GEDİZUNIVERSITY

Faculty of Engineering and Architecture

Computer Engineering Department

COM 102 – OBJECT ORIENTED PROGRAMMING

POSTLAB #9

Academic Year: Spring 2016
Due Date and Hour: May 23, 2016 - Monday , 11.59pm (Submission)
Course Instructor: Asst. Prof. Dr. I.Furkan INCE
Course Assistant: Res. Asst. Arzum KARATAŞ & Res.Asst. Feyza GALIP

MEMBERSHIP CHARGE TRACKER SOFTWARE

Let's assume that your customer has a social club. He needs keep track of maintenance charge paid by the members of the club. In the club, just for the sake of simplicity, let's assume that there are 15 members who pay charge. Thus, he wants to enter maintenance charge information for 2015 year and see the list of charge status for a specific month. He puts a "+" sign for those members who pay while he puts a " - " sign for those members who do not pay the charge for that month.

Note that only + and – signs are allowed as operators. If he enters whatever else as operator, your program should throw an specific exception named "**IllegalOperatorException**" that **extends** Exception class. In this application, your **task** is to write an membership charge tracker application with an exception handling mechanism in Java for him.

Please, create a menu (either GUI or Console) and ask for a selection from him within an appropriate Java application class. According to his choice, your program should perform the related task.The tasks are below.

1- enter montly charge for all members
2- list all member's montly payment info
3- exit from the application.

In your test class , create fifteen members directly without asking him just for testing purpose.

Note that you are supposed to use arrays to store members' payment information. Furthermore, you are required to decide about **possible exceptions** by yourself and **handle** them. Also, you are supposed to create **IllegalOperatorException** class.

In your implementation, use good software engineering and object-oriented programming practices and use a **UML Class diagram** reflecting your design before your implementation.Moreover, do not

forget to **do validity checking** for parameters for each classes.Furthermore, you are required to decide about **possible exceptions** by yourself and **handle** them.

2- CORRECT THE CODE (5x4p = 20 points)

Please, correct the following code fragment in terms of good programming practices and any kind of errors so that you can see the following output and just send a text file(.txt , .doc, .docx or .pdf) explaining errors and corrected versions.

```
public interface Enable {

    void on();
    void off();
}
public class Projector implements Enable {

    private boolean state;
    private int lightDuration;

    public Projector() {
        state = false;
    }

    @Override
    public void on() {
        setState(true);
    }

    @Override
    public void off() {
        setState(false);
    }

    public boolean isState() {
        return state;
    }

    public void setState(boolean state) {
        this.state = state;
    }

    public int getLightDuration() {
        return lightDuration;
    }

    public void setLightDuration(int lightDuration) {
        if(lightDuration>=0){
            this.lightDuration = lightDuration;
        }
        else{
            this.lightDuration = 0;
        }
    }

}
```

```java
public class SimpleAritmeticCalculator implements Enable {

    private boolean state;

    public SimpleAritmeticCalculator() {
        state = false;
    }

    @Override
    public void on() {
        setState(true);
    }

    @Override
    public void off() {
        setState(false);

    }

    public boolean isState() {
        return state;
    }

    public void setState(boolean state) {
        this.state = state;
    }

    public int add(int... numbers){
        int sum = 0;
        for(int number:numbers){
            sum += number;
        }
        return sum;
    }

    public int subtract(int... numbers){
        int difference = numbers[0];
        for(int i = 1; i<numbers.length;i++){
            difference -= numbers[i];
        }
        return difference;
    }

    public int multiply(int... numbers){

        int result = 1;

        for(int number:numbers){
            result *= number;
        }
        return result;
    }
```

```
  public int divide(int numerator, int denominator){
        int result = 0;

        try{
            result = numerator/denominator;

        }
        catch(ArithmeticException arithmeticException){
            System.err.printf( "\nException: %s\n", arithmeticException );
            System.out.println("Zero is an invalid denominator. Please try
again.\n" );

        }
        catch(Exception e){
            System.out.println("An Unknown Error has occured!!");
            e.printStackTrace();
        }

        return result;
    }
}
```

NOTES & SUBMISSION RULES :

1. You are **required to add comment properly**.(It will be graded)

2. You are **strongly advised** to obey the good programming practices (like naming conventions, indentations, commenting your codes and so on.) Using good programming practices is graded.

3. You are **required** to send your source code within a zipped file named :
   COM102_ StudentNumber_YourName_PostLabX.zip
   (e.g., COM102_011XXXX_ArzumKarataş_PostLab9.zip
   COM102_011XXXX_FeyzaGalip_PostLab9.zip)

4. **Be sure whether you attached your work to the e-mail or not**, because it is your responsibility to sending the work on time and in proper format.

5. You are required to **work alone**. Teamwork is **NOT** allowed and **cheating is strictly punished**!

6. You should **submit** your homework to the address following by **e-mail** on time.
   (to com102.2016gediz@gmail.com)

7. **Late submissions** will be graded by using the formula **100 - 10*d^2** where **d** is the number of **late** submission **days**.