**FAÇADE FACTORY**

ItemPurchased.java

package tryFacade;

import java.util.\*;

public class ItemPurchased {

  HashMap<String, Integer> hm;

  int type;

  public ItemPurchased() {

    hm = new HashMap<String, Integer>();

    storePurchase();

  }

  private void storePurchase() {

    Scanner sc = new Scanner(System.in);

    System.out.print("Enter number of types of Items: ");

    type = Integer.parseInt(sc.nextLine());

    for (int i = 1; i <= type; i++) {

      System.out.print("Enter name of Item " + i + ": ");

      String itemName = sc.nextLine();

      System.out.print("Enter Quantity of Item " + i + ": ");

      int quantity = Integer.parseInt(sc.nextLine());

      hm.put(itemName, quantity);

    }

  }

  // It will return the number of types of items user is buying

  int getTypeCount() {

    return type;

  }

  void showItems() {

    System.out.println("Items Purchased:-" + hm);

  }

}

Sport2000ProcessSales.java

package tryFacade;

import java.util.Scanner;

public class Sport2000ProcessSales {

  boolean giftCertificateFlag;

  float amount;

  int quantity;

  public Sport2000ProcessSales(boolean giftCertificateFlag, float amount, int quantity) {

    this.giftCertificateFlag = giftCertificateFlag;

    this.amount = amount;

    this.quantity = quantity;

  }

  void processSales() {

    if (!giftCertificateFlag) {

      System.out.println("You don't have a Gift Certificate!");

      System.out.println("Amount Payable: " + amount);

    } else {

      Scanner sc = new Scanner(System.in);

      System.out.print("Enter Gift Certificate Value: ");

      float certValue = Float.parseFloat(sc.nextLine());

      if (quantity <= 0)

        System.out.println("No Items in Cart!");

      else if (quantity > 1)

        System.out.println("Only one Item can be purchased using Gift Certificate");

      else if (amount > certValue)

        System.out.println("Please pay balance amount sc Cash: Rs." + (amount - certValue));

      else if (amount <= certValue)

        System.out.println("No cashback will be Refunded! Thankyou for your Purchase!");

      sc.close();

    }

  }

}

Sports2000Facade.java

package tryFacade;

public class Sports2000Facade {

  ItemPurchased itemPuchasedObject;

  Sport2000ProcessSales sprorts2000ProcessSalesObject;

  public Sports2000Facade(boolean flag, float amount) {

    itemPuchasedObject = new ItemPurchased();

    sprorts2000ProcessSalesObject = new Sport2000ProcessSales(flag, amount, itemPuchasedObject.getTypeCount());

  }

  void displayItems() {

    itemPuchasedObject.showItems();

  }

  void dispAmount() {

    sprorts2000ProcessSalesObject.processSales();

  }

}

FacadeDemo.java

package tryFacade;

public class FacadeDemo {

  public static void main(String[] args) {

    Sports2000Facade facadeWrapperObject = new Sports2000Facade(true, 100);

    facadeWrapperObject.displayItems();

    facadeWrapperObject.dispAmount();

  }

}

**Output :-**

Enter number of types of Items: 1

Enter name of Item 1: dettol

Enter Quantity of Item 1: 1

Items Purchased:- {dettol=1}

Enter Gift Certificate Value: 50

Please pay balance amount by Cash/Credit Card/Debit Card: Rs.50.0

**ADAPTER PATTERN**

CalcTax.java

package tryAdapter;

public interface CalcTax {

  float taxAmount(int qty, float price);

}

GST.java

package tryAdapter;

public class GST implements CalcTax {

  public float taxAmount(int qty, float price) {

    return qty \* price \* 0.18f;

  }

}

MauriTax.java

package tryAdapter;

public class MauriTax {

  float mauriTaxAmount(int qty, float price) {

    return qty \* price \* 0.1f;

  }

}

MauriTaxAdapter.java

package tryAdapter;

public class MauriTaxAdapter implements CalcTax {

  MauriTax mt = new MauriTax();

  public float taxAmount(int qty, float price) {

    return mt.mauriTaxAmount(qty, price);

  }

}

Item.java

package tryAdapter;

public class Item {

  String name;

  int qty;

  float price;

  CalcTax ct;

  public Item(String name, int qty, float price, CalcTax ct) {

    this.name = name;

    this.price = price;

    this.qty = qty;

    this.ct = ct;

  }

  void setTax(CalcTax ct) {

    this.ct = ct;

  }

  void displayItem() {

    System.out.println("\nName: " + name);

    System.out.println("Quantity: " + qty);

    System.out.println("Price: " + price);

    float tax = ct.taxAmount(qty, price);

    float billAmount = (qty \* price) + tax;

    System.out.println("Tax Amount: " + tax);

    System.out.println("Bill Amount: " + billAmount);

  }

}

AdapterDemo.java

package tryAdapter;

public class AdapterDemo {

  public static void main(String[] args) {

    CalcTax ct = new MauriTaxAdapter();

    Item i1 = new Item("cycle", 2, 100, ct);

    i1.displayItem();

    i1.setTax(new GST());

    i1.displayItem();

  }

}

**Output :-**

Name: cycle Quantity: 2 Price: 100.0 Tax Amount: 20.0 Bill Amount: 220.0

Name: cycle Quantity: 2 Price: 100.0 Tax Amount: 36.0 Bill Amount: 236.0

**STRATEGY PATTERN**

Customer.java

package tryStrategy;

public abstract class Customer {

  String id, name, typeOfCustomer;

  Discount d;

  public Customer(String id, String name) {

    this.id = id;

    this.name = name;

  }

  void printBill(float amt) {

    System.out.println("\nID: " + id);

    System.out.println("Name: " + name);

    System.out.println("Type of Customer: " + typeOfCustomer);

    System.out.println("Gross Amount: " + amt);

    System.out.println("Discount: " + d.calcDiscount(amt));

    System.out.println("Amount Payable: " + (amt - d.calcDiscount(amt)));

  }

}

FTCCustomer.java

package tryStrategy;

public class FTCCustomer extends Customer {

  public FTCCustomer(String id, String name) {

    super(id, name);

    d = new FTCDiscount();

    typeOfCustomer = "First Time Customer";

  }

}

RCCustomer.java

package tryStrategy;

public class RCCustomer extends Customer {

  public RCCustomer(String id, String name) {

    super(id, name);

    d = new RCDiscount();

    typeOfCustomer = "Regular Customer";

  }

}

SCCustomer.java

package tryStrategy;

public class SCCustomer extends Customer {

  public SCCustomer(String id, String name) {

    super(id, name);

    d = new SCDiscount();

    typeOfCustomer = "Senior Customer";

  }

}

Discount.java

package tryStrategy;

public interface Discount {

  float calcDiscount(float amount);

}

FTCDiscount.java

package tryStrategy;

public class FTCDiscount implements Discount {

  public float calcDiscount(float amount) {

    return 0.15f \* amount;

  }

}

RCDiscount.java

package tryStrategy;

public class RCDiscount implements Discount {

  public float calcDiscount(float amount) {

    return 0.12f \* amount;

  }

}

SCDiscount.java

package tryStrategy;

public class SCDiscount implements Discount {

  public float calcDiscount(float amount) {

    return 0.1f \* amount;

  }

}

StrategyDemo.java

package tryStrategy;

public class StrategyDemo {

  public static void main(String[] args) {

    Customer c1 = new RCCustomer("rc1", "modi");

    c1.printBill(100);

    c1 = new SCCustomer("sc1", "trump");

    c1.printBill(100);

    c1 = new FTCCustomer("ftc1", "raga");

    c1.printBill(100);

  }

}

**Output :-**

ID: rc1 Name: modi Type of Customer: Regular Customer Gross Amount: 100.0 Discount: 12.0 Amount Payable: 88.0

ID: sc1 Name: trump Type of Customer: Senior Customer Gross Amount: 100.0 Discount: 10.0 Amount Payable: 90.0

ID: ftc1 Name: raga Type of Customer: First Time Customer Gross Amount: 100.0 Discount: 15.000001 Amount Payable: 85.0

**BRIDGE PATTERN**

Customer.java

package tryBridge;

public abstract class Customer {

  String name;

  int age;

  Discount d;

  String typeOfCust;

  public Customer(String name, int age, Discount d) {

    this.name = name;

    this.age = age;

    this.d = d;

  }

  void setDiscount(Discount d) {

    this.d = d;

  }

  void showBill(float amt) {

    System.out.println("\nName: " + name);

    System.out.println("Age: " + age);

    System.out.println("Type of Customer: " + typeOfCust);

    System.out.println("Gross Cost: " + amt);

    System.out.println("Discount: " + d.getDiscount(amt));

    System.out.println("Payable Amount: " + (amt - d.getDiscount(amt)));

  }

}

FTCCustomer.java

package tryBridge;

public class FTCCustomer extends Customer {

  public FTCCustomer(String name, int age, Discount d) {

    super(name, age, d);

    typeOfCust = "First Time Customer";

  }

}

RCCustomer.java

package tryBridge;

public class RCCustomer extends Customer {

  public RCCustomer(String name, int age, Discount d) {

    super(name, age, d);

    typeOfCust = "Regular Customer";

  }

}

SSCustomer.java

package tryBridge;

public class SSCustomer extends Customer {

  public SSCustomer(String name, int age, Discount d) {

    super(name, age, d);

    typeOfCust = "Senior Customer";

  }

}

Discount.java

package tryBridge;

public interface Discount {

  float getDiscount(float amount);

}

Discount1.java

package tryBridge;

public class Discount1 implements Discount {

  public float getDiscount(float amount) {

    return 0.3f \* amount;

  }

}

Discount2.java

package tryBridge;

public class Discount2 implements Discount {

  public float getDiscount(float amount) {

    return 0.25f \* amount;

  }

}

Discount3.java

package tryBridge;

public class Discount3 implements Discount {

  public float getDiscount(float amount) {

    return 0.2f \* amount;

  }

}

Discount4.java

package tryBridge;

public class Discount4 implements Discount {

  public float getDiscount(float amount) {

    return 0.15f \* amount;

  }

}

BridgeDemo.java

package tryBridge;

public class BridgeDemo {

  public static void main(String[] args) {

    Customer c1 = new RCCustomer("Modi", 61, new Discount1());

    c1.showBill(100);

    c1.setDiscount(new Discount2());

    c1.showBill(100);

    c1.setDiscount(new Discount3());

    c1.showBill(100);

    c1.setDiscount(new Discount4());

    c1.showBill(100);

    Customer c2 = new FTCCustomer("Raga", 6, new Discount1());

    c2.showBill(100);

    c2.setDiscount(new Discount2());

    c2.showBill(100);

    c2.setDiscount(new Discount3());

    c2.showBill(100);

    c2.setDiscount(new Discount4());

    c2.showBill(100);

    Customer c3 = new SSCustomer("Anjaneya", 20, new Discount1());

    c3.showBill(100);

    c3.setDiscount(new Discount2());

    c3.showBill(100);

    c3.setDiscount(new Discount3());

    c3.showBill(100);

    c3.setDiscount(new Discount4());

    c3.showBill(100);

  }

}

**Output :-**

Name: Modi Age: 61 Type of Customer: Regular Customer Gross Cost: 100.0 Discount: 30.000002 Payable Amount: 70.0

Name: Modi Age: 61 Type of Customer: Regular Customer Gross Cost: 100.0 Discount: 25.0 Payable Amount: 75.0

Name: Modi Age: 61 Type of Customer: Regular Customer Gross Cost: 100.0 Discount: 20.0 Payable Amount: 80.0

Name: Modi Age: 61 Type of Customer: Regular Customer Gross Cost: 100.0 Discount: 15.000001 Payable Amount: 85.0

Name: Raga Age: 6 Type of Customer: First Time Customer Gross Cost: 100.0 Discount: 30.000002 Payable Amount: 70.0

Name: Raga Age: 6 Type of Customer: First Time Customer Gross Cost: 100.0 Discount: 25.0 Payable Amount: 75.0

Name: Raga Age: 6 Type of Customer: First Time Customer Gross Cost: 100.0 Discount: 20.0 Payable Amount: 80.0

Name: Raga Age: 6 Type of Customer: First Time Customer Gross Cost: 100.0 Discount: 15.000001 Payable Amount: 85.0

Name: Anjaneya Age: 20 Type of Customer: Senior Customer Gross Cost: 100.0 Discount: 30.000002 Payable Amount: 70.0

Name: Anjaneya Age: 20 Type of Customer: Senior Customer Gross Cost: 100.0 Discount: 25.0 Payable Amount: 75.0

Name: Anjaneya Age: 20 Type of Customer: Senior Customer Gross Cost: 100.0 Discount: 20.0 Payable Amount: 80.0

Name: Anjaneya Age: 20 Type of Customer: Senior Customer Gross Cost: 100.0 Discount: 15.000001 Payable Amount: 85.0

**OBSERVER PATTERN**

Observer.java

package tryObserver;

abstract public class Observer {

  abstract void update(float discount);

}

Customer.java

package tryObserver;

public class Customer extends Observer {

  // Name of the subscriber

  String name;

  // Contains the channel to which the user has subscribed

  Subject store;

  float discount;

  public Customer(Subject store, String name) {

    this.name = name;

    this.store = store;

    this.store.register(this);

  }

  void update(float discount) {

    this.discount = discount;

    System.out.println(name + ",you get a discount of " + this.discount + "%");

  }

  public String toString() {

    return name;

  }

}

Subject.java

package tryObserver;

public abstract class Subject {

  abstract void register(Observer o);

  abstract void unregister(Observer o);

  abstract void notifyObservers();

}

Store.java

package tryObserver;

import java.util.ArrayList;

public class Store extends Subject {

  float discount;

  String name;

  ArrayList<Observer> observerList = new ArrayList<Observer>();

  public Store(String name, float discount) {

    this.name = name;

    this.discount = discount;

  }

  void register(Observer newObserverList) {

    observerList.add(newObserverList);

    System.out.println("Added Customer " + newObserverList + " to Store " + name);

  }

  void unregister(Observer removeObserver) {

    try {

      // indexOf(Object) method returns the index of the first occurrence of the

      // specified element in this list, or -1 if this list does not contain the

      // element.

      observerList.remove(observerList.indexOf(removeObserver));

      System.out.println("Removed Customer " + removeObserver + " from store " + name);

    } catch (NullPointerException e) {

      System.out.println("No such Customer called " + removeObserver + " in store " + name);

    }

  }

  void notifyObservers() {

    for (Observer itr : observerList)

      itr.update(discount);

  }

  void setDiscount(String festival, float d) {

    discount = d;

    System.out.println("New Discount Offer on Account of " + festival);

    notifyObservers();

  }

}

MainClass.java

package tryObserver;

public class MainClass {

  public static void main(String[] args) {

    Store s1 = new Store("Store1", 10);

    Customer c1 = new Customer(s1, "Modi");

    Customer c2 = new Customer(s1, "Trump");

    s1.setDiscount("Holi", 5);

    s1.unregister(c2);

    s1.setDiscount("Diwali", 20);

    Customer c3 = new Customer(s1, "Raga");

    s1.setDiscount("Ugadi", 15);

  }

}

**Output :-**

Added Customer Modi to Store Store1

Added Customer Trump to Store Store1

New Discount Offer on Account of Holi

Modi,you get a discount of 5.0%

Trump,you get a discount of 5.0%

Removed Customer Trump from store Store1

New Discount Offer on Account of Diwali

Modi,you get a discount of 20.0%

Added Customer Raga to Store Store1

New Discount Offer on Account of Ugadi

Modi,you get a discount of 15.0%

Raga,you get a discount of 15.0%

**ABSTRACT FACTORY PATTERN**

OutdoorAdventureSports.java

package tryAbstractFactory;

public abstract class OutdoorAdventureSports {

  abstract void getSportName();

}

BungeeJumpingDiffAbled.java

package tryAbstractFactory;

public class BungeeJumpingDiffAbled extends OutdoorAdventureSports {

  void getSportName() {

    System.out.println("Differently Abled Bungee Jumping");

  }

}

BungeeJumpingRegular.java

package tryAbstractFactory;

public class BungeeJumpingRegular extends OutdoorAdventureSports {

  void getSportName() {

    System.out.println("Regular Bungee-Jumping");

  }

}

ParaglidingDiffAbled.java

package tryAbstractFactory;

public class ParaglidingDiffAbled extends OutdoorAdventureSports {

  void getSportName() {

    System.out.println("Differently Abled Paragliding");

  }

}

ParaglidingRegular.java

package tryAbstractFactory;

public class ParaglidingRegular extends OutdoorAdventureSports {

  void getSportName() {

    System.out.println("Regular Paragliding");

  }

}

TrekkingDiffAbled.java

package tryAbstractFactory;

public class TrekkingDiffAbled extends OutdoorAdventureSports {

  void getSportName() {

    System.out.println("Differently Abled Trekking");

  }

}

TrekkingRegular.java

package tryAbstractFactory;

public class TrekkingRegular extends OutdoorAdventureSports {

  void getSportName() {

    System.out.println("Regular Trekking");

  }

}

OutdoorRegularGames.java

package tryAbstractFactory;

public abstract class OutdoorRegularGames {

  abstract void getSportName();

}

CricketDiffAbled.java

package tryAbstractFactory;

public class CricketDiffAbled extends OutdoorRegularGames {

  void getSportName() {

    System.out.println("Differently Abled Cricket");

  }

}

CricketRegular.java

package tryAbstractFactory;

public class CricketRegular extends OutdoorRegularGames {

  void getSportName() {

    System.out.println("Regular Cricket");

  }

}

IndoorRegularGames.java

package tryAbstractFactory;

public abstract class IndoorRegularGames {

  abstract void getSportName();

}

TableTennisDiffAbled.java

package tryAbstractFactory;

public class TableTennisDiffAbled extends IndoorRegularGames {

  void getSportName() {

    System.out.println("Differently Abled Table Tennis");

  }

}

TableTennisRegular.java

package tryAbstractFactory;

public class TableTennisRegular extends IndoorRegularGames {

  void getSportName() {

    System.out.println("Regular Table Tennis");

  }

}

SportsCategoryFactory.java

package tryAbstractFactory;

public interface SportsCategoryFactory {

  OutdoorAdventureSports getOutdoorAdventureSports(String name);

  OutdoorRegularGames getOutdoorRegularGames();

  IndoorRegularGames getIndoorRegularGames();

}

DiffAbledSportsFactory.java

package tryAbstractFactory;

public class DiffAbledSportsFactory implements SportsCategoryFactory {

  public OutdoorAdventureSports getOutdoorAdventureSports(String name) {

    if(name.equalsIgnoreCase("Bungee Jumping"))

      return new BungeeJumpingDiffAbled();

    else if(name.equalsIgnoreCase("Paragliding"))

      return new ParaglidingDiffAbled();

    else if(name.equalsIgnoreCase("Trekking"))

      return new TrekkingDiffAbled();

    else

      return null;

  }

  public OutdoorRegularGames getOutdoorRegularGames() {

    return new CricketDiffAbled();

  }

  public IndoorRegularGames getIndoorRegularGames() {

    return new TableTennisDiffAbled();

  }

}

RegularSportsFactory.java

package tryAbstractFactory;

public class RegularSportsFactory implements SportsCategoryFactory {

  public OutdoorAdventureSports getOutdoorAdventureSports(String name) {

    if(name.equalsIgnoreCase("Bungee Jumping"))

      return new BungeeJumpingRegular();

    else if(name.equalsIgnoreCase("Paragliding"))

      return new ParaglidingRegular();

    else if(name.equalsIgnoreCase("Trekking"))

      return new TrekkingRegular();

    else

      return null;

  }

  public OutdoorRegularGames getOutdoorRegularGames() {

    return new CricketRegular();

  }

  public IndoorRegularGames getIndoorRegularGames() {

    return new TableTennisRegular();

  }

}

MainClass.java

package tryAbstractFactory;

public class MainClass {

  public static void main(String[] args) {

    //Regular Sports

//Runtime Polymorphism is used

    SportsCategoryFactory reg = new RegularSportsFactory();

//These will execute the methods written in child class

    OutdoorAdventureSports sp1 = reg.getOutdoorAdventureSports("trekking");

    OutdoorRegularGames sp2 = reg.getOutdoorRegularGames();

    IndoorRegularGames sp3 = reg.getIndoorRegularGames();

    sp1.getSportName();

    sp2.getSportName();

    sp3.getSportName();

    //Differently Abled Sports

    SportsCategoryFactory diffabled = new DiffAbledSportsFactory();

//These will execute the methods written in child class

    OutdoorAdventureSports dsp1 = diffabled.getOutdoorAdventureSports("Paragliding");

    OutdoorRegularGames dsp2 = diffabled.getOutdoorRegularGames();

    IndoorRegularGames dsp3 = diffabled.getIndoorRegularGames();

    dsp1.getSportName();

    dsp2.getSportName();

    dsp3.getSportName();

  }

}

**Output :-**

Regular Trekking Regular Cricket Regular Table Tennis

Differently Abled Paragliding Differently Abled Cricket Differently Abled Table Tennis

**TEMPLATE PATTERN**

Menu.java

package tryTemplate;

public interface Menu {

  void displayMenu();

}

Cycle.java

package tryTemplate;

import java.util.ArrayList;

public class Cycle implements Menu {

  ArrayList<Item> array\_List = new ArrayList<Item>();

  public Cycle() {

    array\_List.add(new Item(1, "cycle1", 1000));

    array\_List.add(new Item(2, "cycle2", 2000));

    array\_List.add(new Item(3, "cycle3", 3000));

    array\_List.add(new Item(4, "cycle4", 4000));

    array\_List.add(new Item(5, "cycle5", 5000));

  }

  public void displayMenu() {

    System.out.println("List of Items:-");

    for (Item itr : array\_List)

      System.out.println("\nID: "+itr.id+" Name: "+itr.name+" Price: "+itr.price);

  }

}

Item.java

package tryTemplate;

public class Item {

  String name;

  float price;

  int id;

  public Item(int id, String name, float price) {

    this.name = name;

    this.price = price;

    this.id = id;

  }

  int getID() {

    return id;

  }

}

OrderProcessing.java

package tryTemplate;

public abstract class OrderProcessing {

  abstract Item selectItem();

  abstract void doPayment(Item item);

  abstract void doDelivery();

  void purchaseItem() {

    Item selectedItem = selectItem();

    if (selectedItem != null) {

      doPayment(selectedItem);

      doDelivery();

    }

  }

}

OfflineOrder.java

package tryTemplate;

import java.util.Scanner;

public class OfflineOrder extends OrderProcessing {

  Scanner sc = new Scanner(System.in);

  Cycle menu;

  public OfflineOrder() {

    menu = new Cycle();

  }

  Item selectItem() {

    for (Item itr : menu.array\_List) {

      System.out.println("\nID: "+itr.id + " Name: " + itr.name + " Price: " + itr.price);

      System.out.println("Do you wish to select this product?(yes/no): ");

      String choice = sc.nextLine();

      if (choice.equalsIgnoreCase("yes"))

        return itr;

    }

    System.out.println("No More Items to show!");

    return null;

  }

  void doPayment(Item itr) {

    System.out.print("Selected Item:-  ");

    System.out.print("ID: " + itr.id + " Name: " + itr.name + " Price: " + itr.price);

    System.out.println("\nPayment Modes:-\n1.Cash\n2.Card");

    int choice;

    do {

      System.out.print("Enter Your Choice: ");

      choice = Integer.parseInt(sc.nextLine());

      switch (choice) {

        case 1: cash(); break;

        case 2: card(); break;

        default:   System.out.println("Invalid Payment Option! Try Again!");

      }

    } while (choice != 1 && choice != 2);

  }

  private void card() {

    System.out.println("Thanks for the Card Payment!");

  }

  private void cash() {

    System.out.println("Thanks for the Cash Payment!");

  }

  void doDelivery() {

    System.out.println("Your product will be delivered at your Address!");

  }

}

OnlineOrder.java

package tryTemplate;

import java.util.Scanner;

public class OnlineOrder extends OrderProcessing {

  Cycle menu;

  Scanner sc = new Scanner(System.in);

  public OnlineOrder() {

    menu = new Cycle();

  }

  Item selectItem() {

    menu.displayMenu();

    System.out.println("Enter ID of Product: ");

    int query\_id = Integer.parseInt(sc.nextLine());

    for (Item itr : menu.array\_List)

      if (itr.getID() == query\_id)

        return itr;

    System.out.println("Item not Found!");

    return null;

  }

  void doPayment(Item itr) {

    System.out.println("\nSelected Item:-");

    System.out.println("\nID: " + itr.id + " Name: " + itr.name + " Price: " + itr.price);

    System.out.println("\nPayment Modes:-\n1.Paytm\n2.Card");

    int choice;

    do {

      System.out.print("Enter Your Choice: ");

      choice = Integer.parseInt(sc.nextLine());

      switch (choice) {

        case 1:   paytm();  break;

        case 2:   card(); break;

        default:  System.out.println("Invarray\_Listid Payment Option!Try Again!");

      }

    } while (choice != 1 && choice != 2);

  }

  private void card() {

    System.out.println("Thanks for the Card Payment!");

  }

  private void paytm() {

    System.out.println("Thanks for the Paytm Payment!");

  }

  void doDelivery() {

    System.out.println("Your product will be delivered at your Address!");

  }

}

TemplateDemo.java

package tryTemplate;

public class TemplateDemo {

  public static void main(String[] args) {

    OrderProcessing onlineObject = new OnlineOrder();

    OrderProcessing offlineObject = new OfflineOrder();

    offlineObject.purchaseItem();

  }

}

**Output :-**

ID: 1 Name: cycle1 Price: 1000.0

Do you wish to select this product?(yes/no): yes

Selected Item:-ID: 1 Name: cycle1 Price: 1000.0

Payment Modes:-

1.Cash

2.Card

Enter Your Choice: 1

Thanks for the Cash Payment!

Your product will be delivered at your Address!

**DECORATOR PATTERN**

Sports.java

package tryDecorator;

public abstract class Sports {

  public Sports() {};

  public abstract int getCurrentStock();

}

IndoorSports.java

package tryDecorator;

public class IndoorSports extends Sports{

  public IndoorSports() {};

  public int getCurrentStock() {

    return 0;

  }

}

OutdoorSports.java

package tryDecorator;

public class OutdoorSports extends Sports{

  public OutdoorSports() {};

  public int getCurrentStock() {

    return 0;

  }

}

IndoorSportsDecorator.java

package tryDecorator;

public abstract class IndoorSportsDecorator extends Sports{

  public IndoorSportsDecorator() {};

  public abstract int getCurrentStock();

}

BoardGames.java

package tryDecorator;

public abstract class BoardGames extends IndoorSportsDecorator{

  public BoardGames() {};

}

Carrom.java

package tryDecorator;

public class Carrom extends BoardGames {

  Sports sport;

  public Carrom(Sports sport) {

    this.sport = sport;

  }

  public int getCurrentStock() {

    return 2+sport.getCurrentStock();

  }

}

Chess.java

package tryDecorator;

public class Chess extends BoardGames  {

  Sports sport;

  public Chess(Sports sport) {

    this.sport = sport;

  }

  public int getCurrentStock() {

    return 2+sport.getCurrentStock();

  }

}

CourtGames.java

package tryDecorator;

public abstract class CourtGames extends IndoorSportsDecorator{

  public CourtGames() {};

}

Kabaddi.java

package tryDecorator;

public class Kabaddi extends CourtGames {

  Sports sport;

  public Kabaddi(Sports sport) {

    this.sport = sport;

  }

  public int getCurrentStock() {

    return 2+sport.getCurrentStock();

  }

}

Badminton.java

package tryDecorator;

public class Badminton extends CourtGames{

  Sports sport;

  public Badminton(Sports sport) {

    this.sport = sport;

  }

  public int getCurrentStock() {

    return 2+sport.getCurrentStock();

  }

}

Basketball.java

package tryDecorator;

public class Basketball extends CourtGames{

  Sports sport;

  public Basketball(Sports sport) {

    this.sport = sport;

  }

  public int getCurrentStock() {

    return 2+sport.getCurrentStock();

  }

}

GamesOnTable.java

package tryDecorator;

public abstract class GamesOnTable extends IndoorSportsDecorator{

  public GamesOnTable() {};

}

Billiards.java

package tryDecorator;

public class Billiards extends GamesOnTable{

  Sports sport;

  public Billiards(Sports sport) {

    this.sport = sport;

  }

  public int getCurrentStock() {

    return 2+sport.getCurrentStock();

  }

}

Snooker.java

package tryDecorator;

public class Snooker extends GamesOnTable{

  Sports sport;

  public Snooker(Sports sport) {

    this.sport = sport;

  }

  public int getCurrentStock() {

    return 2+sport.getCurrentStock();

  }

}

TableTennis.java

package tryDecorator;

public class TableTennis extends GamesOnTable {

  Sports sport;

  public TableTennis(Sports sport) {

    this.sport = sport;

  }

  public int getCurrentStock() {

    return 2+sport.getCurrentStock();

  }

}

OutdoorSportsDecorator.java

package tryDecorator;

public abstract class OutdoorSportsDecorator extends Sports{

  public OutdoorSportsDecorator() {};

  public abstract int getCurrentStock();

}

AdventureGames.java

package tryDecorator;

public abstract class AdventureGames extends OutdoorSportsDecorator{

  public AdventureGames() {};

}

Trekking.java

package tryDecorator;

public class Trekking extends AdventureGames{

  Sports sport;

  public Trekking(Sports sport) {

    this.sport = sport;

  }

  public int getCurrentStock() {

    return 2 + sport.getCurrentStock();

  }

}

Paragliding.java

package tryDecorator;

public class Paragliding extends AdventureGames{

  Sports sport;

  public Paragliding(Sports sport) {

    this.sport = sport;

  }

  public int getCurrentStock() {

    return 2 + sport.getCurrentStock();

  }

}

BungeeJumping.java

package tryDecorator;

public class BungeeJumping extends AdventureGames{

  Sports sport;

  public BungeeJumping(Sports sport) {

    this.sport = sport;

  }

  public int getCurrentStock() {

    return 2 + sport.getCurrentStock();

  }

}

Athletics.java

package tryDecorator;

public abstract class Athletics extends OutdoorSportsDecorator{

  public Athletics() {};

}

HighJump.java

package tryDecorator;

public class HighJump extends Athletics{

  Sports sport;

  public HighJump(Sports sport) {

    this.sport = sport;

  }

  public int getCurrentStock() {

    return 2 + sport.getCurrentStock();

  }

}

LongJump.java

package tryDecorator;

public class LongJump extends Athletics{

  Sports sport;

  public LongJump(Sports sport) {

    this.sport = sport;

  }

  public int getCurrentStock() {

    return 2 + sport.getCurrentStock();

  }

}

StadiumGames.java

package tryDecorator;

public abstract class StadiumGames extends OutdoorSportsDecorator {

  public StadiumGames() {};

}

Football.java

package tryDecorator;

public class Football extends StadiumGames{

  Sports sport;

  public Football(Sports sport) {

    this.sport = sport;

  }

  public int getCurrentStock() {

    return 2+sport.getCurrentStock();

  }

}

Baseball.java

package tryDecorator;

public class Baseball extends StadiumGames{

  Sports sport;

  public Baseball(Sports sport) {

    this.sport = sport;

  }

  public int getCurrentStock() {

    return 2+sport.getCurrentStock();

  }

}

Cricket.java

package tryDecorator;

public class Cricket extends StadiumGames{

  Sports sport;

  public Cricket(Sports sport) {

    this.sport = sport;

  }

  public int getCurrentStock() {

    return 2+sport.getCurrentStock();

  }

}

Main.java

package tryDecorator;

public class Main {

  public static void main(String[] args) {

    //Assuming stock of each sport is 2

    Sports sp1=new IndoorSports();

    System.out.println("Total Indoor Sports Stock:"+sp1.getCurrentStock());

    sp1=new Billiards(sp1);

    System.out.println("Total Indoor Sports Stock:"+sp1.getCurrentStock());

    sp1=new Carrom(sp1);

    System.out.println("Total Indoor Sports Stock:"+sp1.getCurrentStock());

    sp1=new Badminton(sp1);

    System.out.println("Total Indoor Sports Stock:"+sp1.getCurrentStock());

    Sports sp2=new OutdoorSports();

    System.out.println("\nTotal Outdoor Sports Stock:"+sp2.getCurrentStock());

    sp2=new Trekking(sp2);

    System.out.println("Total Outdoor Sports Stock:"+sp2.getCurrentStock());

    sp2=new Cricket(sp2);

    System.out.println("Total Outdoor Sports Stock:"+sp2.getCurrentStock());

    sp2=new HighJump(sp2);

    System.out.println("Total Outdoor Sports Stock:"+sp2.getCurrentStock());

    sp2=new LongJump(sp2);

    System.out.println("Total Outdoor Sports Stock:"+sp2.getCurrentStock());

  }

}

**FACTORY PATTERN**

Customer.java

package tryFactory;

public interface Customer {

  void getCustomerType();

}

FTCustomer.java

package tryFactory;

public class FTCustomer implements Customer{

  public void getCustomerType() {

     System.out.println("FTCustomer");

  }

}

RCustomer.java

package tryFactory;

public class RCustomer implements Customer{

  public void getCustomerType() {

    System.out.println("RCustomer");

  }

}

SCCustomer.java

package tryFactory;

public class SCCustomer implements Customer{

  public void getCustomerType() {

    System.out.println("SCCustomer");

  }

}

CutomerFactory.java

package tryFactory;

public class CutomerFactory {

  Customer getObject(String customerType) {

    if(customerType.equalsIgnoreCase("FTCustomer"))

      return new FTCustomer();

    else if(customerType.equalsIgnoreCase("RCustomer"))

      return new RCustomer();

    else if(customerType.equalsIgnoreCase("SCCustomer"))

      return new SCCustomer();

    else

      return null;

  }

}

Main.java

package tryFactory;

public class Main {

  public static void main(String[] args) {

    CutomerFactory customerFactoryObject = new CutomerFactory();

    Customer c1 = customerFactoryObject.getObject("FTCustomer");

    Customer c2 = customerFactoryObject.getObject("RCustomer");

    Customer c3 = customerFactoryObject.getObject("SCCustomer");

    c1.getCustomerType();

    c2.getCustomerType();

    c3.getCustomerType();

  }

}

**SINGLETON PATTERN**

TabWindow.java

package trySingleton;

import java.util.ArrayList;

public class TabWindow {

  //Creating the only object that will be created of this Class

  public static TabWindow sc;

  ArrayList<String> urls;

  //This constructor is Important

  private TabWindow() {

    urls=new ArrayList<String>();

  }

  public static TabWindow getInstance() {

    if(sc==null)

      sc=new TabWindow();

    return sc;

  }

  public void addUrl(String url) {

    urls.add(url);

  }

  public void showUrls() {

    for(String u:urls)

      System.out.println(u);

  }

}

Main.java

package trySingleton;

public class Main {

  public static void main(String[] args) {

    TabWindow s1=TabWindow.getInstance();

    TabWindow s2=TabWindow.getInstance();

    s1.addUrl("www.google.com");

    s2.addUrl("www.facebook.com");

    TabWindow s3=TabWindow.getInstance();

    s3.showUrls();

  }

}