Gebze Technical University

Computer Engineering

Object Oriented Analysis and Design

CSE 443

HOMEWORK 2 REPORT

Değer MANDAL

161044096

Part1
Singleton Design Pattern

1) If the singleton class or its parent class has not implemented the "Clonable" interface, it will give a Java compiler error when the clone() method of the singleton object is called. Because the clone() method is a protected method. If the clone() method of the Object class is called somehow (with the help of the public method) and this operation is tried, then the clone method of the object will throw "CloneNotSupportedException" by default.

2) By implementing the "java.lang.Cloneable" interface, the clone() method for the Singleton class is "override". Inside this method, "CloneNotSupportedException" is thrown. The answer depends on the Parent Class's implementation of the methods of the Clonable interface.

3)

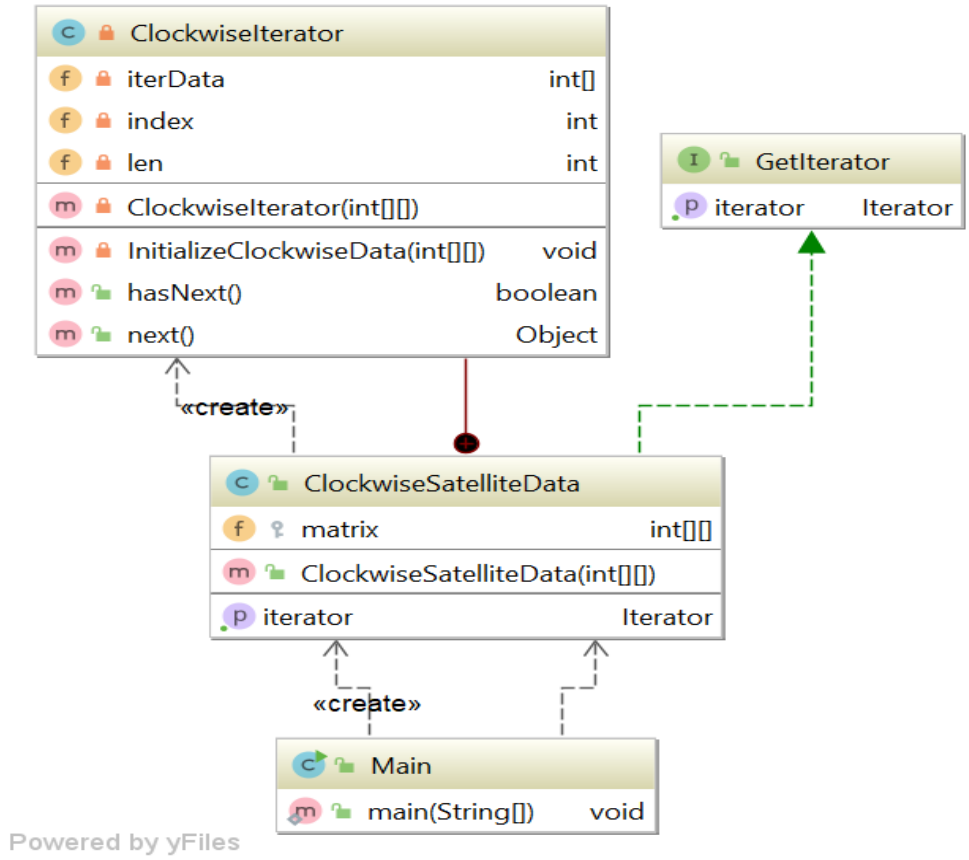
a) If the clone() method of the object is called in the clone method in the Parent class, or if this copy is returned by making an instance copy, a second object from the singleton object will become cloneable. So the Singleton property will be broken. If "CloneNotSupportedException" is thrown in the clone method inside the Parent class, a second object will not be created.

b) To prevent this situation, we override clone() method in Parent class or Singleton class and throw "CloneNotSupportedException".

Part2
Iterator Design Pattern

A class named `ClockwiseSatelliteData` was created and the `GetIterator` interface was implement to the class. The interface has just one method is `getIterator`() method. 2D data (matrix) is kept in `ClockwiseSatelliteData` class. The `ClockwiseIterator` class is coded as an inner class. A new iterator object is returned with the `getIterator`() method. Within the `ClockwiseIterator` class, 2D array has been converted to 1D array to be iterated spirally clockwise. The iterator has been iterated through this array. Also overrided `hasNext`() and `next`() methods.

Class Diagram:



Example Output:

```
int[][] matrix2 = {{1,2,3,4, 5,6,7,8, 9,10},{11,12,13,14, 15,16,17,18, 19,20},{21,22,23,24, 25,26,27,28, 29,30}};
ClockwiseSatelliteData data2 = new ClockwiseSatelliteData(matrix2);
Iterator<Integer> iter2 = data2.getIterator();

System.out.println("Satellite 2D Data: ");
for (int[] dat : matrix2) {
    for (int j = 0; j < matrix2[0].length; j++)
        System.out.print(dat[j] + " ");
    System.out.println();
}
System.out.println("\nClockwise Iterated Data : ");
while(iter2.hasNext())
    System.out.print(iter2.next() + " ");
System.out.println("\n--------------------------------------------------------------------------------");
```

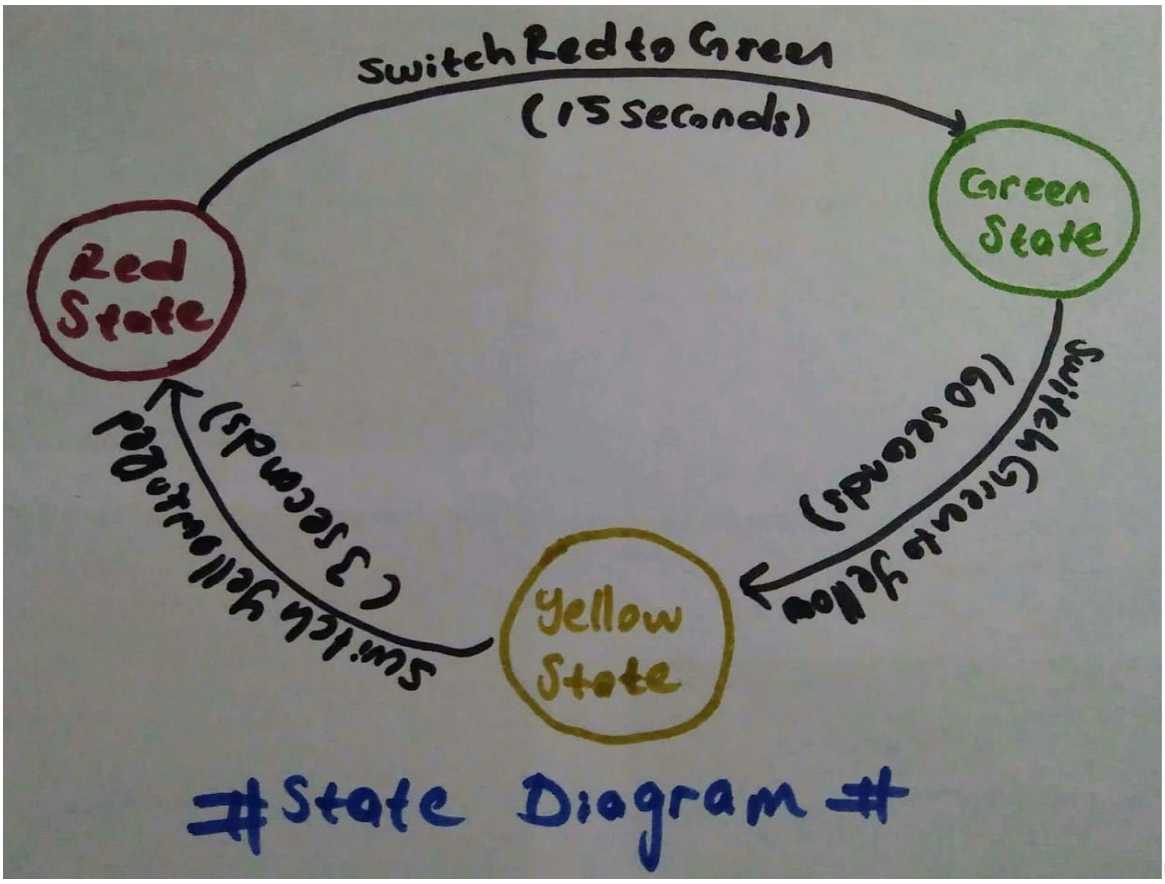
```java
public static void main(String[] args) {
    int[][] matrix = {{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};
    ClockwiseSatelliteData data = new ClockwiseSatelliteData(matrix);
    Iterator<Integer> iter = data.getIterator();

    System.out.println("Satellite 2D Data: ");
    for (int[] dat : matrix) {
        for (int j = 0; j < matrix[0].length; j++)
            System.out.print(dat[j] + " ");
        System.out.println();
    }
    System.out.println("\nClockwise Iterated Data : ");
    while(iter.hasNext())
        System.out.print(iter.next() + " ");
    System.out.println("\n-----------------------------------------------------------------------------------------");
```

```
"C:\Program Files\Java\jdk-14.0.2\bin\java.exe" -Didea.launcher.port=61244 "-Didea.launc
Satellite 2D Data:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16

Clockwise Iterated Data :
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
-----------------------------------------------------------------------------------------
Satellite 2D Data:
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30

Clockwise Iterated Data :
1 2 3 4 5 6 7 8 9 10 20 30 29 28 27 26 25 24 23 22 21 11 12 13 14 15 16 17 18 19
-----------------------------------------------------------------------------------------
```

Part3
State Design Pattern - Observer Design Pattern

The first state requested from us:

State Diagram:



Class Diagram:

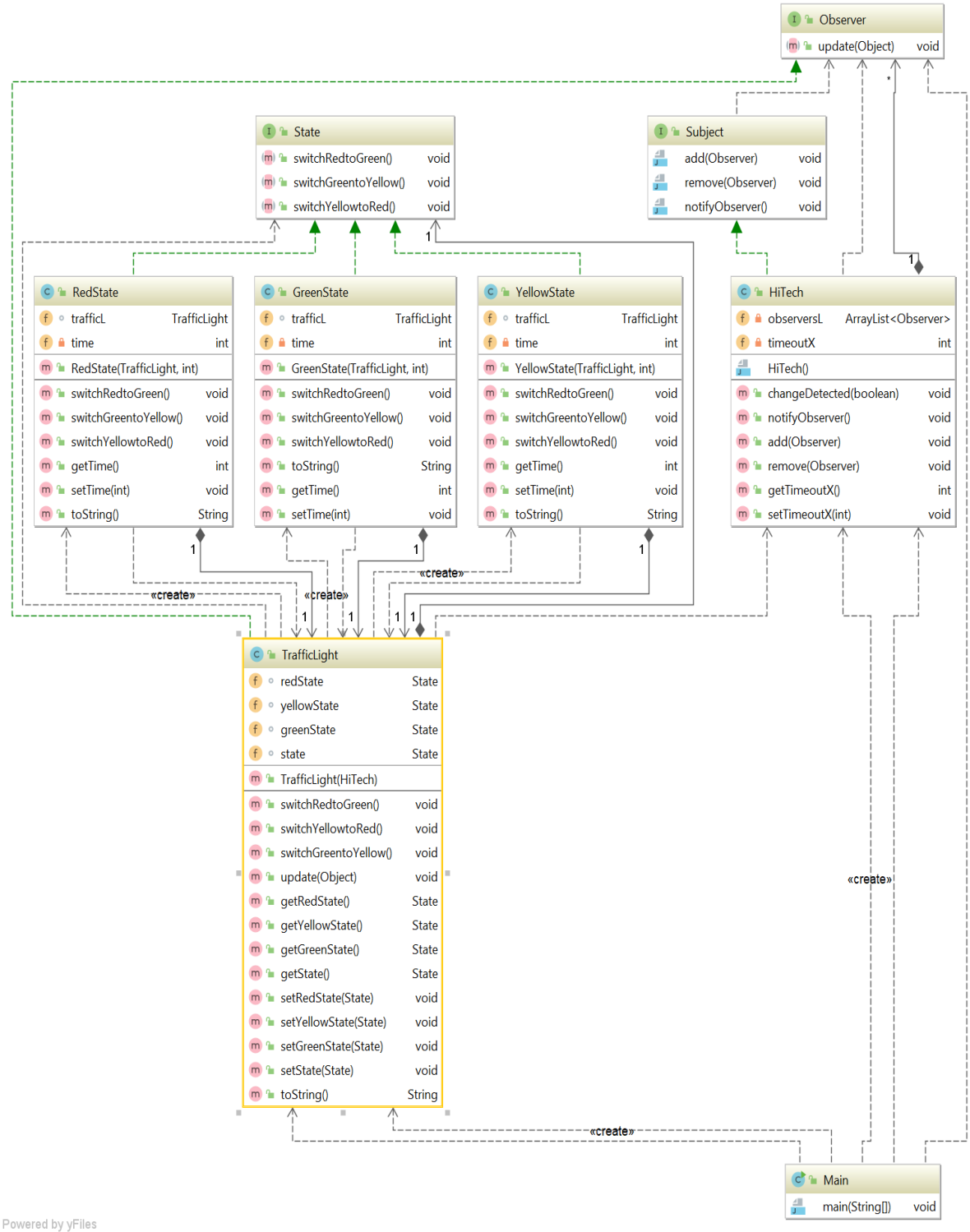
Example Output:

```
public static void main(String[] args) {
    TrafficLight trafficLight = new TrafficLight();

    trafficLight.switchRedtoGreen();
    trafficLight.switchGreentoYellow();
    trafficLight.switchYellowtoRed();
    System.out.println("\n");

    trafficLight.switchGreentoYellow();
    trafficLight.switchYellowtoRed();
    trafficLight.switchRedtoGreen();
    System.out.println("\n");

    trafficLight.switchGreentoYellow();
    trafficLight.switchRedtoGreen();
    trafficLight.switchYellowtoRed();
    System.out.println("\n");

    trafficLight.switchYellowtoRed();
    trafficLight.switchRedtoGreen();
    trafficLight.switchGreentoYellow();

}
```

```
Wait to switch Red to Green (15 seconds) ...
Now traffic light is Green.
Wait to switch Green to Yellow (60 seconds) ...
Now traffic light is Yellow.
Wait to switch Yellow to Red (3 seconds) ...
Now traffic light is Red.

Now traffic light is Red so can not action green to yellow.
Now traffic light is Red so can not action yellow to red.
Wait to switch Red to Green (15 seconds) ...
Now traffic light is Green.

Wait to switch Green to Yellow (60 seconds) ...
Now traffic light is Yellow.
Now traffic light is Yellow so can not action red to green.
Wait to switch Yellow to Red (3 seconds) ...
Now traffic light is Red.

Now traffic light is Red so can not action yellow to red.
Wait to switch Red to Green (15 seconds) ...
Now traffic light is Green.
Wait to switch Green to Yellow (60 seconds) ...
Now traffic light is Yellow.
```

The second state requested from us:

Class Diagram:



Example Output:

```
public static void main(String[] args)  {

    HiTech ht = new HiTech();
    Observer obs1 = new TrafficLight(ht);
    ht.add(obs1);
    TrafficLight trafficLight = new TrafficLight(ht);

    trafficLight.switchRedtoGreen();
    ht.changeDetected( flag: true);
    trafficLight.switchGreentoYellow();
    trafficLight.switchYellowtoRed();

    System.out.println("----------------------------------------");

    trafficLight.switchRedtoGreen();
    ht.changeDetected( flag: false);
    trafficLight.switchGreentoYellow();
    trafficLight.switchYellowtoRed();

    System.out.println("----------------------------------------");
```

```
"C:\Program Files\Java\jdk-14.0.2\bin\java.exe"
Wait to switch Red to Green ( 15 seconds) ...
Now traffic light is Green.
Now lot of traffic and wait 90 seconds ...
Now traffic light is Yellow.
Wait to switch Yellow to Red ( 3 seconds) ...
Now traffic light is Red.
----------------------------------------
Wait to switch Red to Green ( 15 seconds) ...
Now traffic light is Green.
Now traffic is normal and wait 60 seconds ...
Now traffic light is Yellow.
Wait to switch Yellow to Red ( 3 seconds) ...
Now traffic light is Red.
----------------------------------------

Process finished with exit code 0
```