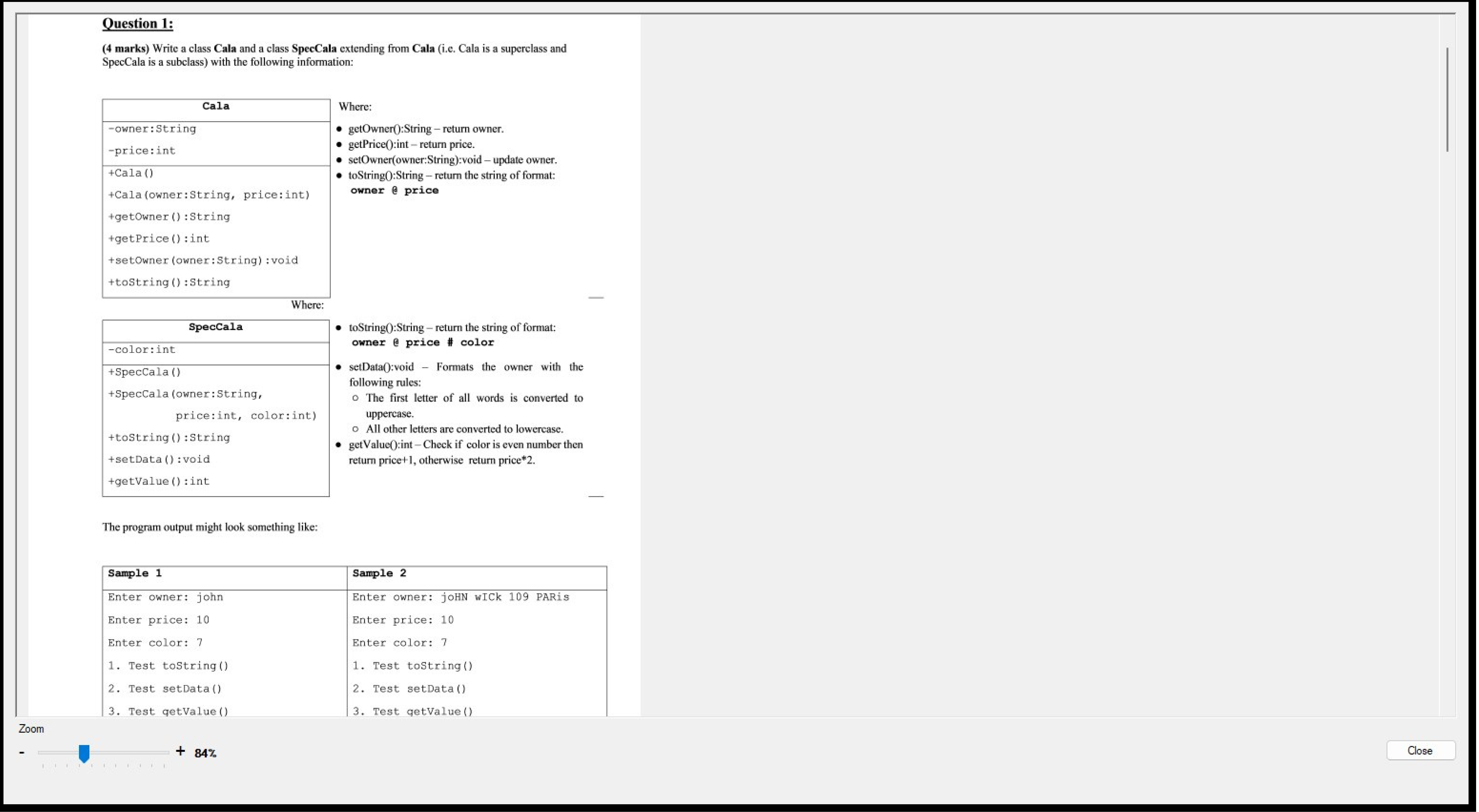


Question 1:

(4 marks) Write a class **Cala** and a class **SpecCala** extending from **Cala** (i.e. Cala is a superclass and SpecCala is a subclass) with the following information:

Cala
-owner:String -price:int
+Cala() +Cala(owner:String, price:int) +getOwner():String +getPrice():int +setOwner(owner:String):void +toString():String

Where:

- getOwner():String – return owner.
- getPrice():int – return price.
- setOwner(owner:String):void – update owner.
- toString():String – return the string of format:
  `owner @ price`

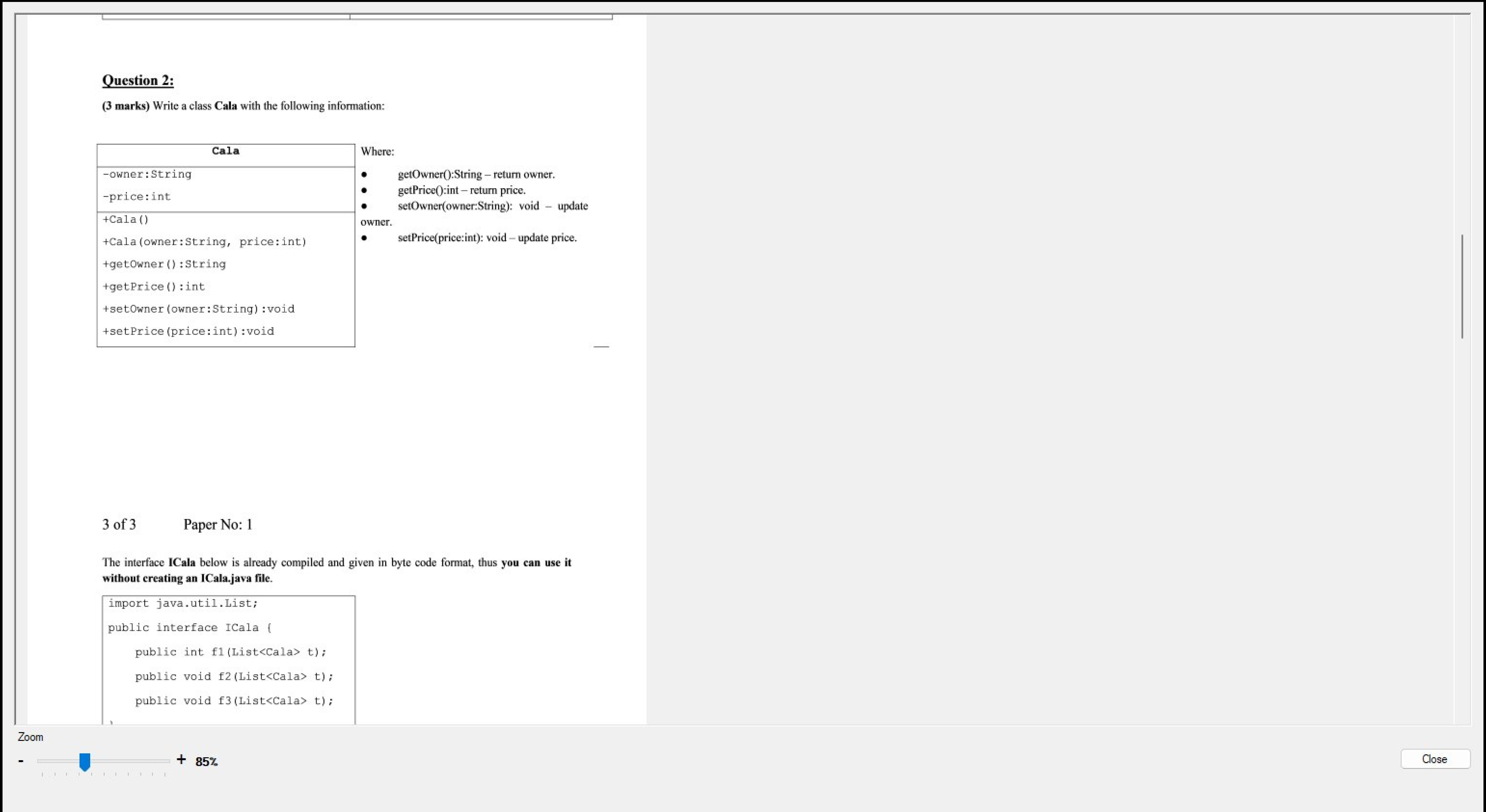
Where:

SpecCala
-color:int
+SpecCala() +SpecCala(owner:String, price:int, color:int) +toString():String +setData():void +getValue():int

- toString():String – return the string of format:
  `owner @ price # color`
- setData():void – Formats the owner with the following rules:
  - The first letter of all words is converted to uppercase.
  - All other letters are converted to lowercase.
- getValue():int – Check if color is even number then return price+1, otherwise return price*2.

The program output might look something like:

Sample 1	**Sample 2**
Enter owner: john	Enter owner: joHN wICk 109 PARis
Enter price: 10	Enter price: 10
Enter color: 7	Enter color: 7
1. Test toString()	1. Test toString()
2. Test setData()	2. Test setData()
3. Test getValue()	3. Test getValue()



Question 2:

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

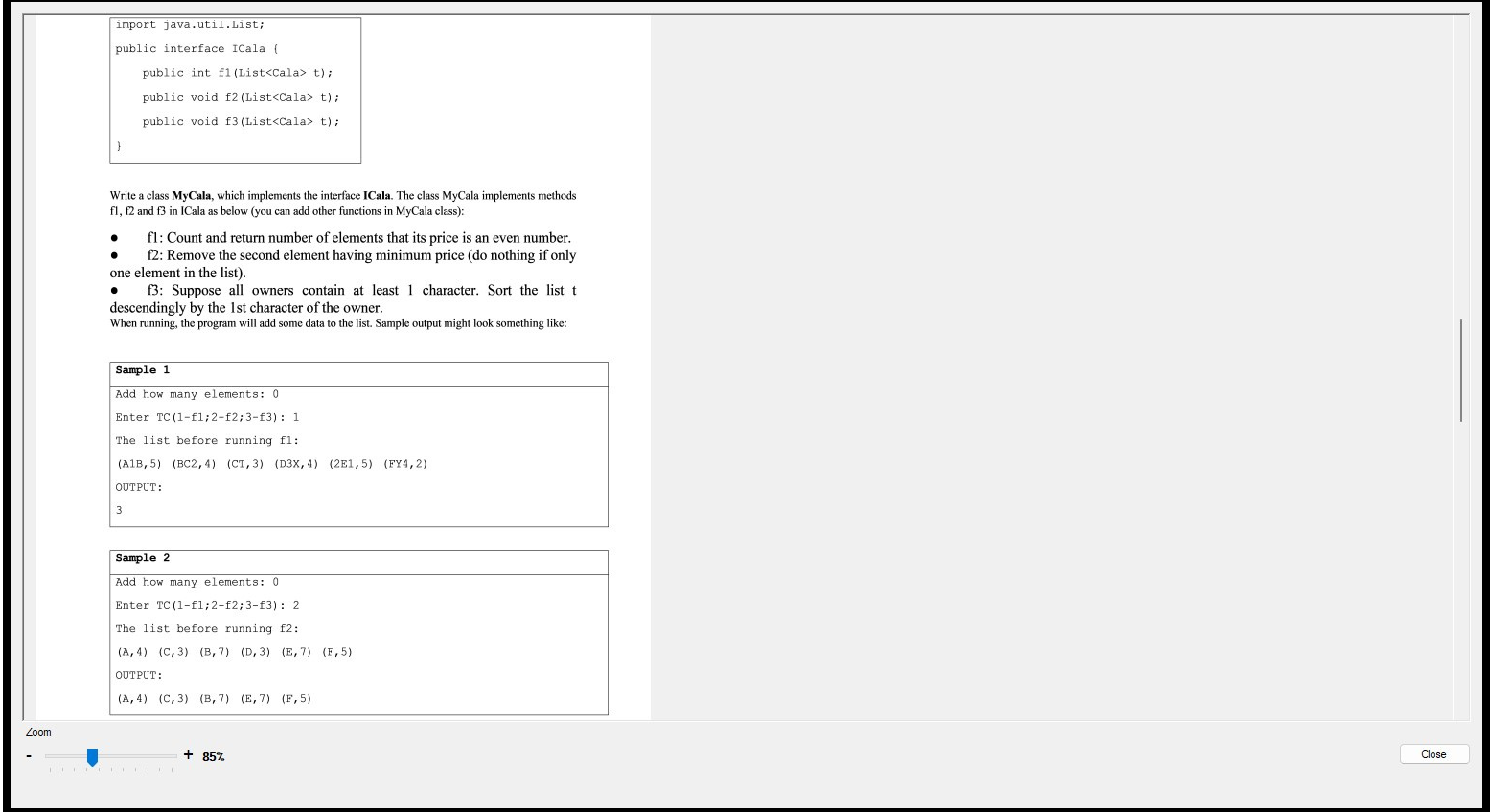
Cala
-owner:String -price:int
+Cala() +Cala(owner:String, price:int) +getOwner():String +getPrice():int +setOwner(owner:String):void +setPrice(price:int):void

Where:

- getOwner():String – return owner.
- getPrice():int – return price.
- setOwner(owner:String): void – update owner.
- setPrice(price:int): void – update price.

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

```
import java.util.List;
public interface ICala {
    public int f1(List<Cala> t);
    public void f2(List<Cala> t);
    public void f3(List<Cala> t);
}
```



```
import java.util.List;

public interface ICala {

    public int f1(List<Cala> t);

    public void f2(List<Cala> t);

    public void f3(List<Cala> t);

}
```

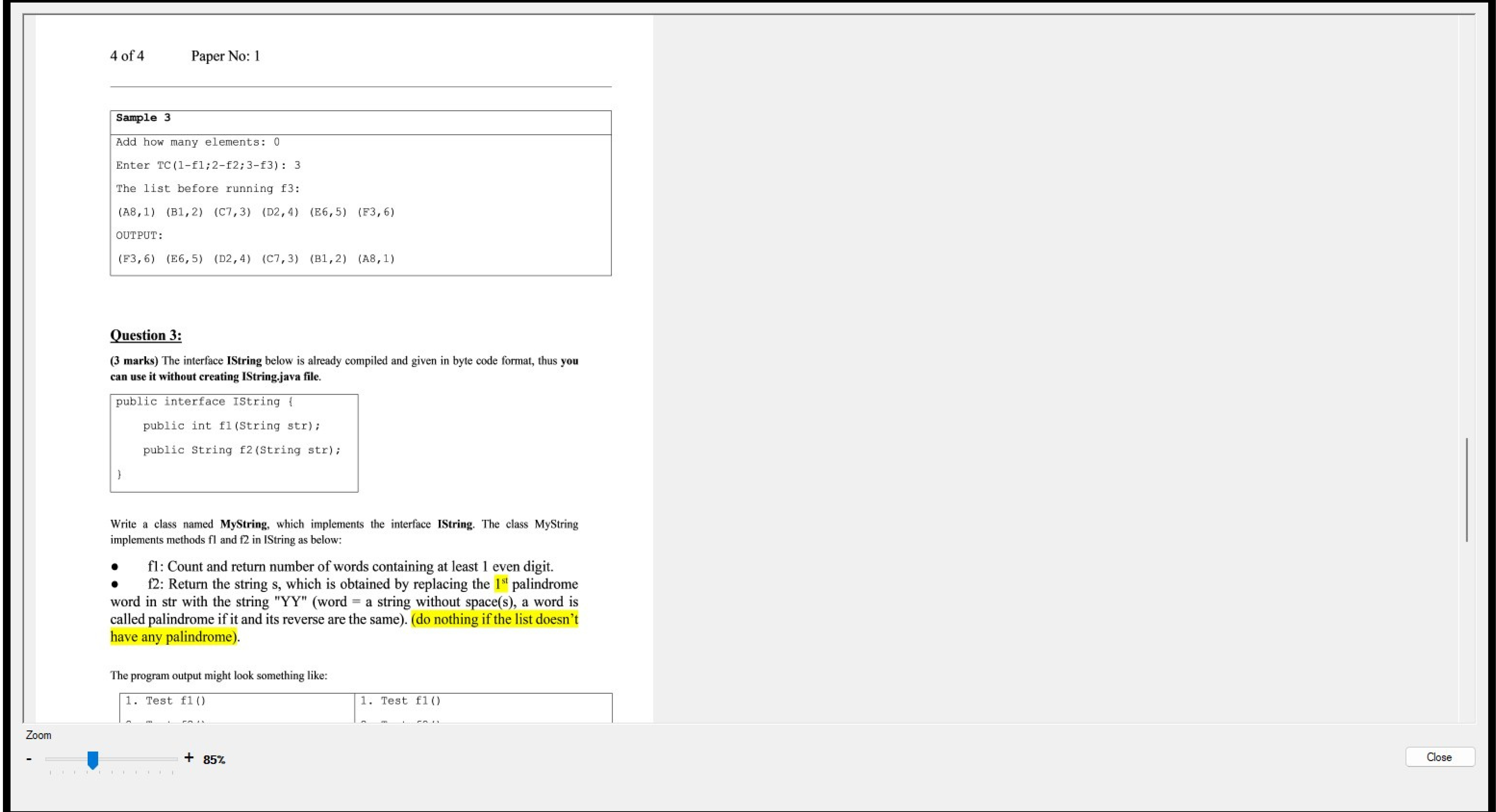
Write a class **MyCala**, which implements the interface **ICala**. The class MyCala implements methods f1, f2 and f3 in ICala as below (you can add other functions in MyCala class):

- f1: Count and return number of elements that its price is an even number.
- f2: Remove the second element having minimum price (do nothing if only one element in the list).
- f3: Suppose all owners contain at least 1 character. Sort the list t descendingly by the 1st character of the owner.

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

```
Sample 1
Add how many elements: 0

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

The list before running f1:

(A1B,5) (BC2,4) (CT,3) (D3X,4) (2E1,5) (FY4,2)

OUTPUT:

3
```

```
Sample 2
Add how many elements: 0

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

The list before running f2:

(A,4) (C,3) (B,7) (D,3) (E,7) (F,5)

OUTPUT:

(A,4) (C,3) (B,7) (E,7) (F,5)
```



Sample 3

```
Add how many elements: 0

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

The list before running f3:

(A8,1) (B1,2) (C7,3) (D2,4) (E6,5) (F3,6)

OUTPUT:

(F3,6) (E6,5) (D2,4) (C7,3) (B1,2) (A8,1)
```

Question 3:

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

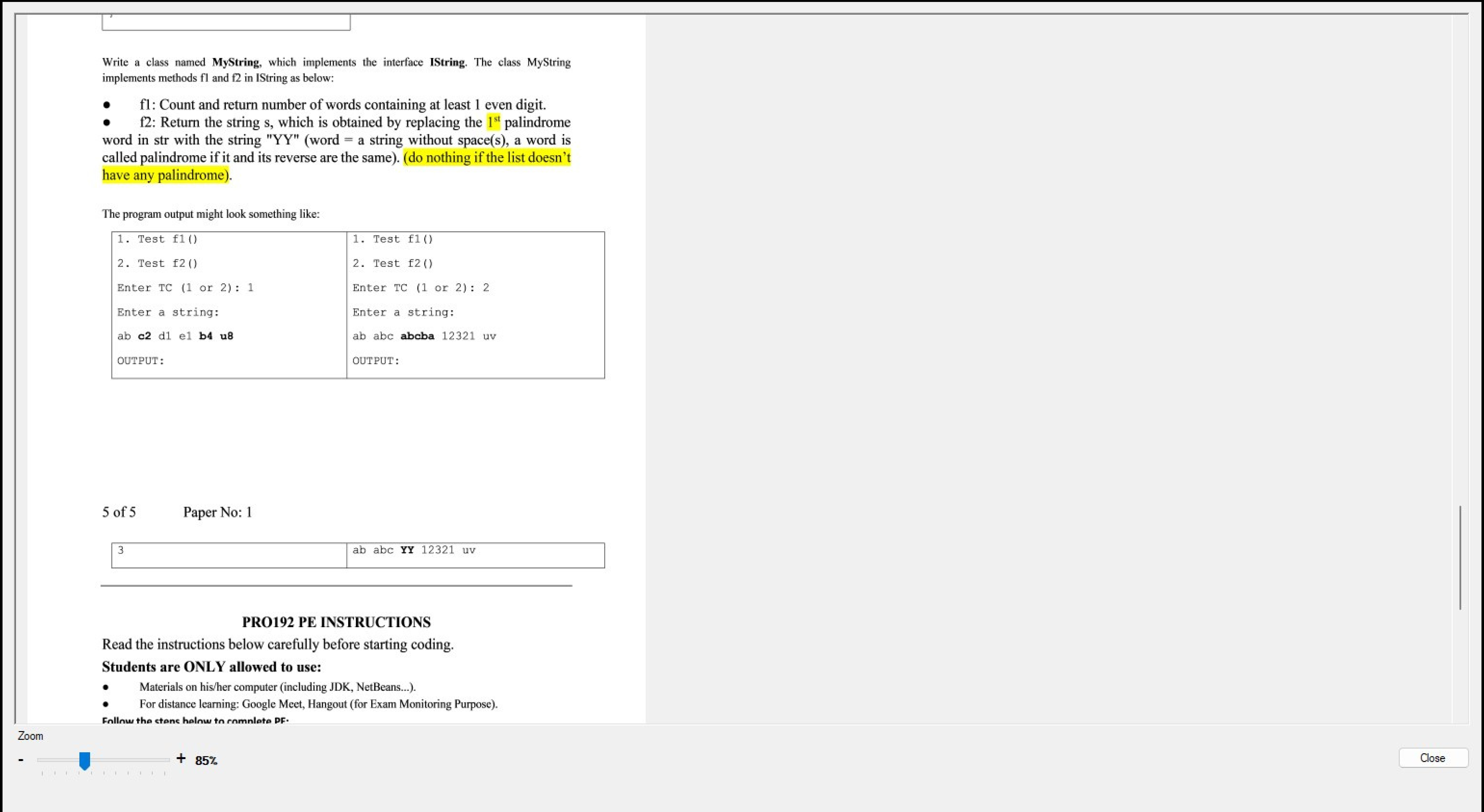
```
public interface IString {

    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 words containing at least 1 even digit.
- f2: Return the string s, which is obtained by replacing the 1st palindrome word in str with the string "YY" (word = a string without space(s), a word is called palindrome if it and its reverse are the same). (do nothing if the list doesn't have any palindrome).

The program output might look something like:

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



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 words containing at least 1 even digit.
- f2: Return the string s, which is obtained by replacing the 1st palindrome word in str with the string "YY" (word = a string without space(s), a word is called palindrome if it and its reverse are the same). (do nothing if the list doesn't have any palindrome).

The program output might look something like:

1. Test f1()	1. Test f1()
2. Test f2()	2. Test f2()
Enter TC (1 or 2): 1	Enter TC (1 or 2): 2
Enter a string:	Enter a string:
ab **c2** d1 e1 **b4** **u8**	ab abc **abcba** 12321 uv
OUTPUT:	OUTPUT:
3	ab abc **YY** 12321 uv

PRO192 PE INSTRUCTIONS

Read the instructions below carefully before starting coding.

Students are ONLY allowed to use:

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

Follow the steps below to complete PE: