**Module 1 – SOLID Principles (Java Examples with All-in-One Classes)  
  
NOTE : Executed all programs in online java compiler**

**1. Single Responsibility Principle (SRP)**

public class Student {

private String name;

public Student(String name) {

this.name = name;

}

public String getName() {

return name;

}

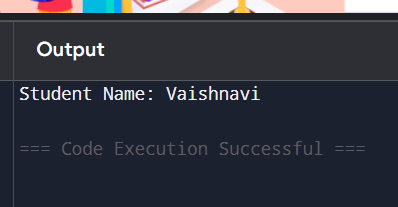
public static void main(String[] args) {

Student student = new Student("Vaishnavi");

System.out.println("Student Name: " + student.getName());

}

}

**OUTPUT**  
****

**2. Open/Closed Principle (OCP)**

abstract class Shape {

abstract double area();

}

class Circle extends Shape {

double radius;

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

double area() { return Math.PI \* radius \* radius; }

}

class Rectangle extends Shape {

double length, breadth;

public Rectangle(double l, double b) { this.length = l; this.breadth = b; }

double area() { return length \* breadth; }

}

public class Main {

public static void main(String[] args) {

Shape circle = new Circle(5);

Shape rectangle = new Rectangle(4, 6);

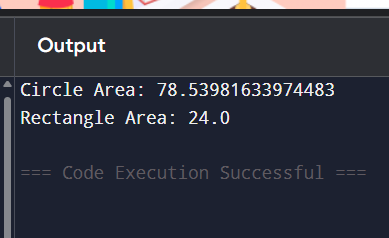
System.out.println("Circle Area: " + circle.area());

System.out.println("Rectangle Area: " + rectangle.area());

}

}

OUTPUT



**3. Liskov Substitution Principle (LSP)**interface Bird {

void makeSound();

}

class Sparrow implements Bird {

public void makeSound() {

System.out.println("Sparrow chirping");

}

}

class Crow implements Bird {

public void makeSound() {

System.out.println("Crow cawing");

}

}

public class Main {

public static void main(String[] args) {

Bird bird1 = new Sparrow();

Bird bird2 = new Crow();

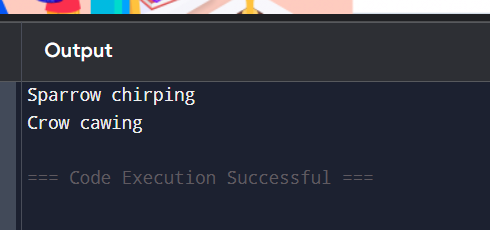
bird1.makeSound();

bird2.makeSound();

}

}

**OUTPUT**

****

**4. Interface Segregation Principle (ISP)**public void print() {

System.out.println("Printing document");

}

public void scan() {

System.out.println("Scanning document");

}

}

class SimplePrinter implements Printer {

public void print() {

System.out.println("Only printing");

}

}

public class Main {

public static void main(String[] args) {

Printer printer = new AllInOnePrinter();

printer.print();

Scanner scanner = new AllInOnePrinter();

scanner.scan();

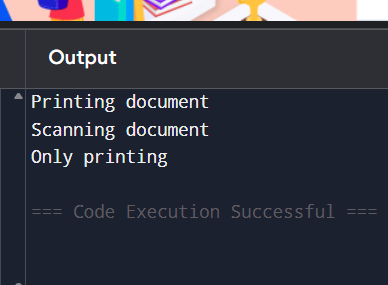
Printer simplePrinter = new SimplePrinter();

simplePrinter.print();

}

}

OUTPUT



**5. Dependency Inversion Principle (DIP)**

interface Keyboard {

void type();

}

class WiredKeyboard implements Keyboard {

public void type() {

System.out.println("Typing on wired keyboard");

}

}

class Computer {

private Keyboard keyboard;

public Computer(Keyboard k) {

this.keyboard = k;

}

public void typeSomething() {

keyboard.type();

}

}

public class Main {

public static void main(String[] args) {

Keyboard keyboard = new WiredKeyboard();

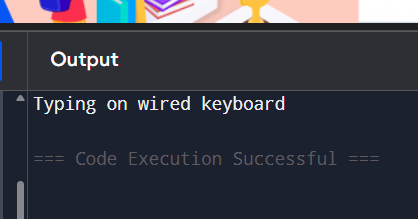
Computer comp = new Computer(keyboard);

comp.typeSomething();

}

}

**OUTPUT**

****