CECS 277

Section 09

Tina L. Vu

11/27/20

Lab Assignment 11

**PART 1:**

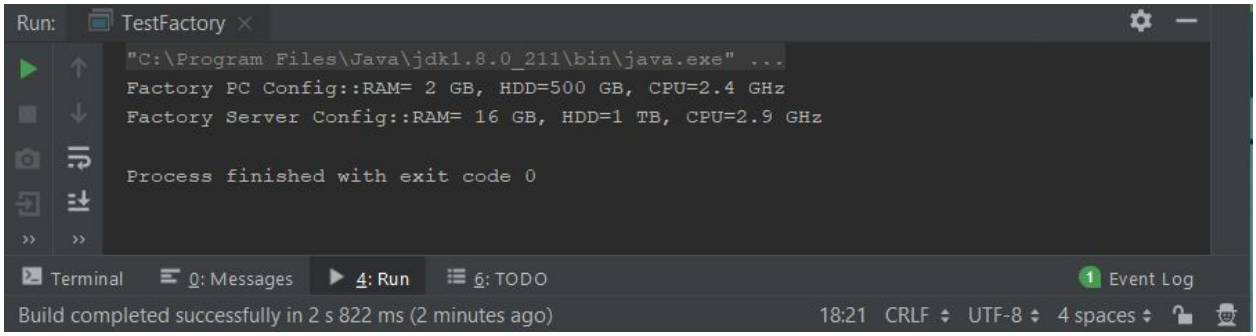
**SingletonComputerFactory CODE:**

```
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
      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;
   }
}
```

**TestFactory CODE:**

```
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);
   }
}
```

**RUNTIME OUTPUT**:




**<u>PART 2:</u>**

**Subject CODE:**

```
public interface Subject
{
    public void attach(Observer o);
    public void detach(Observer o);
    public void notifyUpdate(Message m);
}
```

**MessagePublisher CODE:**

```
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 (Your Code)
        observers.add(o);
    }

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

    @Override
    public void notifyUpdate(Message m) {
        for(Observer o: observers) {
            //Call update method (Your Code)
            int index = observers.indexOf(o);
            observers.get(index).update(m);
        }
```

```
    }
}
```

**Observer CODE:**

```
public interface Observer
{
    public void update(Message m);
}
```

**MessageSubscriberOne CODE:**

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

**MessageSubscriberTwo CODE:**

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

**MessageSubscriberThree CODE:**

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

**Message CODE:**

```
public class Message
{
    final String messageContent;

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

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

```
}
```

**Main CODE:**

```
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);

        //Notify s1 and s2 with the message "First Message
        s1.update(new Message("First Message"));
        s2.update(new Message("First Message"));

        //Deattach s1
        p.detach(s1);

        //just to make it neat
        System.out.println();

        //Attach s3
        p.attach(s3);

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

**RUNTIME OUTPUT:**