AP® Computer Science A Picture Lab Student Guide

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

Picture Lab: Student Guide

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.

Set-up
You will need the `pixLab` folder and a Java Development Kit, also known as a JDK (see http://www.oracle.com/technetwork/java/javase/downloads/index.html). A development environment is also useful. DrJava is a free development environment for Java that allows students to try out code in an interactions pane. It also has a debugger, and can be downloaded from http://drjava.org. However, you can use any development environment with this lab. Just open the files in the `classes` folder and compile them. Please note that there are two small pictures in the `classes` folder that need to remain there: `leftArrow.gif` and `rightArrow.gif`. If you copy the Java source files to another folder you must copy these gif files as well.

Keep the `images` folder and the `classes` folder together in the `pixLab` folder. The `FileChooser` expects the images to be in a folder called `images`, at the same level as the `classes` folder. If it does not find the images there it also looks in the same folder as the class files that are executing. If you wish to modify this, change the `FileChooser.java` class to specify the folder where the pictures are stored. For example, if you want to store the images in "`r://student/images/`," change the following line in the method `getMediaDirectory()` in `FileChooser.java`:

```
URL fileURL = new URL(classURL,"../images/");
```

And modify it to

```
URL fileURL = new URL("r://student/images/");
```

Then recompile.

A4: Two-dimensional arrays in Java

In this activity you will work with integer data stored in a two-dimensional array. Some programming languages use a one-dimensional (1D) array to represent a two-dimensional (2D) array with the data in either *row-major* or *column-major order*. *Row-major order* in a 1D array means that all the data for the first row is stored before the data for the next row in the 1D array. *Column-major order* in a 1D array means that all the data for the first column is stored before the data for the next column in the 1D array. The order matters, because you need to calculate the position in the 1D array based on the order, the number of rows and columns, and the current column and row numbers (indices). The rows and columns are numbered (indexed) and often that numbering starts at 0 as it does in Java. The top left row has an index of 0 and the top left column has an index of 0. The row number (index) increases from top to bottom and the column number (index) increases from left to right as shown below.

	0	1	2
0	1	2	3
1	4	5	6

If the above 2D array is stored in a 1D array in row-major order it would be:

0	1	2	3	4	5
1	2	3	4	5	6

If the above 2D array is stored in a 1D array in column-major order it would be:

0	1	2	3	4	5
1	4	2	5	3	6

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). The AP Computer Science A course specification tells you to assume that all 2D arrays are row-major, which means that the outer array in Java represents the rows and the inner arrays represent the columns.

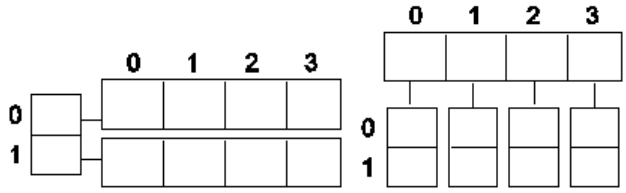

Figure 4: A row-major 2D array (left) and a column-major 2D array (right)

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`

To loop through the values in a 2D array you must have two indexes. One index is used to change the row index and one is used to change the column index. You can use *nested loops,* which is one `for` loop inside of another, to loop through all the values in a 2D array.

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`. Notice the nested `for` loop and how it uses `matrix.length` to get the number of rows and `matrix[0].length` to get the number of columns. Since `matrix[0]` returns the inner array in a 2D array, you can use `matrix[0].length` to get the number of columns.

```
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;
}
```

Because Java two-dimensional arrays are actually arrays of arrays, you can also get the total using nested `for-each` loops as shown in `getTotalNested` below. The outer loop will loop through the outer array (each of the rows) and the inner loop will loop through the inner array (columns in that row). You can use a nested `for-each` loop whenever you want to loop through all items in a 2D array and you don't need to know the row index or column index.

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

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`.