**Day 20 – July 25th**

**TASK 1 – SRP Violation**

public class Customer {

    String name;

    String custID;

    public Customer(String name, String custID) {

        this.name = name;

        this.custID = custID;

    }

    public String getName() {

        return name;

    }

    public String getCustID {

           return custID;

    }

    public void saveData() {

        try{

            FileWriter fw = new FileWriter(name+".txt");

            fw.write("the customer name is "+ name + "\t");

            fw.write("the customer id is "+ custID + "\t");

            System.out.println("the data is saved in

the file with your name");

        }catch(IOException ex) {

            ex.printStacktrace();

        }

    }

    psvm( String[] args) {

         Customer cobj = new Customer(" prasunamba" , "C001");

        cobj.saveData();

    }

}

**Notes:**

**SRP:** *A class should have only one reason to change. Each class should do only one thing.*

Here, Customer class is doing 2 things:

|  |
| --- |
| It stores customer data (name, ID). This is fine. |

|  |  |
| --- | --- |
| **File Handling (Persistence)** | It also handles writing customer data to a file. |

**TASK 2 – Implementation of SRP**

Separate file-saving logic into another class:

// 1. Customer class – only holds data

public class Customer {

private String name;

private String custID;

public Customer(String name, String custID) {

this.name = name;

this.custID = custID;

}

public String getName() { return name; }

public String getCustID() { return custID; }

}

// 2. DataSaver class – handles persistence

import java.io.FileWriter;

import java.io.IOException;

public class CustomerDataSaver {

public void save(Customer customer) {

try (FileWriter fw = new FileWriter(customer.getName() + ".txt")) {

fw.write("Customer Name: " + customer.getName() + "\t");

fw.write("Customer ID: " + customer.getCustID() + "\t");

System.out.println("Data saved to file.");

} catch (IOException ex) {

ex.printStackTrace();

}

}

}

// 3. Main class

public class Main {

public static void main(String[] args) {

Customer c = new Customer("Prasunamba", "C001");

CustomerDataSaver saver = new CustomerDataSaver();

saver.save(c);

}

}

**Notes:**

* If you change the **customer**, only Customer changes.
* If you change the **storage logic**, only CustomerDataSaver changes.

**-------------------------------------------------------------------------------------------------------------------------------**

**TASK3 –** The below is violating SRP complete it and also. please implement the SRP principle and rewrite the code.

// srp violation

public class Book {

    private String title;

    private String author;

    private double price;

    public Book(String title, String author, double price) {

        this.title = title;

        this.author = author;

        this.price = price;

    }

    public String getFormattedTitle() {

        return "Title: " + title.toUpperCase();

    }

    public double calculateDiscountedPrice(double discountPercentage) {

        return price \* (1 - discountPercentage);

    }

    // ... other methods for book details

}

class Book {  
 private String title;  
 private String author;  
 private double price;  
  
 public Book (String title, String author, double price) {  
 this.title = title;  
 this.author = author;  
 this.price = price;  
 }  
  
 // Getters only (no setters unless needed)  
 public String getTitle() {  
 return title;  
 }  
  
 public String getAuthor() {  
 return author;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
}  
// Class: Responsible for formatting book details only  
public class Day20\_SRP1 {  
 public String formatTitle(Book book) {  
 return "Title: " + book.getTitle().toUpperCase();  
 }  
  
 public String formatAuthor(Book book) {  
 return "Author: " + book.getAuthor();  
 }  
  
 public String formatDetails(Book book) {  
 return formatTitle(book) + "\n" + formatAuthor(book) + "\nPrice: ₹" + book.getPrice();  
 }  
}  
// Class: Responsible for price calculation only  
class PriceCalculator {  
 public double calculateDiscountedPrice(Book book, double discountPercentage) {  
 return book.getPrice() \* (1 - discountPercentage);  
 }  
}  
class Book1  
{  
 public static void main(String[] args) {  
 Book book = new Book("Clean Code", "Robert C. Martin", 500);  
  
 Day20\_SRP1 formatter = new Day20\_SRP1();  
 PriceCalculator calculator = new PriceCalculator();  
  
 System.*out*.println(formatter.formatDetails(book));  
  
 double discountedPrice = calculator.calculateDiscountedPrice(book, 0.10); // 10% discount  
 System.*out*.println("Discounted Price: ₹" + discountedPrice);  
 }  
}

**Output**

Title: CLEAN CODE

Author: Robert C. Martin

Price: ₹500.0

Discounted Price: ₹450.0

**TASK 4**

class Employee {

    private String name;

    private String email;

    private double salary;

    // Methods related to employee data

    // Method to generate PDF report

    public void generatePdfReport() {

        // Code to generate PDF report

    }

    // Method to send email

    public void sendEmail() {

        // Code to send email

    }

}

In the above example code, the Employee class violates the SRP because it has multiple responsibilities: managing employee data, generating PDF reports, and sending emails. These responsibilities are not cohesive and may change for different reasons.

class Employee {  
 private String name;  
 private String email;  
 private double salary;  
  
 public Employee(String name, String email, double salary) {  
 this.name = name;  
 this.email = email;  
 this.salary = salary;  
 }  
  
 // Getters  
 public String getName() {  
 return name;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public double getSalary() {  
 return salary;  
 }  
}  
class PdfReportGenerator {  
 public void generate(Employee e) {  
 // Simulated PDF report generation logic  
 System.*out*.println("Generating PDF Report for " + e.getName());  
 // Actual PDF logic would go here  
 }  
}  
class EmailService {  
 public void sendEmail(Employee e) {  
 // Simulated email sending logic  
 System.*out*.println("Sending email to " + e.getEmail());  
 // Actual email logic would go here  
 }  
}  
public class Day20\_SRP2 {  
 public static void main(String[] args) {  
 Employee emp = new Employee("Harsha", "harsha@example.com", 60000);  
  
 PdfReportGenerator reportGenerator = new PdfReportGenerator();  
 EmailService emailService = new EmailService();  
  
 reportGenerator.generate(emp);  
 emailService.sendEmail(emp);  
 }  
}

**Output**

Generating PDF Report for Harsha

Sending email to [harsha@example.com](mailto:harsha@example.com)

**TASK5 – Open closed principle**

class Square() {

  int height;

  int area() { return height \* height; }

}

public class OpenOpenExample {

  public int compareArea(Square a, Square b) {

    return a.area() - b.area();

  }

}

extension code:

class Circle {

  int r;

  int area() { return Math.PI\*r\*r\*;}

}

class OpenOpenExample {

  public int compareArea(Square a, Square b) {

    return a.area() - b.area();

  }

  public int compareArea(Circle x, Circle y) {

   return x.area() - y.area();

  }

}

interface Shape  
{  
 double area();  
}  
class Square implements Shape {  
 private int height;  
  
 public Square(int height) {  
 this.height = height;  
 }  
  
 @Override  
 public double area() {  
 return height \* height;  
 }  
}  
class Circle implements Shape {  
 private int r;  
  
 public Circle(int r) {  
 this.r = r;  
 }  
  
 @Override  
 public double area() {  
 return Math.*PI* \* r \* r;  
 }  
}  
class AreaComparer {  
 public int compareArea(Shape s1, Shape s2) {  
 return Double.*compare*(s1.area(), s2.area());  
 }  
}  
public class Day20\_OCP1 {  
 public static void main(String[] args) {  
 Shape square = new Square(4); // area = 16  
 Shape circle = new Circle(3); // area = ~28.27  
  
 AreaComparer comparer = new AreaComparer();  
 int result = comparer.compareArea(square, circle);  
  
 if (result > 0) {  
 System.*out*.println("Square has a larger area.");  
 } else if (result < 0) {  
 System.*out*.println("Circle has a larger area.");  
 } else {  
 System.*out*.println("Areas are equal.");  
 }  
 }  
}

**Output**

Circle has a larger area.

**HOME TASK –** Write a program to create a class Student. With 3 methods

registrationDetails(), marksCalc(), feesCalc().

// 1. Student class - only holds student data

class Student {

String name;

String studentID;

int[] marks = new int[5];

public Student(String name, String studentID, int[] marks) {

this.name = name;

this.studentID = studentID;

this.marks = marks;

}

}

// 2. MarksCalculator - only calculates marks

class MarksCalculator {

public double calculateTotal(int[] marks) {

double total = 0;

for (int mark : marks) {

total += mark;

}

return total;

}

public double calculateAverage(int[] marks) {

return calculateTotal(marks) / marks.length;

}

}

// 3. FeeCalculator - only calculates fee based on marks

class FeeCalculator {

private final double baseFee = 50000;

public double calculateFees(double averageMarks) {

if (averageMarks > 90) {

System.out.println("🎉 Eligible for 20% scholarship");

return baseFee \* 0.8;

}

return baseFee;

}

}

// 4. Main class

import java.util.Scanner;

public class SRPExample {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Register student

System.out.print("Enter student name: ");

String name = sc.nextLine();

System.out.print("Enter student ID: ");

String id = sc.nextLine();

int[] marks = new int[5];

System.out.println("Enter marks for 5 subjects:");

for (int i = 0; i < 5; i++) {

System.out.print("Subject " + (i + 1) + ": ");

marks[i] = sc.nextInt();

}

Student student = new Student(name, id, marks);

// Calculate marks

MarksCalculator mCalc = new MarksCalculator();

double avg = mCalc.calculateAverage(student.marks);

System.out.println("Average Marks: " + avg);

// Calculate fees

FeeCalculator fCalc = new FeeCalculator();

double finalFees = fCalc.calculateFees(avg);

System.out.println("Final Fees: ₹" + finalFees);

}

}

**Output**

**Average Marks: 90.0**

**Eligible for 20% scholarship**

**Final Fees: ₹40000.0**