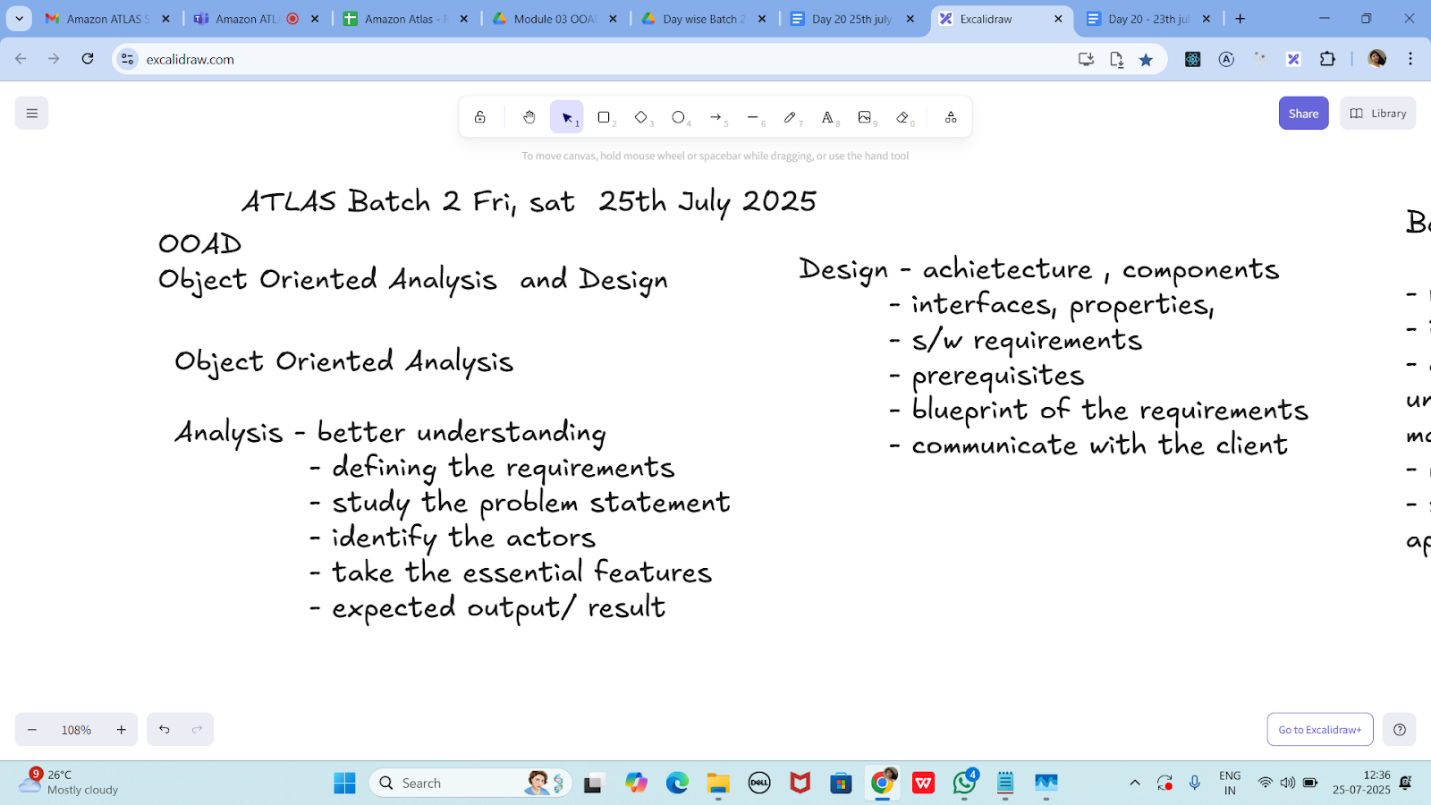
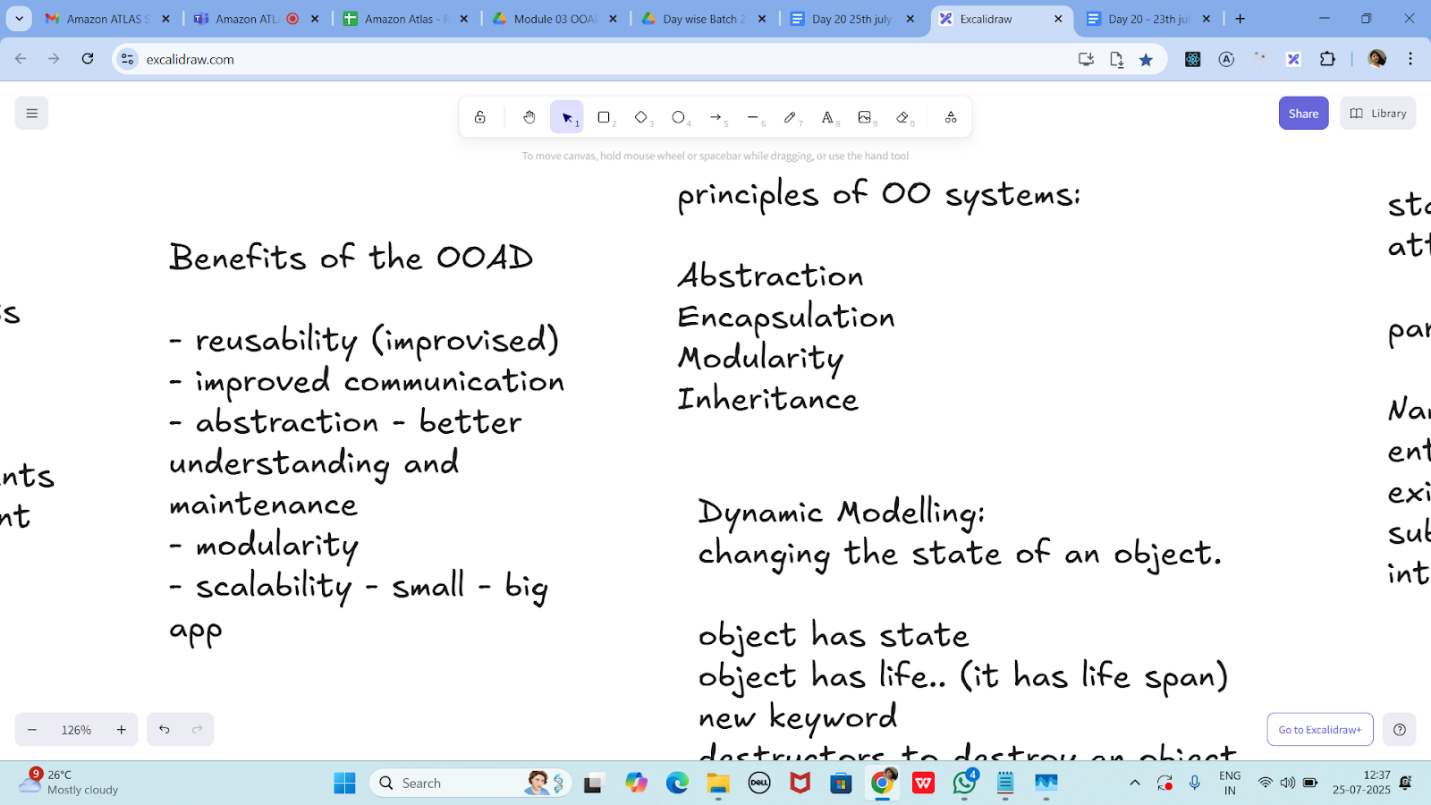
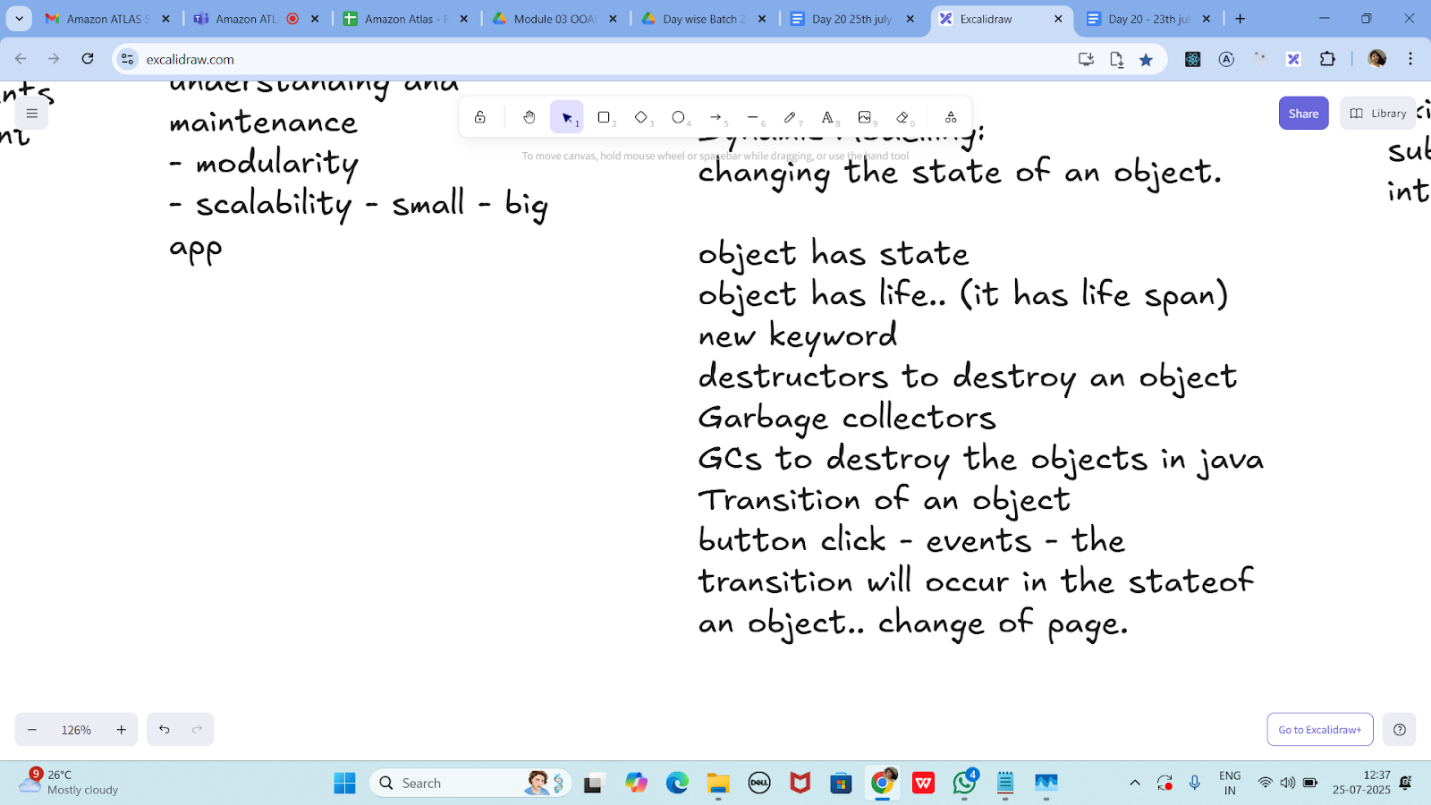
Day 20 25th july 2025

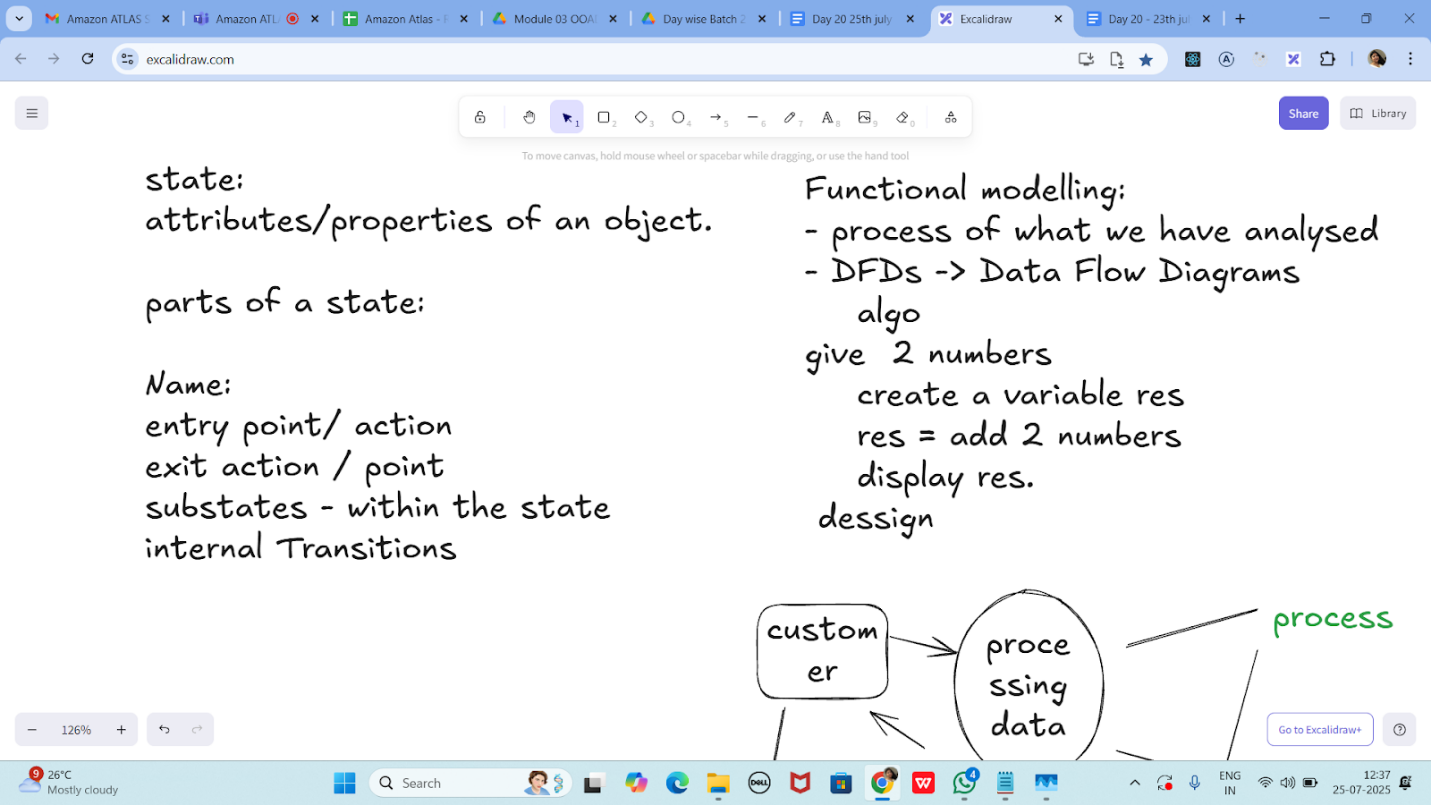
OOAD

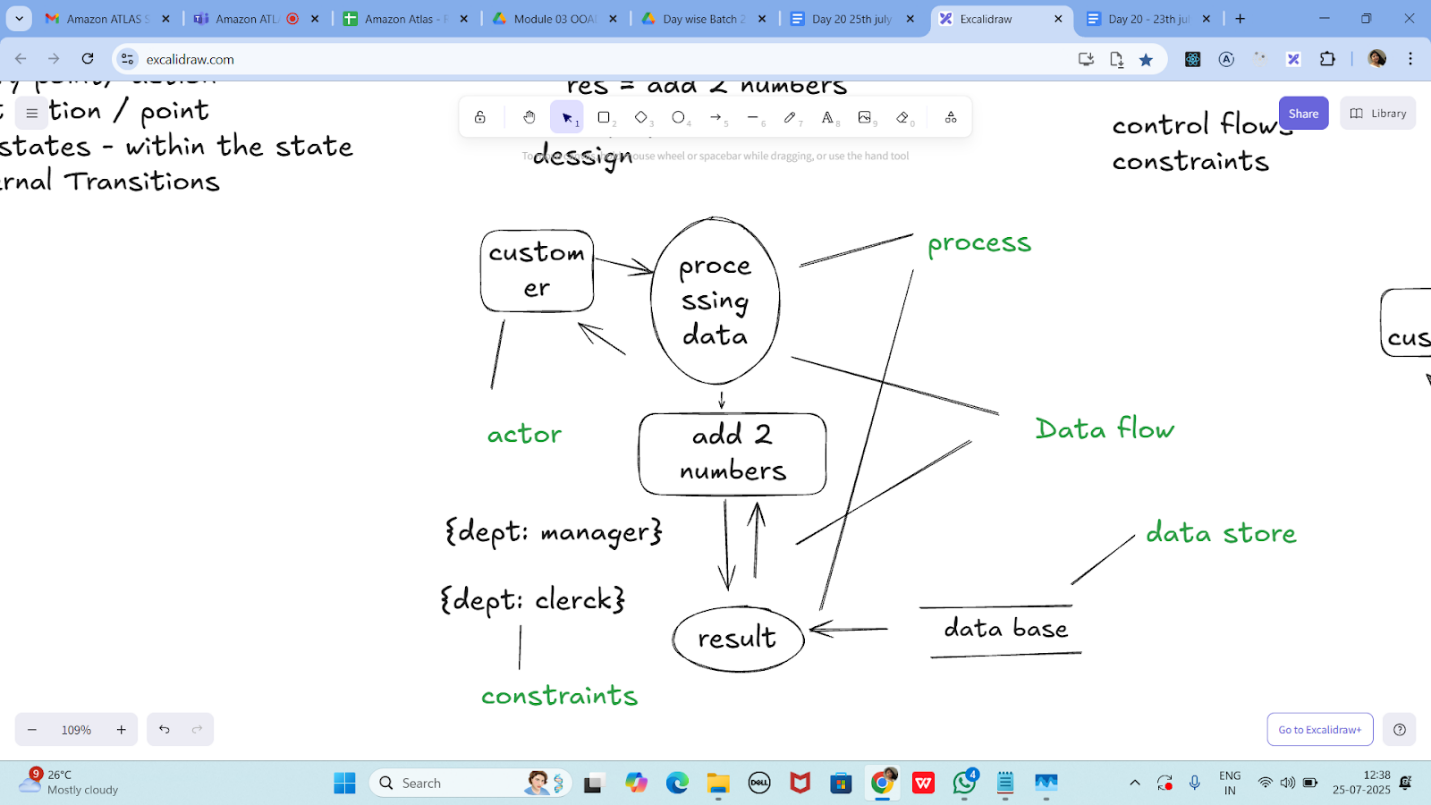
Object Oriented paradigm, Design Patterns, Design Workflow, Refactoring

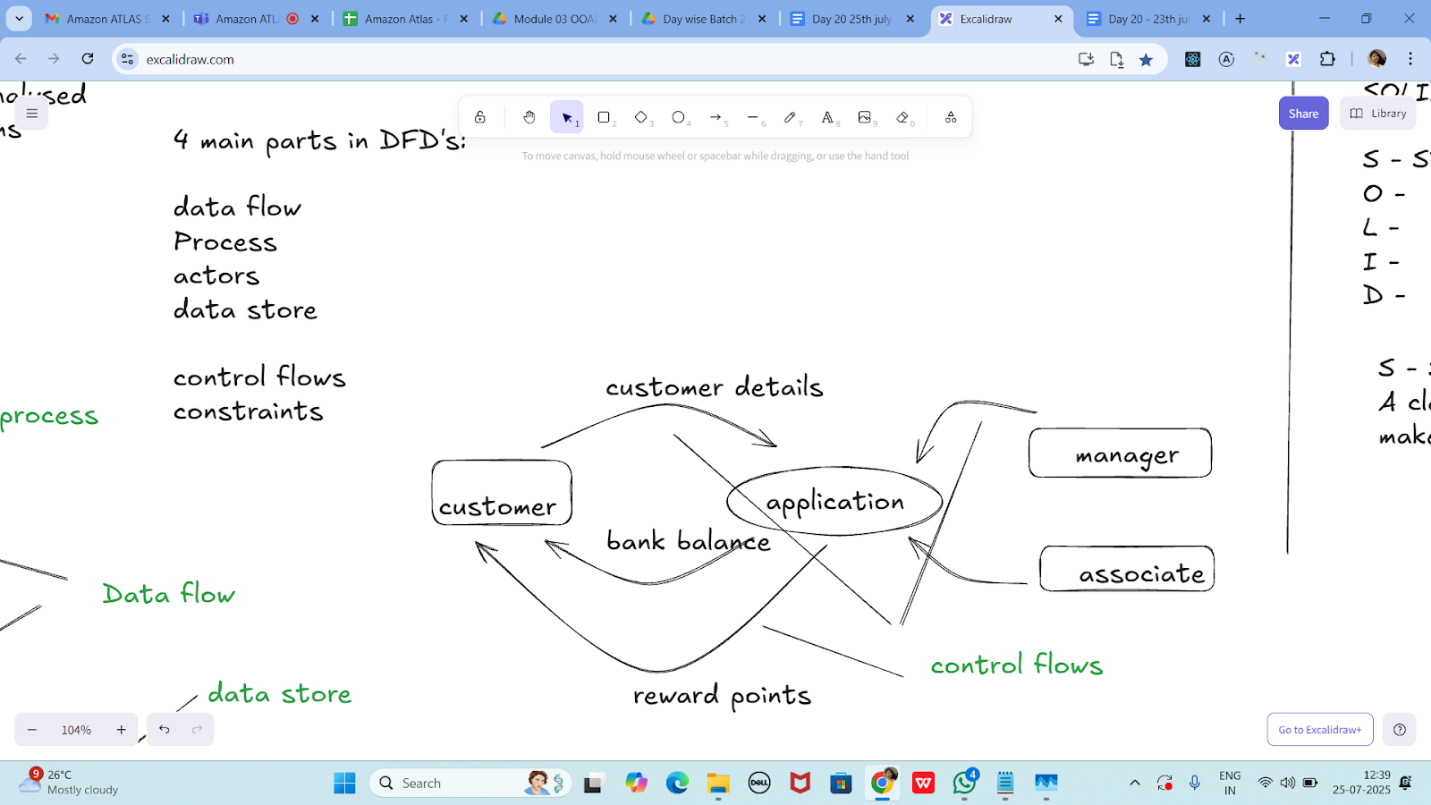


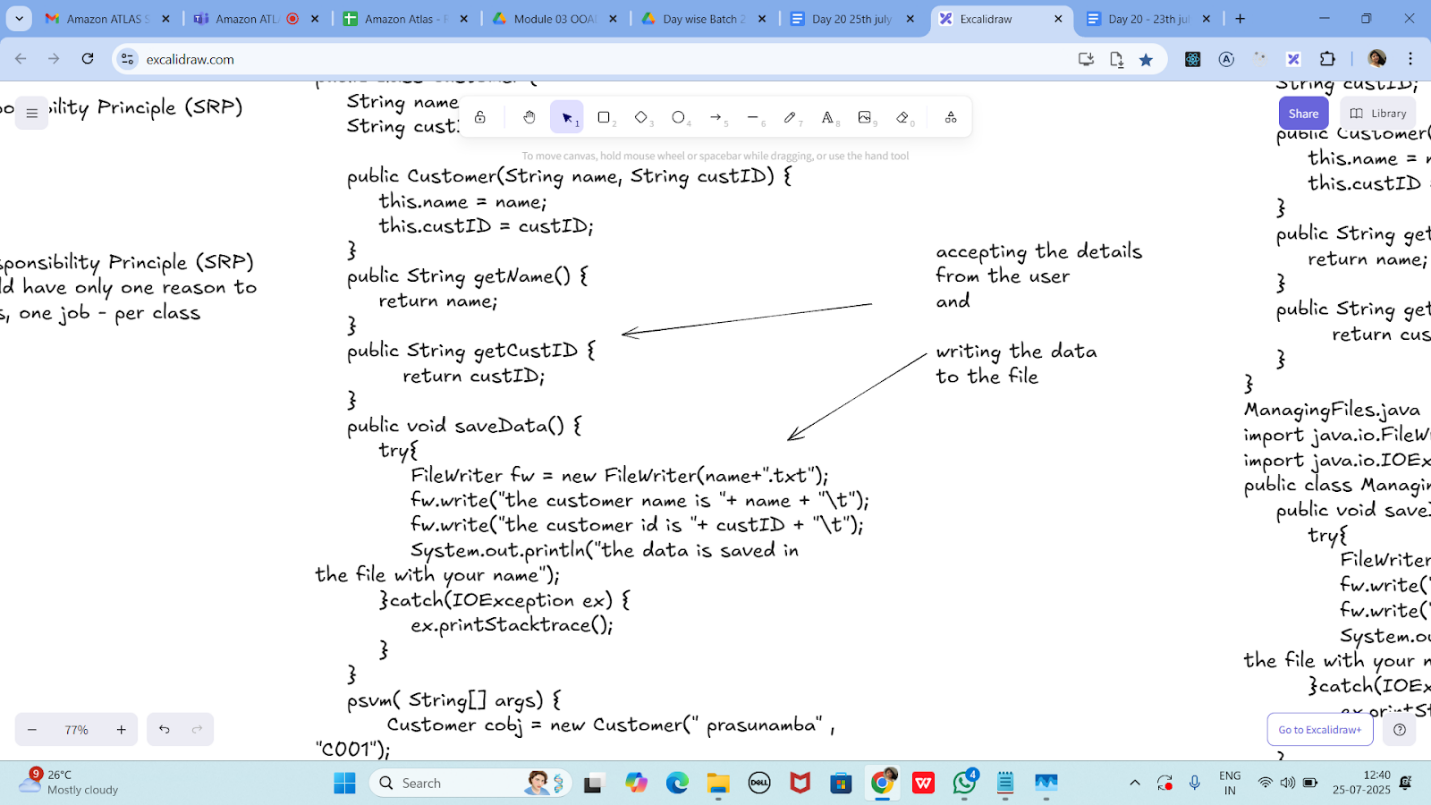












Without SRP:

SRP violation code

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();

    }

}

SRP Implementation:

Customer.java

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;

    }

}

ManagingFiles.java

import java.io.FileWriter;

import java.io.IOException;

public class ManagingFiles{

    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();

        }

    }

}

SRP\_Imple.java

public class SRP\_Imple {

    psvm( String[] args) {

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

         ManagingFiles mobj = new ManagingFiles();

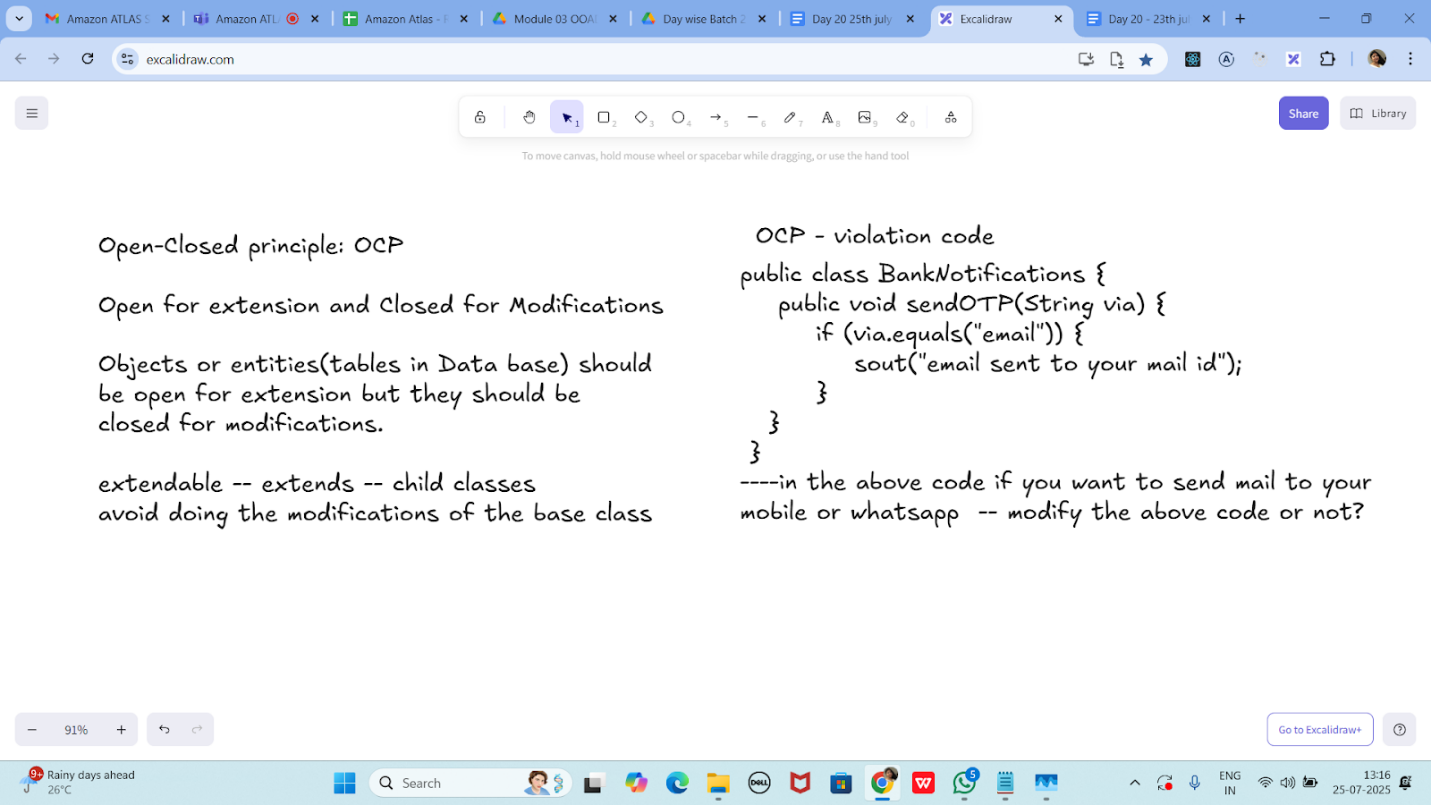
        mobj.saveData();

    }

}

—---------------------------------------------------------------------as of 12.41………………………………

OCP -



public class BankNotifications {

    public void sendOTP(String via) {

        if (via.equals("email")) {

            sout("email sent to your mail id");

        }

   }

 }

----in the above code if you want to send mail to your

mobile or whatsapp  -- modify the above code or not?

OCP - implementation

public interface BankNotifications  {

    public void sendOTP(String via);

    //public void TransactionNotification(Srting via);

//violates srp so .. include another interface

}

class EmailNotify implements BankNotifications  {

    public void sendOTP(String via) {

        sout("email sent to your mail id");

    }

    /\*public void TransactionNotification(String via) {

        sout("email sent to your mail id");

    }\*/

}

class MobileNotify implements BankNotifications  {

    public void sendOTP(String via) {

        sout("msg sent to your Mobile no");

    }

    /\*public void TransactionNotification(String via) {

        sout("msg sent to your Mobile no");

    }\*/

}

class WhatsappNotify implements BankNotifications  {

    public void sendOTP(String via) {

        sout("msg sent to your whatsapp ");

    }

    /\*public void TransactionNotification(String via) {

        sout("msg sent to your whatsapp");

    }\*/

}

// sending a physical notification.. extend here..

**Task 3:**

**The below is violating SRP complete it and also  … plz 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

}

2. 50 to 2.55

Code :

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;

}

// Getters only — no logic

public String getTitle() {

return title;

}

public String getAuthor() {

return author;

}

public double getPrice() {

return price;

}

}

public class BookFormatter {

public String getFormattedTitle(Book book) {

return "Title: " + book.getTitle().toUpperCase();

}

public String getFormattedAuthor(Book book) {

return "Author: " + book.getAuthor();

}

}

public class BookPricing {

public double calculateDiscountedPrice(Book book, double discountPercentage) {

return book.getPrice() \* (1 - discountPercentage);

}

}

public class Main {

public static void main(String[] args) {

Book book = new Book("Clean Code", "Robert C. Martin", 500);

BookFormatter formatter = new BookFormatter();

BookPricing pricing = new BookPricing();

System.out.println(formatter.getFormattedTitle(book)); // Output: Title: CLEAN CODE

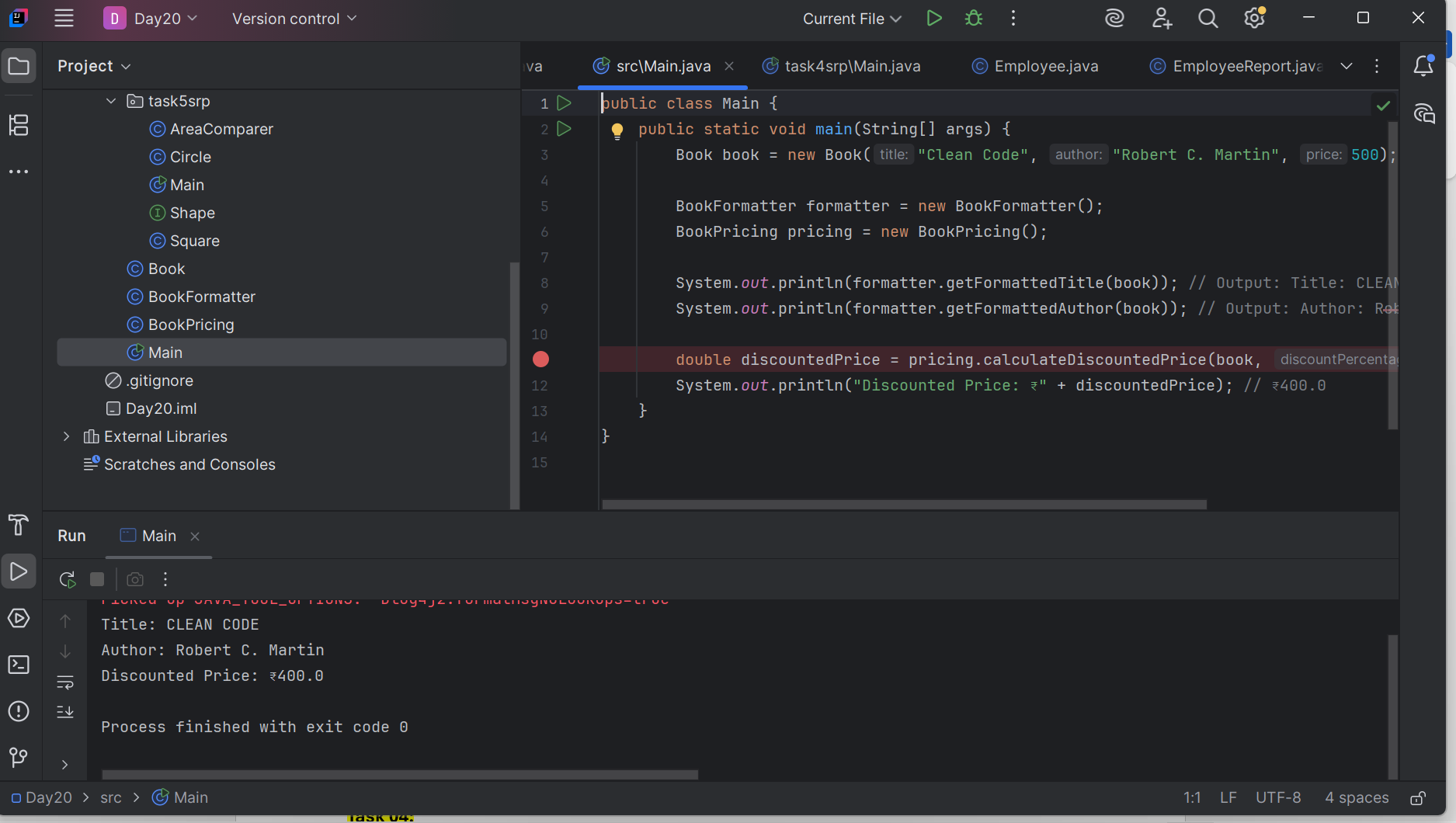
System.out.println(formatter.getFormattedAuthor(book)); // Output: Author: Robert C. Martin

double discountedPrice = pricing.calculateDiscountedPrice(book, 0.2); // 20% off

System.out.println("Discounted Price: ₹" + discountedPrice); // ₹400.0

}

}



**Task 04:**

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.

2.55 to 3pm

Code :

public 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;

}

}

public class EmployeeReport {

public void generatePdfReport(Employee employee) {

// Simulate PDF generation logic

System.out.println("Generating PDF report for " + employee.getName());

}

}

public class EmployeeMailer {

public void sendEmail(Employee employee) {

// Simulate email sending logic

System.out.println("Sending email to " + employee.getEmail());

}

}

public class Main {

public static void main(String[] args) {

Employee emp = new Employee("Alice", "alice@example.com", 50000);

EmployeeReport report = new EmployeeReport();

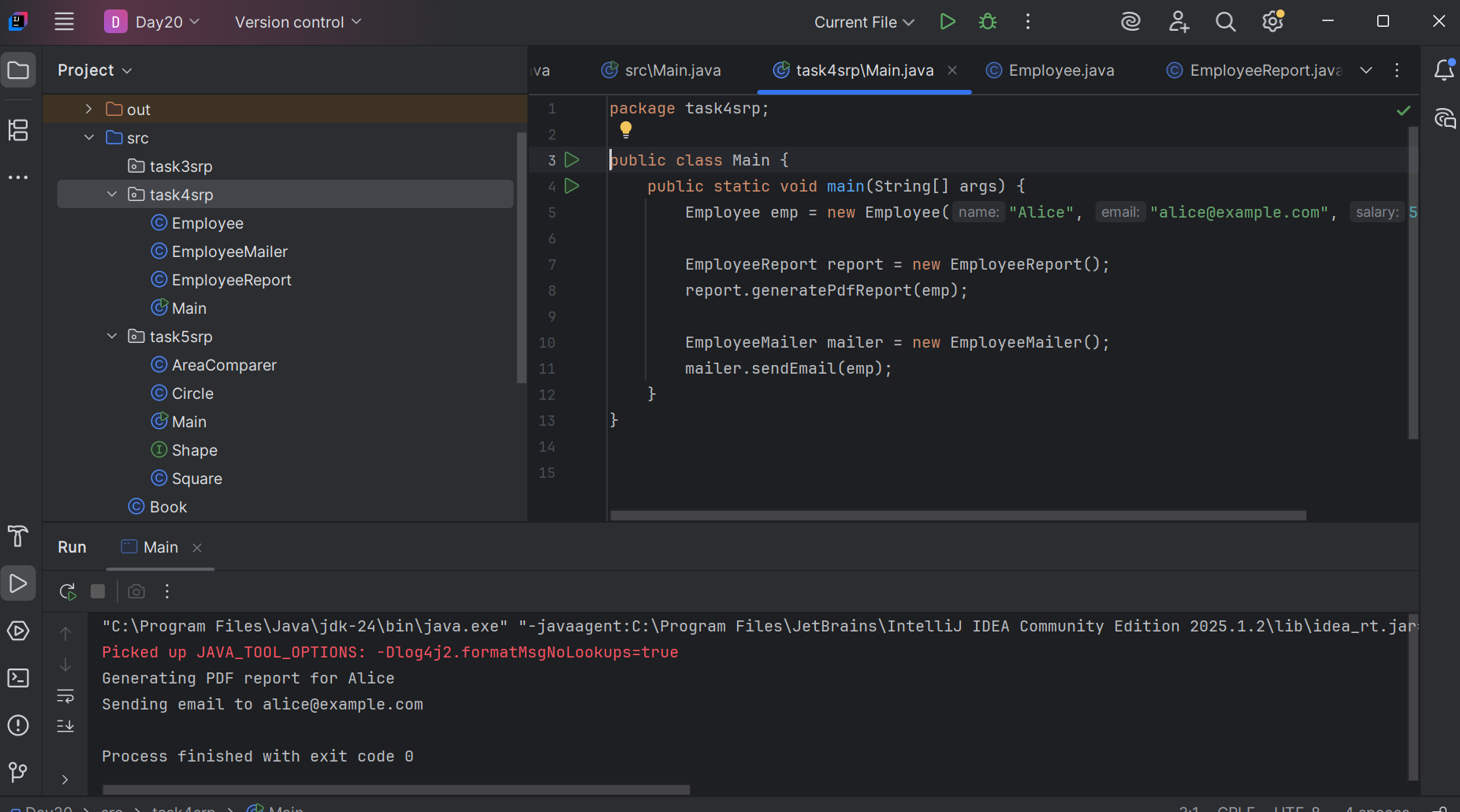
report.generatePdfReport(emp);

EmployeeMailer mailer = new EmployeeMailer();

mailer.sendEmail(emp);

}

}



**Task 05:**

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();

  }

}

3.00 to 3.05

Code :

public interface Shape {

double area();

}

public class Square implements Shape {

private int height;

public Square(int height) {

this.height = height;

}

@Override

public double area() {

return height \* height;

}

}

public class Circle implements Shape {

private int radius;

public Circle(int radius) {

this.radius = radius;

}

@Override

public double area() {

return Math.PI \* radius \* radius;

}

}

public class AreaComparer {

public double compareArea(Shape a, Shape b) {

return a.area() - b.area();

}

}

public class Main {

public static void main(String[] args) {

Shape s1 = new Square(4);

Shape s2 = new Circle(3);

AreaComparer comparer = new AreaComparer();

double result = comparer.compareArea(s1, s2);

if (result > 0)

System.out.println("First shape has larger area");

else if (result < 0)

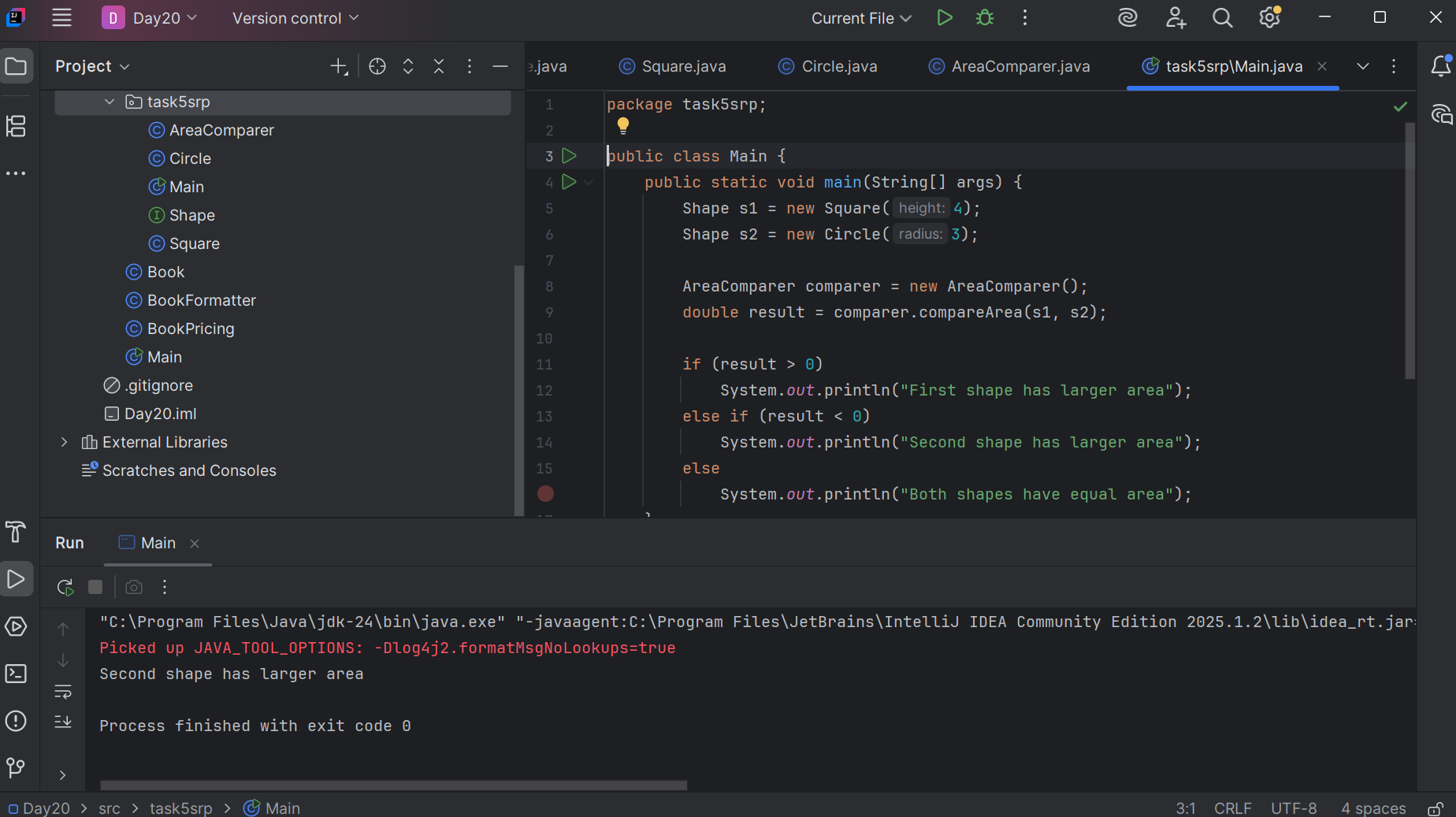
System.out.println("Second shape has larger area");

else

System.out.println("Both shapes have equal area");

}

}



--------------------------------------------------------------------

Applying the open-closed principle in Java

interface Shape {

  int area();

}

class Circle implements Shape {

  int r;

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

}

class Square() implements {

  int height;

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

}

public class OpenClosedExample {

  public int compareArea(Shape a, Shape b) {

    return a.area() - b.area();

  }

}

UML:

UML - Unified modeling Language

plantUML:

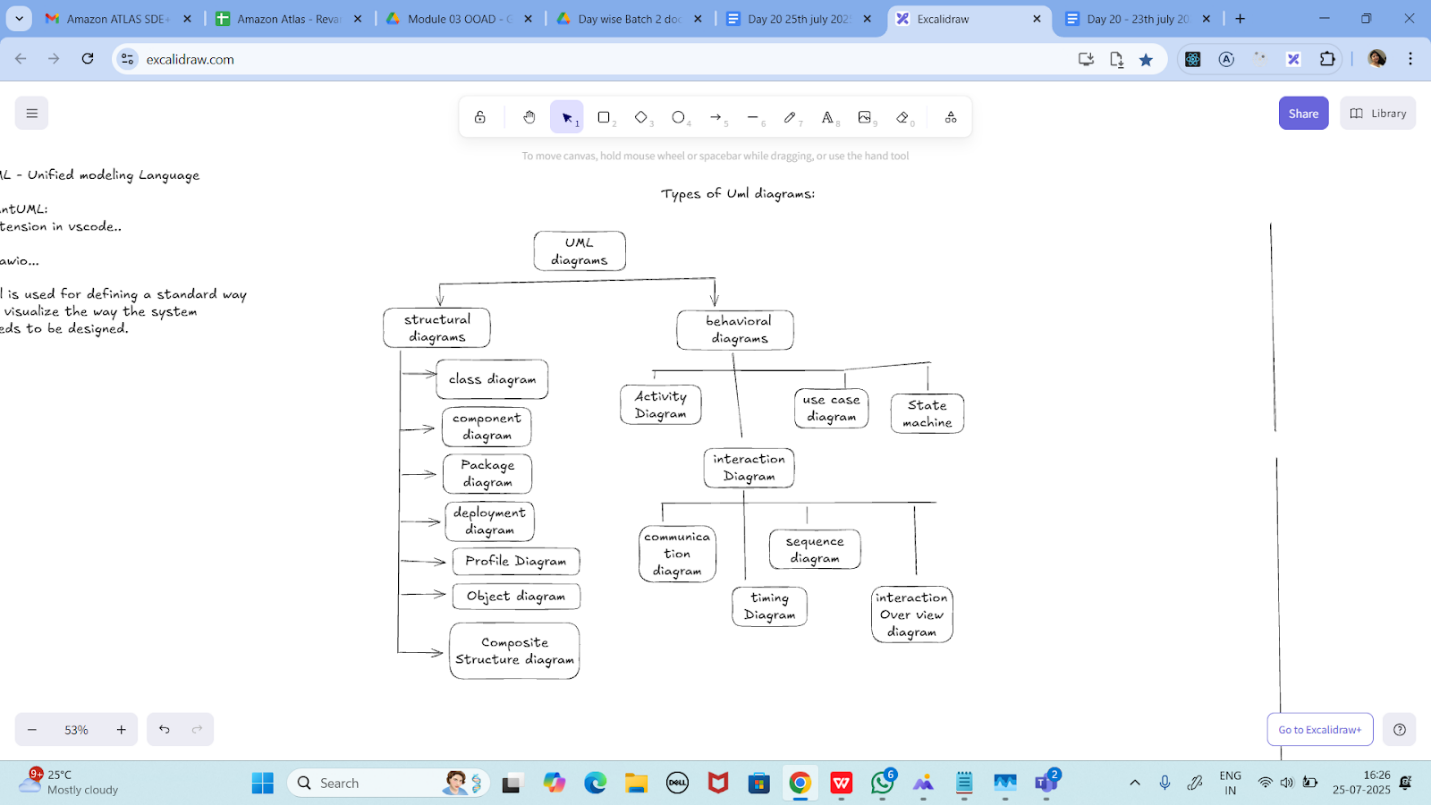
extension in vscode..

Drawio...

uml is used for defining a standard way

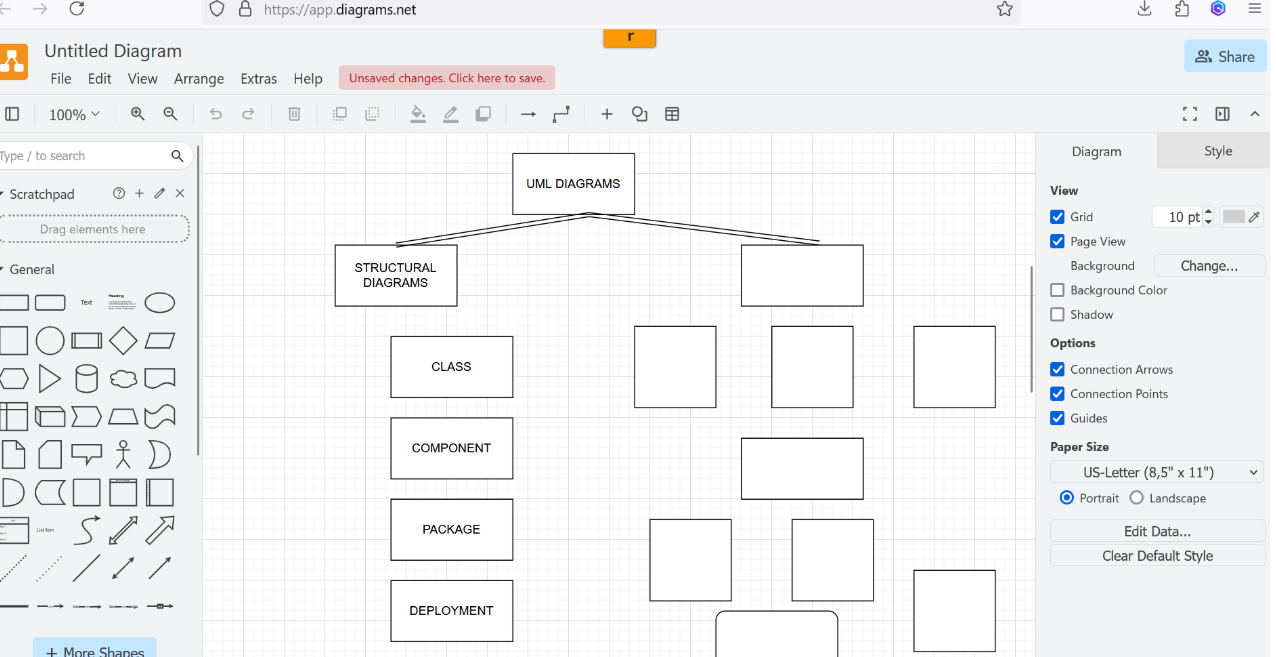
to visualize the way the system

needs to be designed.



**Task 06:**

**Can you guys create diagrams for structural diagrams…**

****

==============================================================

**Info box:**

=============================================================

<https://excalidraw.com/#json=koAtUznEAcXbBLQqRRH1K,SDnPak_Xh1EPkRD7vKH--A>

==============================================================