(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 **Tofu** with the following information:

Tofu
-maker:String -quantity:int
+Tofu() +Tofu(maker:String, quantity:int) +getMaker():String +getQuantity():int +setQuantity(quantity:int):void

Where:

- Tofu() - default constructor.
- Tofu(maker:String, quantity:int) - constructor, which sets values to maker and quantity.
- getMaker():String – returns a string s, which is obtained by lowercase the first and last letters of the maker string.
- getQuantity():int – return quantity.
- setQuantity(quantity:int):void – update quantity=new quantity.

Do not format the result.

The program output might look something like:

Enter maker: TofU Enter quantity: 2 1. Test getMaker() 2. Test setQuantity() Enter TC (1 or 2): 1 OUTPUT: tofu	Enter maker: tofu Enter quantity: 2 1. Test getMaker() 2. Test setQuantity() Enter TC (1 or 2): 2 Enter new quantity: 10 OUTPUT: 10

-name:String -power:int
+Speaker() +Speaker(name:String, power:int) +getName():String +getPower():int +setName(name:String):void +toString():String

- getName():String – return name.
- getPower():int – return power.
- setName(name:String):void – update name.
- toString():String – return the string of format:
  name, power

SpecSpeaker
-sound:String
+SpecSpeaker() +SpecSpeaker(name:String, power:int, sound:String) +toString():String +setData():void +getValue():String

Where:

- toString():String – return the string of format:
  name, sound, power
- setData():void – Remove the last letter of name string.
- getValue():String – Check if the power>10 then return sound, otherwise return sound+power.

The program output might look something like:

Enter name: samsung Enter power: 100 Enter sound: standard	Enter name: samsung Enter power: 100 Enter sound: standard	Enter name: samsung Enter power: 100 Enter sound: standard	Enter name: samsung Enter power: 9 Enter sound: standard

Question 3:

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

Speaker

Where:

-name:String -power:int
+Speaker () +Speaker (name:String, power:int) +getName():String +getPower():int +setName(name:String):void +setPower(power:int):void

- getName():String – return name.
- getPower():int – return power.
- setName(name:String): void – update name.
- setPower(power:int): void – update power.

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

```
import java.util.List;

public interface ISpeaker {

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

```
    public void f2(List<Speaker> t);

    public void f3(List<Speaker> t);

}
```

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

- f1: Count and return the number of speakers with power > 10.
- f2: Sort the first 5 elements in the list ascendingly (other elements are unchanged).
- f3: Check if in the list there are at least 2 elements having minimum power then remove the second one (thus if only one element with minimum power then do nothing).

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: (A2,8) (B1,2) **(C3,16) (D4,17)** (E5,6)	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,29) (E,22)** (F,19) (G,12) (H,5) OUTPUT: (C,2) (A,6) (B,9) (E,22) (D,29) (F,19) (G,12) (H,5)

OUTPUT: 2	

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,8)** (A,65)
OUTPUT:
(H,19) (G,56) (E,8) (F,47) (E,56) (C,65) (A,65)

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 of odd **number**s in the string str (**number** separated by non-numeric characters).
- f2: Reverse the first longest word in str (word = a string without space(s)).

The program output might look something like:

1. Test f1()	1. Test f1()

Enter a string: a1 b2c3 d21e 3c12	Enter a string: a abcd ab ABCD ef

OUTPUT: 28	OUTPUT: a dcba ab ABCD ef