Q.1 Factory Method:

```
import java.util.*;
interface mainorganisation {
        void run();
}
class buyer implements mainorganisation{
        @Override
        public void run() {
                System.out.println("Buyer called");
        }



}
  class seller implements mainorganisation{
        @Override
        public void run() {
                System.out.println("Seller called");
        }
}
  class orgFactory{
        public mainorganisation getorg(String orgType){
                if(orgType==null) return null;
                if(orgType.equalsIgnoreCase("BUYER")){
                        return new buyer();
                } else if (orgType.equalsIgnoreCase("SELLER")) {
                        return new seller();
                }
        return null;
    }
}
public class h1 {
    public static void main(String[] args) {
        orgFactory orgFactory =new orgFactory();


        mainorganisation org1=orgFactory.getorg("Buyer");
        org1.run();
        mainorganisation org2=orgFactory.getorg("SELLER");
        org2.run();


    }
}
```

Haard Shah

21BCP251

CS G-8

Builder Design pattern

```
class org{
	private String name;
	private String city;
	private boolean isBuyer;
	private double pincode;

	org(String name, String city , boolean isBuyer, double pincode){
		super();
		this.name=name;
		this.city=city;
		this.isBuyer=isBuyer;
		this.pincode=pincode;
	}
	public String toString(){
		return "org[name="+name+",city="+city+",IsBuyer ="+isBuyer+",Pincode="+pincode+"]";

	}

}

  class orggettersetter {

	private String name;
	private String city;
	private boolean isBuyer;
	private double pincode;
```

Haard Shah

21BCP251

CS G-8

```
    public orggettersetter setName(String name){
        this.name=name;
        return this;
    }

    public orggettersetter setCity(String city){
        this.city=city;
        return this;
    }

    public orggettersetter setState(boolean isBuyer){
        this.isBuyer=isBuyer;
        return this;
    }

    public orggettersetter setPincode(double pincode){
        this.pincode=pincode;
        return this;
    }

    public org getOrg(){
        return new org(name,city,isBuyer,pincode);
    }

}
public class market{
```

Haard Shah

21BCP251

CS G-8

```
	public static void main(String[] args) {
		org o1=new
orggettersetter().setCity("Ahmedabad").setName("ABCD").setPincode(380008).setState(true).getOrg();
		System.out.println(o1);
	}
}
```

Prototype Method

Class- prototype

Main.java

package com.prototype;

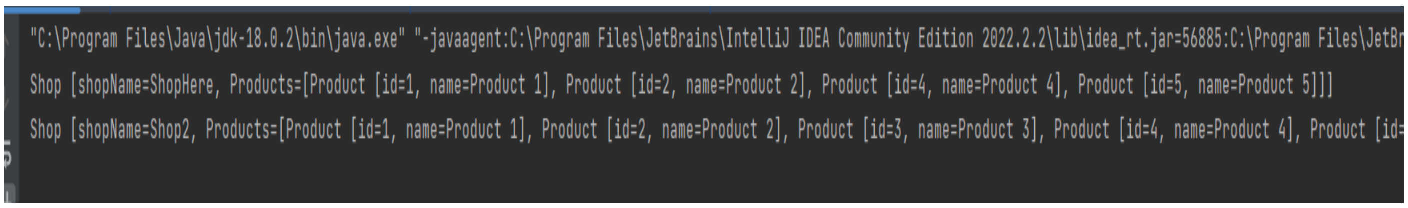
```
public class Main {
	public static void main(String[] args) throws CloneNotSupportedException {

		shop s1= new shop();
		s1.setShopName("ShopHere");
		s1.loadData();



		shop s2= (shop) s1.clone();
		s1.getProductList().remove(2);
		s2.setShopName("Shop2");
		System.out.println(s1);
		System.out.println(s2);
	}
}
```

Products.java

```
package com.prototype;
public class products {
        private int id;
        private String name;

public int getId() {
                return id;
                }
public void setId(int id) {
                this.id= id;
                }
public String getName() {
                return name;
                }
public void setName(String name) {
                this.name= name;
                }
@Override
public String toString() {
                return "Product [id="+ id+ ", name="+ name+ "]";
                }
}
```

Shop.java

```
package com.prototype;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

public class shop implements Cloneable{
        private String shopName;
          List<products> productsList= new ArrayList<>();
        public String getShopName() {
                return shopName;
        }
        public void setShopName(String shopName) {
                this.shopName= shopName;
        }
        public    List<products> getProductList() {
                return productsList;
        }
        public void setProductsList(List<products> productsList) {
                this.productsList= productsList;
        }

        public void loadData(){
                for(int i=1; i<=5; i++){
                        products p= new products();
                        p.setId(i);
                        p.setName("Product "+i);
                getProductList().add(p);
            }
        }
        @Override
        public String toString() {
            return "Shop [shopName="+ shopName+ ", Products="+ productsList+
"]";
        }

        @Override
        protected Object clone() throws CloneNotSupportedException {
            shop shopx= new shop();
            for(products p: this.getProductList()) {
                shopx.getProductList().add(p);
            }return shopx;
        }
}
```



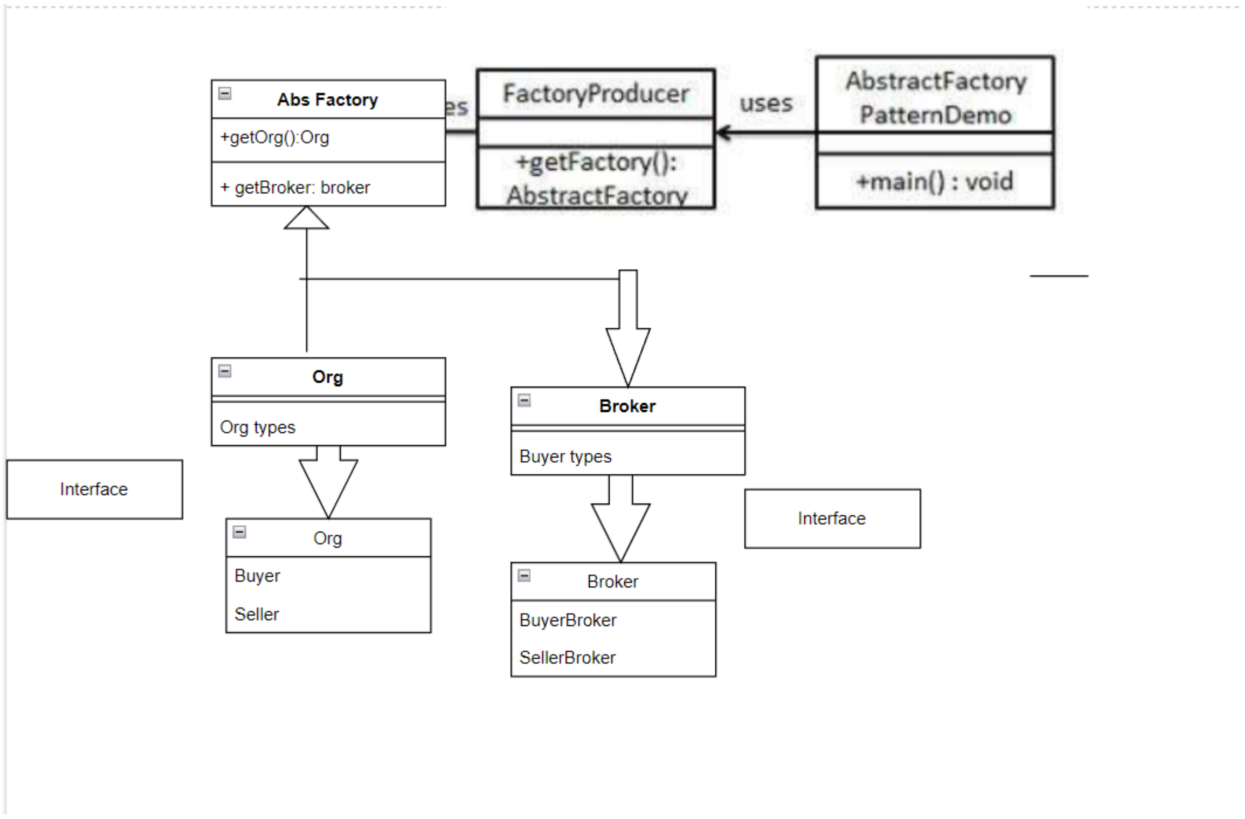
```
"C:\Program Files\Java\jdk-18.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.2.2\lib\idea_rt.jar=56885:C:\Program Files\JetBr
Shop [shopName=ShopHere, Products=[Product [id=1, name=Product 1], Product [id=2, name=Product 2], Product [id=4, name=Product 4], Product [id=5, name=Product 5]]]
Shop [shopName=Shop2, Products=[Product [id=1, name=Product 1], Product [id=2, name=Product 2], Product [id=3, name=Product 3], Product [id=4, name=Product 4], Product [id=
```

Haard Shah

21BCP251

CS G-8

Abstract Factory Method



Code:

```
package abstractFactoryMethod;

interface Organisation {

	void createOrg();

}

interface Broker {

	void createBroker();

}

class Buyer implements Organisation {

	@Override
```

Haard Shah

21BCP251

CS G-8

```
    public void createOrg() {
        System.out.println("Buyer is being created.");
    }
}
class Seller implements Organisation {
    @Override
    public void createOrg() {
        System.out.println("Seller is being created.");
    }
}
class BuyerBroker implements Broker {
    @Override
    public void createBroker() {
        System.out.println("Buyer Broker is being created");
    }
}
class SellerBroker implements Broker {
    @Override
    public void createBroker() {
        System.out.println("Seller Broker is being created");
    }
}
interface AbsFactory{
    Organisation createOrg(String type);
    Broker createBroker(String type);
}
class Buyerfac implements AbsFactory{
```

Haard Shah

21BCP251

CS G-8

```
    @Override
    public Buyer createOrg(String type) {
        if(type.equalsIgnoreCase("buyer")){
            return new Buyer();
        }
        return null;
    }

    @Override
    public BuyerBroker createBroker(String type) {
        if(type.equalsIgnoreCase("BrokerBuyer")){
            return new BuyerBroker();
        }
        return null;
    }
}
class sellerfac implements AbsFactory{

    @Override
    public Seller createOrg(String type) {
        if(type.equalsIgnoreCase("Seller")){
            return new Seller();
        }
        return null;
    }
```

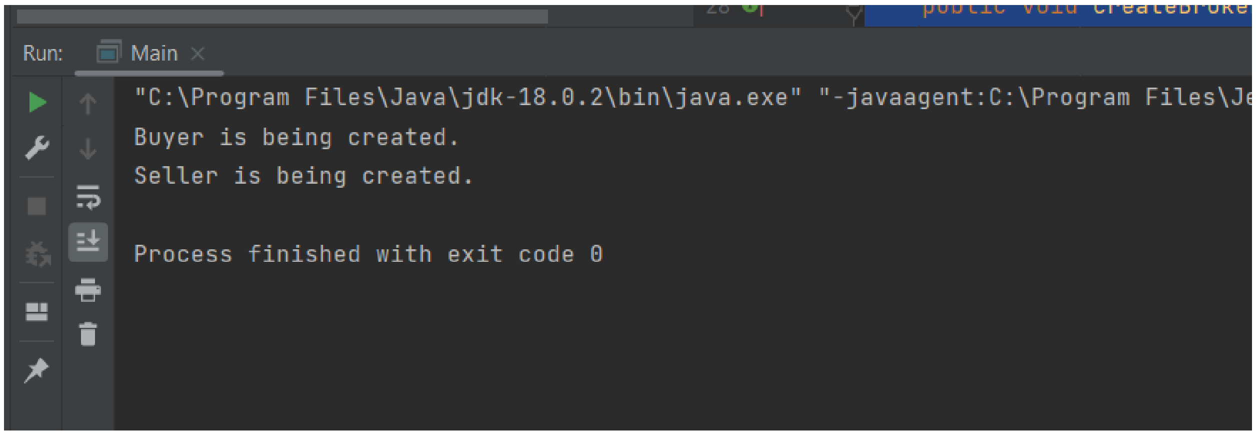
Haard Shah

21BCP251

CS G-8

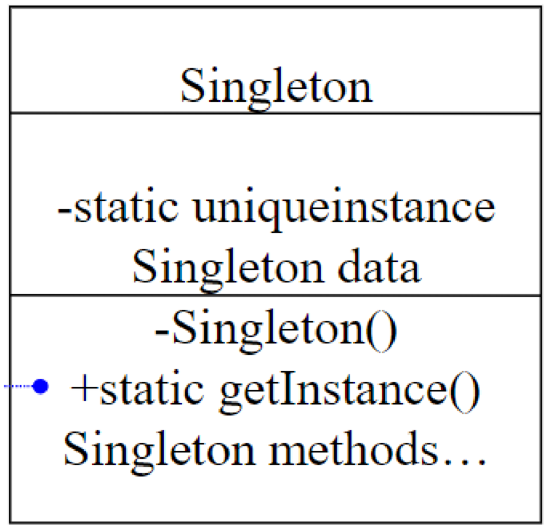
```
    @Override
    public SellerBroker createBroker(String type) {
        if(type.equalsIgnoreCase("BrokerSeller")){
            return new SellerBroker();
        }
        return null;
    }
}
public class Main {
    public static void main(String[] args) {
        Buyerfac of1 = new Buyerfac();
        Buyer c1 =    of1.createOrg("Buyer");
        c1.createOrg();
        sellerfac of2 = new sellerfac();
        Seller oc1 = of2.createOrg("Seller");
        oc1.createOrg();
    }
}
```

Output:



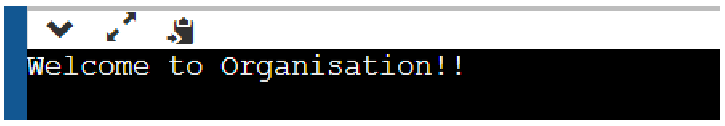
```
"C:\Program Files\Java\jdk-18.0.2\bin\java.exe" "-javaagent:C:\Program Files\Je
Buyer is being created.
Seller is being created.

Process finished with exit code 0
```

Haard Shah
21BCP251
CS G-8



Classical Singleton

```
class Organisation{
    public static Organisation obj = new Organisation(); //initialises and creates an object once
    private Organisation(){
        System.out.println("Welcome to Organisation!!");
    }
    public static Organisation getInstance(){
        return obj;
    }
}
public class Main{
            public static void main(String[] args) {

                //Eager Singleton
                Organisation o1 = Organisation.getInstance();
                Organisation o2 = Organisation.getInstance();
        }
}
```



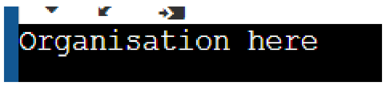
```
Welcome to Organisation!!
```

Lazy Singleton

```
class Organisation{
    public static Organisation obj; //initialises the object
    private Organisation(){
```

Haard Shah
21BCP251
CS G-8

```
        System.out.println("Organisation here");
    }
    public static Organisation getInstance(){
        if(obj==null){ //checks if object of class is not created- singleton concept
            obj = new Organisation(); //creates an object at the time of instantiation
        }
        return obj;
    }
}
public class Main{
            public static void main(String[] args) {

                //Lazy Singleton
                Organisation o3 = Organisation.getInstance();
                Organisation o4 = Organisation.getInstance();
        }
}
```



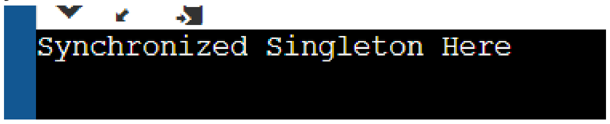
```
Organisation here
```

Synchronized Singleton

```
class Organisation{
    public static Organisation s; //initialises the object
    private Organisation(){
        System.out.println("Synchronized Singleton Here");
    }
    // Synchronized: locks a single thread with the shared data so that no other thread can
access it.
    public static synchronized Organisation getInstance(){
        if(s==null){
            s = new Organisation(); //creates the object
        }
        return s;
    }
}
public class Main{
            public static void main(String[] args) {

                //Synchronized Singleton
                Thread t1 = new Thread(new Runnable(){
                    public void run() { Organisation obj = Organisation.getInstance();}
                });
```

Haard Shah
21BCP251
CS G-8

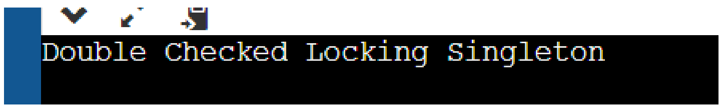
```
        Thread t2 = new Thread(new Runnable(){
            public void run() {Organisation obj = Organisation.getInstance();}
        });
        t1.start();
        t2.start();
    }
}
```



```
Synchronized Singleton Here
```

Double Checked Locking Singleton

```
class Organisation{
    public static Organisation s; //initialises the object
    private Organisation(){
        System.out.println("Double Checked Locking Singleton");
    }
    public static Organisation getInstance(){
        if(s==null){
            synchronized(Organisation.class){ //locking class for a single thread
                if(s==null) s = new Organisation(); //double checking and then creating object
            }
        }
        return s;
    }
}
public class Main{
    public static void main(String[] args) {

        //Double checked locking Singleton
        Thread t1 = new Thread(new Runnable(){
            public void run() {Organisation obj = Organisation.getInstance();}
        });
        Thread t2 = new Thread(new Runnable(){
            public void run() {Organisation obj = Organisation.getInstance();}
        });
        t1.start();
        t2.start();
}
}
```

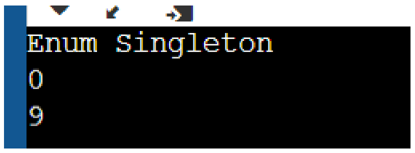


```
Double Checked Locking Singleton
```

Haard Shah
21BCP251
CS G-8

enum Singleton

```
enum Singleton{
   INSTANCE;
   int i;
   public int getI(){
      return i;
   }
   public void setI(int i){this.i = i;}

}
public class Main{
         public static void main(String[] args){

            System.out.println("Enum Singleton");
            Singleton obj1 = Singleton.INSTANCE;
            System.out.println(obj1.getI()); //i=0 default integer value

            Singleton obj2 = Singleton.INSTANCE;
            obj2.setI(11); //i=11 earlier i=0
            obj1.setI(9); //i=9 earlier i=11
            //obj1 & obj2 are one object of same class and so value of i will be updated
            System.out.println(obj2.getI()); //so 9 is the output and not 11
         }
}
```

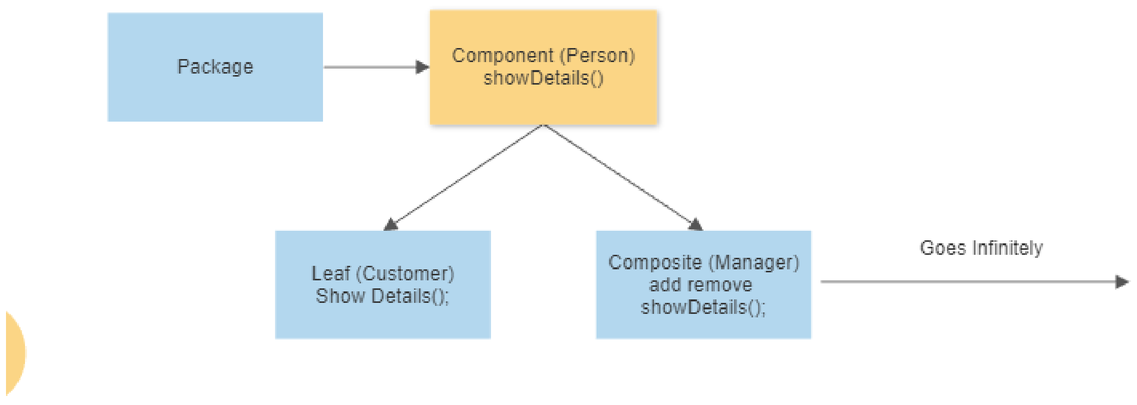


```
Enum Singleton
0
9
```

Haard Shah

21BCP251

CS G-8


Package
Component (Person) showDetails()
Leaf (Customer) Show Details();
Composite (Manager) add remove showDetails();
Goes Infinitely

```
package com.CompositeDP;//creating a package


import java.util.ArrayList;

import java.util.List;


  interface person {

        void showDetails();//declaring operation

}

//package CompositeDP;

class Newcustomer implements person {//leaf

        private String name;

        private int number;

        public Newcustomer(String name, int n) {

                this.name = name;
        this.number = n;
    }
    @Override
    public void showDetails() {//operation of leaf
        System.out.println("Customer_Name:" + name);
        System.out.println("Number:" + number);
    }
}
//package CompositeDP;
class Newmanager implements person {//leaf
    private String name;
    private int number;
    public Newmanager(String name, String e, int n) {
        this.name = name;

        this.number = n;
    }
    @Override
    public void showDetails() {//operation of leaf 2
        System.out.println("Manager_Name:" + name);
        System.out.println("Number:" + number);
    }
}
//package CompositeDP;

class organisation implements person {
    private List<person> personlist = new ArrayList<person>();
```

Haard Shah

21BCP251

CS G-8

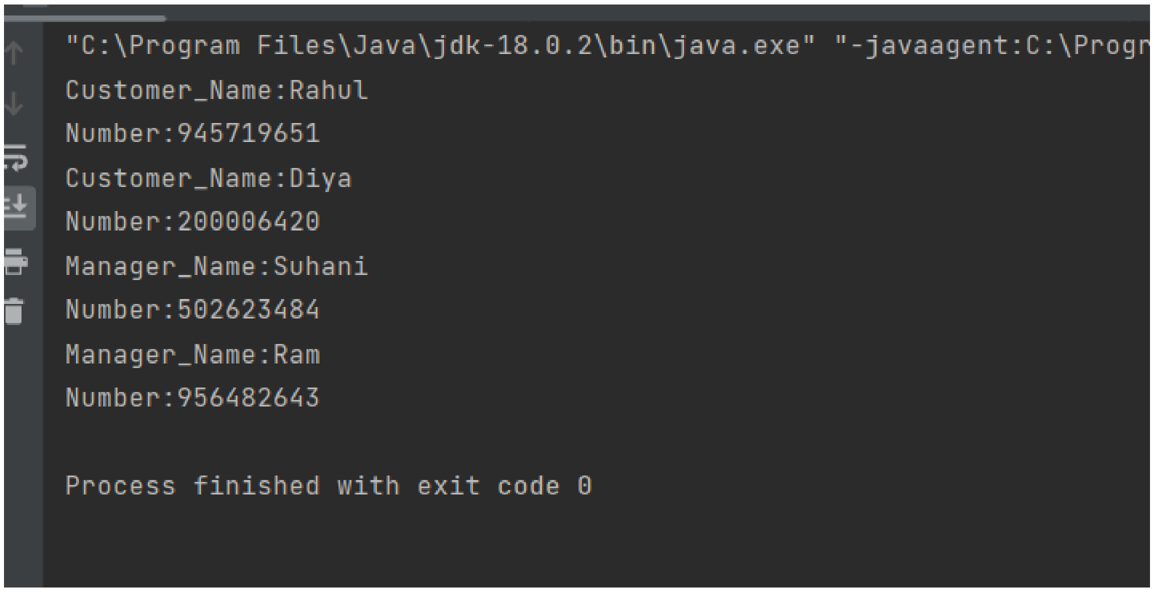
```
    @Override
    public void showDetails() {
        for (person p : personlist) {
            p.showDetails();
        }
    }
    public void addPerson(person p) {
        personlist.add(p);
    }
    public void removePerson(person p) {
        personlist.remove(p);
    }
}
public class Main {
    public static void main(String args[]) {

        Newcustomer c1 = new Newcustomer("Rahul",
                   945719651);
        Newcustomer c2 = new Newcustomer("Diya",
                   200006420);
        organisation org = new organisation();
        org.addPerson(c1);
        org.addPerson(c2);
        Newmanager m1 = new Newmanager("Suhani",
                  "suhani.malhotra@gmail.com", 502623484);
        Newmanager m2 = new Newmanager("Ram", "ram.shah45@gmail.com",
                  956482643);
```

Haard Shah

21BCP251

CS G-8

```
        organisation org2 = new organisation();
        org2.addPerson(m1);
        org2.addPerson(m2);
        organisation org3 = new
                organisation();
        org3.addPerson(org);
        org3.addPerson(org2);
        org3.showDetails();
    }
}
```

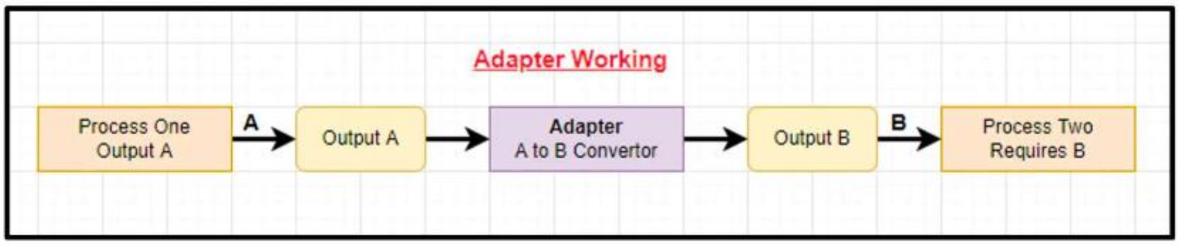


```
"C:\Program Files\Java\jdk-18.0.2\bin\java.exe" "-javaagent:C:\Progr
Customer_Name:Rahul
Number:945719651
Customer_Name:Diya
Number:200006420
Manager_Name:Suhani
Number:502623484
Manager_Name:Ram
Number:956482643

Process finished with exit code 0
```

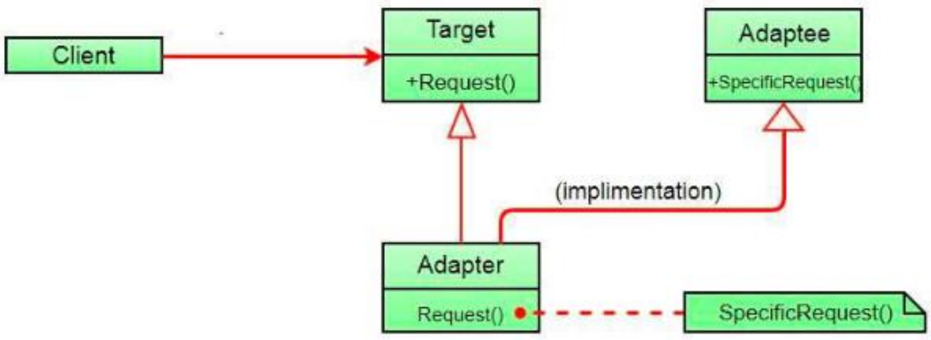
Haard Shah
21BCP251
CS G-8

ADAPTER DESIGN PATTERN

Working:

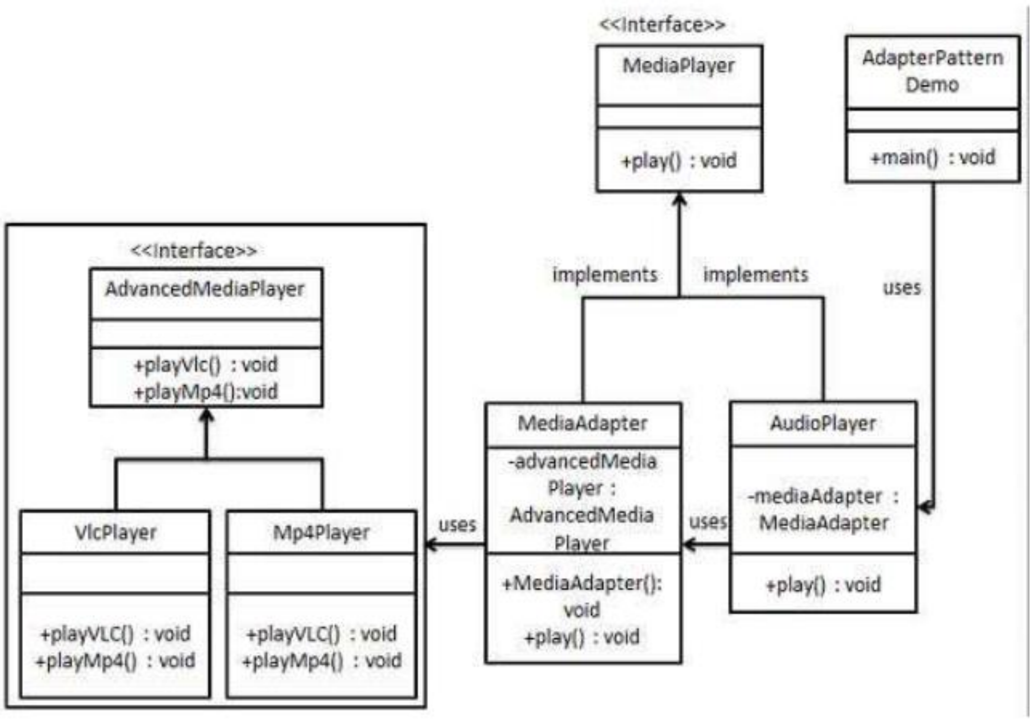


Components:



Haard Shah
21BCP251
CS G-8

Class Diagram:



Code:

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

Haard Shah
21BCP251
CS G-8

```java
interface MathsProcessing{
    public void performMathOperation(String type,int[]
arr,int key,int value);
}

class ArrayToList{
    public  List<Integer> ConvertArrayToList(int[] arr){
        List<Integer> ls = new ArrayList<Integer>();
        for(int i:arr){
            ls.add(i);
        }
        return ls;
    }
}

class BasicMathsProcessing implements MathsProcessing{
    AdvancedMathsProcessing amp = new
AdvancedMathsProcessing();
    ArrayToList atl = new ArrayToList();
    List<Integer> ls = new ArrayList<Integer>();



    public void performMathOperation(String type,int[]
arr,int key,int value) {
        ls= atl.ConvertArrayToList(arr);

        if(type=="+"){
            amp.addition(arr);
        }
        else if(type=="*"){
            amp.multiply(arr);
        }
        else if(type.equalsIgnoreCase("Sort")){
            amp.sort(ls);
        }
```

Haard Shah
21BCP251
CS G-8

```
        else if(type.equalsIgnoreCase("Avg")){
            amp.average(ls);
        }

        else if(type.equalsIgnoreCase("Search")){
            amp.SearchData(ls, key);
        }
        else if(type.equalsIgnoreCase("Replace")){
            amp.replace(ls, key, value);
        }

    }
}

class AdvancedMathsProcessing{
    float result;
    ArrayToList AL1 = new ArrayToList();
    // List<Integer> ls = new ArrayList<Integer>();

    // Iterator itr = ls.iterator();

    public void addition(int[] arr){
        result = 0;
        for(int i=0;i<arr.length;i++){
            result+=arr[i];
        }
        System.out.println("Sum of array element is : " +
(int)result);
    }

    public void multiply(int[] arr){
        result = 1;
        for(int i=0;i<arr.length;i++){
            result*=arr[i];
        }
        System.out.println("Product of array element is : "
+ (int)result);
```

Haard Shah
21BCP251
CS G-8

```
    }

    public void average(List<Integer> ls){
        result = 0;
        for(int i=0;i<ls.size();i++){
            result+=ls.get(i);
        }
        result = (Float) result/(ls.size());
        System.out.println("Average of array element is : "
+ result);
    }

    public void sort(List<Integer> ls){
        System.out.println("Before : ");
        for(int i=0;i<ls.size();i++){
            System.out.print(ls.get(i) + " ");
        }

        Collections.sort(ls);
        System.out.println();
        System.out.println("After");
        for(int i=0;i<ls.size();i++){
            System.out.print(ls.get(i) + " ");
        }

    }

    public void SearchData(List<Integer> ls,int key){
        if(ls.contains(key)){
            System.out.println("Number " + key +" is
present");
        }
        else{
            System.out.println("Number " + key +" is
present");
        }
    }
```

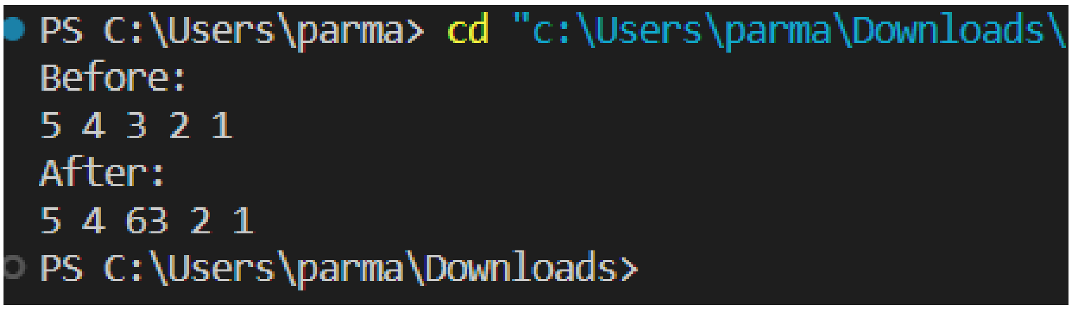
Haard Shah
21BCP251
CS G-8

```java
    public void replace(List<Integer> ls,int key,int value){
        System.out.println("Before:");
        for(int i=0;i<ls.size();i++){
            System.out.print(ls.get(i) + " ");
        }
        ls.set(key, value);
        System.out.println();
        System.out.println("After:");
        for(int i=0;i<ls.size();i++){
            System.out.print(ls.get(i) + " ");
        }
    }
}

public class AdeptorDemo2 {
    public static void main(String[] args) {
        MathsProcessing mp = new BasicMathsProcessing();
        int[] arr = {5,4,3,2,1};

        mp.performMathOperation("Replace", arr, 2, 63);

    }
}

/*
 client interface -   mathprocessing
                      public void performMathOperation(String
typr,int[] data)

class basicMathsProcessing - performmathsoperation() -
>override

class Advance-
    public void calculateavg(list<int> ls)
                sortData(list<int> ls)
                searchdata(same)
```

Haard Shah
21BCP251
CS G-8

```
                replace(same,int key,int replacevalue)




*/
```

Output:



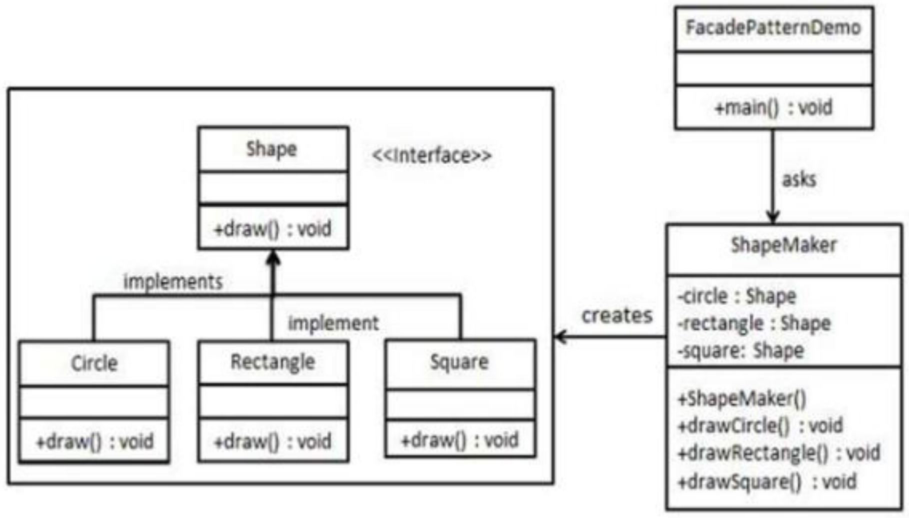
```
PS C:\Users\parma> cd "c:\Users\parma\Downloads\
Before:
5 4 3 2 1
After:
5 4 63 2 1
PS C:\Users\parma\Downloads>
```

Haard Shah

21BCP251

CS G-8

Facade Design Pattern



Code:

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

//implementing façade class
        orgMaker OrgMaker = new orgMaker();

        OrgMaker.showBuyer();
        OrgMaker.showSeller();
        OrgMaker.showWholesaller();
    }
}
```

```
public interface Org {
//creating interface
    void showName();
}
```

Haard Shah

21BCP251

CS G-8

```java
public class Buyer implements Org {
    @Override
    public void showName() {
        System.out.println("Buyer Called");
    }
}
public class seller implements Org{
    @Override
    public void showName() {
        System.out.println("Seller Called");
    }//creating 3 files which will be sharing same function
}
public class wholesaller implements Org{
    @Override
    public void showName() {
        System.out.println("Wholesaller Called");
    }
}
```

```java
public class orgMaker {
//creating façade class
    private Org buyerorg;
    private Org sellerorg;
    private Org wholeSallerorg;

    public orgMaker(){
        buyerorg = new Buyer();
        sellerorg = new seller();
        wholeSallerorg =new wholesaller();

    }

    public void showBuyer(){
        buyerorg.showName();

    }

    public void showSeller(){
        sellerorg.showName();

    }
    public void showWholesaller(){
        wholeSallerorg.showName();

    }
```

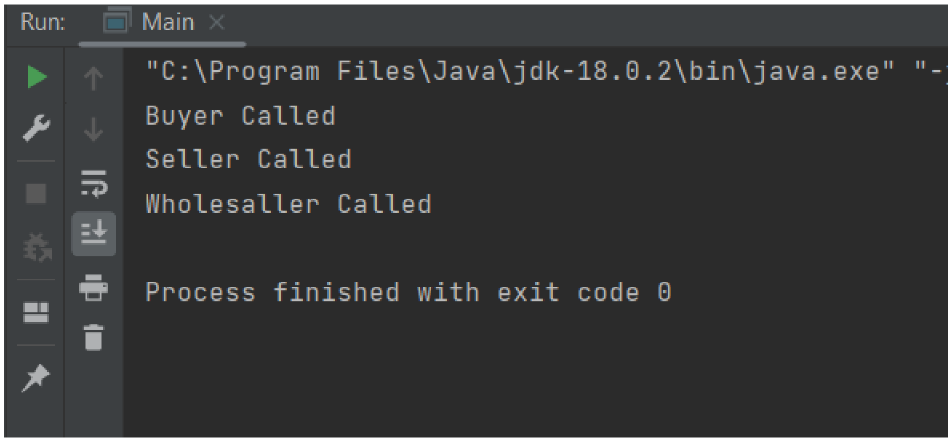
Haard Shah

21BCP251

CS G-8

```
}
```

Output:



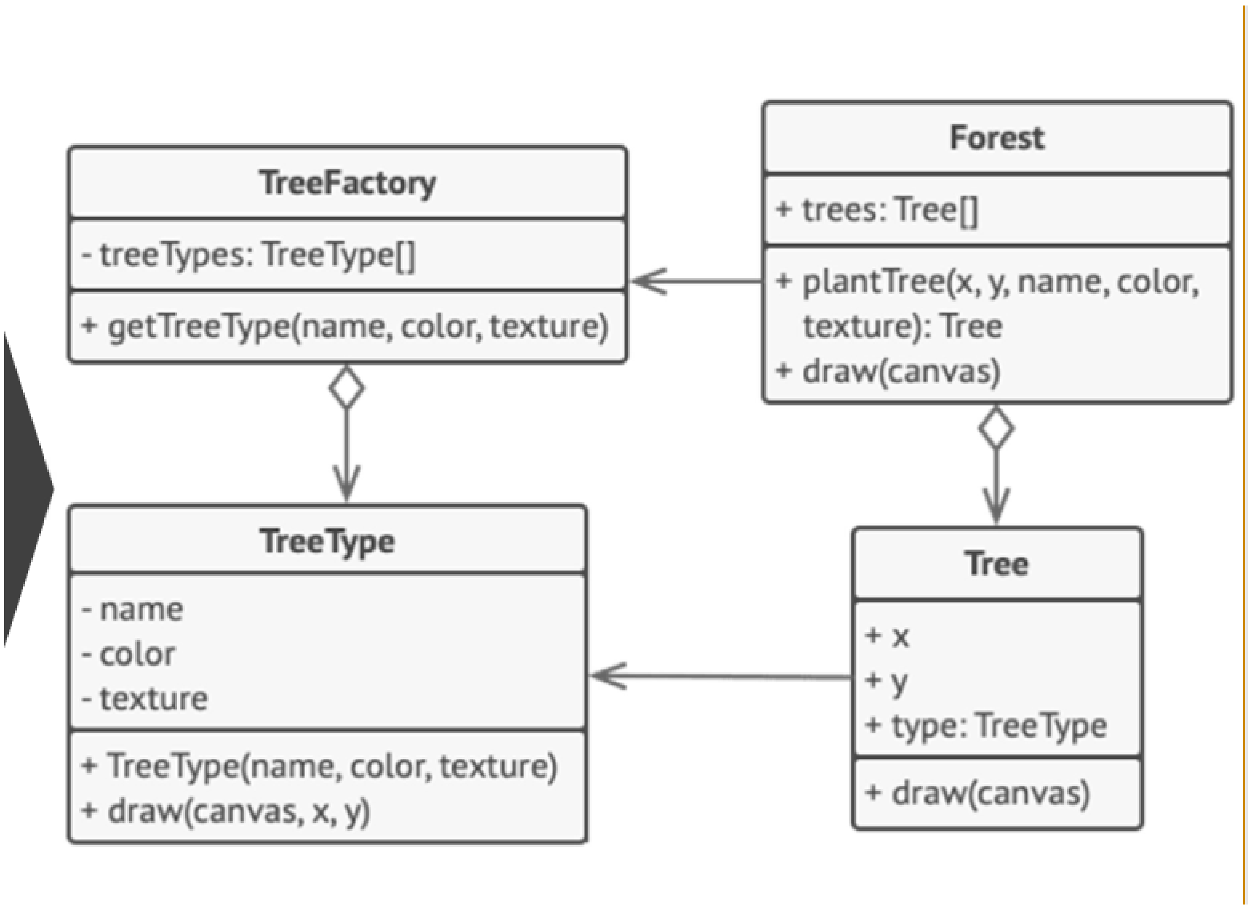
```
"C:\Program Files\Java\jdk-18.0.2\bin\java.exe" "-
Buyer Called
Seller Called
Wholesaller Called

Process finished with exit code 0
```

Haard Shah

21BCP251

CS G-8

Flyweight Design Pattern



Code:

```
import java.util.HashMap;

import java.util.Map;

import java.awt.*;

import javax.swing.*;

import java.util.ArrayList;

import java.util.List;
```

Haard Shah

21BCP251

CS G-8

```
 class Organisation {//creating class which is using type
	private int x;
	private int y;
	private orgType type;

	public Organisation(int x, int y, orgType type) {//constructor
		this.x = x;
		this.y = y;
		this.type = type;
	}

	public void create(Graphics g) {
		type.create(g, x, y);
	}
}

 class orgType {//creating type
	private String name;
	private Color color;
	private String otherOrganisationData;

	public orgType(String name, Color color, String otherOrganisationData) {
        this.name = name;
        this.color = color;
        this.otherOrganisationData = otherOrganisationData;
    }

    public void create(Graphics g, int x, int y) {
        g.setColor(Color.BLACK);
        g.fillRect(x - 1, y, 3, 5);
        g.setColor(color);
        g.fillOval(x - 5, y - 10, 10, 10);
    }
}

  class OrganisationFactory {
    static Map<String, orgType> orgTypes = new HashMap<>();

    public static orgType getorgType(String name, Color color, String
otherOrganisationData) {
        orgType result = orgTypes.get(name);
        if (result == null) {
            result = new orgType(name, color, otherOrganisationData);
            orgTypes.put(name, result);
        }
```

```
                return result;
        }
}

  class Market extends JFrame {//GUI
        private List<Organisation> Organisations = new ArrayList<>();

        public void createOrg(int x, int y, String name, Color color, String
otherOrganisationData) {
                orgType type = OrganisationFactory.getorgType(name, color,
otherOrganisationData);
                Organisation Organisation = new Organisation(x, y, type);
                Organisations.add(Organisation);
        }

        @Override
        public void paint(Graphics graphics) {
                for (Organisation Organisation : Organisations) {
                        Organisation.create(graphics);
                }
        }
}
```

Haard Shah

21BCP251

CS G-8

```
public class Main {
        static int CANVAS_SIZE = 500;
        static int Organisations_TO_CREATE = 1000000;
        static int TREE_TYPES = 2;

        public static void main(String[] args) {
                Market market = new Market();
                for (int i = 0; i < Math.floor(Organisations_TO_CREATE / TREE_TYPES);
i++) {
                        market.createOrg(random(0, CANVAS_SIZE), random(0,
CANVAS_SIZE),
                                        "Summer Oak", Color.GREEN, "Oak texture stub");
                        market.createOrg(random(0, CANVAS_SIZE), random(0,
CANVAS_SIZE),
                                        "Autumn Oak", Color.ORANGE, "Autumn Oak texture
stub");
                }
                market.setSize(CANVAS_SIZE, CANVAS_SIZE);
                market.setVisible(true);

                System.out.println(Organisations_TO_CREATE + " Organisations
Created");
                System.out.println("---------------------");
                System.out.println("Memory usage:");
```

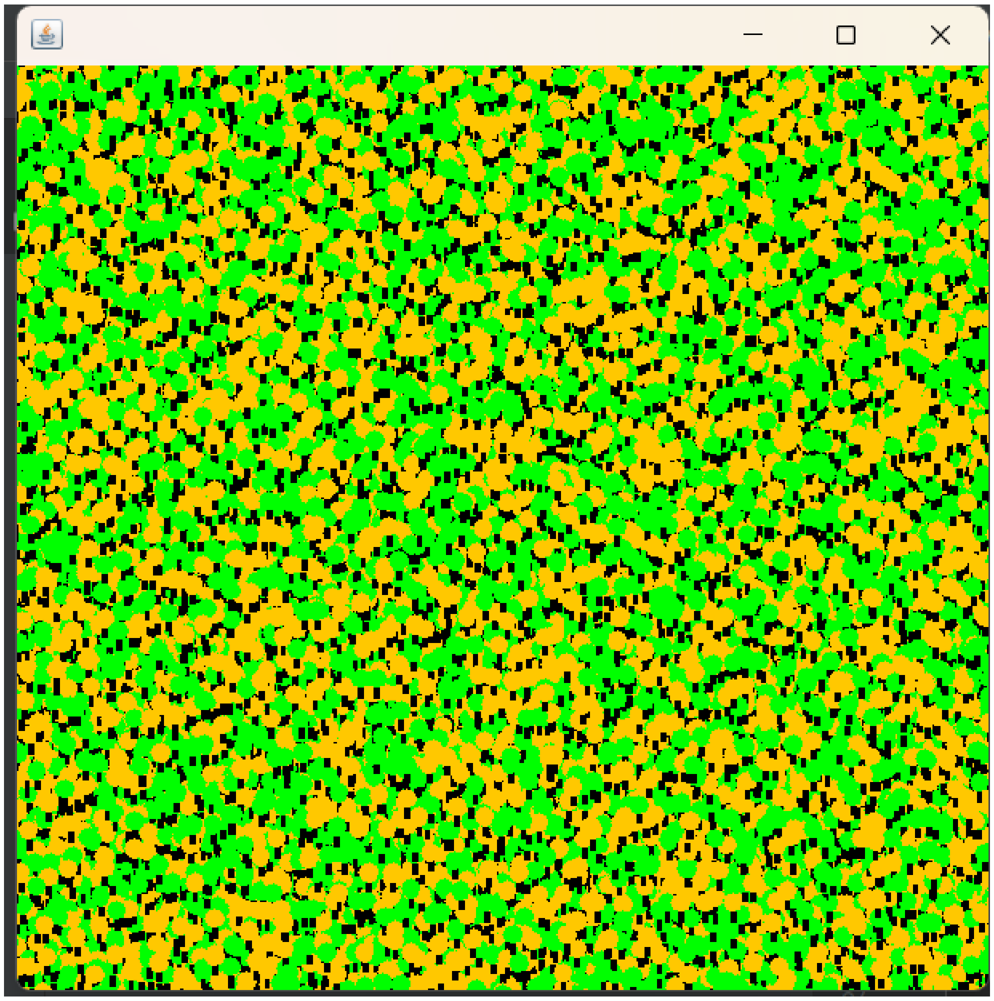
```
		System.out.println("Organisation size (8 bytes) * " +
Organisations_TO_CREATE);
		System.out.println("+ orgTypes size (~30 bytes) * " + TREE_TYPES + "");
		System.out.println("---------------------");
		System.out.println("Total: " + ((Organisations_TO_CREATE * 8 +
TREE_TYPES * 30) / 1024 / 1024) +
				"MB (instead of " + ((Organisations_TO_CREATE * 38) / 1024
/ 1024) + "MB)");
	}

	private static int random(int min, int max) {
		return min + (int) (Math.random() * ((max - min) + 1));
	}
}
```

Output:

Haard Shah

21BCP251

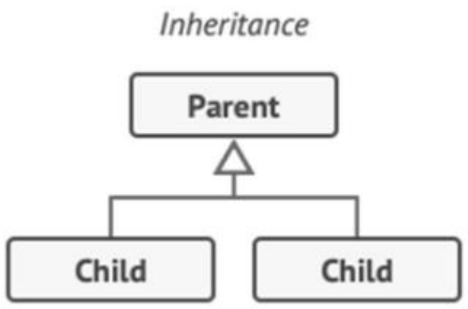
CS G-8

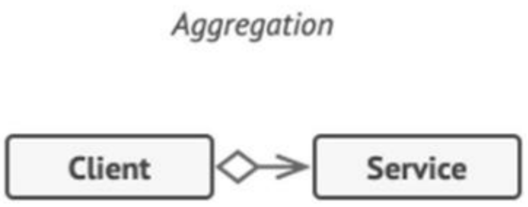


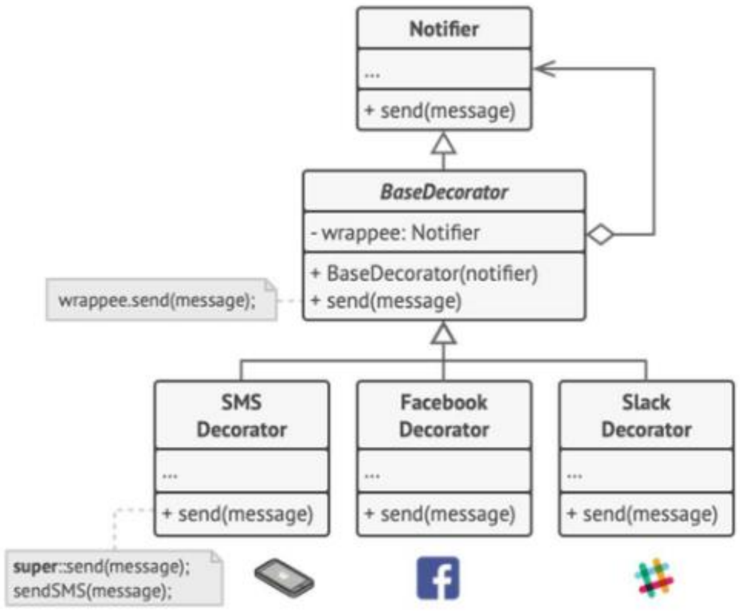
Haard Shah
21BCP251
CS G-8

Assignment 10: DECORATOR DESIGN PATTERN

Chart:







Haard Shah
21BCP251
CS G-8

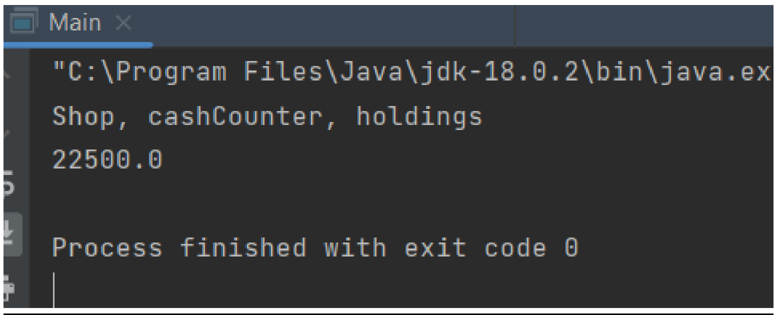
CODE:

```
// Component interface
 interface Shop {
    public String getDescription();
    public double getCost();
}

// Concrete Component
 class basicShop implements Shop {
    public String getDescription() {
        return "Shop";
    }

    public double getCost() {
        return 20000.0;
    }
}

// Decorator
 abstract class shopDecorator implements Shop {
    protected Shop shop;

    public shopDecorator(Shop shop) {
        this.shop = shop;
    }

    public String getDescription() {
        return shop.getDescription();
    }

    public double getCost() {
        return shop.getCost();
    }
}

// Concrete Decorator
 class cashCounter extends shopDecorator {
    public cashCounter(Shop shop) {
        super(shop);
    }

    public String getDescription() {
        return shop.getDescription() + ", cashCounter";
    }

    public double getCost() {
        return shop.getCost() + 1000.0;
    }
}

// Concrete Decorator
class holdings extends shopDecorator {
    public holdings(Shop shop) {
        super(shop);
    }

    public String getDescription() {
        return shop.getDescription() + ", holdings";
```

Haard Shah
21BCP251
CS G-8

```java
    }

    public double getCost() {
        return shop.getCost() + 1500.0;
    }
}

// Client code
public class Main {
    public static void main(String[] args) {
        // Create a basic shop
        Shop shop = new basicShop();

        // Decorate it with leather seats
        shop = new cashCounter(shop);

        // Decorate it with a navigation system
        shop = new holdings(shop);

        // Print the shop's description and cost
        System.out.println(shop.getDescription());
        System.out.println(shop.getCost());
    }
}
```

<u>OUTPUT:</u>

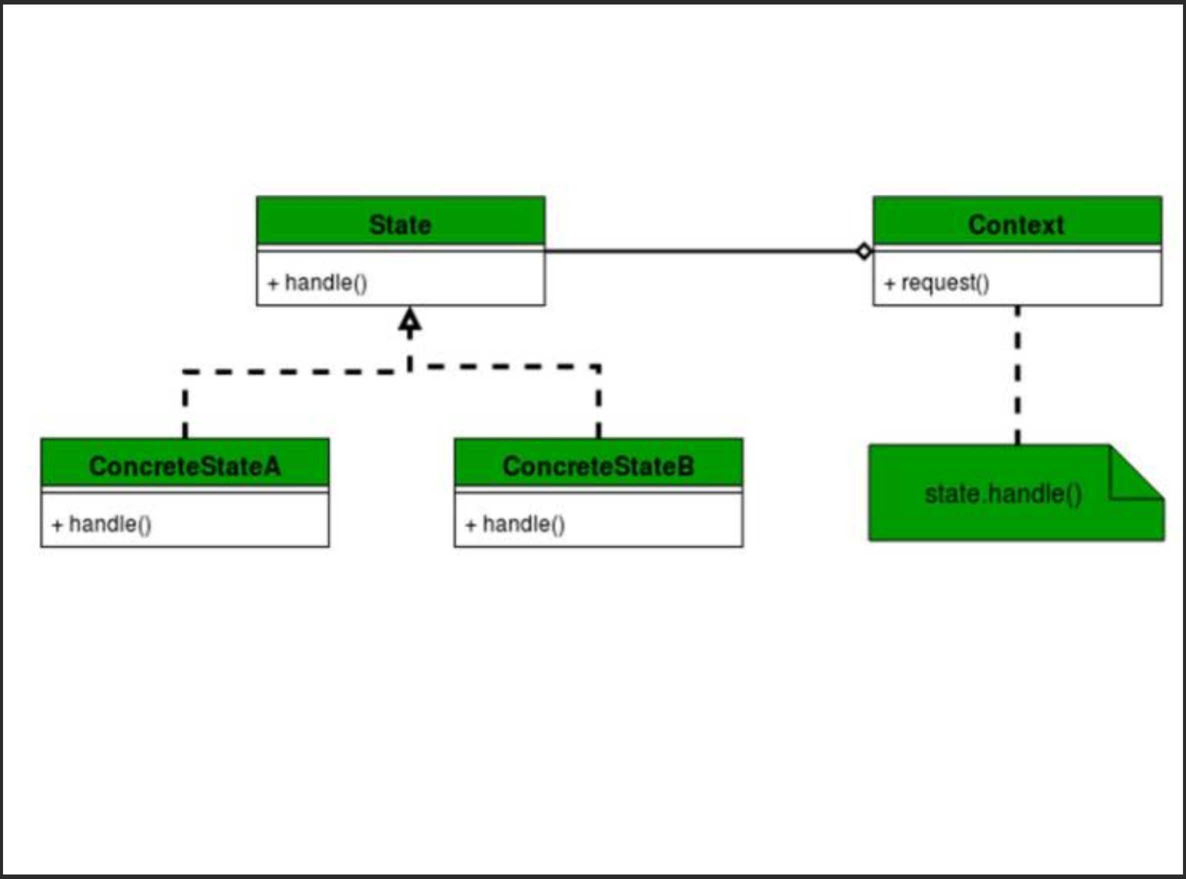


Main

```
"C:\Program Files\Java\jdk-18.0.2\bin\java.ex
Shop, cashCounter, holdings
22500.0

Process finished with exit code 0
```

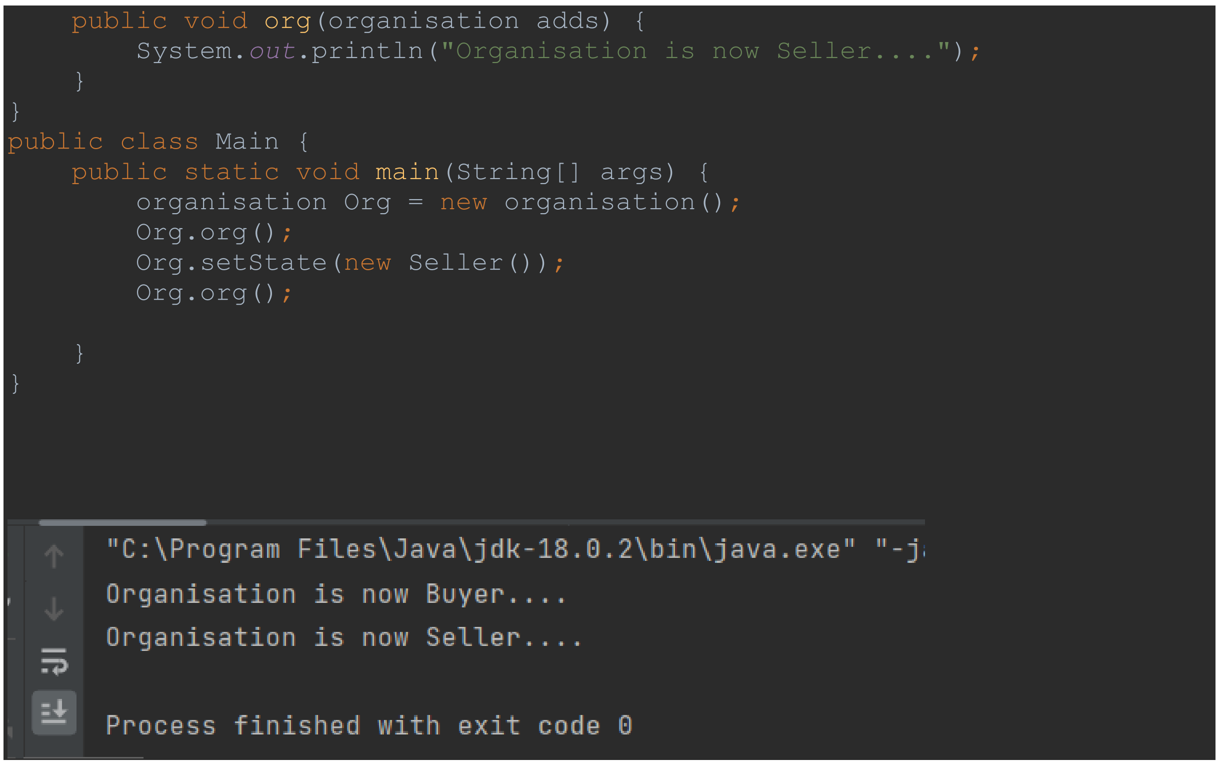
Haard Shah
21BCP251
CS G-8

State Design pattern



```
interface Organisation {
     void org(organisation a);
}
class organisation {
    private Organisation currentState;
    public organisation() {
        currentState = new Buyer();
    }
    public void setState(Organisation as) {
        currentState = as;
    }
    public void org() {
        currentState.org(this);
    }
}
class Buyer implements Organisation {
    @Override
    public void org(organisation adds) {
        System.out.println("Organisation is now Buyer....");
    }
}
class Seller implements Organisation {
    @Override
```

Haard Shah
21BCP251
CS G-8



```
    public void org(organisation adds) {
        System.out.println("Organisation is now Seller....");
    }
}
public class Main {
    public static void main(String[] args) {
        organisation Org = new organisation();
        Org.org();
        Org.setState(new Seller());
        Org.org();

    }
}
```

```
"C:\Program Files\Java\jdk-18.0.2\bin\java.exe" "-j
Organisation is now Buyer....
Organisation is now Seller....

Process finished with exit code 0
```

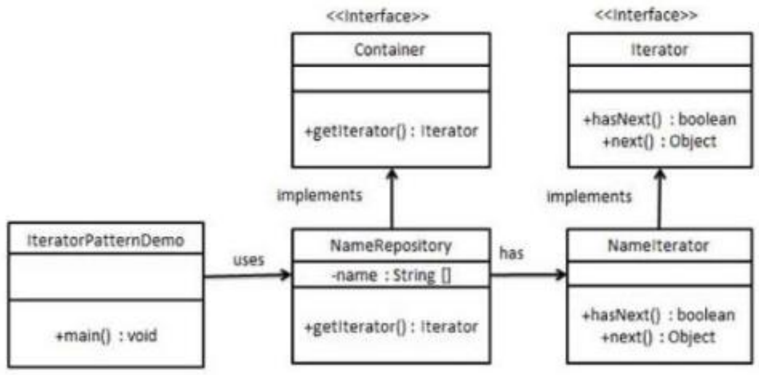
Haard Shah

21BCP251

CS G-8

Iterator Design Pattern

CLASS DIAGRAM



Code:

```
public interface Iterator {
    public boolean hasNext();
    public Object next();
}
```

```
public interface shop {
    public Iterator getIterator();
}
```

```
public class productRepository implements shop {
    public String products[] ={"product 1","product 2","product 3","product 4"};
    @Override
    public Iterator getIterator(){
```

Haard Shah

21BCP251

CS G-8

```
        return new ProductIterator();
    }
//creating iterator
    private class ProductIterator implements Iterator{
        int index;
        @Override
        public boolean hasNext(){
           if(index<products.length){
              return true;
           }
           return false;
        }

        @Override
        public Object next(){
            if(this.hasNext()){
                return products[index++];
            }
            return null;
        }
    }
}
```

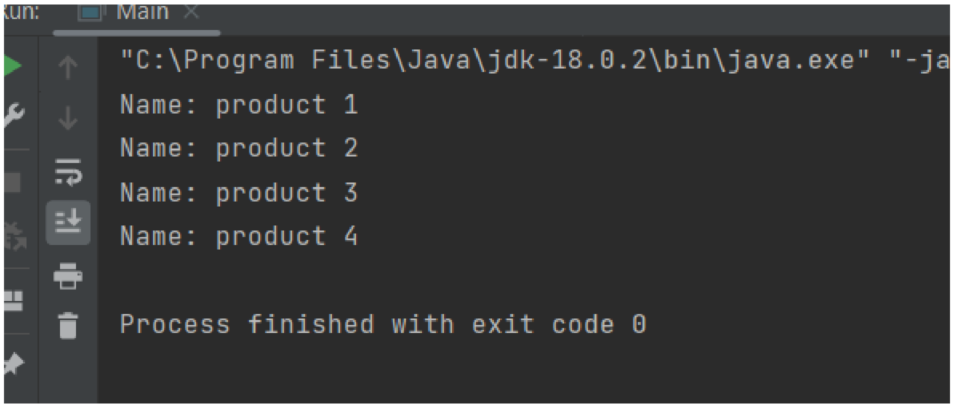
```
public class Main {
    public static void main(String[] args) {
//accessing the product repository as object
        productRepository productsRepository=new productRepository();
        for(Iterator itr=productsRepository.getIterator(); itr.hasNext();){
            String name= (String) itr.next();//saving current name as variable
            System.out.println("Name: "+name);
        }
    }
}
```

Output:

Haard Shah

21BCP251

CS G-8



```
"C:\Program Files\Java\jdk-18.0.2\bin\java.exe" "-ja
Name: product 1
Name: product 2
Name: product 3
Name: product 4

Process finished with exit code 0
```

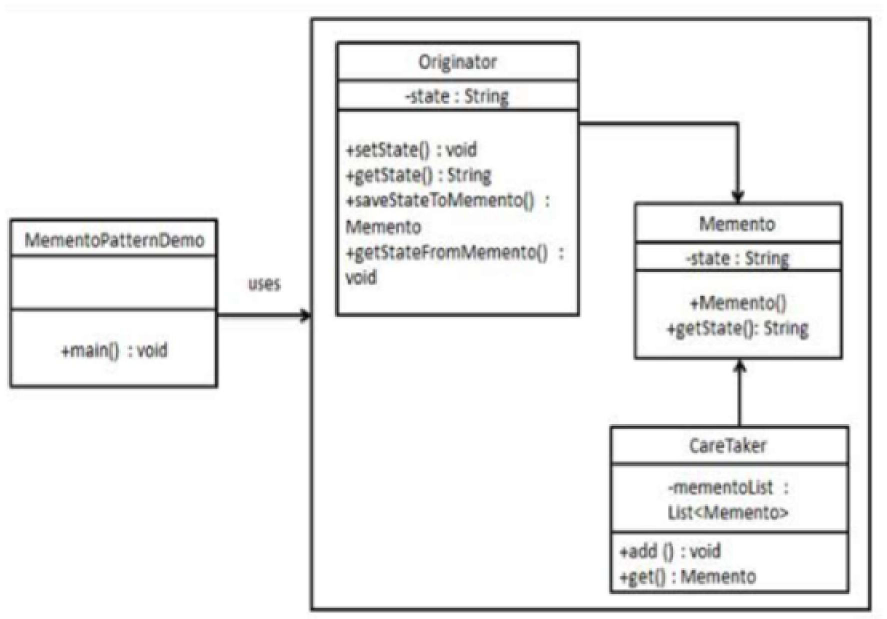
Haard Shah

21BCP251

CS G-8

Memento Design Pattern

Class Diagram:



Code:

```java
public class product {//this will be shared to both Shop and Factory
    private String state;
    public product(String state){
        this.state=state;
    }

    public String getState() {
        return state;
    }
}
```

Haard Shah

21BCP251

CS G-8

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

public class shop {//This will work as caretaker
    private List<product> productList =new ArrayList<product>();

    public void add(product state){
        productList.add(state);//adding state in Arraylist to save the data
    }
    public product get(int index){
        return productList.get(index);
    }
}
```

```
public class factory {//this will work as originator
    private String state;
    public void setState(String state){
        this.state=state;

    }

    public String getState() {
        return state;
    }
    public product SaveStateToProduct(){
        return new product(state);//saving the state

    }
    public void getStateFromProduct(product pr){
        state=pr.getState();
    }
}
```

```
public class Main {
    public static void main(String[] args) {
//initializing the object of class
        factory Factory = new factory();
        shop Shop =new shop();
        Factory.setState("State 1");
        Factory.setState("State 2");//without saving state is being changed
        Shop.add(Factory.SaveStateToProduct());//adding to ARRAYLIST
        Factory.setState("State 3");
        Shop.add(Factory.SaveStateToProduct());
        Factory.setState("State 4");
        System.out.println("Current state:"+Factory.getState());
```

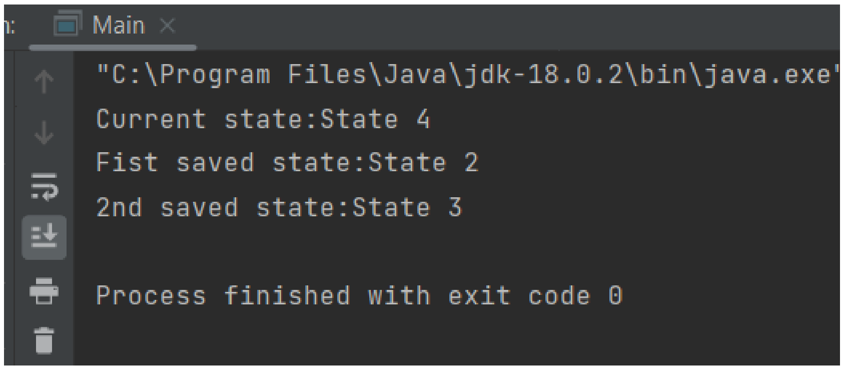
Haard Shah

21BCP251

CS G-8

```
        Factory.getStateFromProduct(Shop.get(0));
        System.out.println("Fist saved state:"+Factory.getState());
        Factory.getStateFromProduct(Shop.get(1));
        System.out.println("2nd saved state:"+Factory.getState());



    }
}
```

Output:



```
"C:\Program Files\Java\jdk-18.0.2\bin\java.exe"
Current state:State 4
Fist saved state:State 2
2nd saved state:State 3

Process finished with exit code 0
```

Haard Shah

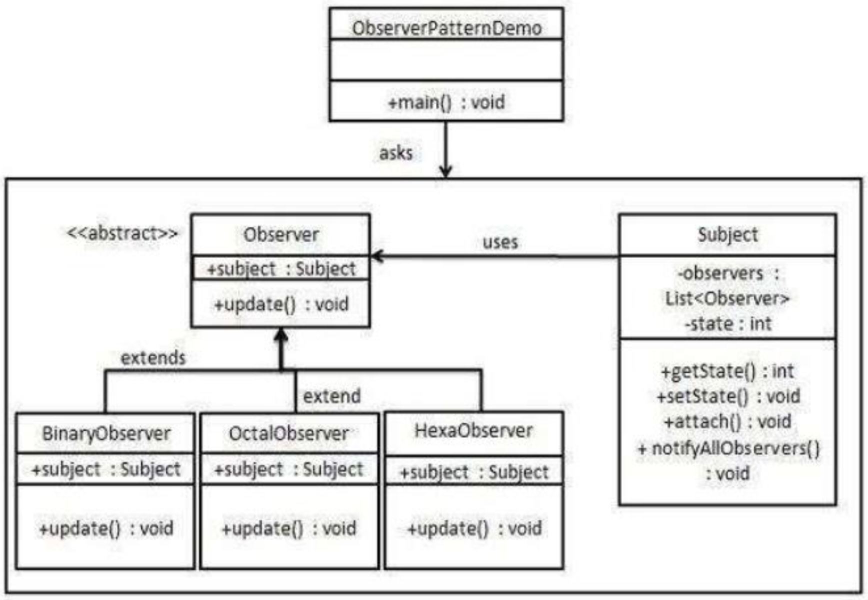
21BCP251

CS G-8

Observer Design Pattern

Class Diagram:

Cpd



Code:

```
package com.observer;
public class Buyer {// work as Subscriber
    private String name;
    private Product product= new Product();
    public Buyer(String name) {
        this.name= name;
    }
    public void update() {
        System.out.println("Hey "+ name+ " Product arrived");
    }
    public void BecomeBuyer(Product pr) {
        product= pr; }
}
```

```
package com.observer;

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

Haard Shah

21BCP251

CS G-8

```
public class Product {//work as channel
    List<Buyer> buyers = new ArrayList<>();//will work as list of subscribers
    private String title;
    public void Purchase(Buyer buyer) {
        buyers.add(buyer);
    }
    public void Return(Buyer buyer) {
        buyers.remove(buyer);
    }
    public void notifyBuyer() {
        for(Buyer buyer:buyers){
            buyer.update();//Send message to buyers about product arrival
        }
    }
    public void upload(String title){
        this.title = title;
        notifyBuyer();
    }
}
```

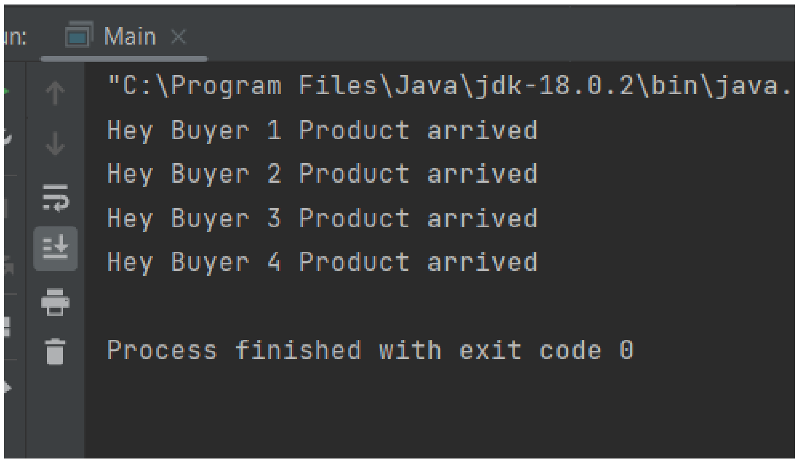
```
package com.observer;

public class Main {
    public static void main(String[] args) {

        Product product1= new Product();
        Buyer b1= new Buyer("Buyer 1");
        Buyer b2= new Buyer("Buyer 2");
        Buyer b3= new Buyer("Buyer 3");
        Buyer b4= new Buyer("Buyer 4");
        product1.Purchase(b1);
        product1.Purchase(b2);
        product1.Purchase(b3);
        product1.Purchase(b4);
        b1.BecomeBuyer(product1);
        b2.BecomeBuyer(product1);
        b3.BecomeBuyer(product1);
        b4.BecomeBuyer(product1);
        product1.upload("PR 1");//this is title of product
    }
}
```

Output:

Haard Shah

21BCP251

CS G-8



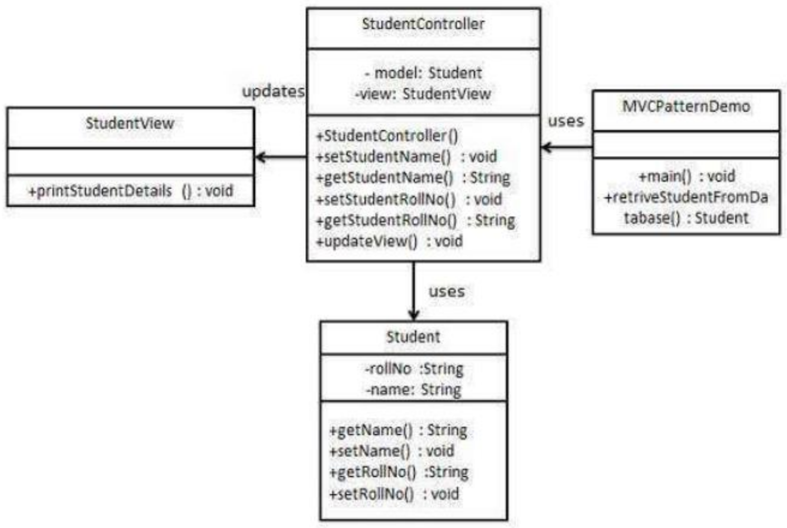
```
"C:\Program Files\Java\jdk-18.0.2\bin\java.
Hey Buyer 1 Product arrived
Hey Buyer 2 Product arrived
Hey Buyer 3 Product arrived
Hey Buyer 4 Product arrived

Process finished with exit code 0
```

Haard Shah

21BCP251

CS G-8

MVC Design Pattern



Code:

```
public class Org {
    private String Name;
    private String Address;
//creating model with getter setter methods
    public String getName() {
        return Name;
    }

    public String getAddress() {
        return Address;
    }

    public void setAddress(String address) {
        Address = address;
    }

    public void setName(String name) {
        Name = name;
    }
}
```

Haard Shah

21BCP251

CS G-8

```java
public class orgView {
    public void printOrgDetails(String orgName, String orgAddress){
        System.out.println("Organisation: ");
        System.out.println("Name: " + orgName);
        System.out.println("Address: " + orgAddress);
    }
}
```

```java
public class orgController {
    private Org model;
    private orgView view;
    public orgController(Org model, orgView view){
        this.model= model;
        this.view= view;
    }//creating getter setter for controller
    public void setOrgName(String name){
        model.setName(name);
    }
    public String getOrgName(){
        return model.getName();
    }
    public void setOrgAddress(String Address){
        model.setAddress(Address);
    }
    public String getOrgAddress(){
        return model.getAddress();
    }
    public void updateView(){
        view.printOrgDetails(model.getName(), model.getAddress());
    }
}
```

```java
public class Main {
    public static void main(String[] args)  {
//access new Org which can be later connected to database too
        Org model = retriveOrgFromDatabase();

        orgView view = new orgView();
//creating objects
        orgController controller = new orgController(model,view);

        controller.updateView();
//changing name only , address remains the same
        controller.setOrgName("Org 1");
        controller.updateView();
    }
    private static Org retriveOrgFromDatabase(){//initializing object from model
        Org org= new Org();
        org.setName("NEW ORG");
        org.setAddress("CITY 1");
```

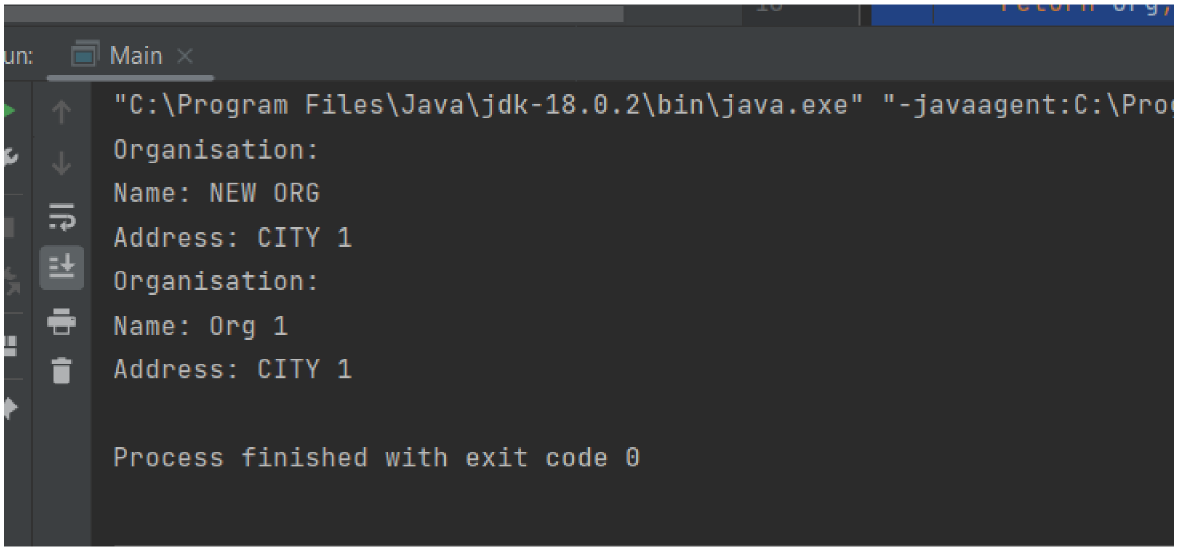
Haard Shah

21BCP251

CS G-8

```
            return org;
        }
}
```

Output:



```
"C:\Program Files\Java\jdk-18.0.2\bin\java.exe" "-javaagent:C:\Pro
Organisation:
Name: NEW ORG
Address: CITY 1
Organisation:
Name: Org 1
Address: CITY 1

Process finished with exit code 0
```