images` folder and then create and explore that picture in the `main` method. If the picture is very large (for instance, one from a digital camera), you can scale it using the `scale` method in the `Picture` class. For example, you can make a new picture (“`smallMyPicture.jpg`” in the images folder) one-fourth the size of the original (“`myPicture.jpg`”) using:

```
Picture p = new Picture("myPicture.jpg");

Picture smallP = p.scale(0.25,0.25);

smallP.write("smallMyPicture.jpg");
```

A4 Two-dimensional arrays in Java

- Java actually uses arrays of arrays to represent 2D arrays. This means that each element in the outer array is a reference to another array. The data can be in either row-major or column-major order (Figure 4).

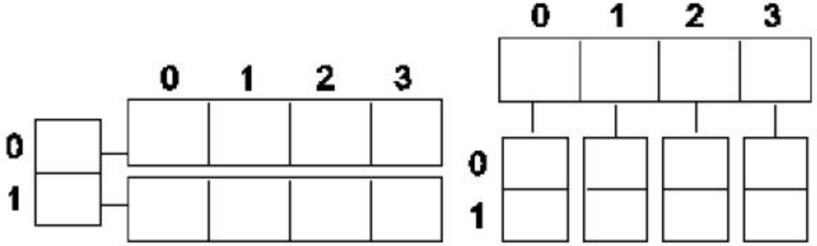
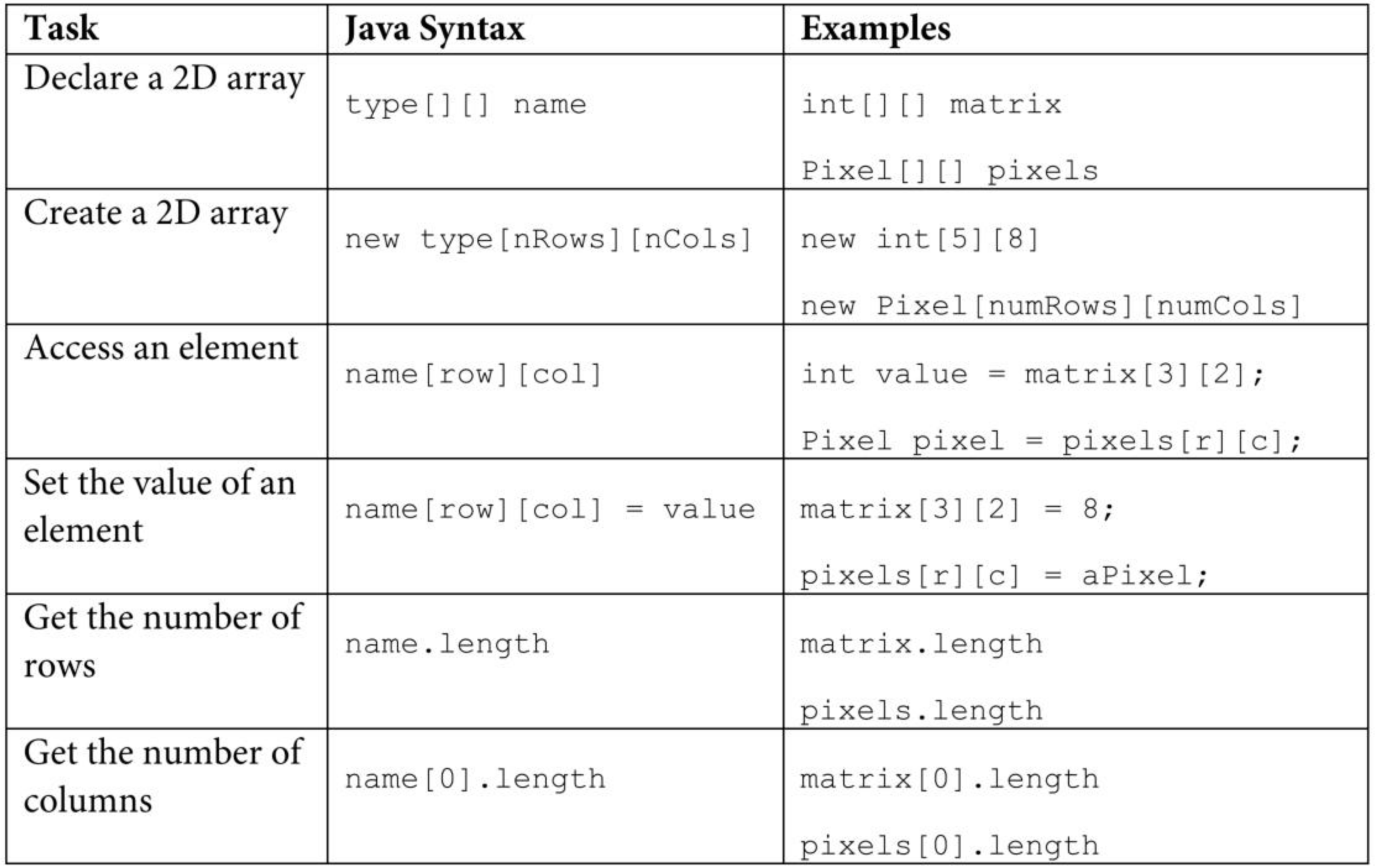


Figure 4: A row-major 2D array (left) and a column-major 2D array (right)

A4 Two-dimensional arrays in Java

- The following table shows the Java syntax and examples for tasks with 2D arrays. Java supports 2D arrays of primitive and object types.



Task	Java Syntax	Examples

Declare a 2D array	`type[][] name`	`int[][] matrix`
		`Pixel[][] pixels`

Create a 2D array	`new type[nRows][nCols]`	`new int[5][8]`
		`new Pixel[numRows][numCols]`

Access an element	`name[row][col]`	`int value = matrix[3][2];`
		`Pixel pixel = pixels[r][c];`

Set the value of an element	`name[row][col] = value`	`matrix[3][2] = 8;`
		`pixels[r][c] = aPixel;`

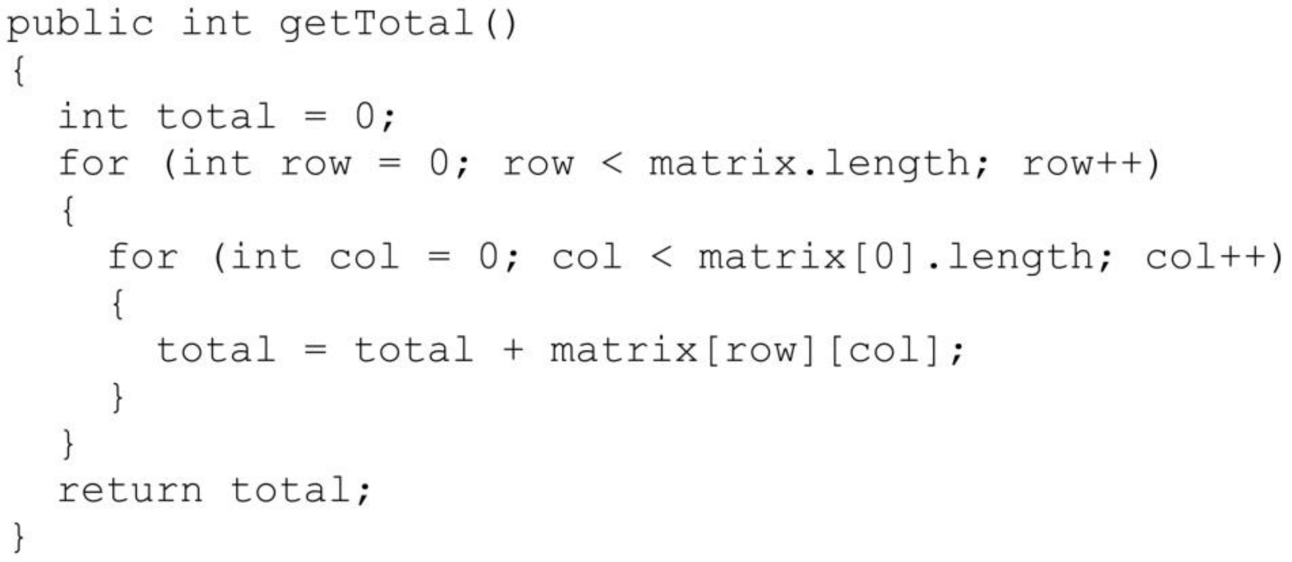
Get the number of rows	`name.length`	`matrix.length` `pixels.length`

Get the number of columns	`name[0].length`	`matrix[0].length`
		`pixels[0].length`

A4 Two-dimensional arrays in Java

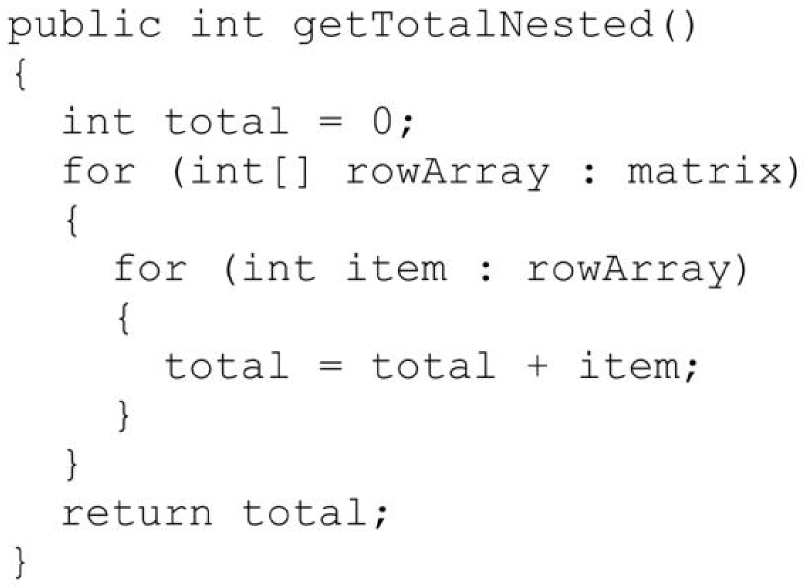
- Here is a method in the `IntArrayWorker` class that totals all the values in a 2D array of integers in a private instance variable (field in the class) named `matrix`.

for loop



```
public int getTotal()
{
  int total = 0;
  for (int row = 0; row < matrix.length; row++)
  {
    for (int col = 0; col < matrix[0].length; col++)
    {
      total = total + matrix[row][col];
    }
  }
  return total;
}
```

for each loop (enhanced for loop)



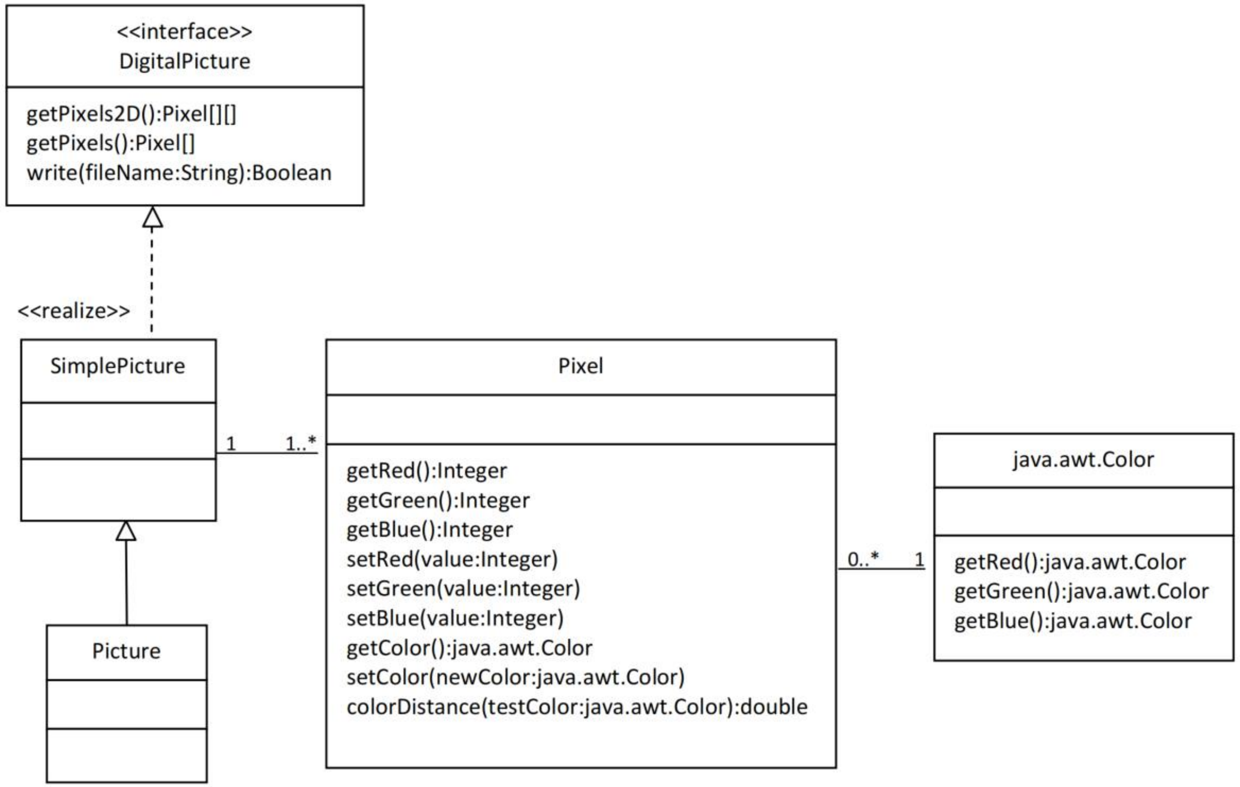
```
public int getTotalNested()
{
  int total = 0;
  for (int[] rowArray : matrix)
  {
    for (int item : rowArray)
    {
      total = total + item;
    }
  }
  return total;
}
```

A4 Two-dimensional arrays in Java

- Exercises
- 1. Write a `getCount` method in the `IntArrayWorker` class that returns the count of the number of times a passed integer value is found in the matrix. There is already a method to test this in `IntArrayWorkerTester`. Just uncomment the method `testGetCount`() and the call to it in the `main` method of `IntArrayWorkerTester`.
- 2. Write a `getLargest` method in the `IntArrayWorker` class that returns the largest value in the matrix. There is already a method to test this in `IntArrayWorkerTester`. Just uncomment the method `testGetLargest`() and the call to it in the main method of `IntArrayWorkerTester`.
- 3. Write a `getColTotal` method in the `IntArrayWorker` class that returns the total of all integers in a specified column. There is already a method to test this in `IntArrayWorkerTester`. Just uncomment the method `testGetColTotal`() and the call to it in the main method of `IntArrayWorkerTester`.

A5 modifying a picture

- You will write methods in the `Picture` class that modify digital pictures.
- The `Picture` class inherits from the `SimplePicture` class and the `SimplePicture` class implements the `DigitalPicture` interface as shown in the Unified Modeling Language (UML) class diagram.



<<interface>>
DigitalPicture

getPixels2D():Pixel[][]
getPixels():Pixel[]
write(fileName:String):Boolean

<<realize>>

SimplePicture

1

1..*

Pixel

getRed():Integer
getGreen():Integer
getBlue():Integer
setRed(value:Integer)
setGreen(value:Integer)
setBlue(value:Integer)
getColor():java.awt.Color
setColor(newColor:java.awt.Color)
colorDistance(testColor:java.awt.Color):double

0..*

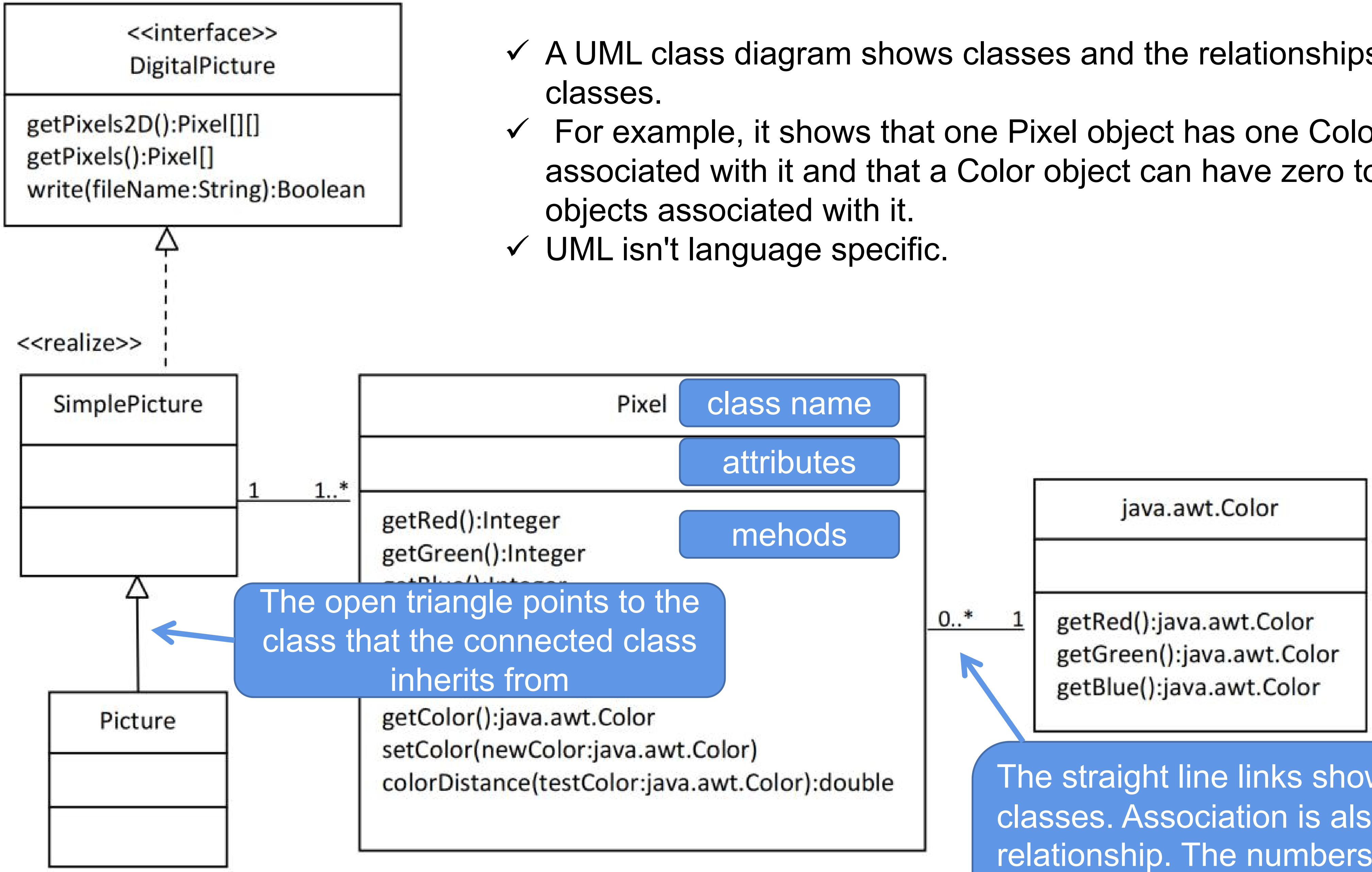
1

java.awt.Color

getRed():java.awt.Color
getGreen():java.awt.Color
getBlue():java.awt.Color

Picture

- ✓ A UML class diagram shows classes and the relationships between the classes.
- ✓ For example, it shows that one Pixel object has one Color object associated with it and that a Color object can have zero to many Pixel objects associated with it.
- ✓ UML isn't language specific.



class name

attributes

mehods

The open triangle points to the class that the connected class inherits from

Figure 5: A UML Class Diagram

The straight line links show associations between classes. Association is also called a “has-a” relationship. The numbers at the end of the association links give the number of objects associated with an object at the other end.

A5 modifying a picture

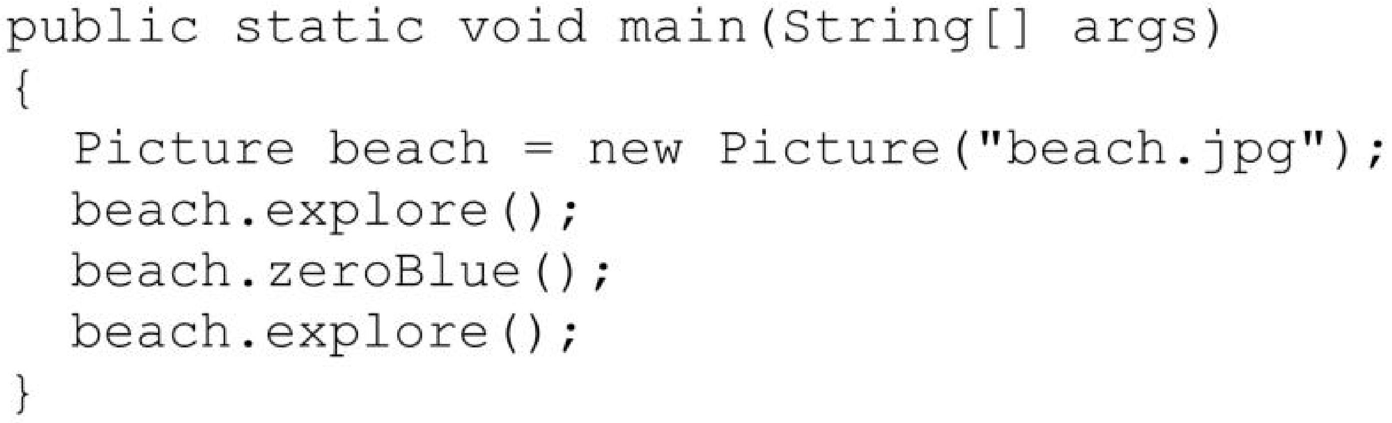
- Questions
- 1. Open `Picture.java` and look for the method `getPixels2D`. Is it there?
- 2. Open `SimplePicture.java` and look for the method `getPixels2D`. Is it there?
- 3. Does the following code compile?
- `DigitalPicture p = new DigitalPicture();`
- 4. Assuming that a no-argument constructor exists for SimplePicture, would the following code compile?
- `DigitalPicture p = new SimplePicture();`
- 5. Assuming that a no-argument constructor exists for Picture, does the following code compile?
- `DigitalPicture p = new Picture();`
- 6. Assuming that a no-argument constructor exists for Picture, does the following code compile?
- `SimplePicture p = new Picture();`
- 7. Assuming that a no-argument constructor exists for SimplePicture, does the following code compile?
- `Picture p = new SimplePicture();`

A5 modifying a picture

- **Extensions----Interface**
- `DigitalPicture` is an **interface**. An interface most often only has public abstract methods. An abstract method is not allowed to have a body. Notice that none of the methods declared in `DigitalPicture` have a body. If a method can't have a body, what good is it?
- Interfaces are useful for separating **what** from **how**. An interface specifies what an object of that type needs to be able to do but not **how** it does it. You cannot create an object using an interface type.
- A class can implement (realize) an interface as `SimplePicture` does. A non-abstract class provides bodies for all the methods declared in the interface, either directly or through inheritance. You can declare a variable to be of an interface type and then set that variable to refer to an object of any class that implements that interface.
- Java has a `List` interface that declares the methods that a list should have such as `add`, `remove`, and `get`, etc. But, if you want to create a `List` object you will create an `ArrayList` object.

A5 modifying a picture

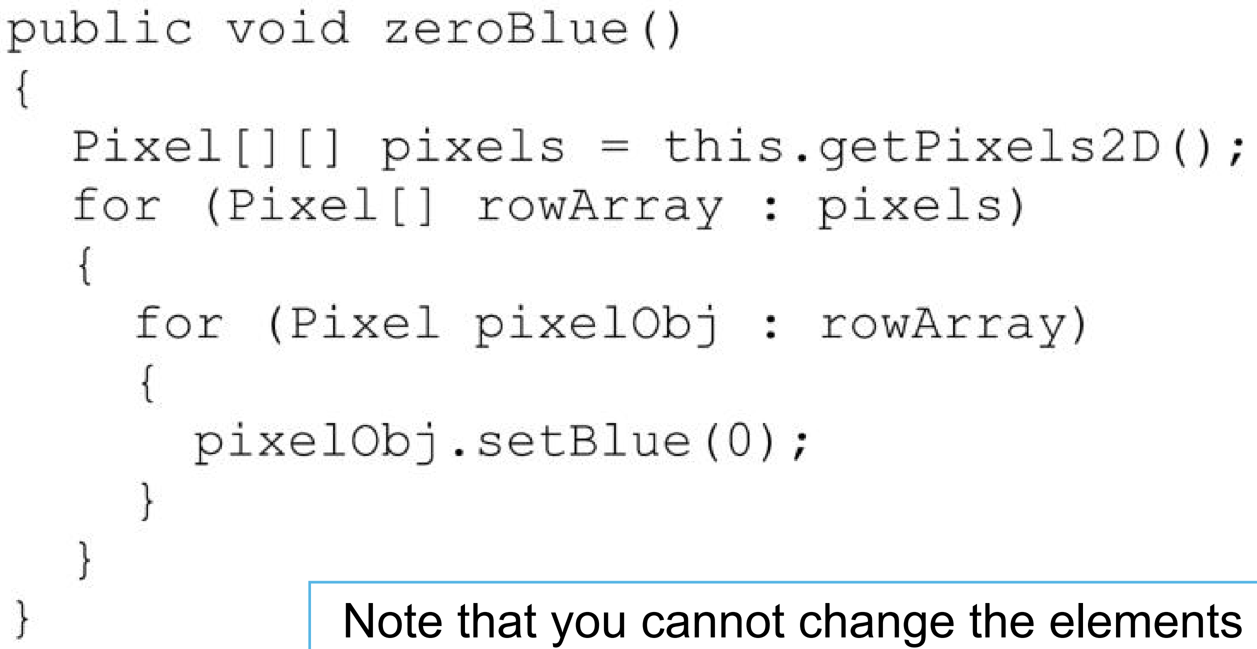
- What do you think you will see if you modify the beach picture in the images folder to set all the blue values to zero? Do you think you will still see a beach?
- Run the `main` method in the `Picture` class.
- The following code is the `main` method from the `Picture` class.



```
public static void main(String[] args)
{
  Picture beach = new Picture("beach.jpg");
  beach.explore();
  beach.zeroBlue();
  beach.explore();
}
```

A5 modifying a picture

- The following code is the `zeroBlue()` method from the `Picture` class.



```
public void zeroBlue()
{
  Pixel[][] pixels = this.getPixels2D();
  for (Pixel[] rowArray : pixels)
  {
    for (Pixel pixelObj : rowArray)
    {
      pixelObj.setBlue(0);
    }
  }
}
```

Note that you cannot change the elements of an array when you use a for-each loop. If, however, the array elements are references to objects that have methods that allow changes,
you can change the internal state of objects referenced in the array (pixels).

A5 modifying a picture

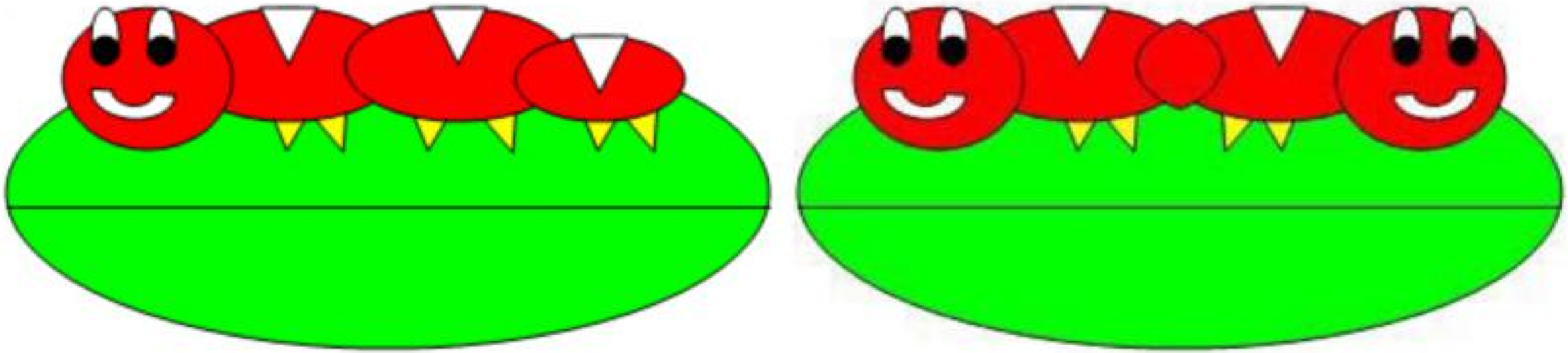
- Exercise:
- 1.Open `PictureTester.java` and run its `main` method. You should get the same results as running the `main` method in the `Picture` class. The `PictureTester` class contains class (static) methods for testing the methods that are in the `Picture` class.
- 2. Uncomment the appropriate test method in the `main` method of `PictureTester` to test any of the other methods in `Picture.java`. You can comment out the tests you don't want to run. You can also add new test methods to `PictureTester` to test any methods you create in the `Picture` class.
- 3.Using the `zeroBlue` method as a starting point, write the method `keepOnlyBlue` that will keep only the blue values, that is, it will set the red and green values to zero.
  - Create a class (static) method to test this new method in the class `PictureTester`. Be sure to call the new test method in the `main` method in `PictureTester`.

A5 modifying a picture

- 4.Write the `negate` method to negate all the pixels in a picture.
    - To negate a picture, set the red value to 255 minus the current red value, the green value to 255 minus the current green value and the blue value to 255 minus the current blue value. Create a class (static) method to test this new method in the class `PictureTester`. Be sure to call the new test method in the main method in `PictureTester`.
- 5. Write the `grayscale` method to turn the picture into shades of gray.
    - Set the red, green, and blue values to the average of the current red, green, and blue values (add all three values and divide by 3).
    - Create a class (static) method to test this new method in the class `PictureTester`. Be sure to call the new test method in the `main` method in `PictureTester`.
- 6. Challenge — Explore the “`water.jpg`” picture in the `images` folder. Write a method `fixUnderwater()` to modify the pixel colors to make the fish easier to see.
    - Create a class (static) method to test this new method in the class `PictureTester`. Be sure to call the new test method in the `main` method in `PictureTester`.

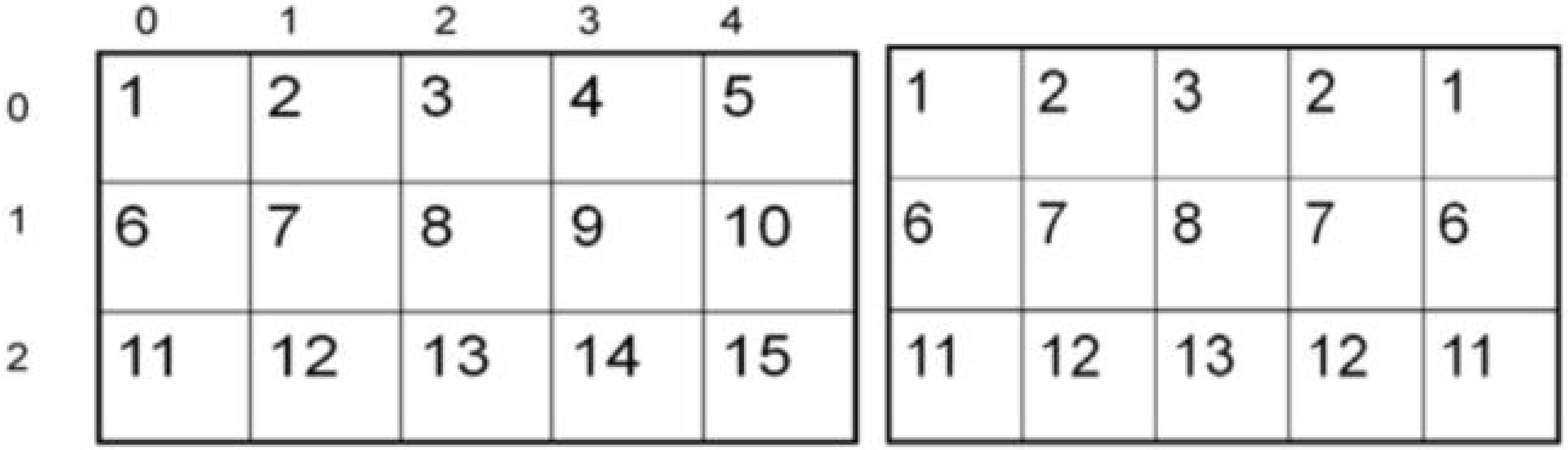
A6 Mirroring pictures

- Car designers at General Motors Research Labs only sculpt half of a car out of clay and then use a vertical mirror to reflect that half to see the whole car. What if we want to see what a picture would look like if we placed a mirror on a vertical line in the center of the width of the picture to reflect the left side.



A6 Mirroring pictures

- How can we write a method to mirror a picture in this way? One way to figure out the algorithm, which is a description of the steps for solving a problem, is to try it on smaller and simpler data. Figure 7 shows the result of mirroring a two-dimensional array of numbers from left to right vertically.



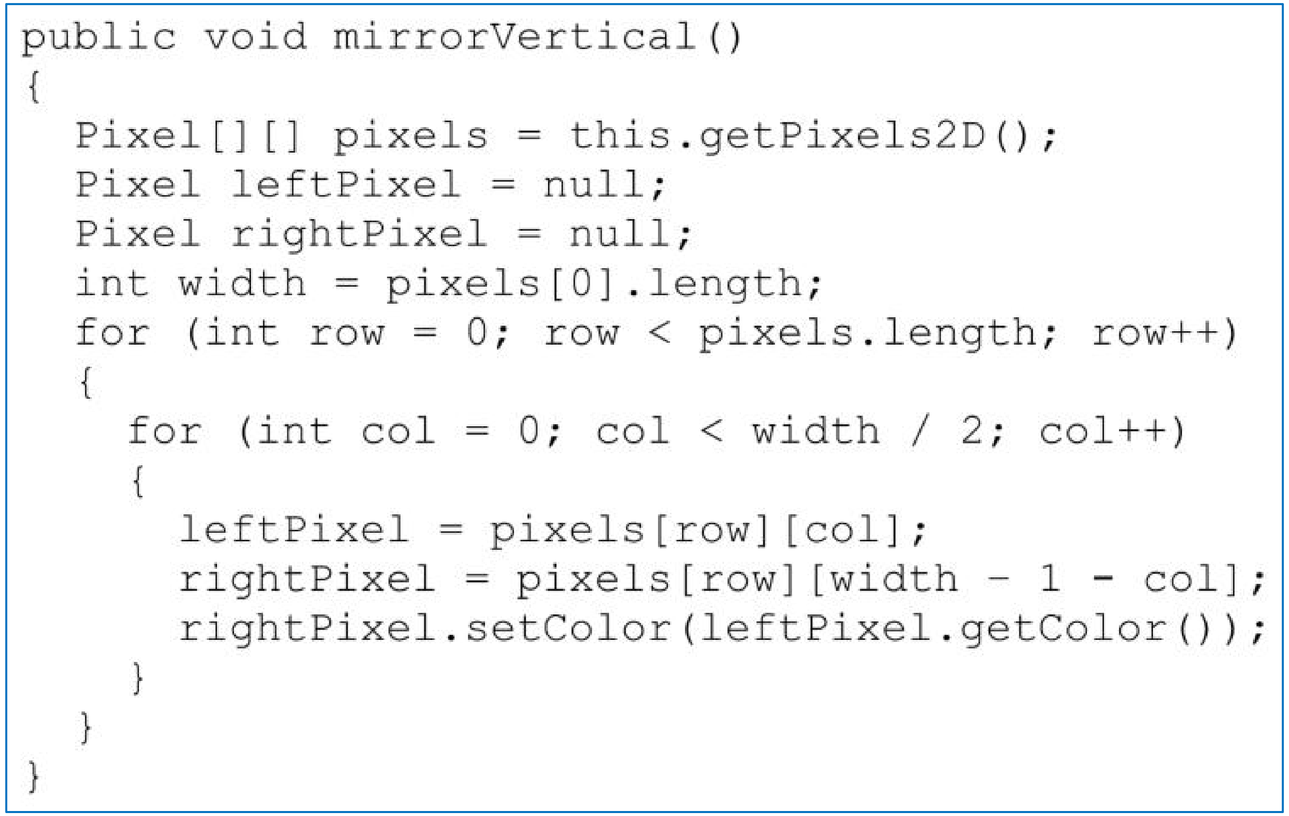
	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15

1	2	3	2	1
6	7	8	7	6
11	12	13	12	11

Two-Dimensional array of numbers (left) and mirrored result (right)

- Can you figure out the algorithm for this process?

A6 Mirroring pictures



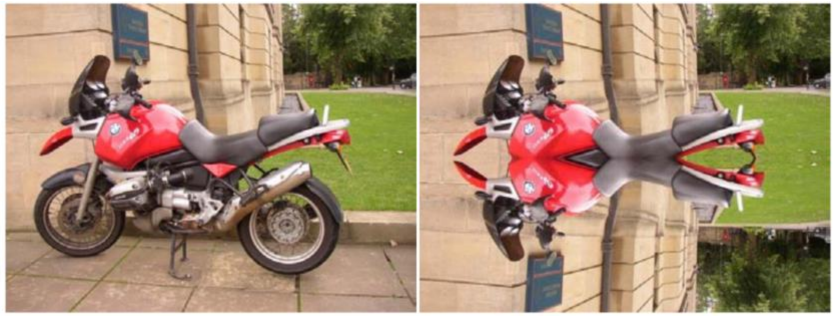
```java
public void mirrorVertical()
{
  Pixel[][] pixels = this.getPixels2D();
  Pixel leftPixel = null;
  Pixel rightPixel = null;
  int width = pixels[0].length;
  for (int row = 0; row < pixels.length; row++)
  {
    for (int col = 0; col < width / 2; col++)
    {
      leftPixel = pixels[row][col];
      rightPixel = pixels[row][width - 1 - col];
      rightPixel.setColor(leftPixel.getColor());
    }
  }
}
```

- ✓ loop through all the rows and half the columns.
- ✓ get a pixel from the left side of the picture and a pixel from the right side of the picture, which is the same distance from the right end as the left pixel is from the left end.
- ✓ Set the color of the right pixel to the color of the left pixel.
- ✓ The column number at the right end is the number of columns, also known as the width, minus one. Each time the left pixel is at (current row value, current column value), the corresponding right pixel is at (current row value, width - 1 - (current column value)).

- You can test this with the `testMirrorVertical` method in `PictureTester`.

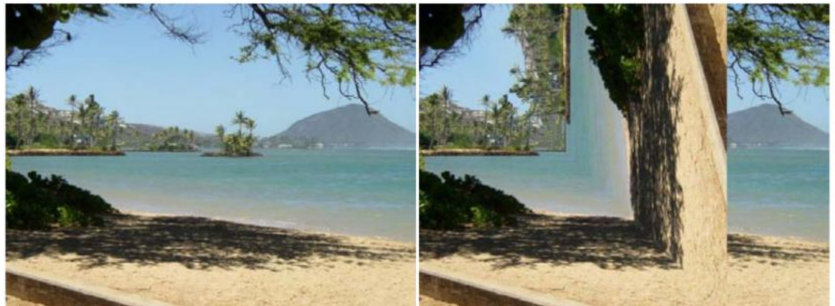
A6 Mirroring pictures

- Exercise
- 1. Write the method `mirrorVerticalRightToLeft` that mirrors a picture around a mirror placed vertically from right to left. Hint: you can copy the body of `mirrorVertical` and only change one line in the body of the method to accomplish this. Write a class (static) test method called `testMirrorVerticalRightToLeft` in `PictureTester` to test this new method and call it in the `main` method.
- 2. Write the method `mirrorHorizontal` that mirrors a picture around a mirror placed horizontally at the middle of the height of the picture. Mirror from top to bottom as hown in the pictures below (Figure 8). Write a class (static) test method in `PictureTester` to test this new method and call it in the `main` method.



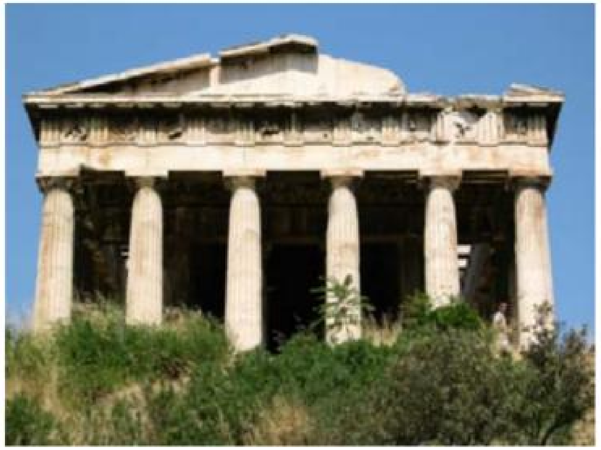
A6 Mirroring pictures

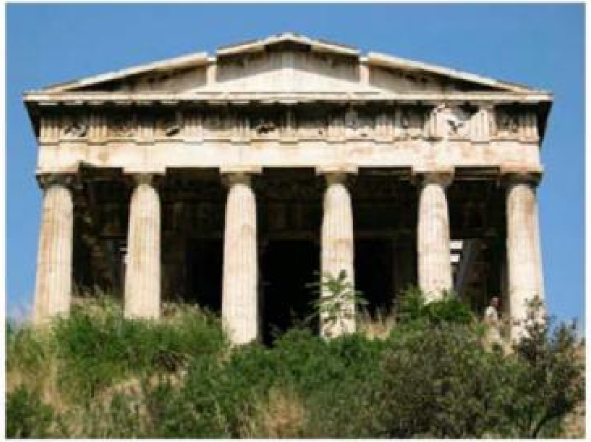
- 3. Write the method `mirrorHorizontalBotToTop` that mirrors the picture around a mirror placed horizontally from bottom to top.
    - Hint: you can copy the body of `mirrorHorizontal` and only change one line to accomplish this. Write a class (static) test method in `PictureTester` to test this new method and call it in the `main` method.
- 4. Challenge — Work in groups to figure out the algorithm for the method `mirrorDiagonal` that mirrors just a square part of the picture from bottom left to top right around a mirror placed on the diagonal line (the diagonal line is the one where the row index equals the column index). This will copy the triangular area to the left and below the diagonal line as shown below. This is like folding a square piece of paper from the bottom left to the top right, painting just the bottom left triangle and then (while the paint is still wet) folding the paper up to the top right again. The paint would be copied from the bottom left to the top right as shown in the pictures below (Figure 9). Write a class (static) test method in `PictureTester` to test this new method and call it in the `main` method.



A7 Mirroring part of a picture

- Sometimes you only want to mirror part of a picture.
- use the explorer tool to find the area that you want to mirror to produce the picture on the right.
- If you do this you will find that you can mirror the rows from 27 to 96 (inclusive) and the columns from 13 to 275 (inclusive).
- You can change the starting and ending points for the row and column values to mirror just part of the picture.



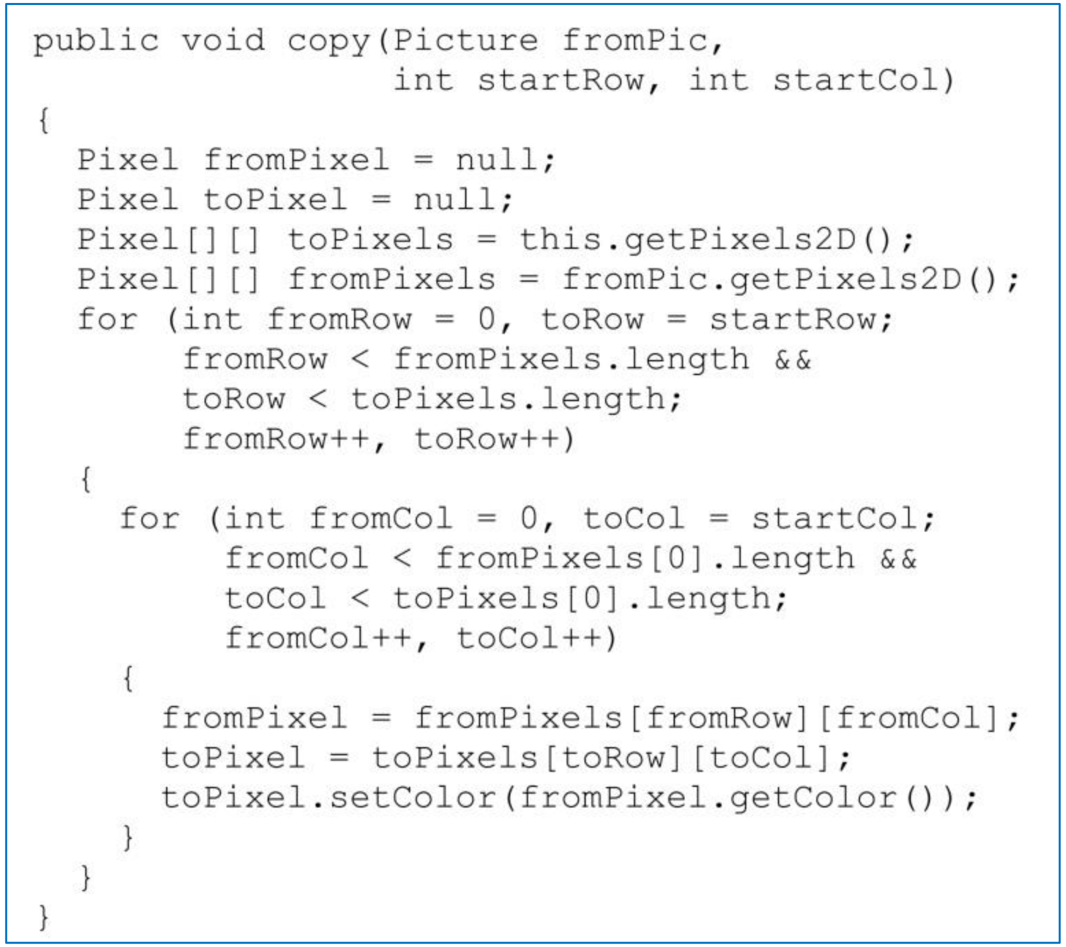


A7 Mirroring part of a picture

- Exercises
- 1. Check the calculation of the number of times the body of the nested loop executes by adding an integer count variable to the `mirrorTemple` method that starts out at 0 and increments inside the body of the loop. Print the value of `count` after the nested loop ends.
- 2. Write the method `mirrorArms` to mirror the arms on the snowman (“`snowman.jpg`”) to make a snowman with 4 arms. Write a class (static) test method in `PictureTester` to test this new method and call it in the `main` method.
- 3. Write the method `mirrorGull` to mirror the seagull (“`seagull.jpg`”) to the right so that there are two seagulls on the beach near each other. Write a class (static) test method in `PictureTester` to test this new method and call it in the `main` method.

A8: Creating a collage

- You can copy one picture to another by copying the color from the pixels in one picture to the pixels in the other picture.

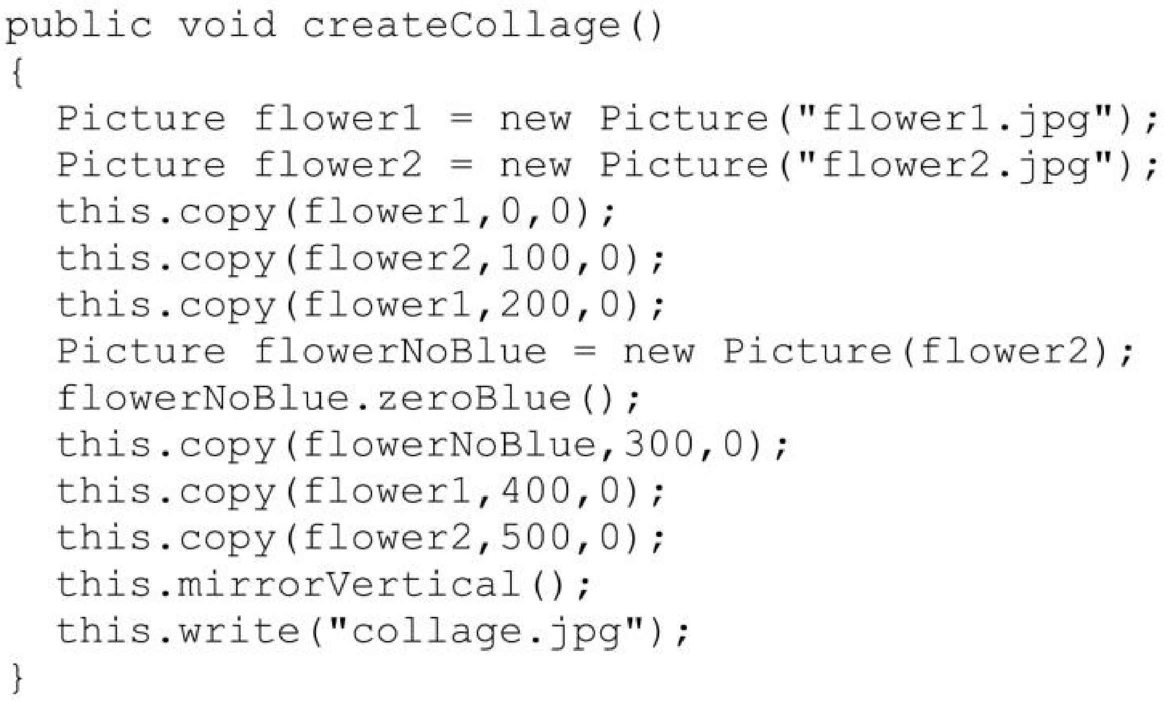


```
public void copy(Picture fromPic,
                 int startRow, int startCol)
{
  Pixel fromPixel = null;
  Pixel toPixel = null;
  Pixel[][] toPixels = this.getPixels2D();
  Pixel[][] fromPixels = fromPic.getPixels2D();
  for (int fromRow = 0, toRow = startRow;
       fromRow < fromPixels.length &&
       toRow < toPixels.length;
       fromRow++, toRow++)
  {
    for (int fromCol = 0, toCol = startCol;
         fromCol < fromPixels[0].length &&
         toCol < toPixels[0].length;
         fromCol++, toCol++)
    {
      fromPixel = fromPixels[fromRow][fromCol];
      toPixel = toPixels[toRow][toCol];
      toPixel.setColor(fromPixel.getColor());
    }
  }
}
```

A `for` loop can have more than one variable declaration and initialization and/or modification. Just separate the items with commas.
Note that the inner loop has both a `fromCol` and a `toCol` declared, initialized, and incremented.

A8: Creating a collage

- You can create a collage by copying several small pictures onto a larger picture. You can do some picture manipulations like zero blue before you copy the picture as well. You can even mirror the result to get a nice artistic effect



```
public void createCollage()
{
  Picture flower1 = new Picture("flower1.jpg");
  Picture flower2 = new Picture("flower2.jpg");
  this.copy(flower1,0,0);
  this.copy(flower2,100,0);
  this.copy(flower1,200,0);
  Picture flowerNoBlue = new Picture(flower2);
  flowerNoBlue.zeroBlue();
  this.copy(flowerNoBlue,300,0);
  this.copy(flower1,400,0);
  this.copy(flower2,500,0);
  this.mirrorVertical();
  this.write("collage.jpg");
}
```

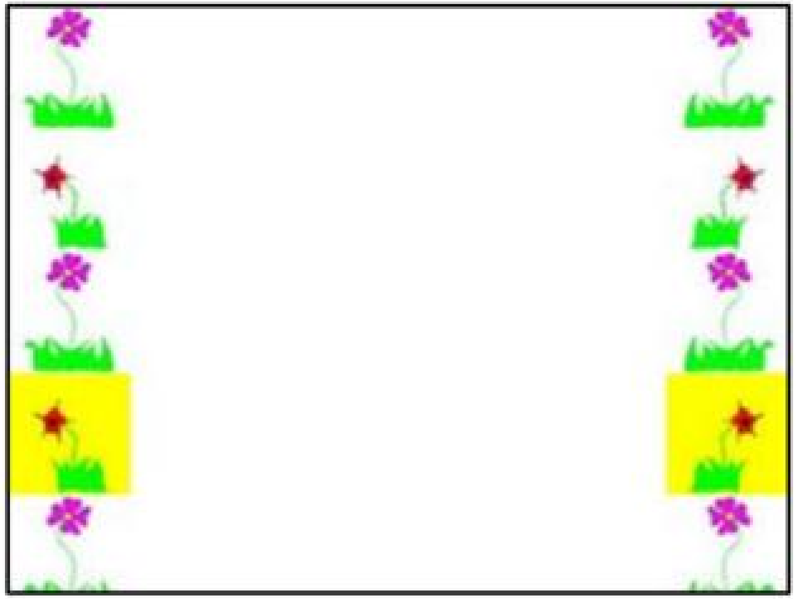


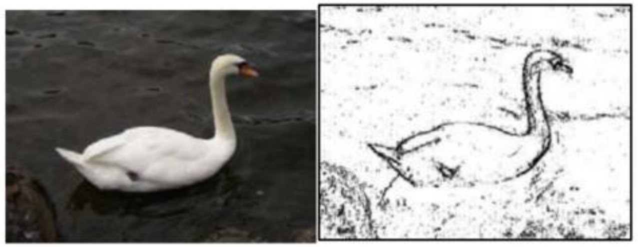
Figure 11: Collage with vertical mirror

A8: Creating a collage

- Exercises
- 1. Create a second `copy` method that adds parameters to allow you to copy just part of the `fromPic`. You will need to add parameters that specify the start row, end row, start column, and end column to copy from. Write a class (static) test method in `PictureTester` to test this new method and call it in the `main` method.
- 2.Create a `myCollage` method that has at least three pictures (can be the same picture) copied three times with three different picture manipulations and at least one mirroring. Write a class (static) test method in `PictureTester` to test this new method and call it in the `main` method.

A9 Simple edge detection

- Detecting edges is a common image processing problem. For example, digital cameras often feature face detection. Some robotic competitions require the robots to find a ball using a digital camera, so the robot needs to be able to “see” a ball.
- One way to look for an edge in a picture is to compare the color at the current pixel with the pixel in the next column to the right. If the colors differ by more than some specified amount, this indicates that an edge has been detected and the current pixel color should be set to black. Otherwise, the current pixel is not part of an edge and its color should be set to white .
- How do you calculate the difference between two colors?
  - The formula for the difference between two points (x_1,y_1) and (x_2,y_2) is the square root of ((x_2 - x_1)^2 + (y_2 - y_1)^2). The difference between two colors (red$_1$,green$_1$,blue$_1$) and (red$_2$,green$_2$,blue$_2$) is the square root of ((red$_2$ - red$_1$)^2 +(green$_2$ - green$_1$)^2 +(blue$_2$ - blue$_1$)^2). The colorDistance method in the Pixel class uses this calculation to return the difference between the current pixel color and a passed color.



A9 Simple edge detection

- Exercises
- 1. Notice that the current edge detection method works best when there are big color changes from left to right but not when the changes are from top to bottom. Add another loop that compares the current pixel with the one below and sets the current pixel color to black as well when the color distance is greater than the specified edge distance.
- 2. Work in groups to come up with another algorithm for edge detection.

Homework

- work in pairs (3 at most)
- complete all the questions and exercises
- presentation （all the following should be involved)
  - About inheritance——the questions in A5 modifying a picture， give the answer and reason
  - All the code and execution result for the exercise
  - Other function about edge detection （code and execution result)
  - What dou you learn from this lab