Lab 11

```
public class SingletonComputerFactory {
   private static SingletonComputerFactory singletonFactory;

// SingletonExample prevents any other class from instantiating

   //YOUR CODE
   private SingletonComputerFactory()
   {

   }
// Providing Global point of access

   public static SingletonComputerFactory getSingletonFactory()
   {
      //YOUR CODE
      if (null == singletonFactory)
      {
         singletonFactory = new SingletonComputerFactory();
      }
      return singletonFactory;

   }

   public Computer getComputer(String type, String ram, String hdd, String cpu)
   {   if("PC".equalsIgnoreCase(type))
      return new PC(ram, hdd, cpu);
   else if("Server".equalsIgnoreCase(type))
      return new Server(ram, hdd, cpu);
      return null;
   }
}
```

```
public class TestFactory {

   public static void main(String[] args) {

      //Create an object of SingletonComputerFactory
      //YOUR CODE
      SingletonComputerFactory fc = SingletonComputerFactory.getSingletonFactory();
      Computer pc = fc.getComputer("pc","2 GB","500 GB","2.4 GHz");
      Computer server = fc.getComputer("server","16 GB","1 TB","2.9 GHz");
      System.out.println("Factory PC Config::"+pc);
      System.out.println("Factory Server Config::"+server);
   }

}
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_231.jdk/Contents/Home/bin/java ...
Factory PC Config::RAM= 2 GB, HDD=500 GB, CPU=2.4 GHz
Factory Server Config::RAM= 16 GB, HDD=1 TB, CPU=2.9 GHz

Process finished with exit code 0
```

```
import java.util.ArrayList;
import java.util.List;
public class MessagePublisher implements Subject {

    private List<Observer> observers = new ArrayList<>();

    @Override
    public void attach(Observer o)
    {
        //ADD o to observers
        observers.add(o);
    }

    @Override
    public void detach(Observer o)
    {
        //REMOVE o from observers
        observers.remove(o);
    }

    @Override
    public void notifyUpdate(Message m)
    {
        for(Observer o: observers) {
            //Call  update method
            o.update(m);
        }
    }
}
```

```
public class MessageSubscriberOne implements Observer
{
    @Override
    public void update(Message m)
    {
        System.out.println("MessageSubscriberOne :: " + m.getMessageContent());
    }
}
```

```
public class MessageSubscriberTwo implements Observer
{
  //YOUR CODE
  public void update(Message m)
  {
     System.out.println("MessageSubscriberTwo :: " + m.getMessageContent());
  }
}
```

```
public class MessageSubscriberThree implements Observer
{
  //YOUR CODE
  public void update(Message m)
  {
     System.out.println("MessageSubscriberThree :: " + m.getMessageContent());
  }
}
```

```
public class Message
{
  final String messageContent;
  public Message (String m)
  {
     this.messageContent = m;
  }

  public String getMessageContent()
  {
     return messageContent;
  }
}
```

```
public class Main
{
  public static void main(String[] args)
  {
     MessageSubscriberOne s1 = new MessageSubscriberOne();
     MessageSubscriberTwo s2 = new MessageSubscriberTwo();
     MessageSubscriberThree s3 = new MessageSubscriberThree();

     MessagePublisher p = new MessagePublisher();
//Attache s1 and s2 to p
     p.attach(s1);
     p.attach(s2);
//YOUR CODE

//Notify s1 and s2 with the message "First Message"
```

```
        p.notifyUpdate(new Message("First Message"));

//Deatach s1
        p.detach(s1);

//Attach s3
        p.attach(s3);

//Notify s2 and s3 with message "Second Message"
        p.notifyUpdate(new Message("Second Message"));
    }
}
```

Lab 12

```
/** Memento class contains functions
*
*/
public class Calculator {
   private int num1;
   private int num2;

   /**
    * Method adds two numbers
    *
    * @return sum
    */
   public int addTwoNumbers() {
       return num1 + num2;
   }

   /**
    * Method replaces numbers with new values
    *
    * @param num1
    * @param num2
    */
```

```
    public void setnumbers(int num1, int num2) {
        this.num1 = num1;
        this.num2 = num2;
    }

    /**
     * Method returns mementoCalc object with current numbers
     *
     * @return mementoCalc object of num1 and num2
     */
    public mementoCalc backUplastCalc() {
        return new mementoCalc(num1, num2);
    }

    /**
     * Restores two numbers from mementoCalc object
     *
     * @param mem mementoCalc object
     */
    public void restorePreviousCalc(mementoCalc mem) {
        num1 = ((mementoCalc) mem).getnum1();
        num2 = ((mementoCalc) mem).getnum2();
    }

    public int add() {
        return num1 + num2;
    }
}



/**
MementoPattern main
method
**
*/

public class MementoPattern {
    public static void main(String[]args) {
        int num1 = 2;
        int num2 = 3;
        Calculator calc = new Calculator();
        calc.setnumbers(num1, num2);
        int sum = calc.add();
        //Code to backup the above addition operation
        //YOUR CODE
        mementoCalc c = calc.backUplastCalc();

        System.out.println("The sum of " + num1 + " + " + num2 + " = " + sum);
        num1 = 5;
        num2 = 6;
        calc.setnumbers(num1, num2);
        sum = calc.add();
        System.out.println("The sum of " + num1 + " + " + num2 + " = " + sum);
```

```
        //Code to restore the previous addition operation
        //YOUR CODE
        calc.restorePreviousCalc(c);
        System.out.println("The last calculation was: " + c.getnum1() + " + " + c.getnum2() + " = " + calc.add());
    }
}
```

```
import java.util.ArrayList;
import java.util.List;

/** Caretaker class
*
*/
public class CalculatorCaretaker {
    List<mementoCalc> list = new ArrayList<mementoCalc>();

    /**Adds mementoCalc object to list
     * @param m mementoCalc object
     */
    public void add(mementoCalc m)
    {
        list.add(m);
    }

    /**Returns last mementoCalc object
     * @return mementoCalc object
     */
    public mementoCalc getLast() {
        return list.get(list.size() - 1);
    }
}
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_231.jdk/Contents/Home/bin/java ...
The sum of 2 + 3 = 5
The sum of 5 + 6 = 11
The last calculation was: 2 + 3 = 5

Process finished with exit code 0
```