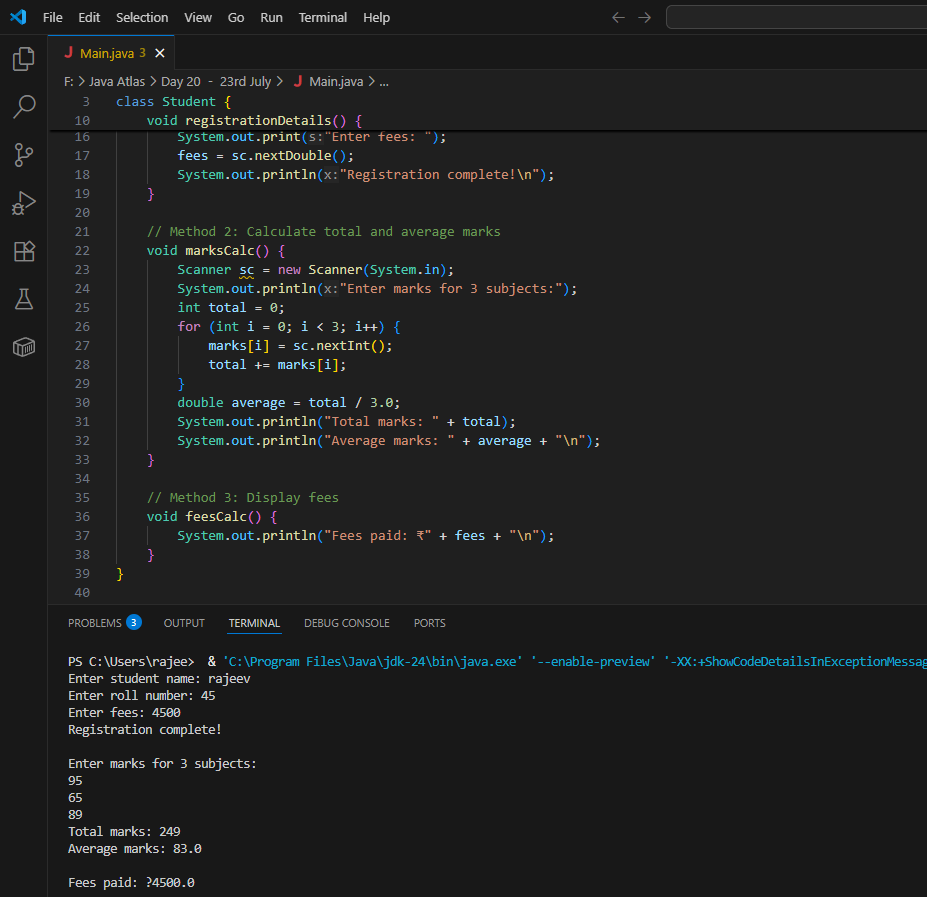
**ID – hrajranj**

**Day 20 - 23th July 2025**

**Task 1:**

**Wap to create a class Student.. With 3 methods**

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

****

**Task 02:**

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.

Implementing SRP:

class Employee {

    private String name;

    private String email;

    private double salary;

    // Methods related to employee data

}

class ReportGenerator {

    public void generatePdfReport(Employee employee) {

        // Code to generate PDF report using employee data

    }

}

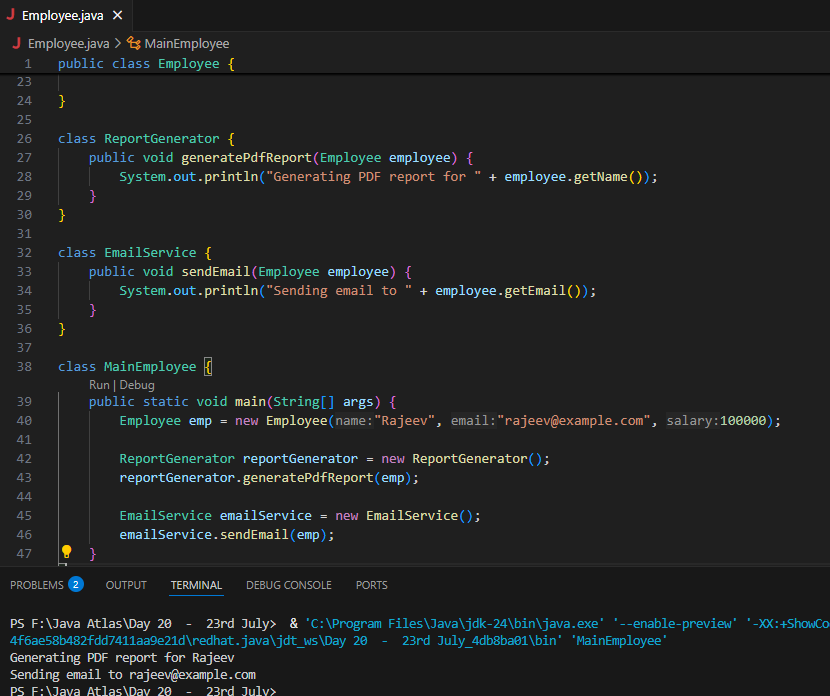
class EmailSender {

    public void sendEmail(String recipient, String message) {

        // Code to send email

    }

}



**Task 03:**

/\*

In the below example, the `PaymentProcessor` class violates the OCP because every time a new payment type needs to be supported, the `processPayment` method needs to be modified, which is against the principle of being closed for modification.

\*/

class PaymentProcessor {

    public void processPayment(String paymentType, double amount) {

        if (paymentType.equals("credit card")) {

            // Process credit card payment

} else if (paymentType.equals("paypal")) {

// Process PayPal payment

} else if (paymentType.equals("bitcoin")) {

// Process Bitcoin payment

}

// ... more payment types

}

}

implementing OCP :

interface PaymentGateway {

    void processPayment(double amount);

}

class CreditCardGateway implements PaymentGateway {

    public void processPayment(double amount) {

        // Process credit card payment

    }

}

class PayPalGateway implements PaymentGateway {

    public void processPayment(double amount) {

        // Process PayPal payment

    }

}

class BitcoinGateway implements PaymentGateway {

    public void processPayment(double amount) {

        // Process Bitcoin payment

    }

}

class PaymentProcessor {

    private PaymentGateway gateway;

    public PaymentProcessor(PaymentGateway gateway) {

        this.gateway = gateway;

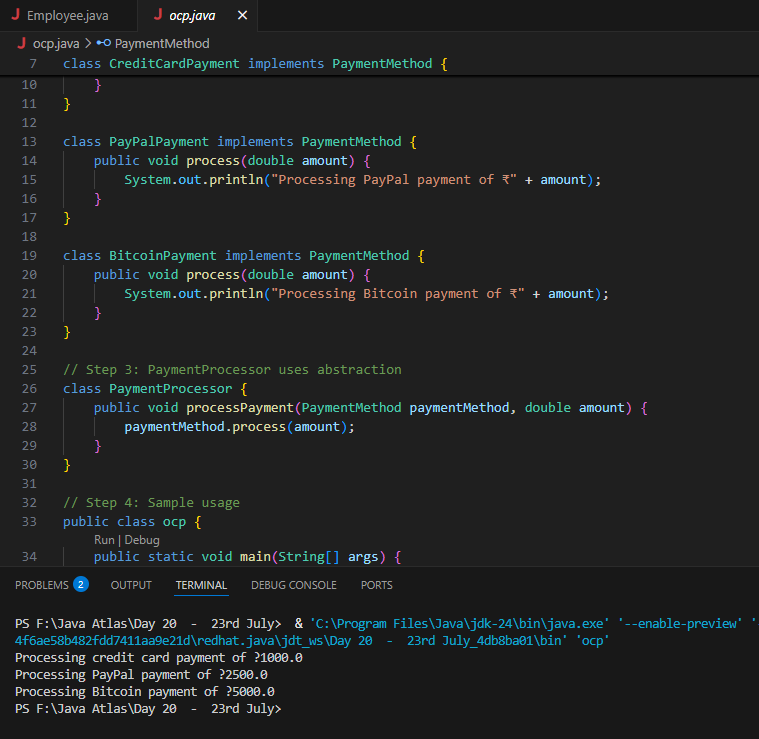
    }

    public void processPayment(double amount) {

        gateway.processPayment(amount);

    }

}



**Task04:**

Sequence diagram:

@startuml

    Prasunamba -> Atlas\_Batch1: Congratulations all

    Atlas\_Batch1 --> Prasunamba: Thank you Meher

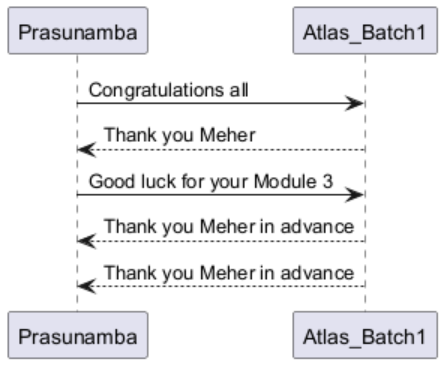
     Prasunamba -> Atlas\_Batch1: Good luck for your Module 3

    Atlas\_Batch1 --> Prasunamba: Thank you Meher in advance

or

    Prasunamba < -- Atlas\_Batch1: Thank you Meher in advance

@enduml



**Task 5:**

@startuml

    pax "A" as A

    pax "B" as B

    A --> B: Message/action

    A ..> B: reply

@enduml

