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.

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.

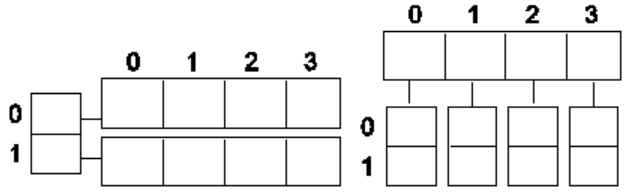

Figure 4: A row-major 2D array (left) and a column-major 2D array (right)

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`.

How image processing is related to new scientific breakthroughs

Many of today's important scientific breakthroughs are being made by large, interdisciplinary collaborations of scientists working in geographically widely distributed locations, producing, collecting, and analyzing vast and complex datasets.

One of the computer scientists who works on a large interdisciplinary scientific team is Dr. Cecilia Aragon. She is an associate professor in the Department of Human Centered Design & Engineering and the eScience Institute at the University of Washington, where she directs the Scientific Collaboration and Creativity Lab. Previously, she was a computer scientist in the Computational Research Division at Lawrence Berkeley National Laboratory for six years, after earning her Ph.D. in Computer Science from UC Berkeley in 2004. She earned her B.S. in mathematics from the California Institute of Technology.

Her current research focuses on human-computer interaction (HCI) and computer-supported cooperative work (CSCW) in scientific collaborations, distributed creativity, information visualization, and the visual understanding of very large data sets. She is interested in how social media and new methods of computer-mediated communication are changing scientific practice. She has developed novel visual interfaces for collaborative exploration of very large scientific data sets, and has authored or co-authored many papers in the areas of computer-supported cooperative work, human-computer interaction, visualization, visual analytics, image processing, machine learning, cyberinfrastructure, and astrophysics.

In 2008, she received the Presidential Early Career Award for Scientists and Engineers (PECASE) for her work in collaborative data-intensive science. Her research has been recognized with four Best Paper awards since 2004, and she was named one of the Top 25 Women of 2009 by Hispanic Business Magazine. She was the architect of the Sunfall data visualization and workflow management system for the Nearby Supernova Factory, which helped advance the study of supernovae in order to reduce the statistical uncertainties on key cosmological parameters that categorize dark energy, one of the grand challenges in physics today.

Cecilia Aragon is also one of the most skilled aerobatic pilots flying today. A two-time member of the U.S. Aerobatic Team, she was a medalist at the 1993 U.S. National Championships and the 1994 World Aerobatic Championships, and was the California State Aerobatic Champion.

Glossary

1. Abstract class — You cannot create an object of an abstract class type. But, you can create an object of a subclass of an abstract class (as long as the subclass is not also an abstract class).
2. Abstract method — An abstract method cannot have a method body in the class where the method is declared to be abstract.
3. Algorithm — A step-by-step description of how to solve a problem.
4. AWT — The Abstract Windowing Toolkit. It is the package that contains the original Graphical User Interface (GUI) classes developed for Java.
5. Binary number — A binary number contains only the digits 0 and 1. Each place is a power of 2 starting with 2^0 on the right. The decimal number 6 would be 110 in binary. That would be $0 * 2^0 + 1 * 2^1 + 1 * 2^2 = 6$.
6. Bit — A **b**inary dig**it**, which means that it has a value of either 0 or 1.
7. Byte — A consecutive group of 8 bits.
8. Column-major order — An order for storing two-dimensional array data in a one-dimensional array, so that all the data for the first column is stored before all the data for the second column and so on. In a two-dimensional array represented using an array of arrays (like in Java) this means that the outer array represents the columns and the inner arrays represent the rows.
9. Digital camera — A camera that can take digital pictures.
10. Digital picture — A picture that can be stored on a computer.
11. Inheritance — In Java, a class can specify the parent class from which it inherits instance variables (object fields) and object methods. Even though instance variables may be inherited, if they are declared to be private they cannot be directly accessed using dot notation in the inheriting class. Private methods that are inherited can also not be directly called in an inheriting class.
12. Inner loop — In a nested loop (a loop inside of another loop) the loop that is inside of another loop is considered the inner loop.
13. Interface — A special type of class that can only have public abstract methods in it and/or static constants.
14. Lossy compression — Lossy compression means that the amount of data that is stored is much smaller than the available data, but the part that is not stored is data that humans would not miss.
15. Media computation — A method of teaching programming by having students write programs that manipulate media: pictures, sounds, text, movies. This approach was developed by Dr. Mark Guzdial at Georgia Tech.
16. Megapixel — One million pixels.
17. Nested loop — One loop inside of another loop.
18. Outer loop — In a nested loop (a loop inside of another loop) the loop that is outside of another loop is considered the outer loop.
19. Package — A package in Java is a group of related classes.
20. Pixel — A picture (abbreviated **pix**) **el**ement.
21. RGB model — Represents color as amounts of red, green, and blue light. It sometimes also includes alpha, which is the amount of transparency.

22. Row-major order — An order for storing two-dimensional array data in a one-dimensional array, so that all the data for the first row is stored before all the data for the second row, and so on. In a two-dimensional array represented using an array of arrays (like in Java) this means that the outer array represents the rows and the inner arrays represent the columns.
23. Subclass — A class that has inherited from another class.
24. Superclass — A class that another class has inherited from.
25. UML —Unified Modeling Language. It is a general purpose modeling language used in object-oriented software development.

References

Dann, W., Cooper, S., & Ericson, B. (2009) *Exploring Wonderland: Java Programming Using Alice and Media Computation*. Englewood, NJ: Prentice-Hall.

Guzdial, M., & Ericson B. (2006) *Introduction to Computing and Programming in Java: A Multimedia Approach*. Englewood, NJ: Prentice-Hall.

Guzdial, M., & Ericson, B. (2009) *Introduction to Computing and Programming in Python: A Multimedia Approach*. (2nd ed.). Englewood, NJ: Prentice-Hall.

Guzdial, M., & Ericson, B. (2010) *Problem Solving with Data Structures using Java: A Multimedia Approach*. Englewood, NJ: Prentice-Hall.

Quick Reference

DigitalPicture Interface

```
Pixel[][] getPixels2D()         // implemented in SimplePicture
void explore()                  // implemented in SimplePicture
boolean write(String fileName)  // implemented in SimplePicture
```

SimplePicture Class (implements Digital Picture)

```
public SimplePicture()
public SimplePicture(int width, int height)
public SimplePicture(SimplePicture copyPicture)
public SimplePicture(String fileName)
public Pixel[][] getPixels2D()
public void explore()
public boolean write(String fileName)
```

Picture Class (extends SimplePicture)

```
public Picture()
public Picture(int height, int width)
public Picture(Picture copyPicture)
public Picture(String fileName)
public Pixel[][] getPixels2D()         // from SimplePicture
public void explore()                  // from SimplePicture
public boolean write(String fileName)  // from SimplePicture
```

Pixel Class

```
public double colorDistance(Color testColor)
public double getAverage()
public int getRed()
public int getGreen()
public int getBlue()
public Color getColor()
public int getRow()
public int getCol()
public void setRed(int value)
public void setGreen(int value)
public void setBlue(int value)
public void setColor(Color newColor)
```

java.awt.Color Class

```
public Color(int r, int g, int b)
public int getRed()
public int getGreen()
public int getBlue()
```