Selected files

16 printable files

src\main\java\org\example\Vending_Machine\AcceptMoneyState.java
src\main\java\org\example\Vending_Machine\CheetosDispenseHandler.java
src\main\java\org\example\Vending_Machine\CokeDispenseHandler.java
src\main\java\org\example\Vending_Machine\DispenseChangeState.java
src\main\java\org\example\Vending_Machine\DispenseItemState.java
src\main\java\org\example\Vending_Machine\DoritosDispenseHandler.java
src\main\java\org\example\Vending_Machine\IdleState.java
src\main\java\org\example\Vending_Machine\ItemSelectionState.java
src\main\java\org\example\Vending_Machine\KitKatDispenseHandler.java
src\main\java\org\example\Vending_Machine\Main.java
src\main\java\org\example\Vending_Machine\PepsiDispenseHandler.java
src\main\java\org\example\Vending_Machine\Snack.java
src\main\java\org\example\Vending_Machine\SnackDispenseHandler.java
src\main\java\org\example\Vending_Machine\SnickersDispenseHandler.java
src\main\java\org\example\Vending_Machine\StateOfVendingMachine.java
src\main\java\org\example\Vending_Machine\VendingMachine.java

src\main\java\org\example\Vending_Machine\AcceptMoneyState.java

```java
package org.example.Vending_Machine;

public class AcceptMoneyState implements StateOfVendingMachine {
    @Override
    public void idle(VendingMachine vendingMachine) {

    }

    @Override
    public void itemSelection(VendingMachine vendingMachine) {

    }

    @Override
    public void acceptMoney(VendingMachine vendingMachine) {

    }

    @Override
    public void dispenseItem(VendingMachine vendingMachine) {

    }

    @Override
    public void dispenseChange(VendingMachine vendingMachine) {
        vendingMachine.setState( new DispenseChangeState() );
        System.out.println("Changing state to Dispensing Change");
    }

    @Override
    public void cancel(VendingMachine vendingMachine) {
        vendingMachine.setState( new IdleState() );
```

```
        System.out.println("Canceling --- Changing state to Idle.\tDispense any change in process");

    }
}

```

src\main\java\org\example\Vending_Machine\CheetosDispenseHandler.java

```
package org.example.Vending_Machine;

public class CheetosDispenseHandler extends SnackDispenseHandler {
    public CheetosDispenseHandler(SnackDispenseHandler next) {
        super(next);
    }

    public void handleSnack(String itemName)
    {
        if( itemName.equals("Cheetos") )
        {
            System.out.println("Dispensing Cheetos");
        }
        else
        {
            System.out.println("I was passed from Cheetos");
            super.handleSnack( itemName );
        }
    }
}

```

src\main\java\org\example\Vending_Machine\CokeDispenseHandler.java

```
package org.example.Vending_Machine;

public class CokeDispenseHandler extends SnackDispenseHandler {
    public CokeDispenseHandler(SnackDispenseHandler next) {
        super(next);
    }

    public void handleSnack(String itemName)
    {
        if( itemName.equals("Coke") )
        {
            System.out.println("Dispensing Coke");
        }
        else
        {
            System.out.println("I was passed from Coke");
            super.handleSnack( itemName );
        }
    }
}
```

src\main\java\org\example\Vending_Machine\DispenseChangeState.java

```java
package org.example.Vending_Machine;

public class DispenseChangeState implements StateOfVendingMachine {
    @Override
    public void idle(VendingMachine vendingMachine) {

    }

    @Override
    public void itemSelection(VendingMachine vendingMachine) {

    }

    @Override
    public void acceptMoney(VendingMachine vendingMachine) {

    }

    @Override
    public void dispenseItem(VendingMachine vendingMachine) {

        vendingMachine.setState( new DispenseItemState() );
        System.out.println("Changing state to Dispensing Items");
    }

    @Override
    public void dispenseChange(VendingMachine vendingMachine) {

    }

    @Override
    public void cancel(VendingMachine vendingMachine) {

    }
}

```

src\main\java\org\example\Vending_Machine\DispenseItemState.java

```java
package org.example.Vending_Machine;

public class DispenseItemState implements StateOfVendingMachine {
    @Override
    public void idle(VendingMachine vendingMachine) {

        vendingMachine.setState( new IdleState() );
        System.out.println("Changing state to Idle");
    }

```

```
    @Override
    public void itemSelection(VendingMachine vendingMachine) {

    }

    @Override
    public void acceptMoney(VendingMachine vendingMachine) {

    }

    @Override
    public void dispenseItem(VendingMachine vendingMachine) {

    }

    @Override
    public void dispenseChange(VendingMachine vendingMachine) {

    }

    @Override
    public void cancel(VendingMachine vendingMachine) {

    }
}

```

src\main\java\org\example\Vending_Machine\DoritosDispenseHandler.java

```
package org.example.Vending_Machine;

public class DoritosDispenseHandler extends SnackDispenseHandler {
    public DoritosDispenseHandler(SnackDispenseHandler next) {
        super(next);
    }

    public void handleSnack(String itemName)
    {
        if( itemName.equals("Doritos") )
        {
            System.out.println("Dispensing Doritos");
        }
        else
        {
            System.out.println("I was passed from Doritos");
            super.handleSnack( itemName );
        }
    }
}

```

src\main\java\org\example\Vending_Machine\IdleState.java

```java
package org.example.Vending_Machine;

public class IdleState implements StateOfVendingMachine{

    @Override
    public void idle(VendingMachine vendingMachine) {
        System.out.println("Vending Machine already in Idle State");
    }

    @Override
    public void itemSelection(VendingMachine vendingMachine) {
        vendingMachine.setState( new ItemSelectionState() );
        System.out.println("Changing state to Item Selection. User can now select snacks.");
    }

    @Override
    public void acceptMoney(VendingMachine vendingMachine) {

    }

    @Override
    public void dispenseItem(VendingMachine vendingMachine) {

    }

    @Override
    public void dispenseChange(VendingMachine vendingMachine) {

    }

    @Override
    public void cancel(VendingMachine vendingMachine) {

    }
}

```

src\main\java\org\example\Vending_Machine\ItemSelectionState.java

```java
package org.example.Vending_Machine;

import org.example.Vending_Machine.StateOfVendingMachine;

public class ItemSelectionState implements StateOfVendingMachine {
    @Override
    public void idle(VendingMachine vendingMachine) {

    }

    @Override
    public void itemSelection(VendingMachine vendingMachine) {
        System.out.println("Vending Machine already in Item Selection State");
    }

```

```
16      @Override
17      public void acceptMoney(VendingMachine vendingMachine) {
18          vendingMachine.setState( new AcceptMoneyState() );
19          System.out.println("Changing state to Accept Money. User can now insert money.");
20      }
21
22      @Override
23      public void dispenseItem(VendingMachine vendingMachine) {
24
25      }
26
27      @Override
28      public void dispenseChange(VendingMachine vendingMachine) {
29
30      }
31
32      @Override
33      public void cancel(VendingMachine vendingMachine) {
34          vendingMachine.setState( new IdleState() );
35          System.out.println("Canceling --- Changing state to Idle");
36      }
37  }
38
```

src\main\java\org\example\Vending_Machine\KitKatDispenseHandler.java

```
1   package org.example.Vending_Machine;
2
3   public class KitKatDispenseHandler extends SnackDispenseHandler {
4       public KitKatDispenseHandler(SnackDispenseHandler next) {
5           super(next);
6       }
7
8       public void handleSnack(String itemName)
9       {
10          if( itemName.equals("KitKat") )
11          {
12              System.out.println("Dispensing KitKat");
13          }
14          else
15          {
16              System.out.println("I was passed from KitKat");
17              super.handleSnack( itemName );
18          }
19      }
20  }
21
```

src\main\java\org\example\Vending_Machine\Main.java

```
1   package org.example.Vending_Machine;
2
3   import java.util.Scanner;
```

```
4
5  public class Main {
6      public static void main(String[] args)
7      {
8          System.out.println("Creating Vending Machine");
9
10         VendingMachine vm = new VendingMachine();
11         vm.idle();
12
13         System.out.println("Filling Vending Machine");
14
15         vm.addSnack( new Snack("Coke", 2.99, 10) );
16         vm.addSnack( new Snack("Pepsi", 2.99, 10) );
17         vm.addSnack( new Snack("Cheetos", 4.48, 8) );
18         vm.addSnack( new Snack("Doritos", 3.98, 12) );
19         vm.addSnack( new Snack("KitKat", 1.99, 20) );
20         vm.addSnack( new Snack("Snickers", 2.29, 10) );
21
22         vm.printVendingMachine();
23
24         vm.startVendingMachine();
25
26         System.out.println("\nVending Machine - Remaining snacks");
27
28         vm.printVendingMachine();
29
30     }
31 }
```

src\main\java\org\example\Vending_Machine\PepsiDispenseHandler.java

```
1  package org.example.Vending_Machine;
2
3  public class PepsiDispenseHandler extends SnackDispenseHandler {
4      public PepsiDispenseHandler(SnackDispenseHandler next) {
5          super(next);
6      }
7
8      public void handleSnack(String itemName)
9      {
10         if( itemName.equals("Pepsi") )
11         {
12             System.out.println("Dispensing Pepsi");
13         }
14         else
15         {
16             System.out.println("I was passed from Pepsi");
17             super.handleSnack( itemName );
18         }
19     }
20 }
21
```

src\main\java\org\example\Vending_Machine\Snack.java

```java
package org.example.Vending_Machine;

public class Snack {
    private String name;
    private double price;
    private int quantity;

    public Snack(String name, double price, int quantity)
    {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }

    public String getName()
    {
        return name;
    }

    public double getPrice()
    {
        return price;
    }

    public int getQuantity()
    {
        return quantity;
    }

    public void setName(String name)
    {
        this.name = name;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public void deductQuantity(int quantity)
    {
        this.quantity = this.quantity - quantity;
    }
    @Override
    public String toString()
    {
        return "[" + this.name + "\t$" + this.price + "\tRemaining: " + this.quantity + "]";
    }
}

```

src\main\java\org\example\Vending_Machine\SnackDispenseHandler.java

```java
package org.example.Vending_Machine;

public abstract class SnackDispenseHandler {
    private SnackDispenseHandler next;

    public SnackDispenseHandler( SnackDispenseHandler next )
    {
        this.next = next;
    }

    public void handleSnack(String itemName)
    {
        if( next != null )
        {
            next.handleSnack( itemName );
        }
    }
}

```

src\main\java\org\example\Vending_Machine\SnickersDispenseHandler.java

```java
package org.example.Vending_Machine;

public class SnickersDispenseHandler extends SnackDispenseHandler {
    public SnickersDispenseHandler(SnackDispenseHandler next) {
        super(next);
    }

    public void handleSnack(String itemName)
    {
        if( itemName.equals("Snickers") )
        {
            System.out.println("Dispensing Snickers");
        }
        else
        {
            System.out.println("I was passed from Snickers");
            super.handleSnack( itemName );
        }
    }
}

```

src\main\java\org\example\Vending_Machine\StateOfVendingMachine.java

```java
package org.example.Vending_Machine;

public interface StateOfVendingMachine {
//- Represents the different states of the vending machine. What are all the
//things vending machine can do?
    void idle(VendingMachine vendingMachine);
```

```
    void itemSelection(VendingMachine vendingMachine);
    void acceptMoney(VendingMachine vendingMachine);
    void dispenseItem(VendingMachine vendingMachine);
    void dispenseChange(VendingMachine vendingMachine);
    void cancel(VendingMachine vendingMachine);

}

```

src\main\java\org\example\Vending_Machine\VendingMachine.java

```
package org.example.Vending_Machine;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class VendingMachine {
    //Should reference state of Vending Machine and SnackDispenser plus hold all snacks.
    private SnackDispenseHandler chain = new CokeDispenseHandler( new PepsiDispenseHandler( new CheetosDispenseHandler( new DoritosDispenseHandler( new KitKatDispenseHandler( new SnickersDispenseHandler(null))))));

    private StateOfVendingMachine state = new IdleState();
    private ArrayList<Snack> snacks;
    private HashMap<Snack,Integer> userSelectedSnacks;
    private double userMoney;
    private double totalPrice;

    public VendingMachine()
    {
        this.snacks = new ArrayList<>();
        this.userSelectedSnacks = new HashMap<>();
        this.userMoney = 0;
        this.totalPrice = 0;
    }

    public void addSnack(Snack snack)
    {
        snacks.add(snack);
    }

    public ArrayList<Snack> getSnacks()
    {
        return this.snacks;
    }

    public void setState(StateOfVendingMachine state)
    {
        this.state = state;
    }

    public StateOfVendingMachine getState()
    {
```

```
43          return state;
44      }
45
46      public void startVendingMachine(){
47          this.itemSelection();
48      }
49
50      public void idle()
51      {
52          getState().idle(this);
53      }
54      public void itemSelection() {
55          getState().itemSelection(this);
56          selectSnack();
57          printSelectedSnack();
58
59          this.acceptMoney();
60      }
61
62      private void selectSnack() {
63          Scanner sc = new Scanner(System.in);
64
65          int exit = 1;
66
67          while(exit == 1 )
68          {
69              System.out.println("Select snack number (0 - exit, 7 - cancel selection): ");
70              int userSelectedSnack = sc.nextInt();
71              sc.nextLine();
72
73              if( userSelectedSnack == 0 ) {
74                  exit = 0;
75                  System.out.println("Exiting item selection...");
76                  continue;
77              }
78
79              if( userSelectedSnack == 7 ) {
80  //                exit = 0;
81                  System.out.println("Canceling item selection...");
82                  cancel();
83                  break;
84              }
85
86              if( userSelectedSnack < 0 || userSelectedSnack > 7)
87              {
88                  System.out.println("Invalid item selection...");
89                  continue;
90              }
91
92              System.out.println("Enter quantity: ");
93              int userSelectedQuantity = sc.nextInt();
94              sc.nextLine();
95
96              userSelectedSnack--;
97              Snack s = snacks.get(userSelectedSnack);
98
```

```
            boolean snackQuantity = checkSnackQuantity(s, userSelectedQuantity);

            if( snackQuantity ) {
                s.deductQuantity(userSelectedQuantity);
                userSelectedSnacks.put(s, userSelectedQuantity);
            }
            else
            {
                System.out.println("Invalid quantity entered. Snack not added.");
            }
        }
    }

    private boolean checkSnackQuantity(Snack snack, int quantity) {
        return quantity <= snack.getQuantity();
    }

    private void printSelectedSnack() {
        System.out.println("User selected the following snacks:\n============================
==");
        for(Map.Entry<Snack, Integer> userSelectedSnack : userSelectedSnacks.entrySet() )
        {
            System.out.println(userSelectedSnack.getKey().getName() + ":\t$" +
userSelectedSnack.getKey().getPrice() + " x " + userSelectedSnack.getValue());
        }
    }
    public void acceptMoney()
    {
        getState().acceptMoney(this);
        getSnackTotal();
        insertMoney();

        this.dispenseChange();
    }

    private void insertMoney() {
        Scanner sc = new Scanner(System.in);

        int exit = 1;

        while(exit == 1 ) {
            System.out.print("Enter inserted money: $");
            this.userMoney += sc.nextDouble();
            sc.nextLine();

            System.out.print("Are you finished entering money? (y/n/cancel)");
            String input = sc.nextLine().toLowerCase();
            if (input.equals("y") && this.userMoney >= this.totalPrice) {
                exit = 0;
                continue;
            }

            if (input.equals("cancel")) {
                exit = 0;
                cancel();
                break;
            }
```

```
154
155             if( this.userMoney < this.totalPrice) {
156                 System.out.println("Insufficient funds. $" + this.userMoney);
157             }
158         }
159     }
160
161     private void getSnackTotal() {
162         double totalPrice = 0;
163         for(Map.Entry<Snack, Integer> userSelectedSnack : userSelectedSnacks.entrySet() )
164         {
165             totalPrice += userSelectedSnack.getKey().getPrice() *
    userSelectedSnack.getValue();
166         }
167
168         System.out.println("Total: $" + totalPrice);
169
170         this.totalPrice = totalPrice;
171     }
172
173     public void dispenseChange()
174     {
175         getState().dispenseChange(this);
176
177         System.out.println("User inserted\t$" + userMoney + "\nTotal price\t\t$" + totalPrice
    + "\n------------------------------");
178         userMoney = userMoney - totalPrice;
179         System.out.println("Change dispensed\t$" + userMoney);
180
181         this.dispenseItem();
182     }
183
184     public void dispenseItem()
185     {
186         getState().dispenseItem(this);
187
188         dispenseSnack();
189
190         this.idle();
191     }
192
193     private void dispenseSnack() {
194
195         for(Map.Entry<Snack, Integer> userSelectedSnack : userSelectedSnacks.entrySet() )
196         {
197             String snackName = userSelectedSnack.getKey().getName();
198             for (int i = 0; i < userSelectedSnack.getValue(); i++) {
199                 handleSnack(snackName);
200             }
201         }
202     }
203
204     public void cancel()
205     {
206         getState().cancel(this);
207     }
208     public void handleSnack(String itemName)
```

```
209     {
210         chain.handleSnack(itemName);
211     }
212 
213     public void printVendingMachine()
214     {
215         int selectNumber = 1;
216         for(Snack snack : snacks)
217         {
218             System.out.println(selectNumber++ + ". " + snack);
219         }
220     }
221 }
222 
223 
```