

## Part 1 (Vending Machine) Implementation

```
1 package edu.cpp.cs5800.VendingMachine;
2
3 public class Driver {
4     public static void main(String[] args) {
5         VendingMachine vendingMachine = new VendingMachine();
6         vendingMachine.printAvailableSnacks();
7
8         vendingMachine.selectSnack(6);
9         vendingMachine.insertMoney(6.5);
10        vendingMachine.dispenseSnack();
11        System.out.println();
12
13        vendingMachine.printAvailableSnacks();
14        vendingMachine.selectSnack(6);
15        vendingMachine.insertMoney(5);
16        vendingMachine.dispenseSnack();
17        System.out.println();
18
19        vendingMachine.selectSnack(3);
20        vendingMachine.insertMoney(2);
21        vendingMachine.dispenseSnack();
22        System.out.println();
23
24        vendingMachine.selectSnack(4);
25        vendingMachine.insertMoney(3.5);
26        vendingMachine.dispenseSnack();
27        System.out.println();
28
29    }
30 }
31
```

## Part 1 (Vending Machine) Implementation

```
1 package edu.cpp.cs5800.VendingMachine;
2
3 import edu.cpp.cs5800.VendingMachine.snacks.*;
4 import edu.cpp.cs5800.VendingMachine.states.Idle;
5 import edu.cpp.cs5800.VendingMachine.states.StateOfVendingMachine;
6
7 import java.util.HashMap;
8 import java.util.Map;
9
10 public class VendingMachine {
11     private StateOfVendingMachine state = new Idle(this);
12     private int selectedSnack;
13     private double insertedAmount;
14     private SnackDispenseHandler snackDispenseHandler;
15
16     private Map<Integer, Snack> stocks = new HashMap<>();
17
18     public VendingMachine() {
19         Snack.resetId();
20         Coke coke = new Coke(15, 3);
21         Pepsi pepsi = new Pepsi(15, 3);
22         Cheetos cheetos = new Cheetos(15, 2.5);
23         Doritos doritos = new Doritos(15, 3.5);
24         KitKat kitkat = new KitKat(2, 5);
25         Snickers snickers = new Snickers(1, 4);
26
27         SnackDispenseHandler snickersHandler = new SnackDispenseHandler(snickers, null);
28         SnackDispenseHandler kitkatHandler = new SnackDispenseHandler(kitkat,
snickersHandler);
29         SnackDispenseHandler doritosHandler = new SnackDispenseHandler(doritos,
kitkatHandler);
30         SnackDispenseHandler cheetosHandler = new SnackDispenseHandler(cheetos,
doritosHandler);
31         SnackDispenseHandler pepsiHandler = new SnackDispenseHandler(pepsi, cheetosHandler
);
32         snackDispenseHandler = new SnackDispenseHandler(coke, pepsiHandler);
33
34         stocks.put(coke.getId(), coke);
35         stocks.put(pepsi.getId(), pepsi);
36         stocks.put(cheetos.getId(), cheetos);
37         stocks.put(doritos.getId(), doritos);
38         stocks.put(kitkat.getId(), kitkat);
39         stocks.put(snickers.getId(), snickers);
40     }
41
42     public void printAvailableSnacks() {
43         for (int key: stocks.keySet().stream().sorted().toList()) {
44             Snack snack = stocks.get(key);
45             System.out.println(key + ": " + snack.getName() + ", price: $" + snack.getPrice
() + ", stock: " + snack.getQuantity());
46         }
47         System.out.println();
48     }
}
```

## Part 1 (Vending Machine) Implementation

```
49
50 public SnackDispenseHandler getSnackDispenseHandler() {
51     return snackDispenseHandler;
52 }
53
54 public Snack getSnack(int number) {
55     return stocks.getDefault(number, null);
56 }
57
58 public void setState(StateOfVendingMachine state) {
59     System.out.println("State transition: " + this.state + "-->" + state);
60     System.out.println("-----");
61     this.state = state;
62 }
63
64 public StateOfVendingMachine getState() {
65     return state;
66 }
67
68 public int getSelectedSnack() {
69     return selectedSnack;
70 }
71
72 public void setSelectedSnack(int selectedSnack) {
73     this.selectedSnack = selectedSnack;
74 }
75
76 public double getInsertedAmount() {
77     return insertedAmount;
78 }
79
80 public void setInsertedAmount(double insertedAmount) {
81     this.insertedAmount = insertedAmount;
82 }
83
84 public void selectSnack(int number) {
85     this.state.selectSnack(number);
86 }
87
88 public void insertMoney(double amount) {
89     this.state.insertMoney(amount);
90 }
91
92 public void dispenseSnack() {
93     this.state.dispenseSnack();
94 }
95 }
96
```

## Part 1 (Vending Machine) Implementation

```
1 package edu.cpp.cs5800.VendingMachine;
2
3 import edu.cpp.cs5800.VendingMachine.snacks.Snack;
4
5 public class SnackDispenseHandler {
6     private Snack snack;
7     private SnackDispenseHandler nextHandler;
8
9     public SnackDispenseHandler(Snack snack, SnackDispenseHandler nextHandler) {
10         this.snack = snack;
11         this.nextHandler = nextHandler;
12     }
13
14     public boolean dispense(int id, double amount) {
15         System.out.print(this.snack.getName() + "Handler");
16         if (this.snack.getId() == id) {
17             System.out.println();
18             if (this.snack.getQuantity() == 0) {
19                 System.out.println("Not enough quantity for item: " + this.snack.getName
20 ());
21                 System.out.printf("Returning money inserted: $%2.2f\n", amount);
22             } else if (this.snack.getPrice() <= amount) {
23                 System.out.println("Dispensing: " + this.snack.dispense() + ", price: $" +
24 this.snack.getPrice());
25                 if (this.snack.getPrice() < amount) {
26                     System.out.printf("Returning Change: $%2.2f\n", amount - this.snack.
27 getPrice());
28                 }
29                 return true;
30             } else {
31                 System.out.println("Money inserted is not enough: " + this.snack.getName
32 () + ", price: $" + this.snack.getPrice());
33             }
34         } else if (this.nextHandler != null) {
35             System.out.print("->");
36             return this.nextHandler.dispense(id, amount);
37         } else {
38             System.out.println("Item id not found: " + id);
39         }
40     }
41
42     return false;
43 }
44 }
```

## Part 1 (Vending Machine) Implementation

```
1 package edu.cpp.cs5800.VendingMachine.snacks;
2
3 public class Coke extends Snack {
4     public Coke(int quantity, double price) {
5         super("Coke", quantity, price);
6     }
7
8 }
9
```

## Part 1 (Vending Machine) Implementation

```
1 package edu.cpp.cs5800.VendingMachine.snacks;
2
3 public class Pepsi extends Snack {
4
5     public Pepsi(int quantity, double price) {
6         super("Pepsi", quantity, price);
7     }
8 }
9
```

## Part 1 (Vending Machine) Implementation

```
1 package edu.cpp.cs5800.VendingMachine.snacks;
2
3 public abstract class Snack {
4     private static int idIncrmnt = 1;
5     private int id;
6     private String name;
7     private int quantity;
8     private double price;
9
10    public Snack(String name, int quantity, double price) {
11        this.id = idIncrmnt++;
12        this.name = name;
13        this.quantity = quantity;
14        this.price = price;
15    }
16
17    public int getId() {
18        return id;
19    }
20
21    public String getName() {
22        return name;
23    }
24
25    public int getQuantity() {
26        return quantity;
27    }
28
29    public double getPrice() {
30        return price;
31    }
32
33    public String dispense() {
34        quantity--;
35        return name;
36    }
37
38    public static void resetId() {
39        idIncrmnt = 1;
40    }
41
42    @Override
43    public String toString() {
44        return "Snack{" +
45            "name='" + name + '\'' +
46            ", quantity=" + quantity +
47            ", price=" + price +
48            '}';
49    }
50 }
51
```

## Part 1 (Vending Machine) Implementation

```
1 package edu.cpp.cs5800.VendingMachine.snacks;
2
3 public class KitKat extends Snack {
4
5     public KitKat(int quantity, double price) {
6         super("KitKat", quantity, price);
7     }
8 }
9
```



## Part 1 (Vending Machine) Implementation

```
1 package edu.cpp.cs5800.VendingMachine.snacks;
2
3 public class Cheetos extends Snack {
4
5     public Cheetos(int quantity, double price) {
6         super("Cheetos", quantity, price);
7     }
8 }
9
```

## Part 1 (Vending Machine) Implementation

```
1 package edu.cpp.cs5800.VendingMachine.snacks;
2
3 public class Doritos extends Snack {
4
5     public Doritos(int quantity, double price) {
6         super("Doritos", quantity, price);
7     }
8 }
9
```

## Part 1 (Vending Machine) Implementation

```
1 package edu.cpp.cs5800.VendingMachine.snacks;
2
3 public class Snickers extends Snack {
4
5     public Snickers(int quantity, double price) {
6         super("Snickers", quantity, price);
7     }
8 }
9
```

## Part 1 (Vending Machine) Implementation

```
1 package edu.cpp.cs5800.VendingMachine.states;
2
3 import edu.cpp.cs5800.VendingMachine.VendingMachine;
4 import edu.cpp.cs5800.VendingMachine.snacks.Snack;
5
6 public class Idle extends StateOfVendingMachine {
7
8     public Idle(VendingMachine vendingMachine) {
9         super(vendingMachine);
10    }
11
12    @Override
13    public String selectSnack(int number) {
14        Snack snack = this.vendingMachine.getSnack(number);
15        String message = "You have selected: " + number + " (" + snack.getName() + ")";
16        System.out.println(message);
17        this.vendingMachine.setSelectedSnack(number);
18        this.vendingMachine.setState(new WaitingForMoney(this.vendingMachine));
19        return message;
20    }
21
22    @Override
23    public String insertMoney(double amount) {
24        String message = "Invalid request: Please first select a snack!";
25        System.out.println(message);
26        return message;
27    }
28
29    @Override
30    public String dispenseSnack() {
31        String message = "Invalid request: Please first select a snack!";
32        System.out.println("Invalid request: Please first select a snack!");
33        return message;
34    }
35
36    public String toString() {
37        return "Idle";
38    }
39 }
40
```

## Part 1 (Vending Machine) Implementation

```
1 package edu.cpp.cs5800.VendingMachine.states;
2
3 import edu.cpp.cs5800.VendingMachine.VendingMachine;
4
5 public class DispensingSnack extends StateOfVendingMachine {
6
7     public DispensingSnack(VendingMachine vendingMachine) {
8         super(vendingMachine);
9     }
10
11     @Override
12     public String selectSnack(int number) {
13         String message = "Invalid request: Currently dispensing snack!";
14         System.out.println("Invalid request: Currently dispensing snack!");
15         return message;
16     }
17
18     @Override
19     public String insertMoney(double amount) {
20         String message = "Invalid request: Currently dispensing snack!";
21         System.out.println("Invalid request: Currently dispensing snack!");
22         return message;
23     }
24
25     @Override
26     public String dispenseSnack() {
27         int selectedItem = this.vendingMachine.getSelectedSnack();
28         double amt = this.vendingMachine.getInsertedAmount();
29         System.out.print("Chain of responsibility: ");
30         this.vendingMachine.getSnackDispenseHandler().dispense(selectedItem, amt);
31         String message = "Completed transaction!";
32         System.out.println(message);
33         this.vendingMachine.setState(new Idle(this.vendingMachine));
34         return message;
35     }
36
37     public String toString() {
38         return "DispensingSnack";
39     }
40 }
41
```

## Part 1 (Vending Machine) Implementation

```
1 package edu.cpp.cs5800.VendingMachine.states;
2
3 import edu.cpp.cs5800.VendingMachine.VendingMachine;
4
5 public class WaitingForMoney extends StateOfVendingMachine {
6
7     public WaitingForMoney(VendingMachine vendingMachine) {
8         super(vendingMachine);
9     }
10
11     @Override
12     public String selectSnack(int number) {
13         String message = "Invalid request: Please inserted money!";
14         System.out.println(message);
15         return message;
16     }
17
18     @Override
19     public String insertMoney(double amount) {
20         String message = "You have inserted: $" + amount;
21         System.out.println(message);
22         this.vendingMachine.setInsertedAmount(amount);
23         this.vendingMachine.setState(new DispensingSnack(this.vendingMachine));
24         return message;
25     }
26
27     @Override
28     public String dispenseSnack() {
29         String message = "Invalid request: Please inserted money!";
30         System.out.println("Invalid request: Please inserted money!");
31         return message;
32     }
33
34     public String toString() {
35         return "WaitingForMoney";
36     }
37 }
38
```

## Part 1 (Vending Machine) Implementation

```
1 package edu.cpp.cs5800.VendingMachine.states;
2
3 import edu.cpp.cs5800.VendingMachine.VendingMachine;
4
5 public abstract class StateOfVendingMachine {
6     protected VendingMachine vendingMachine;
7
8     public StateOfVendingMachine(VendingMachine vendingMachine) {
9         this.vendingMachine = vendingMachine;
10    }
11
12    public abstract String selectSnack(int number);
13
14    public abstract String insertMoney(double amount);
15
16    public abstract String dispenseSnack();
17 }
18
```