WORKSHOP 03

Exercise 1:

1. Design the interface **IString** that consists of methods

<<interface>> **IString**
+ f1(st:String):int; + f2(st:String):String;

2. Design and code the class **MyString**, which implements the interface **IString**, thus it must implement methods f1 and f2 in IString interface:

- f1: calculate and return sum of all digits in st.
- f2: return the string s, which is obtained by reading all characters in st, if a character is a digit between 0 and 8 then increase it by 1 (others characters are unchanged). E.g., if st="a01b2c8d9" then s = "a12b3c9d9"

3. Build Test class contains main function such that the program output might look something like:

```
1. TC = 1 - test f1()
2. TC = 2 - test f2()
Enter TC: 1
Enter a string:
ab3c58d2
OUTPUT:
18
```

```
1. TC = 1 - test f1()
2. TC = 2 - test f2()
Enter TC: 2
Enter a string:
4a5c9u7
OUTPUT:
5a6c9u8
```

Exercise 2:

1. Design the interface **IString** that consists of methods

```
public interface IString {
   public String f1(String str, String s1, String s2);
   public String f2(String str, String s);
}
```

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

- f1: replace all occurrences of a substring (matching argument s1) with replacement s2.
- f2: return the string str, which concatenates the string s at the end of the string str (see sample output), if the string s does not occur in the string str

3. Build Test class contains main function such that the program output might look something like:

```
1. Test f1()
2. Test f2()
Enter TC (1 or 2): 1
Enter a string:
I love them. They love me.
OUTPUT:
Enter a string 1:
love
Enter a string 2:
hate
I hate them. They hate me.
```

```
1. Test f1()
2. Test f2()
Enter TC (1 or 2): 2
Enter a string:
Nguyen Hoang
OUTPUT:
Enter a string 1:
Kim Lien
Nguyen Hoang Kim Lien
```

Exercise 3:

1. Design and code the class **Fan** that holds information about a fan.

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

Where:

- getCode(): String – return code.
- getPrice(): double – return price.
- setCode(code:String): void – set this.code = code
- setPrice(price:double): void – set this.price = price

Design and code the class **Fan** that holds information about a fan.

2. Design the interface **IFan** that consists of methods

<<interface>> ***IFan***
+ void f1(List<Fan> t, String xCode); + void f2(List<Fan> t, double xPrice); + void f3(List<Fan> t);

3. Design and code the class **MyFan**, which implements the interface **IFan**, thus it must implement methods f1, f2 and f3 in IFan interface:

- f1: Increase the price of those fans in the list t, whose code starts with xCode, by 10% of the original price (see sample output).
- f2: count and return the number of fans in the list t, whose price <= xPrice.

- f3: sort all fans in the list t ascendingly by price, in case their prices are the same, sort them ascendingly by their code alphabetically. *The sorting must ignore case during the comparation.*

4. Build Test class contains main function such that the program output might look something like:

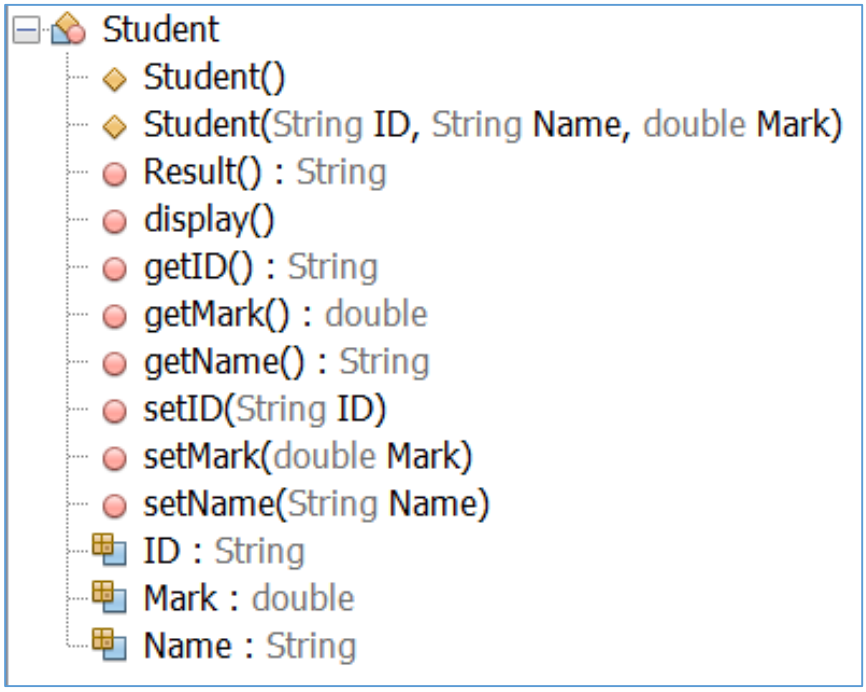
```
Add how many fans: 1
Fan code: ab1
Fan price: 60
Enter TC(1-f1;2-f2;3-f3): 1
The list before running f1:
FS21            80.00
KS20            60.00
FF12            70.00
ab1             60.00
Enter given Fan code: F

OUTPUT:
FS21            88.00
KS20            60.00
FF12            77.00
ab1             60.00
```

```
Add how many fans: 1
Fan code: Ab1
Fan price: 60
Enter TC(1-f1;2-f2;3-f3): 2
The list before running f2:
FS21            80.00
KS20            60.00
FF12            70.00
Ab1             60.00
Enter given Fan price: 70

OUTPUT:
3
```

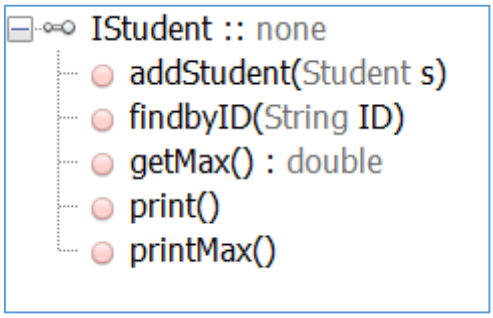
```
Add how many fans: 1
Fan code: Ab1
Fan price: 60
Enter TC(1-f1;2-f2;3-f3): 3
The list before running f3:
FS21            80.00
KS20            60.00
FF12            70.00
Ab1             60.00

OUTPUT:
Ab1             60.00
KS20            60.00
FF12            70.00
FS21            80.00
```

Exercise 4:

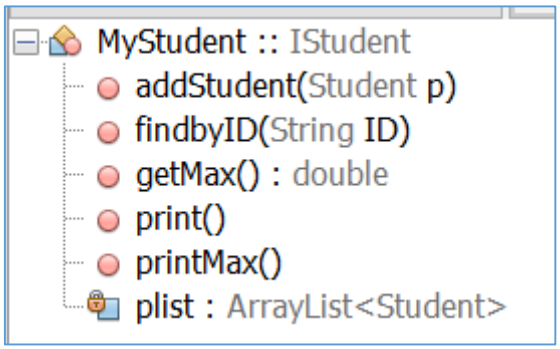
1. Design and code the class **Student** that holds information about a student.



2. Design the interface **IStudent** that consists of methods



3. Design and code the class **MyStudent**, which implements the interface **IStudent**, thus it must implement methods in IStudent interface:

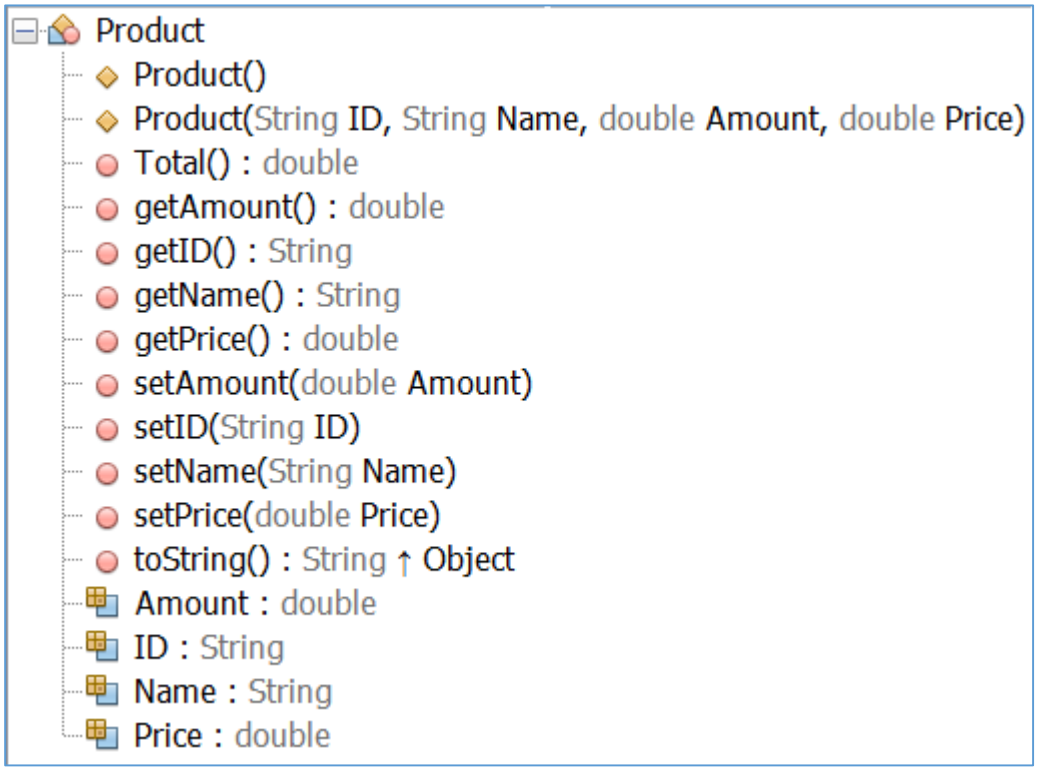


4. Build Test class contains main function such that the program output might look something like:

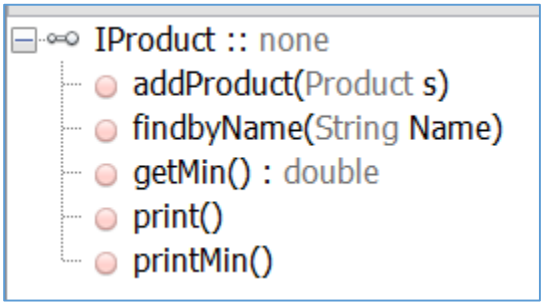
```
Add how many students: 2
Input ID: 01
Input Name: Nam
Input Mark: 19
Input ID: 02
Input Name: Van
Input Mark: 13
The list of students:
Index   ID      Name    Mark    Result
1         01    Nam     19.0    Đỗ
2         02    Van     13.0    Trượt
The list of student with the largest mark
Index   ID      Name    Mark    Result
1         01    Nam     19.0    Đỗ
Input ID: 02
The list of students by ID:
Index   ID      Name    Mark    Result
2         02    Van     13.0    Trượt
```

Exercise 5:

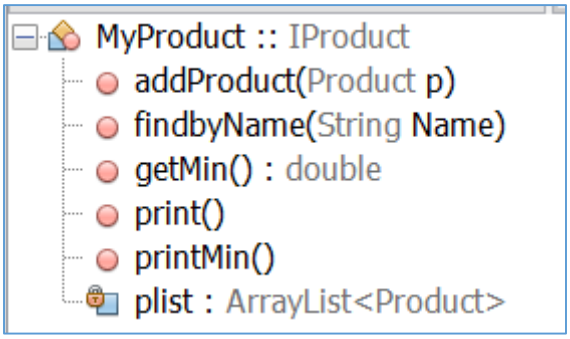
1. Design and code the class **Product** that holds information about a product.



2. Design the interface **IProduct** that consists of methods



3. Design and code the class **MyProduct**, which implements the interface **IProduct**, thus it must implement methods in **IProduct** interface:



4. Build Test class contains main function such that the program output might look something like:

```
Add how many products: 2
Input ID: P1
Input Name: LAPTOP
Input Amount: 12
Input Price: 200
Input ID: P2
Input Name: FAN
Input Amount: 9
Input Price: 100
The list of products:
Index   ID      Name    Amount  Price   Total
1         P1    LAPTOP  12.0    200.0   2160.0
2         P2    FAN     9.0     100.0   900.0
The list of products with the smallest total:
Index   ID      Name    Amount  Price   Total
1         P1    LAPTOP  12.0    200.0   2160.0
Input Name: FAN
The list of products by Name:
Index   ID      Name    Amount  Price   Total
2         P2    FAN     9.0     100.0   900.0
```