OBJECT-ORIENTED PROGRAMMING
LAB 4: REVIEW

I. Objective

After completing this tutorial, we want you to:

- Review about Array in Java;
- Review String in Java;
- Review OOP, Class, and Encapsulation in Java.

II. Array

1. Array is used to store multiple values in a single variable, instead of declaring separate variables for each value:

```
String[] str = {"IT", "TDTU", "HCM"};
int[] nums = {10, 20, 30, 40};
```

2. You access an array element by referring to the index number:

```
int[] nums = {10, 20, 30, 40};
System.out.println(nums[1]); // 20
```

3. To change the value of a specific element, refer to the index number:

```
int[] nums = {10, 20, 30, 40};
nums[1] = 15;
System.out.println(nums[1]); // 15
```

4. To find out how many elements an array has, use the `length` property:

```
int[] nums = {10, 20, 30, 40};
System.out.println(nums.length); // 4
```

5. You can loop through the array elements with the `for` loop, and use the `length` property to specify how many times the loop should run.

```
int[] nums = {10, 20, 30, 40};
for (int i = 0; i < nums.length; i++) {
      System.out.println(nums[i]);
}
```

6. There is also a "for-each" loop, which is used exclusively to loop through elements in the array:

```
int[] nums = {10, 20, 30, 40};
```

```
for (int num : nums) {
      System.out.println(num);
}
```

III. String

1. Strings are used for storing text. A `String` variable contains a collection of characters surrounded by double quotes:

```
String str = "Ton Duc Thang";
```

2. The length of a string can be found with the `length()` method:

```
String str = "Ton Duc Thang";
System.out.println(str.length()); //13
```

3. You can reference the individual characters in a string by using the method `charAt()` with the same index that you would use for an array:

```
String str = "Ton Duc Thang";
System.out.println(str.charAt(0)); //T
```

4. You must compare strings by using the `equals()` method:

```
"Star".equals("star"); // returns false
"abc".equals("abc"); // returns true
```

5. You can use the `concat()` method to concatenate two strings. You can also concatenate two strings to form another string by using the "+" operator:

```
String str1 = "Hello";
str1.concat(" World"); // Hello World

String str2 = "Hello";
str2 = str2 + " World"; // Hello World
```

6. You can use `substring()` to access part of a string:

```
String str = "Hello TDTU";
System.out.println(str.substring(0, 5)); // Hello
```

7. You can split a string into an array of substrings:

```
String str = "Hello TDTU";
String[] arr = str.split(" "); // [Hello,TDTU]
```

Students can explore additional methods of the String class by referring to the Java 11 documentation.

IV. OOP, Class, Encapsulation

1. OOP stands for Object-Oriented Programming.

2. Classes and objects are the two main aspects of object-oriented programming. A class is a template for objects, and an object is an instance of a class. The filename must have the same name as the public class name in that file.

3. A class can contain the following types of variables: Local variables, instance variables, and class variables.

4. A class can also have methods. Method declarations have some components, in order: Modifiers, the return type, the parameter list in parenthesis, and the method body (the method body must be enclosed in curly brackets).

5. Every class has a constructor. A constructor must have the same name as the class. A class can have more than one constructor, but in most cases, you need to define at least three types of the constructor: *Default constructor*, with no parameter; *Parameterized constructor*; *Copy constructor*.

6. Java provides several access modifiers to set access levels for classes, attributes, and methods. The four access levels: are *private*, *protected*, *default*, and *public*.

7. To achieve encapsulation in Java:

- Declare the variables of a class as private/protected.
- Provide public getter and setter methods to modify and view the variable's values.

```
public class Student {
    private String name;
    private String gender;
    private int age;

    public Student() {
        this.name = "";
        this.gender = "male";
        this.age = 0;
    }

    public Student(String name, String gender, int age) {
        this.name = name;
        this.gender = gender;
        this.age = age;
    }

    public Student(Student st) {
        this.name = st.name;
        this.gender = st.gender;
        this.age = st.age;
    }

    public void studying() {
        System.out.println("studying...");
    }

    public void reading() {
        System.out.println("reading...");
    }

    public String getName() {
        return this.name;
    }

    public String getGender() {
        return this.gender;
    }

    public int getAge() {
        return this.age;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public void setAge(int age)  {
        this.age = age;
    }
}
```

V. Exercises

Array

1) Write a Java program:
    a) Write method `public static int maxEven(int[] a)` to find the greatest even number in an array.
    b) Write method `public static int minOdd(int[] a)` to find the smallest odd number in an array.

c) Write method `public static int sumMEMO(int[] a)` to calculate the sum of the greatest even number and the smallest odd number in an array.

d) Write method `public static int sumEven(int[] a)` to calculate the sum of even numbers in an array.

e) Write method `public static int prodOdd(int[] a)` to calculate the product of odd numbers in an array.

f) Write method `public static int idxFirstEven(int[] a)` returns the position of the first even number in the array.

g) Write method `public static int idxLastOdd(int[] a)` returns the position of the last odd number in the array.

h) Write method `public static int[] input(int n)` returns an array with **n** elements which input from keyboard.

i) Write a main method `public static void main(String[] args)`:

- Input **n** and an array with **n** elements from the keyboard.
- Call the above methods and test them with input data.

String

1) Write a Java program:

- Write a method `public static String shortName(String str)` to first and last name.

  Ex: “Nguyen Le Trong Tin” => “Tin Nguyen”.

- Write a method `public static String hashtagName(String str)` to create names with the hashtag.

  Ex: “Nguyen Le Trong Tin” => “#TinNguyen”.

- Write a method `public static String upperCaseAllVowel(String str)` to uppercase all vowel letters in a string.

  Ex: “Nguyen Le Trong Tin” => “NgUyEn LE TrOng TIn”.

- Write a method `public static String upperCaseAllN(String str)` to uppercase all n letters in a string.

  Ex: “Nguyen Le Trong Tin” => “NguyeN Le TroNg TiN”.

- Write a main method `public static void main(String []args)` to test the above methods.

2) For the following paragraph:

"The Edge Surf is of course also a whole lot better, which will hopefully win Microsoft some converts. It offers time trial, support for other input methods like touch and gamepads, accessibility improvements, high scores, and remastered visuals."

- Write method `public static int countWord(String paragraph)` to count the number of words in the paragraph.
- Write method `public static int countSentences(String paragraph)` to count the number of sentences in the paragraph.
- Write method `public static int countAppear(String paragraph, String word)` to count the number of occurrences of the `word` in the paragraph.
- Write a main method `public static void main(String []args)` to test the above methods.

OOP, Class, Encapsulation

1) Implement the **Club** class is defined as the description below:

   Attributes:

   - **name**: String.
   - **wins**: int (number of wins).
   - **draws**: int (number of draws).
   - **losses**: int (number of losses).

   Constructors:

   - Constructor with no parameter **public Club()** (name = "", wins = 0, draws = 0, losses = 0).
   - Constructor with parameters **public Club(String name, int wins, int draws, int losses)**.
   - Copy constructor **public Club(Club club).**

   Methods:

   - **public String getName():** return the name of the club.
   - **public int getWins():** return number of wins.
   - **public int getDraws():** return number of draws.
   - **public int getLosses():** return number of losses.
   - **public void setName(String name):** set the name of the club.
   - **public void setWins(int wins):** set the number of wins.
   - **public void setDraws(int draws):** set the number of draws.

- **public void setLosses(int losses):** set the number of losses.
- **public int numMatchesPlayed():** return the number of matches that the club played

  numMatches = win + draw + lose.

- **public boolean isFinish():** Check if the club has finished the league yet. It is known that the league has 10 matches.
- **public int getPoints():** Return the number of points the club has received

  points = win*3 + draw*1 + lose*0.

- **public String toString()** with the format: "**name** club: **wins/draws/losses - points**"

Example: `Club` cb = new `Club`("Chelsea", 1, 2, 0) will be printed like the following string: *Chelsea club: 1/2/0 - 5*.

Write a test program (called TestClub) to test all the methods defined.

2) Implement the **RegularPolygon** class is defined as the description below:
Attributes:

- **name**: String.
- **edgeAmount**: int (amount of edges).
- **edgeLength**: double (length of edge).

Constructors:

- Constructor with no parameter **public RegularPolygon()** (name = "", edgeAmount= 3, edgeLength = 1).
- Constructor with parameters **public RegularPolygon(String name, int edgeAmount, double edgeLength)**.
- Constructor with parameters **public RegularPolygon(String name, int edgeAmount)** (edgeLength = 1).
- Copy constructor **public RegularPolygon(RegularPolygon polygon)**.

Methods:

- **public String getName():** return the name of the polygon.
- **public int getEdgeAmount():** return the number of edges.
- **public int getEdgeLength():** return the length of an edge.
- **public void setName(String name):** set the name of the polygon.
- **public void setEdgeAmount(int num):** set amount of edges.
- **public void setEdgeLength (double length):** set length of the edge.

- **public String getPolygon():**
  - If the amount of edges equals 3, then return "Triangle",
  - If the amount of edges equals 4, then return "Quadrangle",
  - If the amount of edges equals 5, then return "Pentagon",
  - If the amount of edges equals 6, then return "Hexagon",
  - If the amount of edges is greater than 6, return on "Polygon has the number of edges greater than 6".
- **public double getPerimeter():** return the perimeter of the polygon

  perimeter = edgeLength * edgeAmount
- **public double getArea():** return the area of the polygon

  area = (edgeLength)^2 * a

edgeAmount	**a**
3	0.433
4	1
5	1.72
6	2.595

If a polygon has several edges greater than 6 return area = -1.

- **public String toString()** with the format: "**name - PolygonType - area**"

  Example: RegularPolygon rp = new RegularPolygon("q1", 4, 1.5) will be printed like the following string: *q1 – Quadrangle – 2.25*.

Write a test program (called TestRegularPolygon) to test all the methods defined.

-- END --