OBJECT-ORIENTED PROGRAMMING
LAB 2: JAVA, HOW TO PROGRAM (CONT.)

I. Objective

In this second tutorial, you will:

- Practice with an array in Java.
- Have basic knowledge about the object, how to create the object in Java, and practice it with the sample class defined by Java.

II. Array

Java provides a data structure, the array, which stores a fixed-size sequential collection of elements of the same type. To declare an array, we have 4 possible ways:

```
dataType[] arrayName;
dataType arrayName[];
dataType[] arrayName = new dataType[arraySize];
dataType[] arrayName = {value0, value1, ..., valueK};
```

Example:

```
public class MyFirstProgram {
    public static void main(String[] args) {
        int[] a = {1, 2, 3, 4, 5};
        int sum1 = 0, sum2 = 0;
        for (int i = 0; i < a.length; i++) {
            sum1 = sum1 + a[i];
        }
        System.out.println("sum1 = " + sum1);
        for (int x : a) {
            sum2 = sum2 + x;
        }
        System.out.println("sum2 = " + sum2);
    }
}
```

We have a special for loop in the above sample called *enhanced for* loop. The *enhanced for* loop is mainly used to traverse a collection of elements including arrays. The syntax is as follows.

Example:

```
public class MyFirstProgram {
    public static void main(String[] args) {
        int[] a = {1, 3, 5, 7, 9};
        for (int x : a) {
            System.out.println(x);
        }
    }
}
```

III. Classes and objects in Java

Java is an *Object-Oriented Programming* (OOP) language. Everything you work in Java is through classes and objects. In this lab, we just learn how to create an object from the available class in Java, we will learn about this topic more carefully in Lab 4.

BigDecimal is a class, which is defined in **java.math** package. This class provides operations for arithmetic, scale manipulation, rounding, comparison, hashing, and format conversion. Besides that, the *BigDecimal* class gives its user complete control over rounding behavior. You can read more about this class here.

Object is an instance of a Class.

Example:

```
import java.math.BigDecimal;

class Test {
    public static void main(String[] args) {
        BigDecimal num = new BigDecimal(1);
        BigDecimal num1 = new BigDecimal(4);
        BigDecimal x = num;
        // BigDecimal y;
        System.out.println(num);
        System.out.println(num1);
        System.out.println(x);
        // System.out.println(y);
    }
}
```

Output:

```
1
4
1
```

With the above example, the variables: *num*, *num1*, *x*, and *y* were created from the BigDecimal class. In other words, the variables: *num*, *num1*, *x*, and *y* are the pointers to the objects. However, variable y hasn't been initialized. Therefore, the value of that variable will be undermined until an object is created and assigned to it. If you try using an uninitialized variable, you will get a compiler error.

The format of code when you want to create an object from the class is:

```
ClassName instanceName = new ClassConstructor([paramater1, parameter2, …])
```

Let's observe another example:

```java
import java.math.BigDecimal;

class Test1 {
    public static void main(String[] args) {
        BigDecimal x = new BigDecimal(4);
        BigDecimal y = new BigDecimal(4);

        System.out.println(x == y);
        System.out.println(x.equals(y));
    }
}
```

Output:

```
false
true
```

With objects, we don't use the operator "==" to compare values, because this operator will compare the addresses and there are the pointers so they have different addresses. *BigDecimal* supports the method called **equals** to compare the value of two *BigDecimal* objects.

Next, we will learn how to invoke the **non-static method** and **static method** from a class. To invoke a static method, you do not need to create an object. Instead, you can invoke directly from the class name. Contrarily, with the non-static method, you need to create an object to invoke it.

Example:

```java
import java.math.BigDecimal;

class Test2 {
    public static void main(String[] args) {
        BigDecimal x = new BigDecimal(-4);

        //abs() is a non-static method of the BigDecimal class
        BigDecimal y = x.abs();

        //valueOf() is a static method of the BigDecimal class
        BigDecimal z = BigDecimal.valueOf(20.22);

        System.out.println(y);
        System.out.println(z);
    }
}
```

Output:

```
4
20.22
```

IV. Exercises

1. Write a function `public static int findMax(int arr[])` to find the maximum value of an array.
2. Write a function to find the minimum value of an array.
3. Write a function to sum all even numbers of an array.
4. Write a function to count how many specific elements are in an array.
5. Write a function to count how many prime numbers are in an array.
6. Write a function `public static int find(int arr[], int k)` to find the index of an element *k* in an array, if the element doesn't exist in an array return -1. (the first element index is 0)
7. Write a function `public static void square(int arr[])` to square all elements of an array.
8. Write a function `public static BigDecimal findMax(BigDecimal []arr)` to find the maximum value of a BigDecimal object array.
9. *Write a function `public static int[] divisibleNumbers(int arr[], int k)` to find the elements divisible by *k* in an array. (*Hint: You can use two loops to solve it. The first loop uses to count how many possible elements. Create a new array with a length equal to the number counted in the first loop. The second loop uses to put all possible elements into the array.* Ex: a = [1,2,3,4,5,6,7] with *k* = 2 → [2,4,6]).
10. *Write a function to find the third largest element in an array.

-- END --