OOP MOC Paper

Q1)

a.

Ques1a.java

```
1  package threads.q1;
2
3  class PrintThread extends Thread {
4      public void run() {
5          for (int r = 1; r <= 100; r++) {
6              System.out.println(r);
7          }
8      }
9  }
10
11 class SLIITThread implements Runnable {
12     public void run() {
13         for (int r = 1; r <= 100; r++) {
14             System.out.println(" SLIIT ");
15         }
16     }
17 }
18
19 class Ques1a {
20     public static void main(String args[]) {
21         PrintThread thread1 = new PrintThread();
22         Thread thread2 = new Thread(new SLIITThread());
23         thread1.start();
24         thread2.start();
25     }
26 }
```

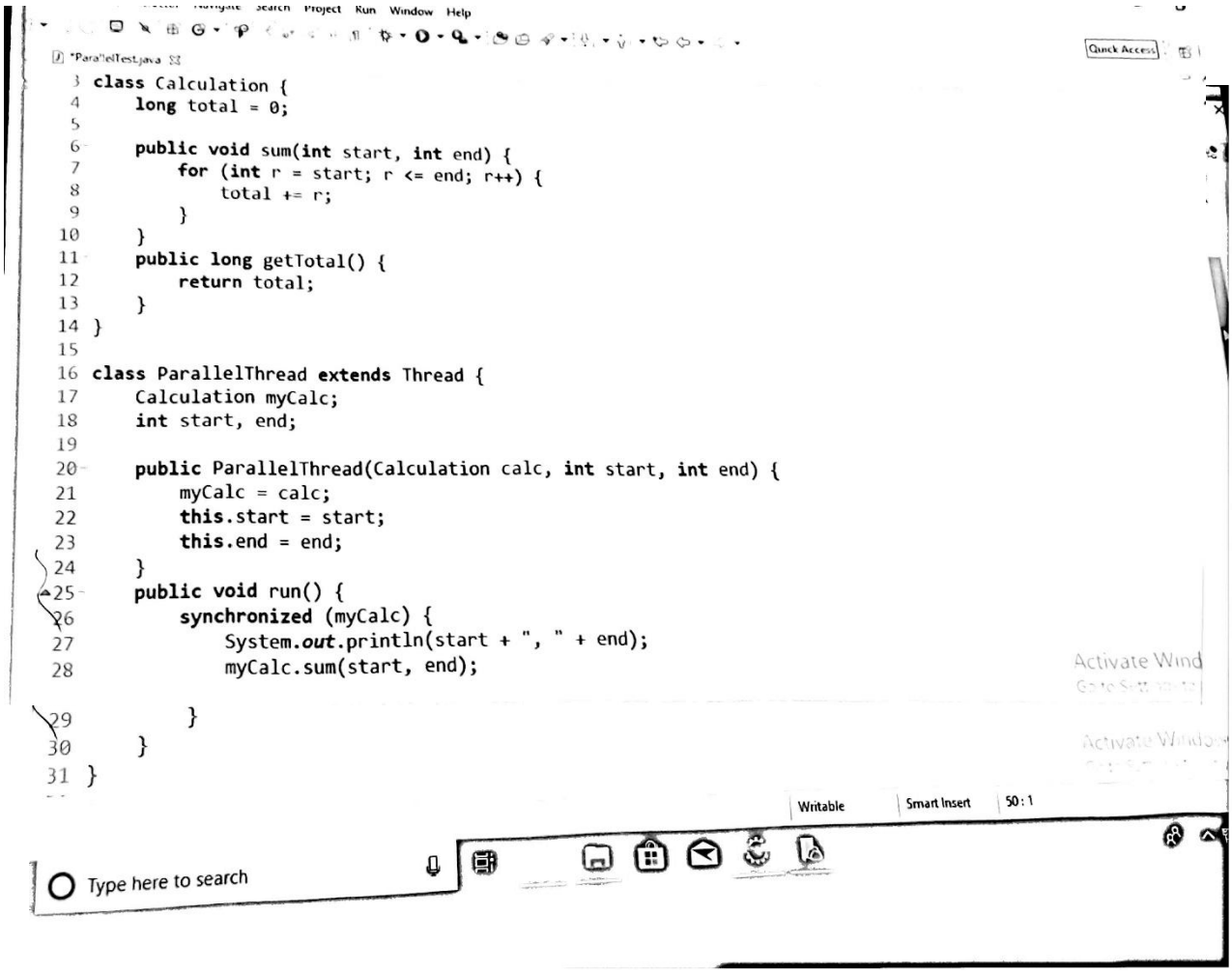
b.

Ques1b.java

```
3  class NumbersThread extends Thread {
4      public NumbersThread(){  }
5      public void run() {
6          synchronized(NumbersThread.class){
7              for (int r = 1; r <= 100; r++) {
8                  System.out.println("Thread Name : " + Thread.currentThread().getName() + " - " + r);
9              }
0          }
1      }
2  }
3  class Ques1b {
4      public static void main(String args[]) {
5          NumbersThread thread1 = new NumbersThread();
6          NumbersThread thread2 = new NumbersThread();
7          NumbersThread thread3 = new NumbersThread();
8          thread1.setName("Red");
9          thread2.setName("Blue");
0          thread3.setName("Green");
1          for (int r = 1; r <= 100; r++){
2              System.out.println("SLIIT");
3          }
4          thread1.start();
5          thread2.start();
6          thread3.start();
7          System.out.println("Thread 1 - State = " + thread1.getState() + " - Alive = " + thread1.isAlive());
8      }
```


Writable | Smart Insert | 16:51

c.



```java
class Calculation {
    long total = 0;

    public void sum(int start, int end) {
        for (int r = start; r <= end; r++) {
            total += r;
        }
    }
    public long getTotal() {
        return total;
    }
}

class ParallelThread extends Thread {
    Calculation myCalc;
    int start, end;

    public ParallelThread(Calculation calc, int start, int end) {
        myCalc = calc;
        this.start = start;
        this.end = end;
    }
    public void run() {
        synchronized (myCalc) {
            System.out.println(start + ", " + end);
            myCalc.sum(start, end);

        }
    }
}
```



```java
        }
    }
}

public class ParallelTest {
    public static void main(String args[]) throws InterruptedException {
        Calculation mycalc = new Calculation();
        ParallelThread thread1 = new ParallelThread(mycalc, 1, 25000);
        ParallelThread thread2 = new ParallelThread(mycalc, 25001, 50000);
        ParallelThread thread3 = new ParallelThread(mycalc, 50001, 75000);
        ParallelThread thread4 = new ParallelThread(mycalc, 75001, 100000);
        thread1.start();
        thread2.start();
        thread3.start();
        thread4.start();

        thread1.join();
        thread2.join();
        thread3.join();
        thread4.join();

        System.out.println("Total " + mycalc.getTotal());

    }
}
```

Q2)

a.

LoginTest.java

```
3  class Login {
4
5      private static Login instance = null;
6      private Login(){}
7
8      public static Login getInstance(){
9          if(instance == null){
10             instance = new Login();
11             System.out.println("Instance created");
12         }
13         return instance;
14     }
15
16     public boolean validateUser(String userName, String password){
17         return userName.equals(password);
18     }
19 }
20 public class LoginTest{
21     public static void main(String[] args) {
22         for (int i = 0; i < 10; i++) {
23             System.out.println(Login.getInstance().validateUser("Manju", "Manju"));
24         }
25     }
26 }
27
28
```

```
<terminated> LoginTest (Java Application) CAP
Instance created
true
true
true
true
true
true
true
true
true
true
```

b.

TVFactory.java

```
1  package design.patterns.q2;
2
3  public class TVFactory extends AbstractFactory{
4
5      @Override
6      public TV getTVmodel(String model) {
7
8          if(model.equals("Alpha40")){
9              return new Alpha40();
0          }
1          else if(model.equals("Theta65")){
2              return new Theta65();
3          }
4          else{
5              return new Gamma50();
6          }
7      }
8
)      @Override
)      public MobilePhone getPhoneModel(String mode
           return null;
       }
```

MobileFactory.java

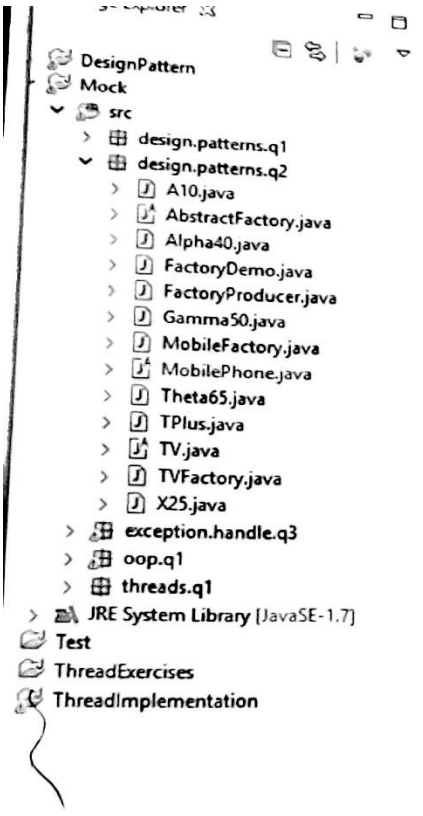
```
1  package design.patterns.q2;
2
3  public class MobileFactory extends AbstractFactory{
4
5
6      @Override
7      public MobilePhone getPhoneModel(String model) {
8
9          if(model.equals("X25")){
10             return new X25();
11         }
12         else if(model.equals("A10")){
13             return new A10();
14         }
15         else {
16             return new TPlus();
17         }
18     }
19
20     @Override
21     public TV getTVmodel(String model) {
22         return null;
23     }
24 }
```

AbstractFactory.java

```java
package design.patterns.q2;

public abstract class AbstractFactory {

    public abstract TV getTVmodel(String model);

    public abstract MobilePhone getPhoneModel(String model);

}
```

FactoryProducer.java

```java
package design.patterns.q2;

public class FactoryProducer {

    public static AbstractFactory getFactory(String type){

        if(type.equals("TV")){
            return new TVFactory();
        }
        else{
            return new MobileFactory();
        }
    }

```



*TV.java

```java
package design.patterns.q2;

public abstract class TV {

    public String model;

    public String size;

    public abstract void display();
}
```

MobilePhone.java

```java
package design.patterns.q2;

public abstract class MobilePhone {

    public String model;

    public double price;

    public abstract void display();

}

```

A10.java | Alpha40.java | X25.java

```java
package design.patterns.q2;

public class X25 extends MobilePhone{

    public X25() {
        super.model = "X25";
        super.price = 82000.00;
    }

    @Override
    public void display() {
        System.out.println("The model is = " + super.model + " and price is = " + super.price)
    }

}
```

A10.java | Alpha40.java | X25.java | Theta65.java

```java
package design.patterns.q2;

public class Theta65 extends TV{

    public Theta65() {
        super.model = "Theta65";
        super.size = "120 inches";
    }

    @Override
    public void display() {
        System.out.println("The TV model = " + super.model + ", the TV size is = " + super.siz
    }

}
```

ra - Java - Mock/src/design/patterns/q2/FactoryDemo.java - Eclipse

dit Source Refactor Navigate Search Project Run Window Help

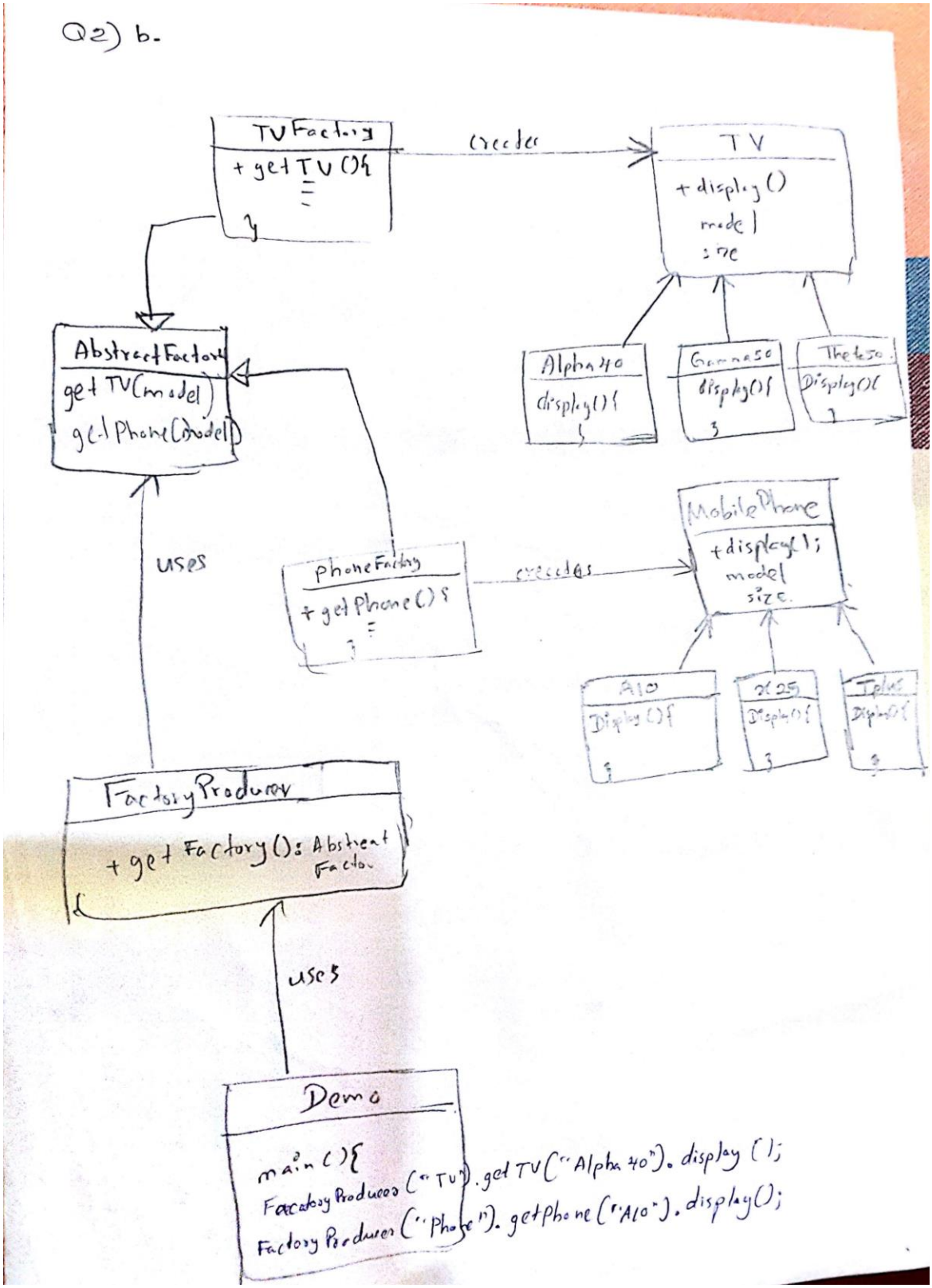
Theta65.java | TPlus.java | FactoryDemo.java

```java
package design.patterns.q2;

public class FactoryDemo {

    public static void main(String[] args) {

        FactoryProducer.getFactory("TV").getTVmodel("Alpha40").display();
        FactoryProducer.getFactory("TV").getTVmodel("Gamma50").display();
        FactoryProducer.getFactory("TV").getTVmodel("Theta65").display();

        FactoryProducer.getFactory("Phone").getPhoneModel("A10").display();
        FactoryProducer.getFactory("Phone").getPhoneModel("X25").display();
        FactoryProducer.getFactory("Phone").getPhoneModel("TPlus").display();
    }

}
```

Problems | Javadoc | Declaration | Console

```
<terminated> FactoryDemo (Java Application) C:\Program Files\Java\j
The TV model = Alpha40, the TV size is = 65 inches
The TV model = Gamma50, the TV size is = 77 inches
The TV model = Theta65, the TV size is = 120 inches
The model is = A10 and price is = 80000.0
The model is = X25 and price is = 82000.0
The model is = TPlus and price is = 76000.0
```

Q2) b.


TVFactory
+getTV(){
}
creates
TV
+display()
model
size
AbstractFactory
getTV(model)
getPhone(model)
Alpha40
display(){
}
Gamma50
display(){
}
Theta50
Display(){
}
uses
PhoneFactory
+getPhone(){
}
creates
MobilePhone
+display();
model
size
A10
Display(){
}
X25
Display(){
}
Iplus
Display(){
}
FactoryProducer
+getFactory(): AbstractFactory
uses
Demo
main(){
FactoryProducer("TV").getTV("Alpha40").display();
FactoryProducer("Phone").getPhone("A10").display();

Q3) Done in Lab 7-Exception Handling

Q4)

Vehicle.java

```java
package oop.q1;

public abstract class Vehicle {

    private double speed;
    private String colour;
    public double regularPrice = 10000.00;

    public Vehicle(double speed, String colour) {
        super();
        this.speed = speed;
        this.colour = colour;
    }

    public Vehicle(double speed, String colour, double regularPrice) {
        super();
        this.speed = speed;
        this.colour = colour;
        this.regularPrice = regularPrice;
    }

    public abstract double getSalePrice();
}
```

Vehicle.java Truck.java

```java
package oop.q1;
public class Truck extends Vehicle{

    private double weight;

    public Truck(double speed, String colour, double weight) {
        super(speed, colour);
        this.weight = weight;
    }

    @Override
    public double getSalePrice() {

        double salesPrice = 0.0;

        if(weight > 2000){
            salesPrice = (super.regularPrice * 90)/100;
        }
        else{
            salesPrice = (super.regularPrice * 80)/100;
        }
        return salesPrice;
    }
}
```

Vehicle.java Truck.java Bus.java

```java
package oop.q1;

public class Bus extends Vehicle{

    private int year;

    private double manufacturerDiscount;

    public Bus(double speed, String colour, int year, double manufacturerDiscount) {
        super(speed, colour);
        this.year = year;
        this.manufacturerDiscount = manufacturerDiscount;
    }

    @Override
    public double getSalePrice() {

        double salesPrice = 0.0;
        salesPrice = (super.regularPrice * (100 - manufacturerDiscount))/100;
        return salesPrice;
    }

}
```

Vehicle.java Truck.java Bus.java MyOwnAutoShop.java

```java
package oop.q1;

public class MyOwnAutoShop {

    public static void main(String[] args) {

        Vehicle vehicle = new Truck(150, "Blue", 1000);
        System.out.println(vehicle.getSalePrice());

        Vehicle truck = new Truck(160, "Red", 3000);
        System.out.println(truck.getSalePrice());

        Vehicle bus = new Bus(250, "White", 2017, 40);
        System.out.println(bus.getSalePrice());

    }

}
```