Mohammad Ali Jinnah University Karachi

Department of Computer Science

LAB MANUAL

CS1421: Object Oriented Programming
LAB 03

Instructors
Safiyah Batool
Samia Bashir

Lab 1
Arrays, String and Math Classes

Activity time-boxing

Task no.	Activity name	Activity time
1	Lab discussion	30 min
2	Walkthrough tasks	30 min
3	Think it on! (Practice Tasks)	60 min
4	Evaluation Tasks	40 min

Table 1: Activity Time-boxing

Overview

Arrays

An array stores a sequence of values that are all of the same type. We want not just to store values but also to be able to quickly access each individual value. The length of an array is established when the array is created. After creation, its length is fixed. Each item in an array is called an *element*, and each element is accessed by its numerical *index*. The method that we use to refer to individual values in an array is to number and then *index* them—if we have *n* values, we think of them as being numbered from 0 to *n*−1.

Making an array in a Java program involves three distinct steps:

- Declare the array name.
- Create the array.
- Initialize the array values.

We refer to an array element by putting its index in square brackets after the array name.
To use an array in a program, you must declare a variable to reference the array and specify the array's *element type*. Here is the syntax for declaring an array variable:

```
elementType[] arrayRefVar;
```

The **elementType** can be any data type, and all elements in the array will have the same data type.

Unlike declarations for primitive data type variables, the declaration of an array variable does not allocate any space in memory for the array. It creates only a storage location for the reference to an array. If a variable does not contain a reference to an array, the value of the variable is **null**. You cannot assign elements to an array unless it has already been created. After an array variable is declared, you can create an array by using the **new** operator and assign its reference to the variable with the following syntax:

```
arrayRefVar = new elementType[arraySize];
```

```
public class ArrayExample {
   public static void main(String[] args) {
        int[] arr;            //Declaration
        arr=new int[10];      //creation
        arr[0]=2;             //initialization
        arr[1]=3;
        arr[2]=43;
        arr[3]=54;
        arr[4]=43;
        arr[5]=53;
        arr[6]=21;
        arr[7]=2;
        arr[8]=6;
        arr[9]=-3;
       for(int i=0;i<arr.length;i++) {
           System.out.println(arr[i]);
       }
   }
}
```

Java has a shorthand notation, known as the *array initializer*, which combines the declaration, creation, and initialization of an array in one statement using the following syntax:

```
elementType[] arrayRefVar = {value0, value1, ..., valuek};
```

```
public class Array2Example {
   public static void main(String[] args) {
        float[] arr={2.3f,4.5f,3.6f,3.2f,8.4f};
        for(int i=0;i<arr.length;i++) {
```

```
            System.out.println(arr[i]);
        }
    }
}
```

Arrays Class

Arrays class which is in `java.util.Arrays` package, is a provision by Java that provides you a number of methods through which arrays can be manipulated. This class also lets you perform sorting and searching operations on an array.

Exaxmple:

```
//Sort and print an array using java.util.Arrays
import java.util.Arrays;
public class ArraySorting {
   public static void main(String[] args) {
        int[] arr={5,3,6,3,2,8,0,7,1};
        Arrays.sort(arr);
        System.out.println("Sorted Array[]: "+Arrays.toString(arr));

   }
}
```

Eample;

```
//Array Searching
public class ArraySearching {
   public static void main(String[] args) {
        String[] dept={"CS","EE","CSE","BIO-MEDICAL"};
        System.out.println(Arrays.binarySearch(dept,"CS"));
        System.out.println(Arrays.binarySearch(dept,"EE"));
   }
}
```

```
// To Sort sub Array
public static void main(String[] args) {
        int[] arr={5,3,6,3,2,8,0,7,1};
        Arrays.sort(arr,0,4);// Sort 0-3 index
        System.out.println("Sorted Array[]:
"+Arrays.toString(arr));
```

```
    }
}
```

```
//To check if two arrays are equal or not.
public class ArrayEquals {
   public static void main(String[] args) {
        int[] arr={34,54,23,53,53};
        int[] arr2={43,23,54,75,32};
        System.out.println("Is arr1 equals to arr2:
"+Arrays.equals(arr,arr2));
    }
}
```

```
//Deep Equals Method to Check the equality for multi-dimensional Arrays and print MD array
public static void main(String[] args) {
   int marks1[][]={{23,32},{23,32}};
   int marks2[][]={{23,32},{23,32}};
   int marks3[][]={{27,32},{63,32}};
   System.out.println("Is marks1[] equals to marks2[] "+
Arrays.deepEquals(marks1,marks2));
   System.out.println("Is marks2[] equals to marks3[] "+
Arrays.deepEquals(marks2,marks3));
   System.out.println(Arrays.deepToString(marks1));
}
```

Java Math Class

The Java programming language supports basic arithmetic with its arithmetic operators: +, -, *, /, and %. The Math class provides methods and constants for doing more advanced mathematical computation.

The Math is located in the java.lang package, and not in the java.math package. Thus, the fully qualified class name of the Math class is java.lang.Math .

The methods in the Math class are all static, so you call them directly from the class, like this:
Math.cos(angle);

Note: Using the static import language feature, you don't have to write Math in front of every math function:
import static java.lang.Math.*;

This allows you to invoke the Math class methods by their simple names. For example: cos(angle);

```
Example:
public class MathClassExample {
   public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.println("Enter any Number to find it's
Cosine");
        int num=input.nextInt();
        System.out.println("Cosine of "+num+" is
"+Math.cos(num));
   }
}
```

Constants

The Math class includes two constants:

- Math.E, which is the base of natural logarithms, and
- Math.PI, which is the ratio of the circumference of a circle to its diameter.

Basic Math Mehtods

The Math class includes more than 40 static methods. They can be categorized as *trigonometric methods*, *exponent methods*, and *service methods*. Service methods include the rounding, min, max, absolute, and random methods.

Trigonometric Methods

The `Math` class contains the following methods.

The parameter for `sin`, `cos`, and `tan` is an angle in radians. The return value for `asin`, `acos`, and `atan` is a degree in radians in the range between -pi/2 and pi/2. One degree is equal to pi/180 in radians, 90 degrees is equal to pi/2 in radians, and 30 degrees is equal to pi/6 in radians.

Exponent Methods

There are five methods related to exponents in the `Math` class.

The Rounding Methods

The `Math` class contains five rounding methods

The Service Methods

The **`min`**, **`max`**, and **`abs`** Methods

The **min** and **max** methods return the minimum and maximum numbers of two numbers (**int**, **long**, **float**, or **double**). For example, **max(4.4, 5.0)** returns **5.0**, and **min(3, 2)** returns **2**.

The **abs** method returns the absolute value of the number (**int, long, float,** or **double**). This method generates a random **double** value greater than or equal to 0.0 and less than 1.0 (**0 <= Math.random() < 1.0**). You can use it to write a simple expression to generate random numbers in any range.

Example:

```
import java.util.Scanner;

public class MathClassExample {
   public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.println("Enter any Number");
        int num=input.nextInt();
        System.out.println("Square root: "+Math.sqrt(num));
        System.out.println("Maximum is "+Math.max(num,100));
        System.out.println("Minimum is "+Math.min(num,0));
        System.out.println("Absolute is "+Math.abs(num));
        System.out.println("cube is "+ Math.pow(num,4));
        System.out.println("Cuberoot is "+Math.cbrt(num));
        System.out.println("Radian is "+ Math.toRadians(num));
        System.out.println("Inverse sine is "+Math.asin(num));
        System.out.println("Cosine of "+num+" is
"+Math.cos(num));
    }
}
```

Example:

```
public class Circumference {
   public static void main(String[] args) {
        Scanner input= new Scanner(System.in);
        System.out.println("Enter Radius");
        int r=input.nextInt();
        System.out.println("Circumference is "+(2*Math.PI*r));
    }
}
```

Example

```
public static void main(String[] args) {
   double randomNumber=(Math.random());
   System.out.println(randomNumber);
   System.out.println(Math.floor(randomNumber));
   System.out.println(Math.ceil(randomNumber));
   System.out.println(Math.log(randomNumber));

}
```

Example:
To generate random number in range

```
public class RandomNumber {
   public static void main(String[] args) {
        //to generate random number from 1 to 10
        // minimun+math.random()*maximum
        int randomNumber=(int)(1+Math.random()*10);
        System.out.println(randomNumber);
   }
}
```

Java String Class

The **`char`** type represents only one character. To represent a string of characters, use the data type called **`String`**. For example, the following code declares **message** to be a string with the value **"Welcome to Java"**.

```
String message = "Welcome to Java";
```

`String` is a predefined class in the Java library, just like the classes **`System`** and **`Scanner`**. The **`String`** type is not a primitive type. It is known as a *reference type*. Any Java class can be used as a reference type for a variable. The variable declared by a reference type is known as a reference variable that references an object. Here, **`message`** is a reference variable that references a string object with contents **`Welcome to Java`**.

The `java.lang.String` class provides a lot of methods to work on string. By the help of these methods, we can perform operations on string such as trimming, concatenating, converting, comparing, replacing strings etc.

Getting String Length

You can use the **`length()`** method to return the number of characters in a string. For example, the following code

```
String message = "Welcome to Java";
System.out.println("The length of " + message + " is "+
message.length());
```

Getting Characters from a String

The **`s.charAt(index)`** method can be used to retrieve a specific character in a string **s**, where the index is between **`0`** and **`s.length()-1`**. For example, **`message.charAt(0)`** returns the character **W**, as shown in figure. Note that the index for the first character in the string is **`0`**.

Example:

```
public static void main(String[] args) {
    String uni="Muhammad Ali Jinnah University";
    String uni1="Muhammad Ali Jinnah University";

    String uni2=",Karachi";
    String ends="y";
    System.out.println(uni.concat(uni2));
    System.out.println(uni.length());
    System.out.println(uni.compareTo(uni1));
    System.out.println(uni.compareTo(ends));
    System.out.println(uni.charAt(2));
    System.out.println(uni);
    System.out.println(uni.endsWith(ends));
}
```

Converting Strings

The **`toLowerCase()`** method returns a new string with all lowercase letters and the **`toUpperCase()`** method returns a new string with all uppercase letters.

For example,

`"Welcome".toLowerCase()` returns a new string **welcome**.

`"Welcome".toUpperCase()` returns a new string **WELCOME**.

The **`trim()`** method returns a new string by eliminating whitespace characters from both ends of the string. The characters **' '**, **\t**, **\f**, **\r**, or **\n** are known as *whitespace characters*.

Example:

```
public static void main(String[] args) {
    String upperCase=" ABC ";
    String str1="MA ";
```

```
    String str2=" JU";
    System.out.println("Upper Case: "+upperCase);
    String lowerCase=upperCase.toLowerCase();
    System.out.println("Lower Case: "+lowerCase);
    System.out.println(str1+str2);
    System.out.println(str1.trim()+str2.trim());
}
```

Practice Tasks

What is the output of the following code?

```
int[] values = new int[5];
for (int count = 0; count < 5; count++)
      values[count] = count + 1;
for (int count = 0; count < 5; count++)
      System.out.println(values[count]);
```

Task 1

Generate vehicle plate numbers

Assume a vehicle plate number consists of three uppercase letters followed by three digits. Write a program to randomly generate a plate number.

Task 2

Rainfall

Write a RainFall class that stores the total rainfall for each of 12 months into an array of doubles. The program should return the following:

- the total rainfall for the year
- the average monthly rainfall
- the month with the most rain
- the month with the least rain

Input Validation: Do not accept negative numbers for monthly rainfall figures.

Evaluation Tasks:

Task 3a

Random Number Guessing Game

Write a program that generates a random number and asks the user to guess what the number is. If the user's guess is higher than the random number, the program should display "Too high, try again." If the user's guess is lower than the random number, the program should display "Too low, try again." The program should use a loop that repeats until the user correctly guesses the random number.

Task 3b

Random Number Guessing Game Enhancement

Enhance the program that you wrote for Task 3a so it keeps a count of the number of guesses that the user makes. When the user correctly guesses the random number, the program should display the number of guesses.

Task 4

Write a program that plays a simple dice game between the computer and the user. When the program runs, a loop should repeat 10 times. Each iteration of the loop should do the following:

•Generate a random integer in the range of 1 through 6. This is the value of the computer's die.

•Generate another random integer in the range of 1 through 6. This is the value of the user's die.

•The dice with the highest value wins. (In case of a tie, there is no winner for that particular roll of the dice.)

As the loop iterates, the program should keep count of the number of times the computer wins, and the number of times that the user wins. After the loop performs all of its iterations, the program should display who was the grand winner, the computer or the user.

Task 5

(Reverse the numbers entered)

Write a program that reads ten integers and displays them in the reverse of the order in which they were read.

Task 6

Beautifying the sentences

Write an application that takes a sentence from user and checks if the sentence stars from a capital letter and ends with a full stop. If it doesn't, the program should add it.