MID-TERM REVIEW

SUBJECT: OBJECT-ORIENTED PROGRAMMING

INSTRUCTIONS:

1. Students create a folder named **Student ID_ Full Name**.
2. **Students will receive 02 folders: Question 1 and Question 2** and copy these 02 folders to the folder named **Student ID_ Full Name** above.
3. When finished, students save the file and compress the folder created in section 1. The correct compress file will contains folder **Student ID_ Full Name** with folder Question 1 and Question 2 inside.

QUESTIONS

Students must ensure that your program can be compiled, executed and placed in the correct directory as instructed. All cases of not following the request or causing errors during compiling/executing are given 0 points for the whole sentence.

The main function given in each folder is only for students to test your code; students don't have to remove the main function when submitting, but must ensure that the main function does not cause errors for your program.

Students code directly in the received files, but DO NOT edit the file name, function name, parameter order.

Question 1:

Folder **Question1** contains the **JavaBasic.java** file, students open this file and do the following requirement:

a. Implement the function with prototype: **public static int sumNegativeElements(int[] a)** to return the sum of all negative elements in array **a**.

b. Implement the function with prototype: **public static String uppercaseFirstVowels(String str)** to return the string uppercase the first letter of each word if it's the vowel. Knowing that, each word will be separated by one space character.

   For example: given a string as "nguyen tran anh oanh", this method will return "nguyen tran Anh Oanh".

c. Implement the function with prototype: **public static int findMinNegativeElement(int[] a)** to return the index of the minimum negative element in array **a**. If these is no negative element then return -1. The first index of an array is 0.

d. Implement the function with prototype: **public static String getName(String str)** to return the full name in the string with format: "Name: <full name>". For example, given a string:

   "Name: Tran Van Tai"

   Method will return a string: "Tran Van Tai"

e. Implement the function with prototype: **public static int findFirstMod3Element(int[] a)** to return the first element which is divisible by 3. If these is no element meets the condition, then return -1. The first index of an array is 0.

f. Implement the function with prototype: **public static int countString(String str, String k)** to return the occurrence of word **k** in string **str**. There is no case-sensitive when counting the occurrence.

For example: given a string **str**: "Pham Thi Uyen Uyen" and a string **k**: "Uyen", method will return 2.

Question 2:

Folder **Question2** contains **02 files**: ***Rectangle*.java** and ***TestRectangle*.java**, students open file ***Rectangle*.java** and define the **Rectangle** class**:**

a. Properties:

- Property for rectangle name: **name** (**String** type)
- Properties for width and length: **width**, **length** (**double** type)

b. Methods:

- Parameterized constructor method:

  Rectangle(String name, String color, double wit, double len)

- Getter methods:

  Return rectangle's name, color: **getName(), getColor()**

  Return rectangle's width and rectangle's length: **getWidth(), getLength()**

- Setter methods:

  Accept the parameter values and assign them to the properties of Rectangle class: **setName(String name), setColor(String color), setWidth(double width), setLength(double length)**

- Return the perimeter of the rectangle:

  public double getPerimeter()

- Return the type of the rectangle:

  public String getType()

  - Rectangle's area >= 10 then return "A".
  - Rectangle's area < 10 and >= 5 then return "B".
  - Rectangle's area < 5 then return "C".

  Note: Return String data only, do not print on the screen. The strings are case-sensitive.

- Check the rectangle is a square or not:

  public boolean isSquare()

  Knowing that, a square is a rectangle which have the length equals the width.

- Calculate and return the length of the diagonal line:

  public double calDiagonalLine()

- Create a new rectangle with the new rate:

  public Rectangle resize(double rate)

  The parameter will receive a decimal number representing the rate of change, the new rectangle will have length = length of original rectangle * rate, width = width of original rectangle * rate. If the rate is 1, the new shape will have the same length and width as the original rectangle.

- The method **public String toString()** returns a string in the following format (The space characters and punctuation characters in the return string must exactly match the format):

  public String toString()

  Rectangle[name, length, width, area, type]

  For example: Given an object **Rectangle rec = new Rectangle("Ruby", "Red", 6, 8)** then the toString() method of this object will return a string **Rectangle[Ruby, 8, 6, 48, A]**

You are provided file **TestRectangle.java**, using this file to test your **Rectangle** class.