```java
package com.restaurant;

import java.util.ArrayList;

/// Component interface
public interface Food {
    void viewInfo();
    void addTopping();
    ArrayList<String> getToppings();
    double getCost();
}
```

```java
package com.restaurant;

import com.restaurant.*;
import com.restaurant.burger.*;
import com.restaurant.customer.CustomerOrder;
import com.restaurant.fries.*;
import com.restaurant.hot_dog.*;
import com.restaurant.coffee.*;

public class Main {
    public static void main(String[] args) {
        Food burger = new OriginalBurger();
        Food coffee = new OriginalCoffee();
        Food fries = new OriginalFries();
        Food hotDog = new OriginalHotDog();
        double totalPrice;

        burger = new BurgerCheeseDecorator(burger);
        burger.addTopping();
        burger = new BurgerLettuceDecorator(burger);
        burger.addTopping();
        burger = new BurgerOnionDecorator(burger);
        burger.addTopping();
        burger = new BurgerPicklesDecorator(burger);
        burger.addTopping();



        coffee = new CoffeeCreamDecorator(coffee);
        coffee.addTopping();
        coffee = new CoffeeMilkDecorator(coffee);
        coffee.addTopping();
        coffee = new CoffeeSyrupDecorator(coffee);
        coffee.addTopping();

        fries = new FriesKetchupDecorator(fries);
        fries.addTopping();
        fries = new FriesSaltDecorator(fries);
        fries.addTopping();

        hotDog = new HotDogKetchupDecorator(hotDog);
```

```
42          hotDog.addTopping();
43          hotDog = new HotDogMustardDecorator(hotDog);
44          hotDog.addTopping();
45          hotDog = new HotDogRelishDecorator(hotDog);
46          hotDog.addTopping();
47
48          CustomerOrder customer = new CustomerOrder(
   159);
49          customer.addFood(burger);
50          customer.addFood(coffee);
51          customer.addFood(fries);
52          customer.addFood(hotDog);
53
54          customer.applyLoyaltyDiscount(true);
55
56          customer.viewOrder();
57
58          System.out.println("Customer loyalty points
   available: " + customer.getLoyaltyPoints());
59
60          totalPrice = customer.getTotalPrice();
61
62          System.out.println("Total price: " +
   totalPrice);
63          System.out.println("Discount applied: $" +
   customer.getDiscountApplied());
64          System.out.println("Remaining loyalty points
   : " + customer.getLoyaltyPoints());
65      }
66  }
67
```

```java
package com.restaurant.fries;

import com.restaurant.Food;

import java.util.ArrayList;

/// Concrete component
public class OriginalFries implements Food {
    @Override
    public double getCost(){
        return 1.20;
    }

    @Override
    public void addTopping(){}

    @Override
    public ArrayList<String> getToppings(){
        return new ArrayList<>();
    }

    @Override
    public void viewInfo(){
        System.out.println("An order of Fries.");
    }
}

```

```java
package com.restaurant.fries;

import com.restaurant.Food;

import java.util.ArrayList;

/// Abstract decorator
public abstract class FriesDecorator implements Food {
    protected Food fries;
    protected ArrayList<String> toppings;
    protected double cost;

    protected FriesDecorator(Food fries) {
        this.fries = fries;
        this.toppings = fries.getToppings();
        this.cost = fries.getCost();
    }

    // Equivalent to init() accessed by the client.
    public void addTopping(){
        toppings = getToppings();
        cost = getCost();
    }

    @Override
    public ArrayList<String> getToppings(){
        return toppings;
    }

    @Override
    public double getCost(){
        return cost;
    }

    @Override
    public void viewInfo() {
        System.out.print("An order of Fries. ");

        if(!toppings.isEmpty()) {
            System.out.println("With toppings: ");
```

```
41
42              for (String topping : toppings) {
43                  System.out.print(topping + "    ");
44              }
45          }
46
47          System.out.println("\nCost: $" + cost + "\n"
    );
48      }
49  }
50
```

```java
package com.restaurant.fries;

import com.restaurant.Food;

/// Concrete decorator
public class FriesSaltDecorator extends FriesDecorator {
    public FriesSaltDecorator(Food fries) {
        super(fries);
    }

    // Equivalent to init() accessed by the client.
    @Override
    public void addTopping(){
        getToppings().add("Salt, $0.00");
        cost += 0;  // For uniformity
    }
}
```

```java
package com.restaurant.fries;

import com.restaurant.Food;

/// Concrete decorator
public class FriesKetchupDecorator extends FriesDecorator {
    public FriesKetchupDecorator(Food fries) {
        super(fries);
    }

    // Equivalent to init() accessed by the client.
    @Override
    public void addTopping(){
        getToppings().add("Ketchup, $0.20");
        cost += 0.2;
    }
}
```

```java
package com.restaurant.burger;

import com.restaurant.Food;

import java.util.ArrayList;

/// Concrete component
public class OriginalBurger implements Food {
    @Override
    public double getCost(){
        return 4.15;
    }

    @Override
    public void addTopping(){}

    @Override
    public ArrayList<String> getToppings(){
        return new ArrayList<>();
    }

    @Override
    public void viewInfo(){
        System.out.println("An order of Burger.");
    }
}

```

```java
package com.restaurant.burger;

import com.restaurant.Food;

import java.util.ArrayList;

/// Abstract decorator
public abstract class BurgerDecorator implements Food {
    protected Food burger;
    protected ArrayList<String> toppings;
    protected double cost;

    protected BurgerDecorator(Food burger) {
        this.burger = burger;
        this.toppings = burger.getToppings();
        this.cost = burger.getCost();
    }

    // Equivalent to init() accessed by the client.
    public void addTopping(){
        toppings = getToppings();
        cost = getCost();
    }

    @Override
    public ArrayList<String> getToppings(){
        return toppings;
    }

    @Override
    public double getCost(){
        return cost;
    }

    @Override
    public void viewInfo() {
        System.out.print("An order of Burger. ");

        if(!toppings.isEmpty()) {
            System.out.println("With toppings: ");
```

```
41
42              for (String topping : toppings) {
43                  System.out.print(topping + "    ");
44              }
45          }
46
47          System.out.println("\nCost: $" + cost + "\n"
    );
48      }
49  }
50
```

```java
package com.restaurant.burger;

import com.restaurant.Food;

/// Concrete decorator
public class BurgerOnionDecorator extends BurgerDecorator {
    public BurgerOnionDecorator(Food burger) {
        super(burger);
    }

    // Equivalent to init() accessed by the client.
    @Override
    public void addTopping(){
        toppings.add("Onion, $0.20");
        cost += 0.2;
    }
}
```

```java
package com.restaurant.burger;

import com.restaurant.Food;

/// Concrete decorator
public class BurgerCheeseDecorator extends BurgerDecorator {
    public BurgerCheeseDecorator(Food burger) {
        super(burger);
    }

    // Equivalent to init() accessed by the client.
    @Override
    public void addTopping(){
        toppings.add("Cheese, $0.50");
        cost += 0.5;
    }
}
```

```java
package com.restaurant.burger;

import com.restaurant.Food;

/// Concrete decorator
public class BurgerLettuceDecorator extends BurgerDecorator {
    public BurgerLettuceDecorator(Food burger) {
        super(burger);
    }

    // Equivalent to init() accessed by the client.
    @Override
    public void addTopping(){
        toppings.add("Lettuce, $0.10");
        cost += 0.1;
    }
}
```

```java
package com.restaurant.burger;

import com.restaurant.Food;

/// Concrete decorator
public class BurgerPicklesDecorator extends BurgerDecorator {
    public BurgerPicklesDecorator(Food burger) {
        super(burger);
    }

    // Equivalent to init() accessed by the client.
    @Override
    public void addTopping(){
        toppings.add("Pickles, $0.30");
        cost += 0.3;
    }
}
```

```java
package com.restaurant.coffee;

import com.restaurant.Food;

import java.util.ArrayList;

/// Concrete component
public class OriginalCoffee implements Food {
    @Override
    public double getCost(){
        return 1.00;
    }

    @Override
    public void addTopping(){}

    @Override
    public ArrayList<String> getToppings(){
        return new ArrayList<>();
    }

    @Override
    public void viewInfo(){
        System.out.println("An order of Coffee.");
    }
}

```

```java
package com.restaurant.coffee;

import com.restaurant.Food;

import java.util.ArrayList;

/// Abstract decorator
public abstract class CoffeeDecorator implements Food {
    protected Food coffee;
    protected ArrayList<String> toppings;
    protected double cost;

    protected CoffeeDecorator(Food coffee) {
        this.coffee = coffee;
        this.toppings = coffee.getToppings();
        this.cost = coffee.getCost();
    }

    // Equivalent to init() accessed by the client.
    public void addTopping(){
        toppings = getToppings();
        cost = getCost();
    }

    @Override
    public ArrayList<String> getToppings(){
        return toppings;
    }

    @Override
    public double getCost(){
        return cost;
    }

    @Override
    public void viewInfo() {
        System.out.print("An order of Coffee. ");

        if(!toppings.isEmpty()) {
            System.out.println("With toppings: ");
```

```
41
42              for (String topping : toppings) {
43                  System.out.print(topping + "    ");
44              }
45          }
46
47          System.out.println("\nCost: $" + cost + "\n"
    );
48      }
49  }
50
```

```java
package com.restaurant.coffee;

import com.restaurant.Food;

/// Concrete decorator
public class CoffeeMilkDecorator extends
CoffeeDecorator {
    public CoffeeMilkDecorator(Food coffee) {
        super(coffee);
    }

    // Equivalent to init() accessed by the client.
    @Override
    public void addTopping(){
        getToppings().add("Milk, $1.20");
        cost = getCost() + 1.2;
    }
}
```

```java
package com.restaurant.coffee;

import com.restaurant.Food;

/// Concrete decorator
public class CoffeeCreamDecorator extends CoffeeDecorator {
    public CoffeeCreamDecorator(Food coffee) {
        super(coffee);
    }

    // Equivalent to init() accessed by the client.
    @Override
    public void addTopping(){
        toppings.add("Cream, $0.50");
        cost += 0.5;
    }
}
```

```java
package com.restaurant.coffee;

import com.restaurant.Food;

/// Concrete decorator
public class CoffeeSyrupDecorator extends CoffeeDecorator {
    public CoffeeSyrupDecorator(Food coffee) {
        super(coffee);
    }

    // Equivalent to init() accessed by the client.
    @Override
    public void addTopping(){
        getToppings().add("Syrup, $0.70");
        cost += 0.7;
    }
}
```

```java
package com.restaurant.hot_dog;

import com.restaurant.Food;

import java.util.ArrayList;

/// Concrete component
public class OriginalHotDog implements Food {
    @Override
    public double getCost(){
        return 3.50;
    }

    @Override
    public void addTopping(){}

    @Override
    public ArrayList<String> getToppings(){
        return new ArrayList<>();
    }

    @Override
    public void viewInfo(){
        System.out.println("An order of Hot Dog.");
    }
}

```

```java
package com.restaurant.hot_dog;

import com.restaurant.Food;

import java.util.ArrayList;

/// Abstract decorator
public abstract class HotDogDecorator implements Food {
    protected Food hotDog;
    protected ArrayList<String> toppings;
    protected double cost;

    protected HotDogDecorator(Food hotDog) {
        this.hotDog = hotDog;
        this.toppings = hotDog.getToppings();
        this.cost = hotDog.getCost();
    }

    // Equivalent to init() accessed by the client.
    public void addTopping(){
        toppings = getToppings();
        cost = getCost();
    }

    @Override
    public ArrayList<String> getToppings(){
        return toppings;
    }

    @Override
    public double getCost(){
        return cost;
    }

    @Override
    public void viewInfo() {
        System.out.print("An order of Hot Dog. ");

        if(!toppings.isEmpty()) {
            System.out.println("With toppings: ");
```

```
41
42              for (String topping : toppings) {
43                  System.out.print(topping + "    ");
44              }
45          }
46
47          System.out.println("\nCost: $" + cost + "\n"
    );
48      }
49  }
50
```

```java
package com.restaurant.hot_dog;

import com.restaurant.Food;

/// Concrete decorator
public class HotDogRelishDecorator extends HotDogDecorator{
    public HotDogRelishDecorator(Food hotDog) {
        super(hotDog);
    }

    // Equivalent to init() accessed by the client.
    @Override
    public void addTopping(){
        getToppings().add("Relish, $0.25");
        cost += 0.25;
    }
}
```

```java
package com.restaurant.hot_dog;

import com.restaurant.Food;

/// Concrete decorator
public class HotDogKetchupDecorator extends HotDogDecorator{
    public HotDogKetchupDecorator(Food hotDog) {
        super(hotDog);
    }

    // Equivalent to init() accessed by the client.
    @Override
    public void addTopping(){
        getToppings().add("Ketchup, $0.20");
        cost += 0.2;
    }
}
```

```java
package com.restaurant.hot_dog;

import com.restaurant.Food;

/// Concrete decorator
public class HotDogMustardDecorator extends HotDogDecorator{
    public HotDogMustardDecorator(Food hotDog) {
        super(hotDog);
    }

    // Equivalent to init() accessed by the client.
    @Override
    public void addTopping(){
        getToppings().add("Mustard, $0.20");
        cost += 0.2;
    }
}
```

```java
package com.restaurant.customer;

public class Loyalty {
    // loyalty points
    private int points;

    Loyalty() {}

    public void setPoints(int points) {
        this.points = points;
    }

    public int getPoints() {
        return points;
    }

    /// The customer gets 5 dollars off for every 100 points
    /// Very simplified model
    public double getDiscount(){
        int discountMultiplier = points / 100;
        points = points % 100;

        return (double)discountMultiplier * 5;
    }
}

```

```java
package com.restaurant.customer;

import com.restaurant.Food;

import java.util.ArrayList;

public class CustomerOrder {
    ArrayList<Food> foods;
    Boolean discountUsage = false;
    Loyalty loyalty;
    double discount = 0;

    /// The argument here is customer's loyalty points.
    /// This stat is the only variable likely to be static.
    public CustomerOrder(int points) {
        foods = new ArrayList<>();
        loyalty = new Loyalty();
        loyalty.setPoints(points);
    }

    public void addFood(Food food) {
        foods.add(food);
    }

    public void applyLoyaltyDiscount(Boolean discountUsage) {
        this.discountUsage = discountUsage;
    }

    public double getTotalPrice(){
        double total = 0;

        for(Food food : foods)
            total += food.getCost();

        if(discountUsage) {
            discount = loyalty.getDiscount();
            total = total - discount;
        }
```

```
39
40          return total;
41      }
42
43      public void viewOrder(){
44          for(Food food : foods)
45              food.viewInfo();
46      }
47
48      public int getLoyaltyPoints(){
49          return loyalty.getPoints();
50      }
51
52      public double getDiscountApplied(){
53          return discount;
54      }
55  }
```