SOLID Documentation

ARJUN S

January 2024

1 LISKOV SUBSTITUTION PRINCIPLE

Base class:

```
1
2 public class FoodRecommendation {
3 public List<String> recommendFood() {
4     return List.of();
5 }
6 }
```

Child classes :

```
1
2 public class UserBasedFoodRecommendation extends
    FoodRecommendation {
3 @Override
4 public
5 List<String> recommendFood() {
6     return List.of("Pizza", "Burger", "Salad");
7 }
```

```
1
2 public class AIModelBasedFoodRecommendation extends
    UserBasedFoodRecommendation {
3 //using function of the  UserBasedFoodRecommendation
4 }
```

```
1
2 public class NewRecommendation extends
    FoodRecommendation { //for new users to avoid cold
    start problem
3 @Override
4 public List<String> recommendFood() {
5    System.out.println("This is a new method specific
       to NewRecommendation.");
6     return List.of("New Food 1", "New Food 2", "New
       Food 3");
7 }
8 }
```

2 INTERFACE SEGREGATION PRINCIPLE

Interface:

```
1
2
3 public interface BasicFoodBooking {
4  void bookFood(String foodItem, int quantity, Date
     deliveryDate);
5
6  //void bookSpecialFood(String specialFoodItem, int
     quantity, Date deliveryDate);
7  //If i add bookSpecialFood function also here it
     violates ISP.
8 }
```

```
1 import java.util.Date;
2
3 public
4 interface VIPFoodBooking {
5     void bookSpecialFood(String specialFoodItem, int
        quantity, Date deliveryDate);
6 }
```

Implementation classes

```java
import java.util.Date;

public
class RegularFoodBooking implements BasicFoodBooking {
    @Override
    public void bookFood(String foodItem, int
        quantity, Date deliveryDate) {
         System.out.println("Regular␣user␣booked␣" +
            quantity + "␣" + foodItem + "(s)␣for␣
            delivery␣on␣" + deliveryDate);
    }
}
```

```java
import java.util.Date;

public
class VIPFoodBookingImpl implements VIPFoodBooking {
    @Override
    public void bookSpecialFood(String
        specialFoodItem, int quantity, Date
        deliveryDate) {

         System.out.println("VIP␣user␣booked␣" +
            quantity + "␣" + specialFoodItem + "(s)␣
            for␣delivery␣on␣" + deliveryDate);
    }
}
```

3 DEPENDENCY INVERSION PRINCIPLE

INTERFACE

```

import java.util.Date;

public

interface FoodBooking {
 void bookFood(String foodItem, int quantity, Date
    deliveryDate);
}
```

```

import java.util.Date;

public
class RegularFoodBooking implements FoodBooking {
    @Override
    public void bookFood(String foodItem, int
        quantity, Date deliveryDate) {

        System.out.println("Regular user booked " +
            quantity + " " + foodItem + "(s) for
            delivery on " + deliveryDate);
    }
}
```

```

import java.util.Date;

public
class VIPFoodBookingImpl implements FoodBooking {
    @Override
    public void bookFood(String foodItem, int
        quantity, Date deliveryDate) {

        System.out.println("VIP user booked " +
            quantity + " " + foodItem + "(s) for
            delivery on " + deliveryDate);
    }
}
```

Main page

```
import java.util.Date;

import com.ilp.service.*;

public class FoodBookingUtility {

    public static void main(String[] args) {

        FoodBooking regularBooking = new
            RegularFoodBooking();
        regularBooking.bookFood("Pizza", 2, new
            Date());

        FoodBooking vipBooking = new
            VIPFoodBookingImpl();
        vipBooking.bookFood("Exclusive␣Meal", 1, new
            Date());

//         RegularFoodBooking regularFoodBooking = new
    RegularFoodBooking();
//         regularFoodBooking.bookFood("Pizza", 2, new
    Date());
        // this code violates the DIP
    }
}
```

4 OPEN CLOSE PRINCIPLE

Interface

```

  package com.ilp.service;

import java.util.Date;

public interface FoodBooking {
 void bookFood(String foodItem, int quantity, Date
    deliveryDate);

 //public void VegBookFood(String foodItem, int
    quantity, Date deliveryDate) {}

//public void NonVegBookFood(String foodItem, int
    quantity, Date deliveryDate) {}
```

```
}
```

Implementation of interface

```

import java.util.Date;

public class NonVegFoodBooking implements FoodBooking
    {
 @Override
 public void bookFood(String foodItem, int quantity,
     Date deliveryDate) {

     System.out.println("Non veg Food Booking: " +
             "Item: " + foodItem +
             ", Quantity: " + quantity +
             ", Delivery Date: " + deliveryDate);
 }
}
```

```
package com.ilp.service;

import java.util.Date;

public class VegFoodBooking implements FoodBooking {
 @Override
 public void bookFood(String foodItem, int quantity,
     Date deliveryDate) {
     System.out.println("Regular Food Booking: " +
             "Item: " + foodItem +
             ", Quantity: " + quantity +
             ", Delivery Date: " + deliveryDate);
 }

// public void VegBookFood(String foodItem, int
    quantity, Date deliveryDate) {
//     System.out.println("Regular Food Booking: " +
//             "Item: " + foodItem +
//             ", Quantity: " + quantity +
//             ", Delivery Date: " + deliveryDate);
// }
//
// public void NonVegBookFood(String foodItem, int
    quantity, Date deliveryDate) {
//     System.out.println("Regular Food Booking: " +
//             "Item: " + foodItem +
```

```
//                ", Quantity: " + quantity +
//                ", Delivery Date: " + deliveryDate);
// }

 //violates Open Close principle
}
```

5 SINGLE RESPONSIBIILITY PRINCIPLE

All the above programs follows Single Responsibility principle.