# Object-Oriented Development (CIS1056-N) Worksheet 05: Arrays

## Before You Start

Remember: You are not expected to complete the entire brief within the allotted two hours, but to make a start and continue outside of the class.

Ensure you have completed all assessment tasks from Worksheet 1 and are comfortable with Java primitive types before beginning this worksheet. Attempt to complete this set of tasks before your next session. Any issues seek help from your tutors.

**Hint:** It is good practice to plan your programs first on paper using pseudocode. When we say pseudocode, we mean code that is halfway between English and a programming language, such as Python.

Read more about it here: <https://en.wikipedia.org/wiki/Pseudocode>

## Introduction

Last week we were introduced to arrays to store collections of (typically) related data. Arrays are a fixed size and can only store data of the same data type. Prior to arrays, we’d discovered and practiced with *Iteration* using while, for and do … while loops, and more recently the for-each loop – hopefully you have a good understanding of *Iteration* generally, and also when you might choose one particular loop over another. Arrays and Iteration work together, as you use Iteration to process (by iterating over) a collection of values in an array. The *indexed* for and the for-each are the most common looping constructs that are used with sequenced collection types such as arrays. This exercise sheet is to practice the use of arrays and iteration to process the arrays.

## 1. Basic Arrays

**public class WK05Arrays {**

**public static void main(String[] args) {**

**int[] array = { 12, 3, 5, -1, 0, 5, 99, 10 };**

**for(int index = 0; index < 8; index++) {**

**System.out.printf(**

**"Index %d: Value: %d\n",**

**index,**

**array[index]);**

**}**

**}**

**}**

1. Create a new Netbeans project and copy the code above into it. Run it and test it.
2. Remove a value from the **array**, for example:  
     
   **int[] array = { 12, 3, 5, 0, 5, 99, 10 };** // -1 removed  
     
   Re-run and test it.
   1. What happened and why?
   2. If there was a problem, fix it.
3. Generate another random number in the range **1** to **100** and test if this number appears in the **array**.

## 2. Integer Array – Lottery Numbers

1. Create a new project called ‘**WK05\_LotteryNumbers**’. Type in and then test the following Java code that allows the user to enter 7 numbers (integers) and print them to the console – You will need to create a new class called **Lottery**.

**public class Lottery {**

**public static void main(String[] args) {**

**Scanner keyboard = new Scanner(System.in);**

**int[] lotteryNumbers = new int[7];**

**System.out.println("Please enter 7 numbers");**

**for(int index = 0; index < lotteryNumbers.length; index++) {**

**System.out.println(**

**String.format("Number %d:", index + 1));**

**int number = keyboard.nextInt();**

**lotteryNumbers[index] = number;**

**}**

**for(int index = 0; index < lotteryNumbers.length; index++) {**

**System.out.println(**

**String.format(**

**"Number %d: %3d",**

**index + 1,**

**lotteryNumbers[index]));**

**}**

**}**

}

1. Comment the code.
2. Update the program developed in **part (a)** so that duplicate numbers are not allowed. The user should be informed that the number has already been entered and asked to re-enter the number. For example (underlined number are inputs):

Please enter 7 numbers

Number 1: **10**

Number 2: **7**

Number 3: **10**

The number (10) has already been entered – please re-enter:

1. Update your program to change the prompts to the following:

|  |  |  |
| --- | --- | --- |
| Please  First | enter 7  Number: | numbers  **10** |
| Second | Number: | **7** |
| Third | Number: | **8** |
| Fourth | Number: | **21** |
| Fifth | Number: | **48** |
| Sixth | Number: | **19** |

Bonus Ball Number: **7**

Hint: Use an array to store the position name (eg: First, Second, etc).

## 3. Populations

Create and test a new Java Application called Population. This application will produce some basic statistics about ages.

You will create an array of 10 elements that will store ages of people. The user will be asked for the age of a person which is put into the array, this is repeated until all 10 elements of the array have been set with a value.

You should validate the input to prevent the user adding erroneous data (e.g. negative numbers, impossible ages).

The program will now compute the following information about the age data held in the array:

1. The minimum age.
2. The maximum age.
3. The mean average age.
4. The median average age.

## 4. Array exercises

Create a new Java Application called ArrayExercises. Create a new Java class with its own **main()** method for each of the exercises below.

Create a 20-element array that is populated with randomly generated numbers in the range **1** to **30**. Do the following to the array:

1. Display the values stored in the array.
2. Modify (1) to print the index in the array.
3. Ask the user to input a number between **1** to **30**. Count the total number times this number appears in the array.
4. Display all the indexes in the array where the number is found.
5. Ask the user for a number in the range **1** to **30**. Replace all instances of this number in the array with **0**. Display the contents of the array, ignoring any elements of the array that have been set to **0**.
6. Create a new array whose size is that of original array minus the number of elements that contain **0**. Copy all the non-zero values into the new array, then display its contents.
7. Create an array that is populated with random numbers in the range **-10** to **10**. Display a suitable message to the user if the array is dominated by negative values.

## 5. Integer Array

For this exercise you are required to use the Java Arrays API: <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Arrays.html>

1. Declare an array of integers. This array should have enough room for 100 values. Use the Arrays.fill() to fill every slot in this array with the value 19. Use a for loop to iterate through this array, printing each element to console.
2. Look at the API for the Arrays class. Find the method that allows you to fill slots 51-100 with the value 25, without disturbing slots 1-50. Check your work. Use an enhanced for loop to iterate through the array, printing each element to console.

## 6. Sorting Arrays

For this exercise you are required to use the Java Arrays API: <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Arrays.html>

1. Declare an array of 10 integers using the shortcut declaration:  
     
   **int[] nums = {5, 4, 76, 12, 54, 1, 6, 999, 998, 821};**
2. Use **Arrays.sort()** to sort this array. Print the contents of the array to console.
3. Look at the API for the **Arrays** class. Identify the method that allows you to search for a specific value. Search for the value 5. Print out the index position of this int value.

## 7. Employee Manager (Revisited)

In week 2 (‘if’ conditions) you created a project called EmployeeManager that calculated and displays the weekly wage (without tax) from the number of hours worked and hourly rate. Copy this project, changing the project name to EmployerManagerV2.

This version of the program will:

1. Ask the user how many employees they have.
2. Create an array big enough to store the wages for the user specified number of employees.
3. The user will then enter hours worked, including overtime, and calculate the total wage for each employee. This will be stored in the array. Revisit *Worksheet 2*, *exercise 3 (c)* for more information.
4. Once all the wages have been calculated and their values stored in the array the program will output all the wages and the total wages.  
   The format should be:  
     
   **Employee #1: 341.25  
   Employee #2: 175.50  
   Employee #3: 447.00  
   ----------------------  
   Total: 963.75  
   ----------------------**

## 8. Lookup Table

Create 2 arrays of type String.

The first array will be 7 elements in size and will contain the names of the days of the week in order, for example:

**String[] daysOfTheWeek = {**

**"Monday", "Tuesday", "Wednesday",**

**"Thursday", "Friday",**

**"Saturday", "Sunday"**

**};**

The second array hold 12 elements and will contain the names of the months of the year in order.

1. Ask the user to input 2 numbers, the ordinal day of the week (1 to 7) and ordinal month of the year 1 to 12. All user input should be validated. With these two values you will now display the corresponding day name and month name.
2. **Extended version (1)**: Change the program so that instead of reading 2 integer values, the user inputs the day and month in the format: **D-MM**. For example, if the user entered **7-06**, the program would output: **SUNDAY JUNE**.
3. **Extended version (2)**: Remove **"Saturday"** and **"Sunday"** from the **daysOfTheWeek** array. Change the behaviour of the program so that if the user inputs **6** or **7** for the day of the week, then it simple displays **"Weekend"**.

## 9. Coin Game.

Design, write and test a coin-tossing program. This program should have a variable that indicates whether a result was heads or tails. What type should this instance variable be?  
  
The method to toss the coin requires a random number, either 0 or 1.

We can get a random number using a method of the Math class. Math.random() returns a double value between 0 and 1. To convert this value to an integer, either 0 or 1, the following code can be used:

int randomNum = (int)(Math.random() \*2); //returns an integer

1. Run your coin-tossing program. It should print out the result (heads or tails).
2. Now alter the main method to toss the coin 100 times and count the number of times the coin toss results in heads and the number of time it result in tails. You will need a loop for this, iterating 100 times.
3. Display the heads count and tails count.

Link to the Random class documentation: <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Random.html>

## 10. Consecutive values

Write a program that given an array of 100 elements of random numbers in the range 0 to 20 will determine the maximum number of consecutive array elements that contain the same number. For example, if we had this small array:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 18 | 18 | 2 | **1** | **1** | **1** | 3 | 1 | 1 |

The maximum number of consecutive array elements with the same value would be 3 (the 3 1’s in red).

1. **Extended version 1**: Keep track the value that has the longest run of consecutive entries in the array. Output this value.
2. **Extended version 2**: Keep track of the indices of the array where the longest run of consecutive values start and end. Output these values. For example, from small array above the indices would reported as **4** and **6**.
3. Can you re-apply your solution to the Coin Game exercise?

## Document History

Revision 0 (25-Sep-22): This is the initial version of the 2022/23 exercise.

Revision 1 (25-Oct-22): Fixed some typos. Added some additional extension exercises.

Revision 2 (27-Oct-22): Modified code example for Lottery exercise to use expected naming convention for variable name.

Revision 3: (27-Oct-22): Rephrased question 8 to make it clearer.