PRO192 PE INSTRUCTIONS

Students are ONLY allowed to use:

- Materials on his/her computer (including JDK, NetBeans...).
- For distance learning: Google Meet, Hangout (for Exam Monitoring Purpose).

<u>Instructions for completing PE</u>:

(In the followings the name of a folder run and src are written in uppercase for the purpose of emphasis, in a project their case is not important)

✔ For each quesion:

o DO NOT create new project. You must use the given project and modify it according to question's requirements. DO NOT delete given file(s) and DO NOT create file(s) with the same name as them. DO NOT use package in your code.

o You CAN create more classes/interfaces/methods/variables (if needed).

o **Submission**: Submission is performed by project: each time only one project is submitted. Just before submission, you must run the option "Clean and Build Project", then rename the folder dist to RUN. You must select the question number (1 for Q1, 2 for Q2,...), browse and select the whole project (do not compress) and click the button "Submit".

(The submitted folder must contain 2 subfolders: SRC contains source files and RUN contains jar file. SRC is already contained in the project. The RUN folder can be created by 2 ways: rename the folder dist, or create the folder RUN and copy the jar file to it).

✔ If a project is submitted incorrectly, it will get ZERO

<u>LAB 6 (SLOT 17)</u>

<u>Question 1</u>: (1 marks)

Do not pay attention to real meaning of objects, variables and their values in the questions below.

Write a class named **Book** that holds information of a book.

Book
-title:String -price:int
+Book() +Book(title:String, price:int) +getTitle():String +getPrice():int +setPrice(price:int):void

Where:

- Book() - default constructor.
- Book(title:String, price:int) - constructor, which sets values to title and price.
- getTitle():String – return title in **uppercase** format.
- getPrice():int – return price.
- setPrice(price:int):void – update price.

Do not format the result.

The program output might look something like:

Enter title: atlanta Enter price: 12 1. Test getTitle() 2. Test setPrice() Enter TC (1 or 2): 1 OUTPUT: ATLANTA	Enter title: atlanta Enter price: 12 1. Test getTitle() 2. Test setPrice() Enter TC (1 or 2): 2 Enter new price: 20 OUTPUT: 20

<u>Question 2</u>: (1.5 marks)

Write a class named **Car** that holds information about a car and class named **SpecCar** which is derived from **Car** (i.e. Car is super class and SpecCar is sub class).

Car
-maker:String -price:int
+Car() +Car(maker:String, price:int) +getMaker():String +getPrice():int +setMaker(maker:String):void +toString():String

Where:

- getMaker():String – return maker.
- getPrice():int – return price.
- setMaker(maker:String):void – update maker.
- toString():String – return the string of format:
  maker, price

SpecCar
-type:int
+SpecCar() +SpecCar(maker:String, price:int, type:int) +toString():String +setData():void +getValue():int

Where:

- toString():String – return the string of format:
  maker, price, type
- setData():void – Add string "XZ" to the head of maker and increase price by 20.//maker="XZ"+maker; price+=20;
- getValue():int – Return price+inc, where if type<7 then inc=10, otherwise inc=15.

The program output might look something like:

Enter maker: hala Enter price: 500 Enter type: 7 1. Test toString() 2. Test setData() 3. Test getValue() Enter TC (1,2,3): 1 OUTPUT: hala, 500 hala, 500, 7	Enter maker: hala Enter price: 500 Enter type: 7 1. Test toString() 2. Test setData() 3. Test getValue() Enter TC (1,2,3): 2 OUTPUT: XZhala, 520	Enter maker: hala Enter price: 500 Enter type: 6 1. Test toString() 2. Test setData() 3. Test getValue() Enter TC (1,2,3): 3 OUTPUT: 510	Enter maker: hala Enter price: 500 Enter type: 8 1. Test toString() 2. Test setData() 3. Test getValue() Enter TC (1,2,3): 3 OUTPUT: 515

Question 3: (1.5 marks)

Write a class named **Car** that holds information about a car.

Car
-maker:String -rate:int
+Car () +Car (maker:String, rate:int) +getMaker():String +getRate():int +setMaker(maker:String):void +setRate(rate:int):void

Where:

- getMaker():String – return maker.
- getRate():int – return rate.
- setMaker(maker:String): void – update maker.
- setRate(rate:int): void – update rate.

The interface **ICar** below is already compiled and given in byte code format, thus **you can use it without creating ICar.java file**.

```
import java.util.List;
```

```
public interface ICar {
   public int f1(List<Car> t);
   public void f2(List<Car> t);
   public void f3(List<Car> t);
}
```

Write a class named **MyCar**, which implements the interface **ICar**. The class MyCar implements methods f1, f2 and f3 in ICar as below (you can add other functions in MyCar class):

- f1: Return the whole part of average rate of all cars (e.g. the whole part of 3.7 is 3).
- f2: Find the first max and min rates in the list and swap their positions.
- f3: Sort the list by maker alphabetically, in case makers are the same, sort them descendingly by rate.

When running, the program will add some data to the list. Sample output might look something like:

```
Add how many elements: 0
Enter TC(1-f1;2-f2;3-f3): 1
The list before running f1:
(A,3) (B,7) (C,6) (D,7) (E,6)
OUTPUT:
5
```

```
Add how many elements: 0
Enter TC(1-f1;2-f2;3-f3): 2
The list before running f2:
(A,6) (B,2) (C,9) (D,17) (E,8) (F,17) (G,2)
OUTPUT:
(A,6) (D,17) (C,9) (B,2) (E,8) (F,17) (G,2)
```

```
Add how many elements: 0
Enter TC(1-f1;2-f2;3-f3): 3
The list before running f3:
(H,1) (G,2) (E,3) (F,4) (E,15) (C,6) (B,7) (A,8)
OUTPUT:
(A,8) (B,7) (C,6) (E,15) (E,3) (F,4) (G,2) (H,1)
```

Question 4: (1 marks)
The interface **IString** below is already compiled and given in byte code format, thus **you can use it without creating IString.java file**.

```
public interface IString {
   public int f1(String str);
   public String f2(String str);
}
```

Write a class named **MyString**, which implements the interface **IString**. The class MyString implements methods f1 and f2 in IString as below:

- f1: Count and return number of prime digits in str.
- f2: Reverse order of all words in str (word = a string without space).

The program output might look something like:

```
1. Test f1()
2. Test f2()
Enter TC (1 or 2): 1
Enter a string:
a32b 95cd b67
OUTPUT:
4
```

```
1. Test f1()
2. Test f2()
Enter TC (1 or 2): 2
Enter a string:
a9 b1 a8 a7 a6 a5
OUTPUT:
a5 a6 a7 a8 b1 a9
```

Question 5: (5 marks)