Question 1.

(2 Point) Design and code a class `Brick` that holds information of a Brick.

Brick
-price:double -code:String
+Brick () +Brick (code:String,price:double) +getCode():String +setPrice(price:double):void +getPrice():double

- getCode(): String – return a code of the Brick where every letters are in uppercase.
- getPrice(): double – return price of the Brick.
- setPrice(price:double): void – set current price of the Brick to a given price.

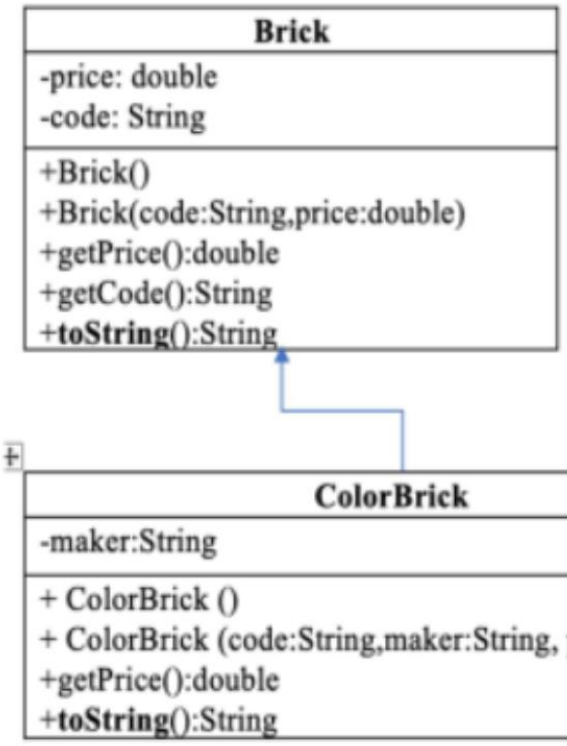
Do not format the result

The program output might look something like:

No of test case	Correct output
1	Enter Brick code: Clay Brick Enter Brick price: 12.0 1. TC = 1 - test getCode() 2. TC = 2 - test setPrice() Enter TC: 1 OUTPUT: CLAY BRICK
2	Enter Brick code: Clay Brick Enter Brick price: 12.0 1. TC = 1 - test getCode() 2. TC = 2 - test setPrice() Enter TC: 2 Enter new Brick price: 12.5 OUTPUT: 12.50

Question 2:

(3 Point) Design and code a class **`Brick`** that holds information about a **`Brick and`** class **`ColorBrick`** which is derived from **`Brick`**.



Where:
1. getPrice():double – return the price of a Brick.
2. getCode():String – return the code of a Brick.
3. toString():String – return information of a Brick as a string as format of *code price*

- toString():String - return information of a ColorBrick as a string as format of *code maker price*
- getPrice():double – use to determine price of a ColorBrick, *price = original price + increment*, where:
  - increment = 10 percent out of original price if maker starts with "J" or "j".
  - otherwise increment = 0.

Do not format the format the result.

The program output might look something like:

```
Enter Brick code: Clay Brick
Enter Brick price: 20
Enter Brick maker: Italian
1. TC = 1 - test toString()
2. TC = 2 - test getPrice()
Enter TC: 1
OUTPUT:
Clay Brick      20.0
Clay Brick      Italian 20.0
```

```
Enter Brick code: Clay Brick
Enter Brick price: 100
Enter Brick maker: Japanese
1. TC = 1 - test toString()
2. TC = 2 - test getPrice()
Enter TC: 2
OUTPUT:
110.00
```

Enter Brick code: Clay Brick Enter Brick price: 100 Enter Brick maker: Chinese 1. TC = 1 - test toString() 2. TC = 2 - test getPrice() Enter TC: 2 OUTPUT: 100.00	

Question 3:

(3 Point)
Design and code a class **Brick** that holds information about a Brick.

Brick
-price:double -code:String
+Brick() +Brick(code:String,price:double) +getCode():String +getPrice():double +setCode(code:String):void +setPrice(price:double):void

Where:
1. getCode():String – return code of a Brick
2. getPrice():double– return price of a Brick
3. setPrice(price:double):void – set current price to a given price
4. setCode(code:String):void – set curent code to a given code

The interface IBrick is given (DO NOT EDIT THIS ONE).

Design and code a class **MyBrick** which will implement interface IBrick and complete 3 methods which were declared in IBrick:

- Method named f1: increment price by 10 percent out of original price for all Bricks which are in the list "list" and have price is less than the largest Brick.
- Method named f2: count and return the number of Bricks in the list "list" which price is less than or equals to given price "d".
- Method named f3: remove the second largest Brick price in the list "list".

Given some data which is added to list "list" in the Main already:

Brick code	Brick price
FS21	60
KS20	68
FF12	52

By using given data, the program output might look something like:

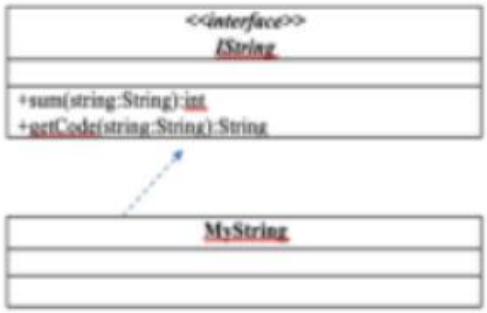
```
Add more how many Brick: 1

Brick code: AK12
Brick price: 75

Enter test function (1-f1;2-f2;3-f3): 1

OUTPUT:
FS21            66.00
KS20            74.80
FF12            57.20
AK12            75.00
```

```
Add more how many Brick: 1

Brick code: AK12
Brick price: 75.0

Enter test function (1-f1;2-f2;3-f3): 2
Enter given Brick price: 60

OUTPUT:
2
```

```
Add more how many Brick: 1

Brick code: AK12
Brick price: 75.0

Enter test function (1-f1;2-f2;3-f3): 3

OUTPUT:
FS21            60.00
FF12            52.00
AK12            75.00
```

Question 4.

(2 Point) You are given an interface named **IString (DO NOT EDIT THIS ONE)**. Design a class **MyString** which will implement the interface IString.



1. int sum(string:String) – this function return summation of a given string "string" as the rule:

   sum = sum of all digits of the first and the last numbers in the given "string".

2. String getCode(string:String) – this function return code of "string" as the rule:

   - Code of string = reverse the first and the last number in the given string "string" only.

   Only one empty space between two words in the result.

The program output might look something like:

No of test case	Correct output
1	1. TC = 1 – test sum() 2. TC = 2 – test getCode() Enter TC: 1 Enter a value in a string: 1a 123 ab 23 abc 10 OUTPUT: 7
2	1. TC = 1 – test sum() 2. TC = 2 – test getCode() Enter TC: 2 Enter a value of rc: 1a 123 ab 145 cd 201 OUTPUT: 321 102