Question 1:

(2 marks) Read PE instructions at the bottom of the exam paper.
Do not pay attention to real meaning of objects, variables and their values in the questions below.

Write a class named **Engine** with the following information:

Engine
-designer:String -power:int
+Engine() +Engine(designer:String, power:int) +getDesigner():String +getPower():int +setPower(power:int):void

Where:

- Engine() - default constructor.
- Engine(designer:String, power:int) - constructor, which sets values to designer and power.
- getDesigner():String – returns a string s, which is obtained by taking the 3 first characters of string and lowercase the first character in the s string.
- getPower():int – return power.
- setPower(power:int):void – update power.

  Do not format the result.

The program output might look something like:

```
Enter designer: Tom123

Enter power: 10

1. Test getDesigner()

2. Test setPower()

Enter TC (1 or 2): 1

OUTPUT:

tom
```

```
Enter designer: Tom123

Enter power: 10

1. Test getDesigner()

2. Test setPower()

Enter TC (1 or 2): 2

Enter new power: 12

OUTPUT:

12
```

Question 2:

(3 marks) Write a class **Robot** and a class **SpecRobot** extending from **Robot** (i.e. Robot is a superclass and SpecRobot is a subclass) with the following information:

Robot
-label:String -type:int

Where:

- getLabel():String – return label.
- getType():int – return type.

+Robot() +Robot(label:String, type:int) +getLabel():String +getType():int +setLabel(label:String):void +toString():String

- setLabel(label:String):void – update label.
- toString():String – return the string of format: **label, type**

SpecRobot
-step:int
+SpecRobot() +SpecRobot(label:String, type:int, step:int) +toString():String +setData():void +getValue():int

Where:

- toString():String – return the string of format: **label, type, step**
- setData():void – Insert step into the 2nd character of the label.
- getValue():int – Check if the type < 3 and the label contains 'A' character then return step, otherwise return step

+toString():String

SpecRobot
-step:int
+SpecRobot() +SpecRobot(label:String, type:int, step:int) +toString():String +setData():void +getValue():int

Where:

- toString():String – return the string of format:
  label, type, step
- setData():void – Insert step into the 2nd character of the label.
- getValue():int – Check if the type < 3 and the label contains 'A' character then return step, otherwise return step + 2.

The program output might look something like:

Enter label: asimo Enter type: 1 Enter step: 2 1. Test toString() 2. Test setData() 3. Test getValue() Enter TC (1,2,3): 1 OUTPUT: asimo, 1 asimo, 1, 2	Enter label: asimo Enter type: 1 Enter step: 2 1. Test toString() 2. Test setData() 3. Test getValue() Enter TC (1,2,3): 2 OUTPUT: a2simo, 1	Enter label: Asimo Enter type: 1 Enter step: 3 1. Test toString() 2. Test setData() 3. Test getValue() Enter TC (1,2,3): 3 OUTPUT: 3	Enter label: Asimo Enter type: 3 Enter step: 3 1. Test toString() 2. Test setData() 3. Test getValue() Enter TC (1,2,3): 3 OUTPUT: 5

Question 3:

(3 marks) Write a class **Robot** with the following information:

Robot
-label:String -step:int
+Robot () +Robot (label:String, step:int) +getLabel():String +getStep():int +setLabel(label:String):void +setStep(step:int):void

Where:

- getLabel():String – return label.
- getStep():int – return step.
- setLabel(label:String): void – update label.
- setStep(step:int): void – update step.

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

```
import java.util.List;

public interface IRobot {

   public int f1(List<Robot> t);
```

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:
```

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

```
(A,3) (B,7) (CAB,6) (D,7) (E,6)

OUTPUT:

13
```

```
Add how many elements: 0
Enter TC(1-f1;2-f2;3-f3): 3
The list before running f3:
(H,19) (G,56) (E,8) (F,47) (E,56) (C,65) (B,74) (A,83)
OUTPUT:
(H,19) (E,8) (F,47) (G,56) (E,56) (C,65) (B,74) (A,83)
```

Question 4:

(2 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: Sum length of words are not palindrome string and have at least two characters (word = a string without space(s); *The palindrome string is the string after reversal and the original string is exactly the same, for example "aba" is a palindrome string*).
- f2: Return the string by removing characters that appear more than once in the string and keep only the first character, for example bananas → bans.

The program output might look something like:

1. Test f1() 2. Test f2()	1. Test f1() 2. Test f2()