

## CMIS 141 Hands-on Lab Week 5

### Overview

The week we continue our study of Java by working with arrays. Arrays allow us to store and manipulate data. For example, we can create an array of integers and then sort those in ascending or descending order. We can also search the array to see if a specific value is in the array and where it is located.

It is assumed the JDK 8 or higher programming environment is properly installed and the associated readings for this week have been completed.

### Submission requirements

Hands-on labs are designed for you to complete each week as a form of self-assessment. You do not need to submit your lab work for grading. However; you should post questions in the weekly questions area if you have any trouble or questions related to completing the lab. These labs will help prepare you for the graded assignments, therefore; it is highly recommended you successfully complete these exercises.

### Objectives

The following objectives will be covered by successfully completing each exercise:

1. Create 1D arrays in Java
2. Use existing algorithms and the Arrays class to sort arrays
3. Use the Arrays class to search for values in an array
4. Create Multidimensional arrays in Java

### Exercise 1 – Create 1D arrays in Java

Java arrays help us be more efficient in our coding by allowing one variable name to be associated with multiple values. For example, instead of having to create names such as value1, value2, value3, value4 ... value100, we can create an array named values and allocate enough space or memory for 100 values.

```
int[] values = new int[100];
```

We can create arrays of all of the Java primitives using the same approach:

```
short[] values = new short[100];  
byte[] values = new byte[100];  
long[] values = new long[100];  
float[] values = new float[100];  
double[] values = new double[100];  
char[] values = new char[100];
```

With arrays you use the array index to reference specific values. In Java, indexes start with 0. So an array of 100 elements has indexes from 0 to 99. This is a critical concept because if you attempt to access beyond element 99 in a 100 element array, you will receive errors.

You can also create arrays of Objects. For example, you can construct an array of String objects:

```
String[] lastNames = new String[5];
```

Then you can assign Strings for each of the 5 elements of the array.

```
lastNames[0] = "Jones";
lastNames[1] = "Smith";
lastNames[2] = "Applebee";
lastNames[3] = "Adams";
lastNames[4] = "Johnson";
```

You can also create arrays of objects you have created or of any objects available to you in the Java API or from another programmer. For example, consider the Point classes we made earlier this semester. The following code would create an array of five Point classes and then create a point for each element.

```
Point[] pentagon = new Point[5];
pentagon[0] = new Point(0.0,0.0);
pentagon[1] = new Point(1.0,0.0);
pentagon[2] = new Point(1.0,1.0);
pentagon[3] = new Point(0.5,1.5);
pentagon[4] = new Point(0.0,1.0);
```

In this exercise we will create, initialize and then print the values of several primitive data arrays. We will use loops to both initialize and display the results. We will also create some arrays of existing Java objects.

- a. Open your favorite text editor and type (or copy and paste) the following Java:

```
/*
 * File: ArrayDemo.java
 * Author: Dr. Robertson
 * Date: January 1, 2015
 * Purpose: This program demonstrates the creation
 * and use of Java arrays
 */

// Import Color class
import java.awt.Color;

public class ArrayDemo {
    // Define constant array size
    static final int ARRAYSIZE = 20;
    static final int STRARRAYSIZE = 5;

    // Main method
    public static void main(String args[]){
        System.out.println("Welcome to the Array Demo");

        // Create array of ints
        int[] intValues = new int[ARRAYSIZE];
        // Assign random values
        for (int i=0; i<intValues.length; i++) {
            intValues[i] = (int) (Math.random() * 10);
        }

        // Create array of String
        String[] stringValues = new String[STRARRAYSIZE];
```

```

stringValues[0] = "Have a Nice Day!";
stringValues[1] = "I'll be back!";
stringValues[2] = "Welcome again!";
stringValues[3] = "Work hard! Play hard!";
stringValues[4] = "I can program arrays in Java!";

    // Create array of doubles
double[] doubleValues = new double[ARRAYSIZE];
// Assign random values
for (int i=0; i<doubleValues.length; i++) {
    doubleValues[i] = Math.random() * 100.0;
}

// Create an array of Color
Color[] colorValues = new Color[ARRAYSIZE];
for (int i=0; i<colorValues.length; i++) {
    int red = (int) (Math.random()*255);
    int green = (int) (Math.random()*255);
    int blue = (int) (Math.random()*255);
    colorValues[i] = new Color(red, green, blue);
}

// Display the results
System.out.println("Int values");
for (int i=0; i<intValues.length; i++) {
    // Print new line every 10 items
    if ((i % 10)==0 ) {
        System.out.println("");
    }
    System.out.print(intValues[i] +"\t");
}

System.out.println("");
System.out.println("Double values");
for (int i=0; i<doubleValues.length; i++) {
    // Print new line every 10 items
    if ((i % 10)==0 ) {
        System.out.println("");
    }
    // Use String.format to show 2 decimals
    System.out.print(String.format("%.2f\t", doubleValues[i]));
}

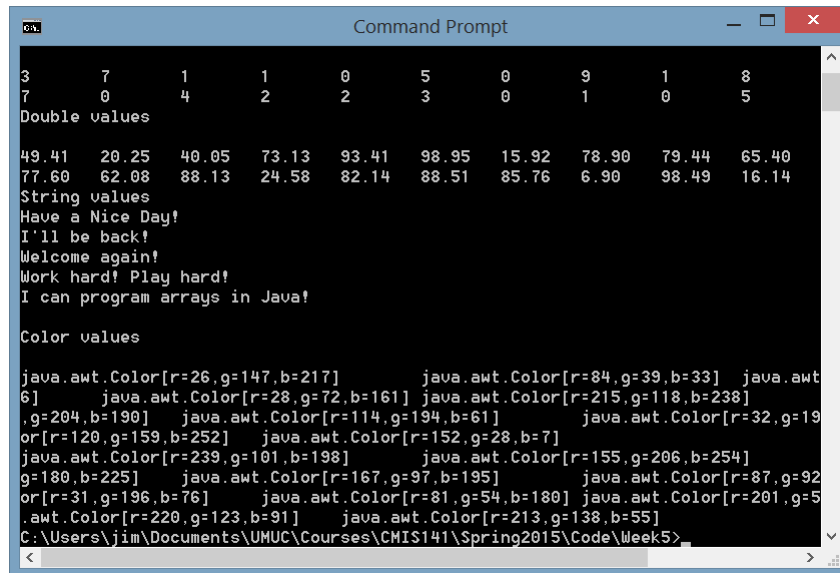
System.out.println("\nString values");
for (int i=0; i<stringValues.length; i++) {
    System.out.println(stringValues[i]);
}

System.out.println("\nColor values");
for (int i=0; i<colorValues.length; i++) {
    // Print new line every 10 items
    if ((i % 10)==0 ) {
        System.out.println("");
    }
    // toString shows rgb values
    System.out.print(colorValues[i].toString() +"\t");
}

```

```
}
}
```

- Save the file as “ArrayDemo.java” in a location of your choice.
- To compile the file, type `javac ArrayDemo.java` at the command prompt. Executing the code will yield an output similar to this:



```

3      7      1      1      0      5      0      9      1      8
7      0      4      2      2      3      0      1      0      5
Double values
49.41  20.25  40.05  73.13  93.41  98.95  15.92  78.90  79.44  65.40
77.60  62.08  88.13  24.58  82.14  88.51  85.76  6.90  98.49  16.14
String values
Have a Nice Day!
I'll be back!
Welcome again!
Work hard! Play hard!
I can program arrays in Java!
Color values
java.awt.Color[r=26,g=147,b=217]  java.awt.Color[r=84,g=39,b=33]  java.awt
6]  java.awt.Color[r=28,g=72,b=161]  java.awt.Color[r=215,g=118,b=238]
,g=204,b=190]  java.awt.Color[r=114,g=194,b=61]  java.awt.Color[r=32,g=19
or[r=120,g=159,b=252]  java.awt.Color[r=152,g=28,b=7]
java.awt.Color[r=239,g=101,b=198]  java.awt.Color[r=155,g=206,b=254]
g=180,b=225]  java.awt.Color[r=167,g=97,b=195]  java.awt.Color[r=87,g=92
or[r=31,g=196,b=76]  java.awt.Color[r=81,g=54,b=180]  java.awt.Color[r=201,g=5
.awt.Color[r=220,g=123,b=91]  java.awt.Color[r=213,g=138,b=55]
C:\Users\jim\Documents\UMUC\Courses\CHIS141\Spring2015\Code\Week5>

```

As you analyze and experiment with the code, note the following:

- You can use static constants to define your array sizes where appropriate. In this case we declare the `ARRAYSIZE` to be 20. This generated 20 of the int, double and Color items and 5 of the String items.

```
static final int ARRAYSIZE = 20;
static final int STRARRAYSIZE = 5;
```

- Loops and the `Math.random()` method are an excellent way to quickly generate and populate pseudorandom data for your arrays.

```
int[] intValues = new int[ARRAYSIZE];
// Assign random values
for (int i=0; i<intValues.length; i++) {
    intValues[i] = (int) (Math.random() * 10);
}
```

3. We used a new class from the java.awt package called the Color class. We could have picked any class from the Java 8 API and applied the constructors to demonstrate the use of arrays of objects. Notice the constructor in the API we used was:

```
Color(int r, int g, int b)
```

The Color class allows us to create Color objects with different red, green and blue components. For this constructor, the 3 arguments are integers with values between 0 and 255. We used the Math.random() method to generate pseudorandom values for each color component using a loop:

```
for (int i=0; i<colorValues.length; i++) {  
    int red = (int) (Math.random()*255);  
    int green = (int) (Math.random()*255);  
    int blue = (int) (Math.random()*255);  
    colorValues[i] = new Color(red, green, blue);  
}
```

4. We used the String.format() method to display only the first 2 decimal places of the double values when we printed the doubleValues array. The format() method can be used for floating and doubles by using the %f and for integers using the %i. There are many possibilities for formatting results. Typically, you use a decimal point and the number of decimal places you need. For example, "%.4f" would provide 4 decimal placed.
5. Finally, notice we called the Color.toString() method to display the Color's red, blue and green values.

Now it is your turn. Try the following exercise:

Create a Java class named MyArrayDemo using your favorite text editor. Be sure you name the file "MyArrayDemo.java". Add code to the file in the main() method that will construct 50 random int values between -50 and 50, and 100 **Boolean** objects with random values of true or false. Use a loop structure of your choice to initialize and print the results of your arrays.

## Exercise 2 – Use existing algorithms and the Arrays class to sort arrays

As mentioned in the reading and videos for in the Sorting arrays topic, sorting is a common task in computer programming. Multiple algorithms and strategies have been developed for sorting algorithms. These algorithms include selection sort, insertion sort, bubble sort, quick sort and others. This course is not focused on algorithm performance, so we won't discuss speed or efficiency at this point.

We will use the available algorithms to sort data. In this exercise, we will use the selection sorting algorithm and the Arrays quick sort algorithm. The Java Arrays class provides both searching and sorting algorithms for most primitive data types.

- a. Open your favorite text editor and type (or copy and paste) the following Java:

```
/*
 * File: SortDemo.java
 * Author: Dr. Robertson
 * Date: January 1, 2015
 * Purpose: This program demonstrates how to sort
 * primitives using the selection sort
 * and the Arrays quick sort algorithm
 */

// Import Color class
import java.util.Arrays;

public class SortDemo {
// Define constant array size
static final int ARRAYSIZE = 10;

// Method from Notes for Selection Sort
public static void selectionSort(int[] array) {
    // Iterate for each position in the array. Note
    // that because < (not <= ) is used,
    // i < (array.length - 1) iterates
    // i from 0 to (array.length - 2).
    for (int i = 0; i < (array.length - 1); i++) {

        int min = i;

        // Find the smallest value in the array. Note that
        // after each pass, the smallest value is in place,
        // so the loop starts at the position following
        // the last pass in the loop.
        for (int j = i; j < (array.length); j++) {
            if (array[j] < array[min])
                min = j;
        } // end for

        // Swap the next selected array position
        // with the smallest value found on this pass.
        int temp = array[min];
        array[min] = array[i];
        array[i] = temp;
    } // end for
} // end method

// Main method
public static void main(String args[]){
    System.out.println("Welcome to the Array Demo");

    // Create array of ints
    int[] intValues = new int[ARRAYSIZE];
    // Assign random values
    for (int i=0; i<intValues.length; i++) {
```

```

        intValues[i] = (int) (Math.random() * 100);
    }

    System.out.println("*****Unsorted int array*****");
    // Print the unsorted array
    for (int i=0; i<intValues.length; i++) {
        System.out.println(intValues[i]);
    }

    System.out.println("*****");

// Call the Selection Sort method
    selectionSort(intValues);

    System.out.println("*****Sorted int array*****");
    // Print the sorted array
    for (int i=0; i<intValues.length; i++) {
        System.out.println(intValues[i]);
    }

    System.out.println("*****");

    // Generate an array of doubles
    double[] doubleValues = new double[ARRAYSIZE];
    // Assign random values
    for (int i=0; i<doubleValues.length; i++) {
        doubleValues[i] = (Math.random() * 100);
    }

    System.out.println("*****Unsorted double array*****");
    // Print the unsorted array
    for (int i=0; i<doubleValues.length; i++) {
        System.out.println(doubleValues[i]);
    }

    System.out.println("*****");
    // Sort the double value
    Arrays.sort(doubleValues);

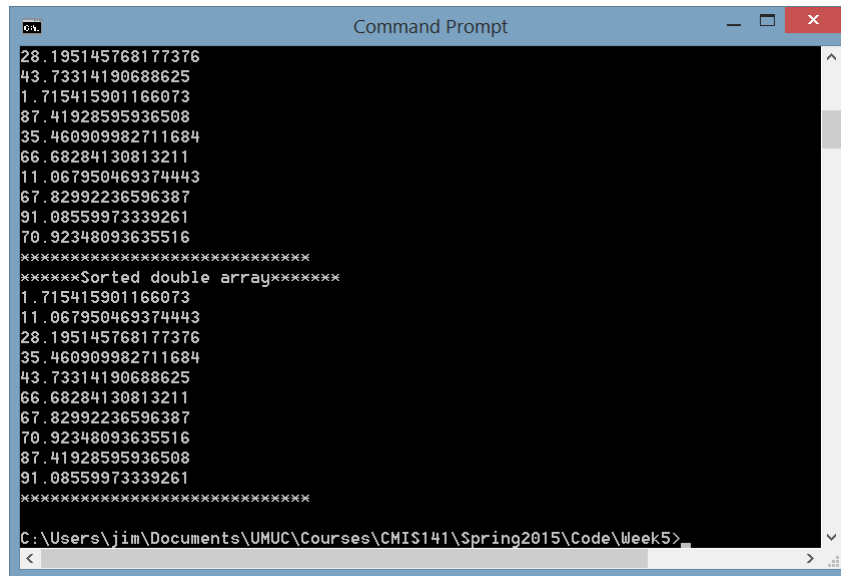
    System.out.println("*****Sorted double array*****");
    // Print the sorted array
    for (int i=0; i<doubleValues.length; i++) {
        System.out.println(doubleValues[i]);
    }

    System.out.println("*****");
}

}

```

- b. Save the file as “SortDemo.java” in a location of your choice.
- c. To compile the file, type `javac SortDemo.java` at the command prompt. Executing the code will yield an output similar to this:



```
28.195145768177376
43.73314190688625
1.715415901166073
87.41928595936508
35.460909982711684
66.68284130813211
11.067950469374443
67.82992236596387
91.08559973339261
70.92348093635516
*****
*****Sorted double array*****
1.715415901166073
11.067950469374443
28.195145768177376
35.460909982711684
43.73314190688625
66.68284130813211
67.82992236596387
70.92348093635516
87.41928595936508
91.08559973339261
*****
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week5>
```

As you analyze and experiment with the code, note the following:

1. Different sorting algorithms produce the same results for the same array of values.
2. When the array is passed into the method, it is passed by reference. This results in the operations being performed on the original array. Hence the order of the original array is not preserved.
3. Using the existing `sort()` method in the `java.util.Arrays` class is straight forward. If you review the Java API, you will see the `sort()` method is static. This is why we can call it using `Arrays.sort()`. Also notice, the method is overloaded with the `sort()` method taking multiple data types.

Now it is your turn. Try the following exercise:

Create a Java class named `MySortDemo` using your favorite text editor. Be sure you name the file "`MySortDemo.java`". Add code to the file in the `main()` method that will construct 50 random float values. Sort the array using the selection and quick sort algorithms. You will need to make some modifications to the selection sort algorithm to adjust for the float values. Display the array values before and after the sorting routines.

### Exercise 3 – Use the `Arrays` class to search for values in an array

Searching is another common programming task. Different algorithms exist including linear and binary searches. In this exercise, we will take advantage of the binary search algorithms already prepared for us in the `Arrays` class.



- a. Open your favorite text editor and type (or copy and paste) the following Java:

```
/*
 * File: SearchDemo.java
 * Author: Dr. Robertson
 * Date: January 1, 2015
 * Purpose: This program demonstrates how to search
 * primitives using the
 * Arrays binary search algorithm
 */

// Import Color class
import java.util.Arrays;

public class SearchDemo {
    // Define constant array size
    static final int ARRAYSIZE = 100;

    // Main method
    public static void main(String args[]){
        System.out.println("Welcome to the Search Demo");

        // Create array of ints
        int[] intValues = new int[ARRAYSIZE];
        // Assign random values
        for (int i=0; i<intValues.length; i++) {
            intValues[i] = (int) (Math.random() * 50);
        }
        System.out.println("*****Unsorted int array*****");
        // Print the unsorted array
        for (int i=0; i<intValues.length; i++) {
            System.out.println(intValues[i]) ;
        }
        System.out.println("*****");

        // Sort the int array value
        Arrays.sort(intValues);

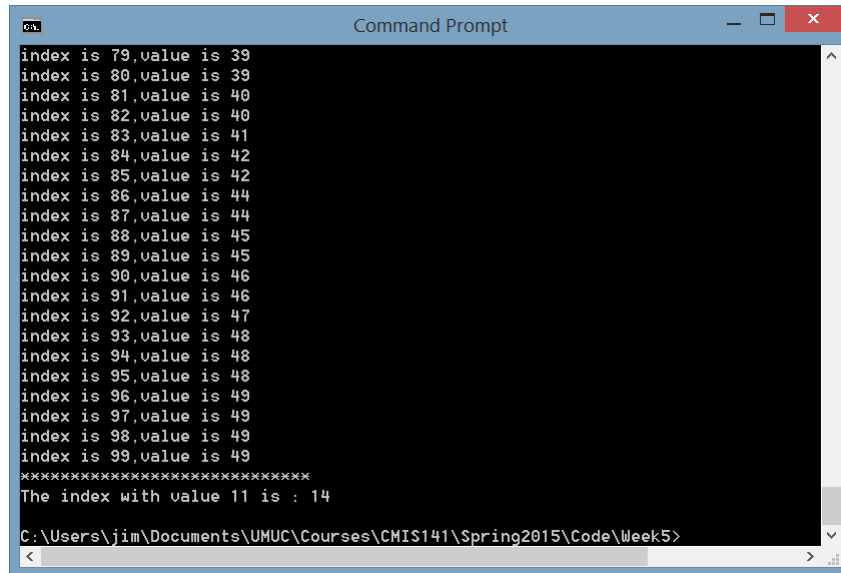
        System.out.println("*****Sorted int array*****");
        // Print the sorted array
        for (int i=0; i<intValues.length; i++) {
            System.out.println("index is " +i+ ", " + "value is " + intValues[i]);
        }
        System.out.println("*****");

        // Now we can search
        // Remember we must sort before we can search
        // when using a binary search approach
        int lookFor = 11;

        int arrIndex = Arrays.binarySearch(intValues,lookFor);

        System.out.println("The index with value 11 is : " + arrIndex);
    }
}
```

- b. Save the file as “SearchDemo.java” in a location of your choice.
- c. To compile the file, type `javac SearchDemo.java` at the command prompt. Executing the code will yield an output similar to this:



```
index is 79,value is 39
index is 80,value is 39
index is 81,value is 40
index is 82,value is 40
index is 83,value is 41
index is 84,value is 42
index is 85,value is 42
index is 86,value is 44
index is 87,value is 44
index is 88,value is 45
index is 89,value is 45
index is 90,value is 46
index is 91,value is 46
index is 92,value is 47
index is 93,value is 48
index is 94,value is 48
index is 95,value is 48
index is 96,value is 49
index is 97,value is 49
index is 98,value is 49
index is 99,value is 49
*****
The index with value 11 is : 14
C:\Users\jim\Documents\UMUC\Courses\CMIS141\Spring2015\Code\Week5>
```

As you analyze and experiment with the code, note the following:

1. To perform the binary search you must sort the data first.
2. If you have duplicates in your array, the binary search using in the Arrays may not return the first occurrence.
3. If no values are found you will most likely receive an index less than 0.

Now it is your turn. Try the following exercise:

Create a Java class named `MySearchDemo` using your favorite text editor. Be sure you name the file “`MySearchDemo.java`”. Add code to the file in the `main()` method that will construct 20 random int values between 1 and 100. Sort the array using the Arrays quick sort algorithm. Prompt the user to enter a value they want to search and then perform the Arrays binary search method to perform the search. Return the index containing the value. If no results are found output a message that the data was not found.

#### Exercise 4 - Create Multidimensional arrays in Java

Multi-dimensional arrays are useful for storing data that represents more than one dimension. For example, the average precipitation for one month can easily be stored in a 1D array. If you wanted to keep track of the specific month as well as the precipitation each day, a 2D array would be more useful.

Consider the following 2D array that uses indexes to keep track of the month of the year and the day of the month for each data value.

```
double[][] precipitation = new double[12][31];
```

This array will allow for the storage of 12 months and 31 days for each month. Of course, some months won't need all 31 days but at least the capacity is there.

You can also move to a 3D array if you wanted to store years. For example, if you wanted to store 7 years of precipitation data, the following array declaration would suffice.

```
double[][][] precipitation = new double[7][12][31];
```

For this exercise we will create and populate a 2D and 3D array for storing precipitation data. We will use the `Math.random()` method to generate data for each of the cells.

- a. Open your favorite text editor and Type (or copy and paste) the following Java:

```
/*
 * File: MultiDimensionDemo.java
 * Author: Dr. Robertson
 * Date: January 1, 2015
 * Purpose: This program demonstrates how to create
 * and populate 2D and 3D arrays in Java
 */

// Import Color class
import java.util.Arrays;

public class MultiDimensionDemo {
    // Define constant array size
    static final int NUMYEARS = 7;
    static final int NUMMONTHS = 12;
    static final int NUMDAYS = 31;

    // Main method
    public static void main(String args[]){
        System.out.println("Welcome to the MultiDimensional Array Demo");

        // Create array of doubles 2 D
        double[][] precip2D = new double[NUMMONTHS][NUMDAYS];
        // Assign random values
        for (int i=0; i<NUMMONTHS; i++) {
            for (int j=0; j<NUMDAYS; j++) {
                precip2D[i][j] =
Double.parseDouble(String.format("%.2f",Math.random() * 10));
            }
        }

        // Print the results for each cell in a table 31x12
        // Switching the printing order to display months in columns
        for (int j=0; j<NUMDAYS; j++) {
            for (int i=0; i<NUMMONTHS; i++) {
                System.out.print(precip2D[i][j] + "\t");
            }
        }
    }
}
```

```

        System.out.println("");
    }

    // Create array of doubles 3D
    double[][][] precip3D = new double[NUMYEARS][NUMMONTHS][NUMDAYS];
    // Assign random values
    for (int k=0; k<NUMYEARS; k++) {
        for (int i=0; i<NUMMONTHS; i++) {
            for (int j=0; j<NUMDAYS; j++) {
                precip3D[k][i][j] =
Double.parseDouble(String.format("%.2f",Math.random() * 10));
            }
        }
    }

    // Print results with Year upfront
    // Define BaseYear
    int baseYear = 2000;
    for (int k=0; k<NUMYEARS; k++) {
        System.out.println("Year is " + (baseYear+k));
        for (int j=0; j<NUMDAYS; j++) {
            for (int i=0; i<NUMMONTHS; i++) {
                System.out.print(precip3D[k][i][j] + "\t");
            }
            System.out.println("");
        }
    }
}
}

```

- b. Save the file as “MultiDimensionalDemo.java” in a location of your choice.
- c. To compile the file, type `javac MultiDimensionalDemo.java` at the command prompt. Executing the code will yield an output similar to this:

```

Command Prompt
4.92 7.52 7.87 8.99 2.49 4.86 4.02 2.19 7.13 9.82 3.55 8.11
9.63 3.02 4.97 4.09 2.43 5.32 3.95 0.88 0.92 1.92 2.52 6.67
2.19 5.44 9.54 4.27 6.75 1.48 6.74 1.85 4.62 1.88 0.16 3.85
2.9 1.6 3.75 4.61 7.19 4.89 0.78 3.9 1.29 1.66 3.05 3.53
5.09 5.03 9.0 2.17 5.69 4.72 8.42 0.96 0.96 4.27 2.75 6.43
2.04 4.42 7.74 1.98 5.27 5.51 3.38 8.45 3.5 4.23 8.07 5.48
5.13 3.86 4.08 0.89 7.15 7.31 2.6 4.1 0.09 8.2 9.88 3.77
Year is 2006
4.09 7.3 2.88 8.98 5.45 4.77 3.98 5.13 3.4 8.09 2.76 5.49
6.92 5.68 2.95 4.48 9.92 1.34 9.33 3.36 1.32 0.83 9.16 6.58
5.27 3.98 1.33 9.38 4.08 6.95 4.46 6.01 7.37 9.93 8.35 6.41
4.29 3.53 1.65 3.23 8.16 8.58 5.45 0.84 4.47 3.14 2.12 6.32
5.18 6.2 2.96 0.25 9.2 4.82 9.27 4.63 9.38 4.3 2.56 1.29
2.51 8.72 7.7 2.01 5.51 5.44 2.32 1.85 4.64 1.74 8.29 4.5
1.23 1.27 6.15 7.85 2.93 3.51 1.62 7.13 8.92 6.18 4.67 8.08
6.22 5.81 6.09 9.43 0.84 4.52 8.06 8.58 7.72 5.03 2.63 4.89
3.85 6.08 0.94 4.41 3.11 7.27 7.53 0.69 8.5 5.6 4.12 4.68
1.9 3.78 6.3 8.48 8.31 9.81 7.98 2.43 3.05 7.56 8.83 9.62
4.99 7.43 5.78 3.81 1.32 4.72 8.17 0.45 9.75 1.86 1.1 8.09
8.8 7.85 4.2 6.6 4.73 4.0 8.22 3.36 7.58 2.91 6.31 1.27
2.88 9.45 9.67 1.71 3.48 6.01 5.86 6.23 0.04 9.05 3.53 8.03
4.58 7.27 2.57 2.69 5.83 2.44 9.58 6.63 6.81 7.88 5.17 9.23
4.26 3.08 3.45 9.45 4.02 4.4 9.02 9.01 4.62 8.82 9.44 3.19
3.22 5.68 2.35 2.53 7.85 4.08 5.27 1.63 8.65 7.32 4.76 0.78
1.24 1.93 8.96 1.72 6.27 3.91 7.56 6.85 7.33 5.28 4.56 5.96
2.26 6.01 8.89 0.22 2.48 8.09 6.62 6.85 1.2 0.08 2.28 4.12
0.07 6.91 1.69 4.23 2.11 8.76 6.72 4.84 7.99 9.52 1.6 6.25

```

As you analyze and experiment with the code, note the following:

1. Constants can be used to define the array size values:

```
static final int NUMYEARS = 7;
static final int NUMMONTHS = 12;
static final int NUMDAYS = 31;
```

2. Nested loops are used to populate the 2D and 3D arrays:
3. The `String.format()` method along the `Double.parseDouble()` are used to store the data using only 2 decimal places. `String.format()` returns a `String` requiring the `Double.parseDouble()` method to take that string and convert it to a double value.

```
precip2D[i][j] = Double.parseDouble(String.format("%.2f",Math.random()*10));
```

4. Depending upon the desired format of the output, you may need to switch the nested loops. For example, to display the data in 12 columns representing each month the inner and outer loops are switched.

```
for (int j=0; j<NUMDAYS; j++) {
    for (int i=0; i<NUMMONTHS; i++) {
        System.out.print(precip2D[i][j] + "\t");
    }
    System.out.println("");
}
```

Now it is your turn. Try the following exercise:

Create a Java class named `My3DDemo` using your favorite text editor. Be sure you name the file `"My3DDemo.java"`. Add code to the file in the `main()` method that will construct 3D array consisting of random int values between 1 and 30. The size of the 3D array should be `5x20x10`. Print the results in an organized table of 10 columns.