Gebze Technical University
Computer Engineering Department
CSE443 - Object Oriented Analysis and Design
Fall 2020-2021
Homework 2 – v1

Rule 1: Detected cases of plagiarism will lead to a penalty of your grade at the end of the semester.

Rule 2: no late submissions! Even if it is late by one minute, it will be ignored. Learning to plan your schedule according to deadlines is part of your education and an invaluable professional asset.

What to submit: a) the source code of your project *fully documented (with javadoc)*, b) a nicely formatted pdf report of your design decision explanations and class diagrams and c) an executable demo that fully illustrates your program's capabilities *whenever code is requested*.

Java version >= 8

****

Question 1 (15 points): You have implemented the Singleton design pattern as a class named (very originally) "Singleton".

1. What happens if someone tries to clone a Singleton object using the `clone()` method inherited from Object? Does it lead to the creation of a second distinct Singleton object? Justify your answer. **(5 points)**
2. Clonin Singletons should not be allowed. How can you prevent the cloning of a Singleton object? **(5 points)**

Let's assume the class Singleton is a subclass of class Parent, that fully implements the Cloneable interface. How would you answer questions 1 and 2 in this case? **(5 points)**

Question 2 (20 points): It's the year 2023, and the Göktürk 3 satellite is in orbit. You work as a data analyst for the Turkish Space Agency. The satellite transmits data as 2D arrays of integers. You are tasked with coding two iterator classes for these data: one iterator that will print a 2D array spirally clockwise and another one that will print it spirally anti-clockwise, both starting at the top left element.

Example: if the data is

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

the first iterator should iterate it in the order: 1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10 and the second iterator as 1 5 9 13 14 15 16 12 8 4 3 2 6 10 11 7. Code in java **one of these iterators**, and provide a main class showing it in action. Do not implement the remove method of the iterator.

Question 3 (30 points): There is a traffic light on the way to campus. It has three states (initially RED):

a) RED: switches to GREEN after 15 seconds.
b) YELLOW: switches to RED after 3 seconds.
c) GREEN: switches to YELLOW after 60 seconds (timeout_X).

Draw the **state diagram** of this traffic light.

Draw the class diagram and implement in Java this state diagram **using the state design pattern**. Make sure you have a main method where you illustrate every state and every transition with text outputs on the terminal.

The traffic department has noticed that the roads surrounding the campus are getting sometimes overwhelmed with traffic due to unexpected events. And the situation is getting worse as they have to wait at this traffic light. That is why they have installed a MOBESE camera on top of the traffic light that measures the amount of cars under it. When it detects a lot of traffic timeout_X is increased from 60 to 90 seconds.

More specifically, whenever the camera detects a change of traffic the method `changeDetected` of the class `HiTech` (provided by the software library of the camera) is called automagically by the hardware:

```
public class HiTech{
      public void changeDetected(boolean flag){
            // ????
      }
}
```

if `flag` is true, it means the traffic has increased substantially, otherwise (if false), everything is normal, so timeout_X returns to its initial value. It is up to you to fill the method.

Implement this new component into your existing traffic light code, **using the Observer design pattern** and redraw the class diagram. Make sure your traffic light subscribes to the camera's software to receive updates on traffic, and set its timeout accordingly. Your main method should illustrate its functionality.

Question 4 (35 points): Your company has an old software library providing a class DataBaseTable, but this class does not provide the capability to allow clients to lock individual table rows. Thus, two clients might end up modifying the same row simultaneously, and we don't want that happening. Moreover, you do not have the source code for this class library, but you have the complete documentation and know about the interface ITable implemented by the DataBaseTable class.

```
public interface ITable{
      public Object getElementAt(int row, int column);
      public void setElementAt(int row, int column, Object o);
      public int getNumberOfRows();
      public int getNumberOfColumns();
}
```

a) **Use the proxy design pattern** in order to equip the DataBaseTable class' objects with synchronization capability. Make sure no **reader thread** calls getElementAt while a **writer thread** is executing setElementAt. Provide your source code and class diagram. **(15 points)**

b) Clients are unhappy with your DataBaseTable synchronization proxy. There are too many getElementAt calls that keep "locking" the Table's rows, and this way the clients that need to modify them using setElementAt wait too long to acquire the table lock. Solve this problem and design a new synchronization solution that **prioritizes writers** of DataBaseTable more than readers in terms of table row access. **(20 points)**