Laiba Binte Tahir

FA21-BSE-019

Lab Assignment-01

Question 1: How does the following code violates "Single responsibility Principle"? Also Write the correct code.

class Marker{
String name;
String color;
int year;
int price;
 	
public Marker(String name, String color, int year, int price) {
this.name = name;
this.color = color;
this.year = year;
this.price = price;
}
	class Invoice{
	private Marker marker;
	private int quantity;
	
	public Invoice(Marker marker, int quantity) {
	this.marker = marker;
	this.quantity = quantity;
	}
	public int calculateTotal(){
	int price = ((marker.price)*this.quantity);
	return price;
	}
	public void printInvoice(){
	// print the invoice
	}
	public void saveToDB(){
	//save the data into DB
	}
	}

Violation:

A class only handles one responsibility in a single responsibility pattern so in above code we should make:

- InvoiceClass to handle only calculations.
- InvoicePrinter Class to handle only printing work.
- InvoiceRepository To handle database thus saving invoices to DB.

1. Invoice Class

```java
package org.example;

class Invoice {
    private Marker marker;
    private int quantity;

    public Invoice(Marker marker, int quantity) {
        this.marker = marker;
        this.quantity = quantity;
    }

    public int calculateTotal() {
        int price = (marker.price * quantity);
        return price;
    }
}
```

2. MarkerClass

```java
package org.example;

class Marker {
    String name;
    String color;
    int year;
    int price;

    public Marker(String name, String color, int year, int price) {
        this.name = name;
        this.color = color;
        this.year = year;
        this.price = price;
    }
}
```

3. InvoicePrinter Class

```java
package org.example;

class InvoicePrinter {
    public void printInvoice(Invoice invoice) {
        System.out.println("Invoice total: " + invoice.calculateTotal());
    }
}
```

4.InvoiceRegistory

```java
package org.example;

class InvoiceRepository {
    public void saveToDB(Invoice invoice) {
        System.out.println("Invoice saved to database.");
    }
}
```

5. MainClass

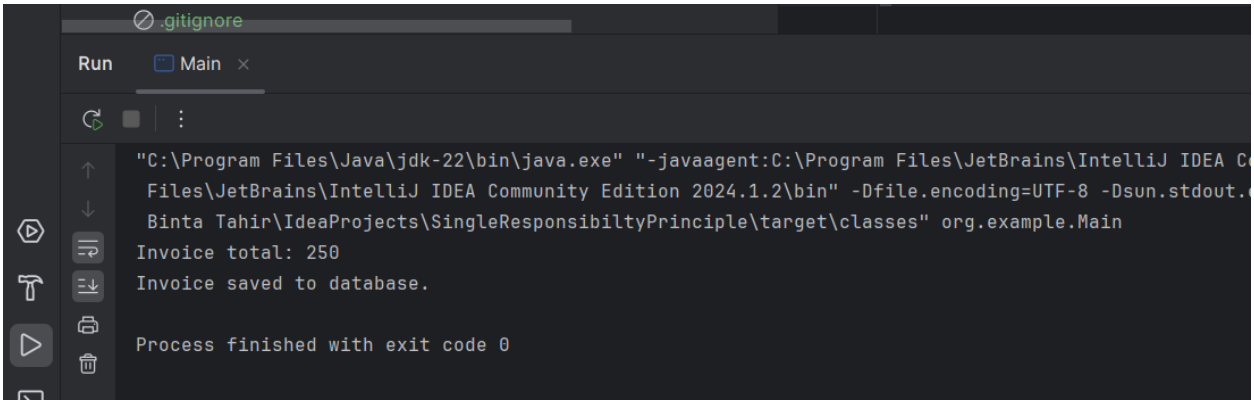
```java
package org.example;

To Run code, press Shift F10 or click the ▷ icon in the gutter.
public class Main {
    public static void main(String[] args) {
        Marker marker = new Marker( name: "MarkerBrand",  color: "Black",  year: 2024,  price: 50);

        Invoice invoice = new Invoice(marker,  quantity: 5);
        InvoicePrinter printer = new InvoicePrinter();
        printer.printInvoice(invoice);

        InvoiceRepository repository = new InvoiceRepository();
        repository.saveToDB(invoice);
    }
}
```

6. Output



Question 2: Write the correct code by applying Liskov substitution principle.

```
public class Bird {
    public void fly() {
        // Flying logic
    }
}

public class Ostrich extends Bird {
    // Ostriches cannot fly, but they inherit the
fly method
}
```

The superclass can easily substitute with other classes without effecting implantation. In above scenario we must make:

- BirdClass without Fly() method
- Make another flyingbird and orchid classes with appropriate methods.

BirdClass

```java
package org.example;

public abstract class Bird {
    public abstract void move();
}
```

<u>FlyingBirdClass</u>

```java
package org.example;

class FlyingBird extends Bird {
    @Override
    public void move() {
        fly();
    }

    public void fly() {
        System.out.println("Flying");
    }
}
```

OrchidClass

```java
package org.example;

// Ostrich Class (Non-Flying Bird)
class Ostrich extends Bird {
    @Override
    public void move() {
        run();
    }

    public void run() {
        System.out.println("Running");
    }
}
```

MainClass

```java
package org.example;
public class Main {
    public static void main(String[] args) {
        Bird sparrow = new FlyingBird();
        sparrow.move();

        Bird ostrich = new Ostrich();
        ostrich.move();
    }
}
```

Output:

```
Run    Main
"C:\Program Files\Java\jdk-22\bin\java.e
 Files\JetBrains\IntelliJ IDEA Community
 Binta Tahir\IdeaProjects\LiskovSubstitu
Flying
Running

Process finished with exit code 0
```