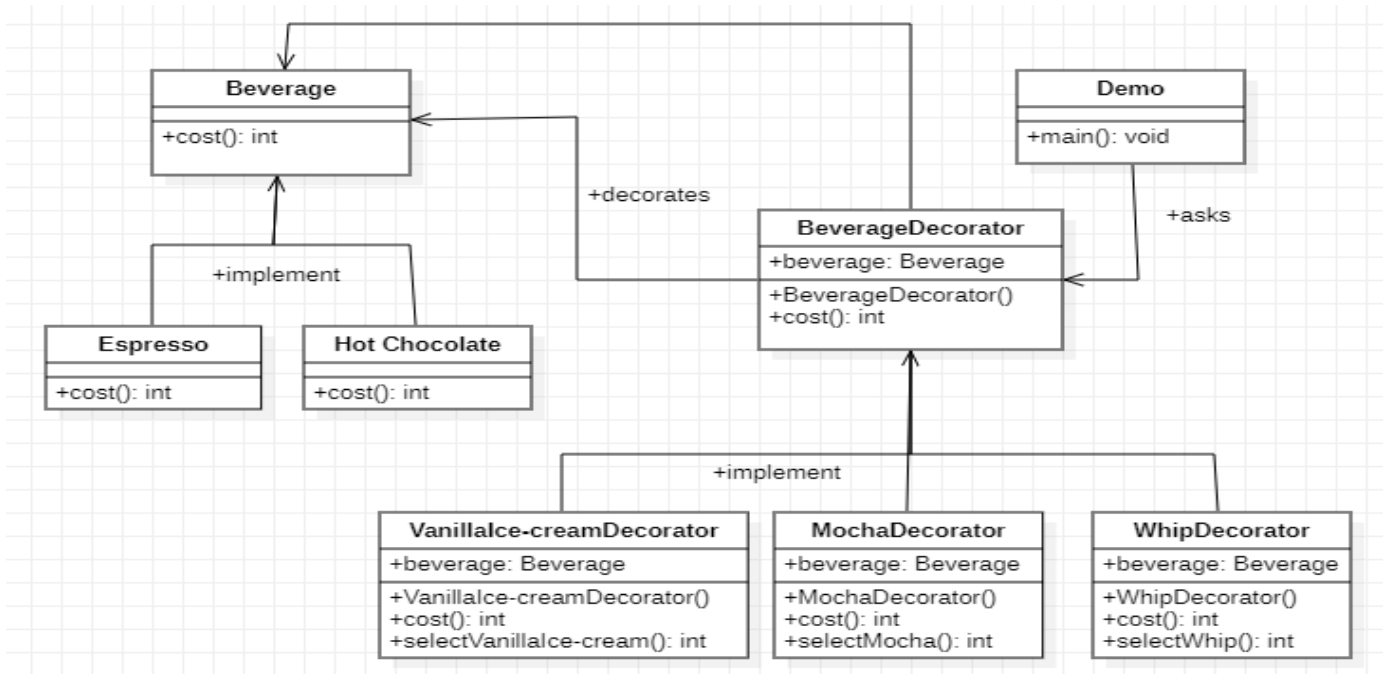


2. Suppose you are planning to start a side business of a Coffee Shop. To automate the order processing, you intend to seek help from the OO decorator pattern. Create a Beverage class and decorate it with condiments at run time and estimate the overall cost of Beverage at the end. For example, a customer wants Espresso with mocha and whip or he wants Hot Chocolate with Vanilla ice cream. Draw a class diagram for the given situation and implement it in Java. Also attach print outs.



Step 1: Beverage.java interface

```

Beverage.java
1 package Lab3_q2;
2
3 public interface Beverage {
4     int cost();
5 }
6
7

```

Step 2: Create concrete classes implementing the interface Beverage

```

Espresso.java
1 package Lab3_q2;
2
3 public class Espresso implements Beverage {
4
5     @Override
6     public int cost() {
7         int price = 100;
8         return price;
9     }
10
11 }
12

```

```

HotChocolate.java
1 package Lab3_q2;
2
3 public class HotChocolate implements Beverage {
4
5     @Override
6     public int cost() {
7         int price = 150;
8         return price;
9     }
10
11 }
12

```

Step 3: Create the abstract decorator class implementing the Beverage interface

```
*BeverageDecorator.java
1 package Lab3_q2;
2
3 public abstract class BeverageDecorator implements Beverage {
4     protected Beverage decoratedBeverage;
5     public BeverageDecorator(Beverage decoratedBeverage) {
6         this.decoratedBeverage = decoratedBeverage;
7     }
8     @Override
9     public int cost() {
10         return decoratedBeverage.cost();
11     }
12 }
```

Step 4: Create concrete class implementing the BeverageDecorator abstract class

```
*MochaDecorator.java
1 package Lab3_q2;
2
3 public class MochaDecorator extends BeverageDecorator {
4     public MochaDecorator(Beverage decoratedBeverage) {
5         super(decoratedBeverage);
6     }
7     public int cost() {
8         return decoratedBeverage.cost()+selectMocha(decoratedBeverage);
9     }
10    private int selectMocha(Beverage decoratedBeverage) {
11        int price=50;
12        System.out.println("Mocha cost:"+price);
13        return price;
14    }
15 }
```

```
*WhipDecorator.java
1 package Lab3_q2;
2
3 public class WhipDecorator extends BeverageDecorator {
4     public WhipDecorator(Beverage decoratedBeverage) {
5         super(decoratedBeverage);
6     }
7     public int cost() {
8         return decoratedBeverage.cost()+selectWhip(decoratedBeverage);
9     }
10    private int selectWhip(Beverage decoratedBeverage) {
11        int price=100;
12        System.out.println("Whip cost:"+price);
13        return price;
14    }
15 }
```

*VanillaIceCreamDecorator.java

```
1 package Lab3_q2;
2
3 public class VanillaIceCreamDecorator extends BeverageDecorator {
4     public VanillaIceCreamDecorator(Beverage decoratedBeverage) {
5         super(decoratedBeverage);
6     }
7     public int cost() {
8         return decoratedBeverage.cost()+selectVanillaIceCream(decoratedBeverage);
9     }
10    private int selectVanillaIceCream(Beverage decoratedBeverage) {
11        int price=80;
12        System.out.println("Vanilla Ice-cream cost:"+price);
13        return price;
14    }
15 }
```

*Demo.java

```
1 package Lab3_q2;
2
3 public class Demo {
4     public static void main(String[] args) {
5         Beverage espresso= new Espresso();
6         Beverage mochaEspresso = new MochaDecorator(new Espresso());
7         Beverage whipEspresso = new WhipDecorator(new Espresso());
8         Beverage mwEspresso = new WhipDecorator(new MochaDecorator(new Espresso()));
9         Beverage hotChocolate= new HotChocolate();
10        Beverage vanillaHotChocolate = new VanillaIceCreamDecorator(new Espresso());
11
12        System.out.println("Espresso cost:"+espresso.cost());
13        System.out.println("Espresso with Mocha cost:" + mochaEspresso.cost());
14        System.out.println("\n");
15        System.out.println("Espresso cost:"+espresso.cost());
16        System.out.println("Espresso with Whip cost:" + whipEspresso.cost());
17        System.out.println("\n");
18        System.out.println("Espresso cost:"+espresso.cost());
19        System.out.println("Espresso with Mocha & Whip cost:" + mwEspresso.cost());
20        System.out.println("\n");
21        System.out.println("Hot Chocolate cost:"+hotChocolate.cost());
22        System.out.println("Hot Chocolate with Vanilla Ice-cream cost:" + vanillaHotChocolate.cost());
23    }
24 }
25 }
```

Step 5: Create the main class demonstrating the decorator design pattern

Output:

```
Espresso cost:100
Mocha cost:50
Espresso with Mocha cost:150

Espresso cost:100
Whip cost:100
Espresso with Whip cost:200

Espresso cost:100
Mocha cost:50
Whip cost:100
Espresso with Mocha & Whip cost:250

Hot Chocolate cost:150
Vanilla Ice-cream cost:80
Hot Chocolate with Vanilla Ice-cream cost:180
```