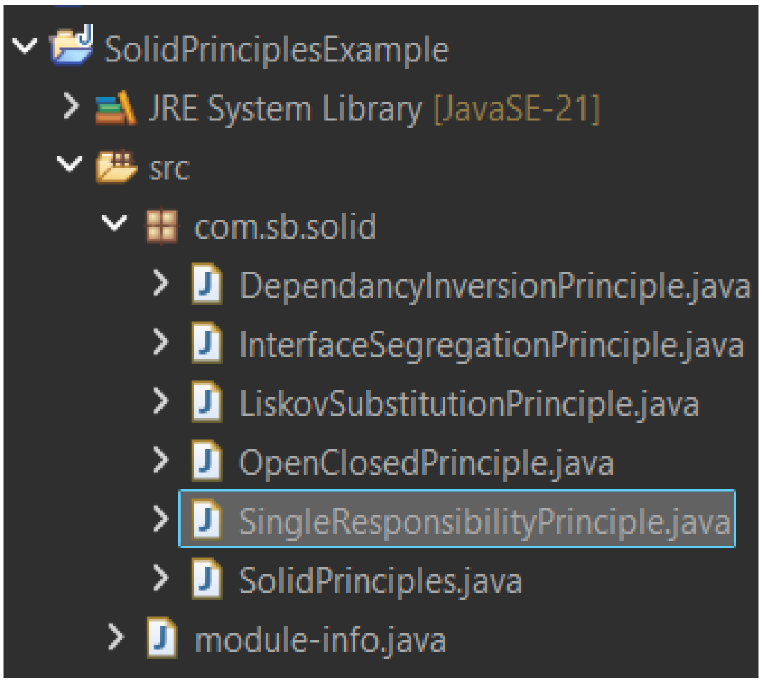
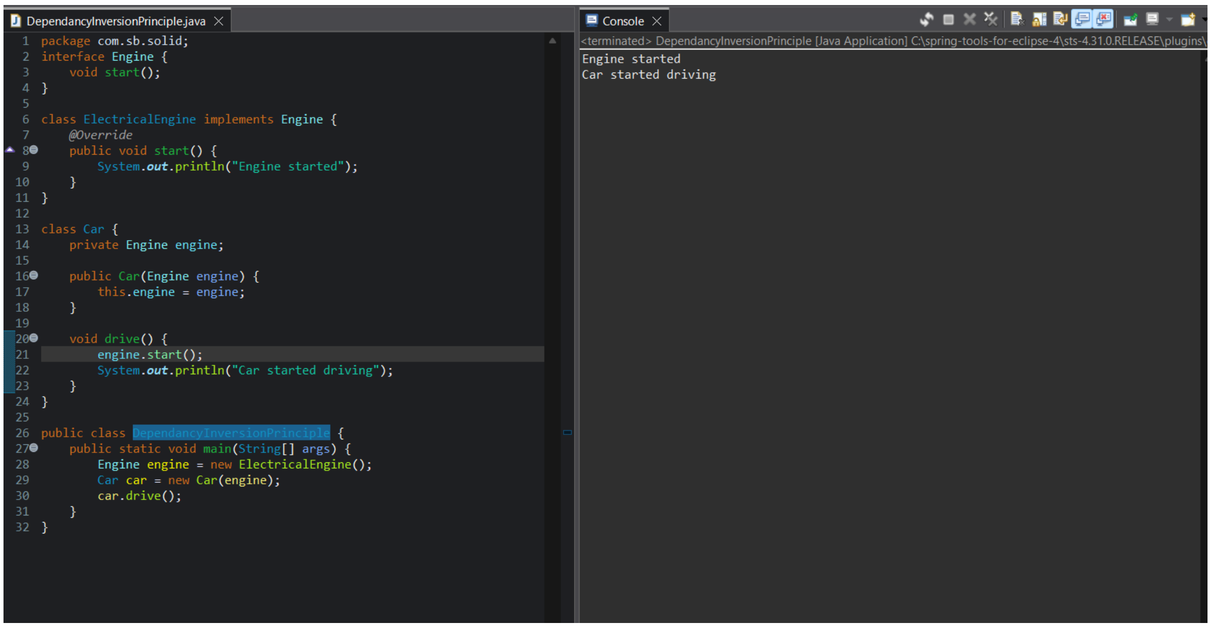
Solid Principles:

Project Structure->



```
SolidPrinciplesExample
  JRE System Library [JavaSE-21]
  src
    com.sb.solid
      DependancyInversionPrinciple.java
      InterfaceSegregationPrinciple.java
      LiskovSubstitutionPrinciple.java
      OpenClosedPrinciple.java
      SingleResponsibilityPrinciple.java
      SolidPrinciples.java
    module-info.java
```

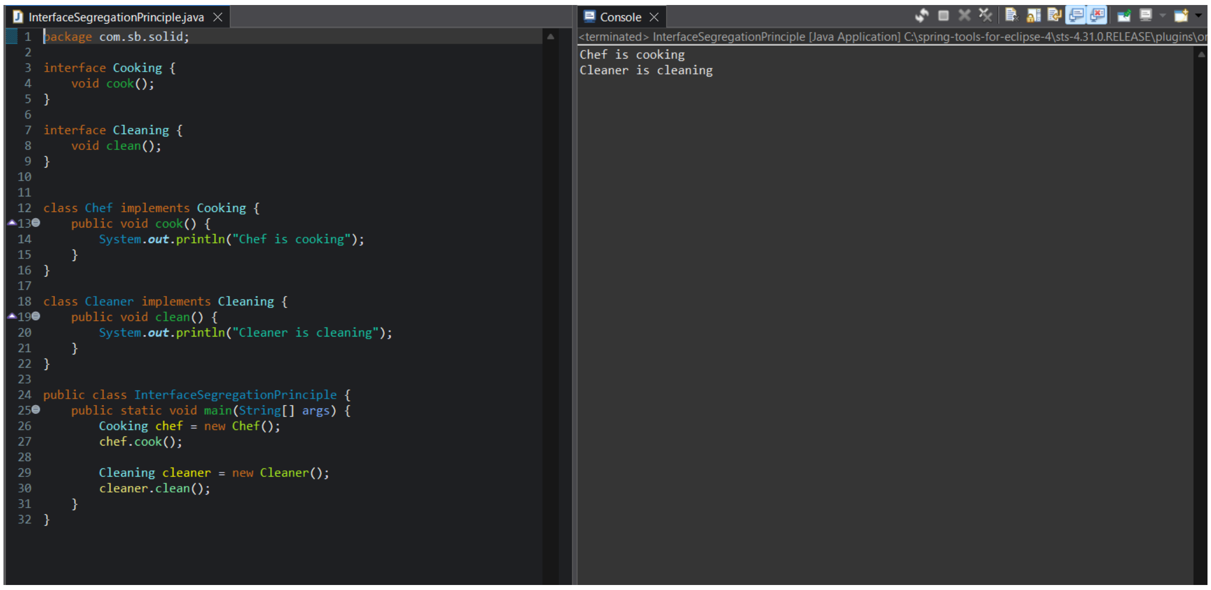
Code and Their Outputs given below:


DependancyInversionPrinciple.java

```java
package com.sb.solid;
interface Engine {
    void start();
}

class ElectricalEngine implements Engine {
    @Override
    public void start() {
        System.out.println("Engine started");
    }
}

class Car {
    private Engine engine;

    public Car(Engine engine) {
        this.engine = engine;
    }

    void drive() {
        engine.start();
        System.out.println("Car started driving");
    }
}

public class DependancyInversionPrinciple {
    public static void main(String[] args) {
        Engine engine = new ElectricalEngine();
        Car car = new Car(engine);
        car.drive();
    }
}
```

Console

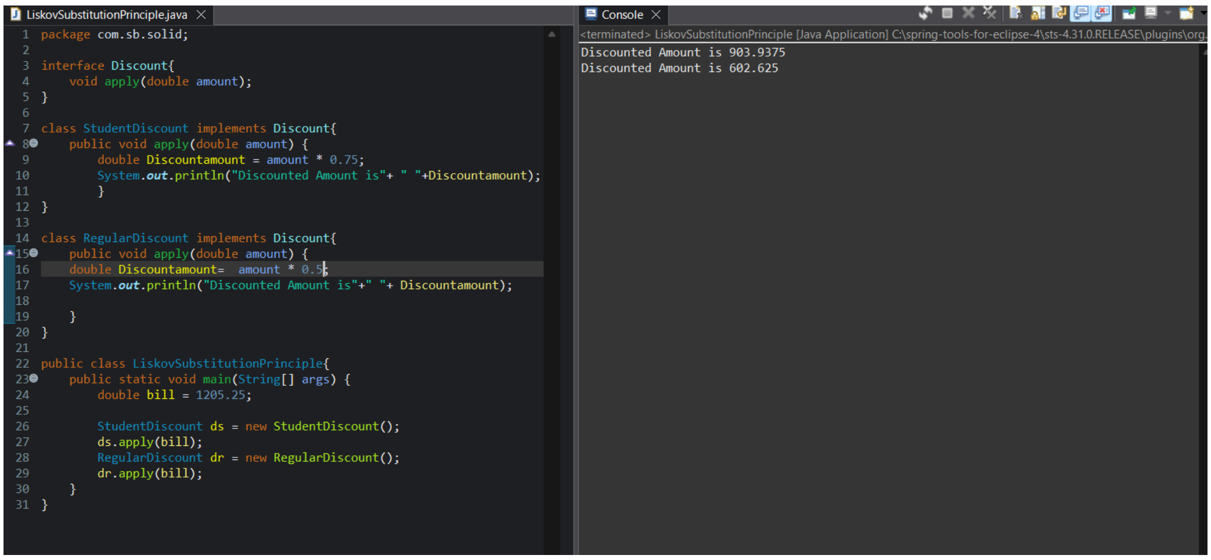
```
<terminated> DependancyInversionPrinciple [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins\
Engine started
Car started driving
```


InterfaceSegregationPrinciple.java

```java
package com.sb.solid;

interface Cooking {
    void cook();
}

interface Cleaning {
    void clean();
}


class Chef implements Cooking {
    public void cook() {
        System.out.println("Chef is cooking");
    }
}

class Cleaner implements Cleaning {
    public void clean() {
        System.out.println("Cleaner is cleaning");
    }
}

public class InterfaceSegregationPrinciple {
    public static void main(String[] args) {
        Cooking chef = new Chef();
        chef.cook();

        Cleaning cleaner = new Cleaner();
        cleaner.clean();
    }
}
```

Console

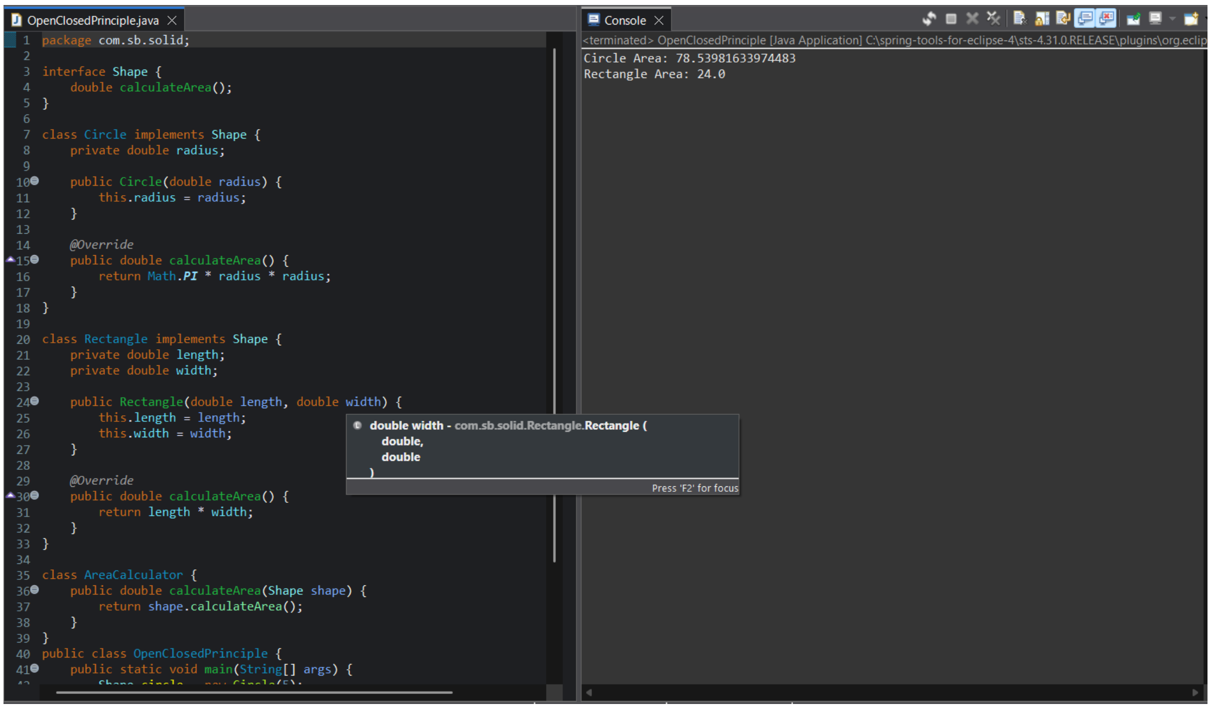
```
<terminated> InterfaceSegregationPrinciple [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins\or
Chef is cooking
Cleaner is cleaning
```


LiskovSubstitutionPrinciple.java

```java
package com.sb.solid;

interface Discount{
    void apply(double amount);
}

class StudentDiscount implements Discount{
    public void apply(double amount) {
        double Discountamount = amount * 0.75;
        System.out.println("Discounted Amount is"+ " "+Discountamount);
        }
}

class RegularDiscount implements Discount{
    public void apply(double amount) {
    double Discountamount=  amount * 0.5;
    System.out.println("Discounted Amount is"+" "+ Discountamount);

    }
}

public class LiskovSubstitutionPrinciple{
    public static void main(String[] args) {
        double bill = 1205.25;

        StudentDiscount ds = new StudentDiscount();
        ds.apply(bill);
        RegularDiscount dr = new RegularDiscount();
        dr.apply(bill);
    }
}
```

Console

```
<terminated> LiskovSubstitutionPrinciple [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins\org.
Discounted Amount is 903.9375
Discounted Amount is 602.625
```



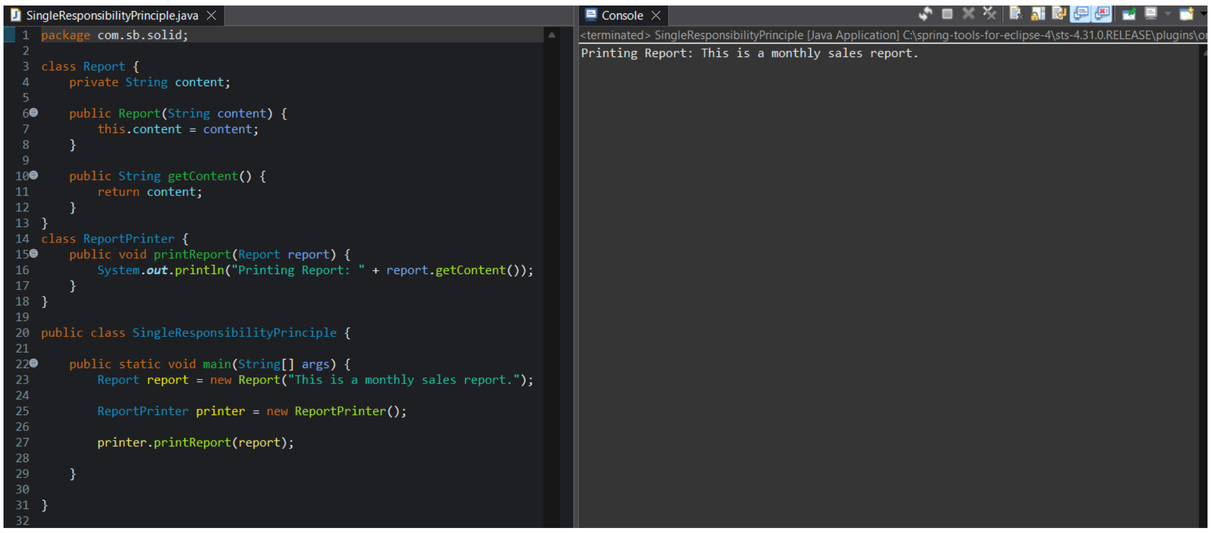
OpenClosedPrinciple.java

```java
package com.sb.solid;

interface Shape {
    double calculateArea();
}

class Circle implements Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double calculateArea() {
        return Math.PI * radius * radius;
    }
}

class Rectangle implements Shape {
    private double length;
    private double width;

    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    @Override
    public double calculateArea() {
        return length * width;
    }
}

class AreaCalculator {
    public double calculateArea(Shape shape) {
        return shape.calculateArea();
    }
}
public class OpenClosedPrinciple {
    public static void main(String[] args) {
        Shape circle = new Circle(5);
```

```
double width - com.sb.solid.Rectangle.Rectangle(
    double,
    double
)
Press 'F2' for focus
```

Console

```
<terminated> OpenClosedPrinciple [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins\org.eclip
Circle Area: 78.53981633974483
Rectangle Area: 24.0
```

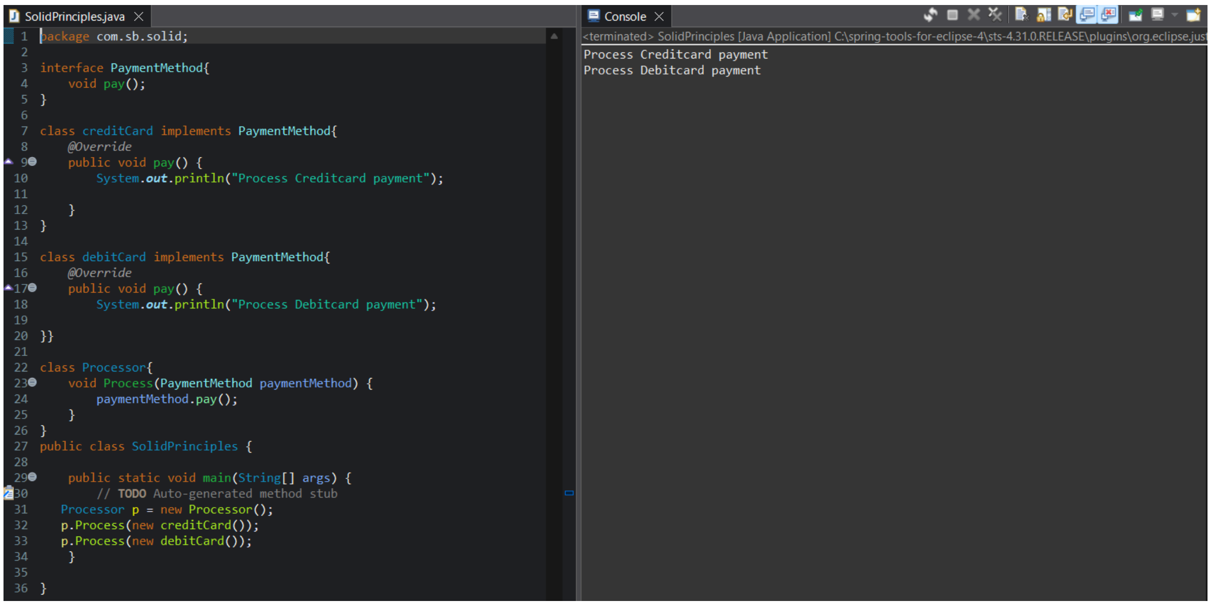


SingleResponsibilityPrinciple.java

```java
package com.sb.solid;

class Report {
    private String content;

    public Report(String content) {
        this.content = content;
    }

    public String getContent() {
        return content;
    }
}
class ReportPrinter {
    public void printReport(Report report) {
        System.out.println("Printing Report: " + report.getContent());
    }
}

public class SingleResponsibilityPrinciple {

    public static void main(String[] args) {
        Report report = new Report("This is a monthly sales report.");

        ReportPrinter printer = new ReportPrinter();

        printer.printReport(report);

    }

}
```

Console

```
<terminated> SingleResponsibilityPrinciple [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins\o
Printing Report: This is a monthly sales report.
```



SolidPrinciples.java

```java
package com.sb.solid;

interface PaymentMethod{
    void pay();
}

class creditCard implements PaymentMethod{
    @Override
    public void pay() {
        System.out.println("Process Creditcard payment");

    }
}

class debitCard implements PaymentMethod{
    @Override
    public void pay() {
        System.out.println("Process Debitcard payment");

}}

class Processor{
    void Process(PaymentMethod paymentMethod) {
        paymentMethod.pay();
    }
}
public class SolidPrinciples {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
   Processor p = new Processor();
   p.Process(new creditCard());
   p.Process(new debitCard());
    }

}
```

Console

<terminated> SolidPrinciples [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins\org.eclipse.just

```
Process Creditcard payment
Process Debitcard payment
```