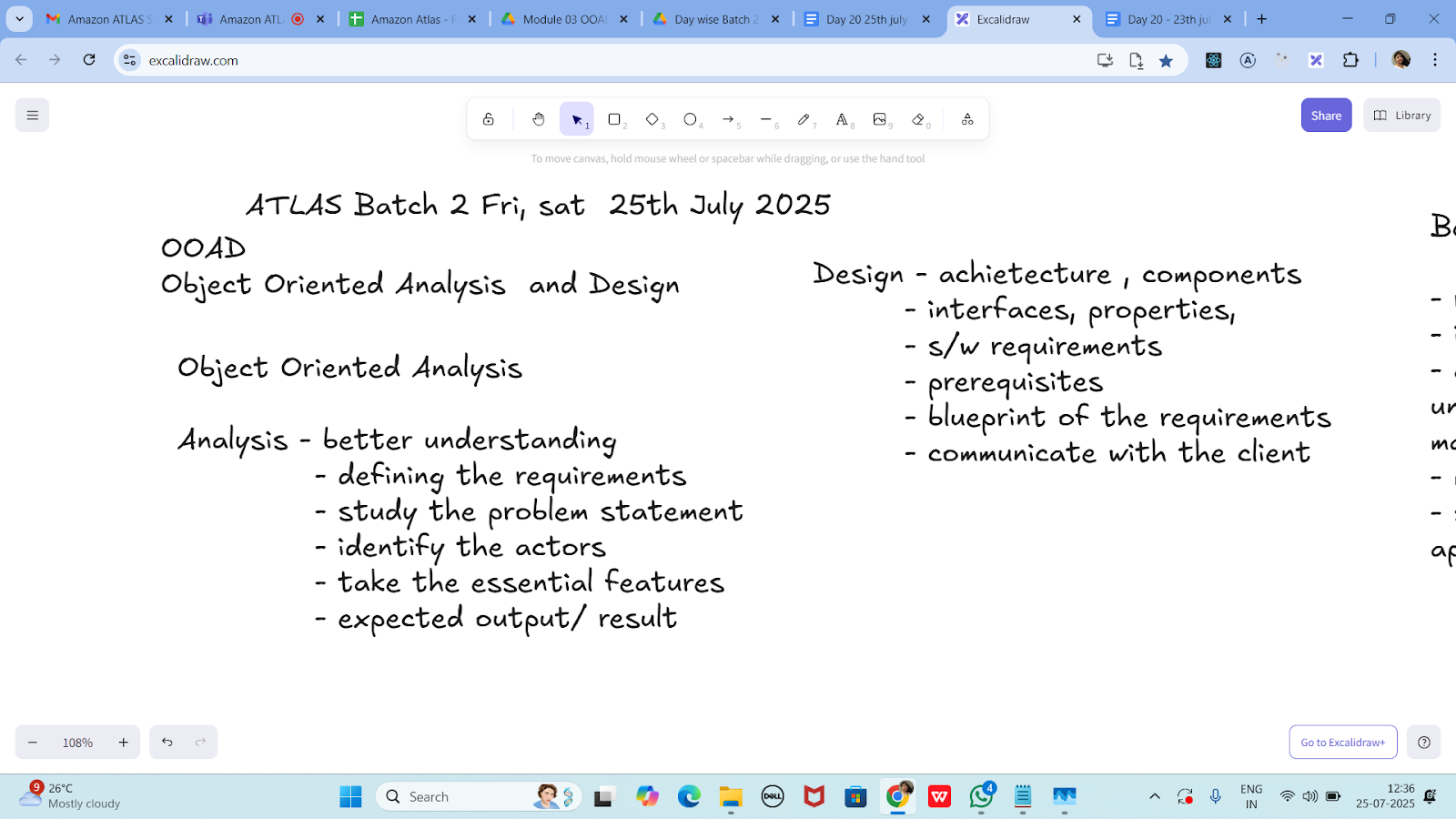
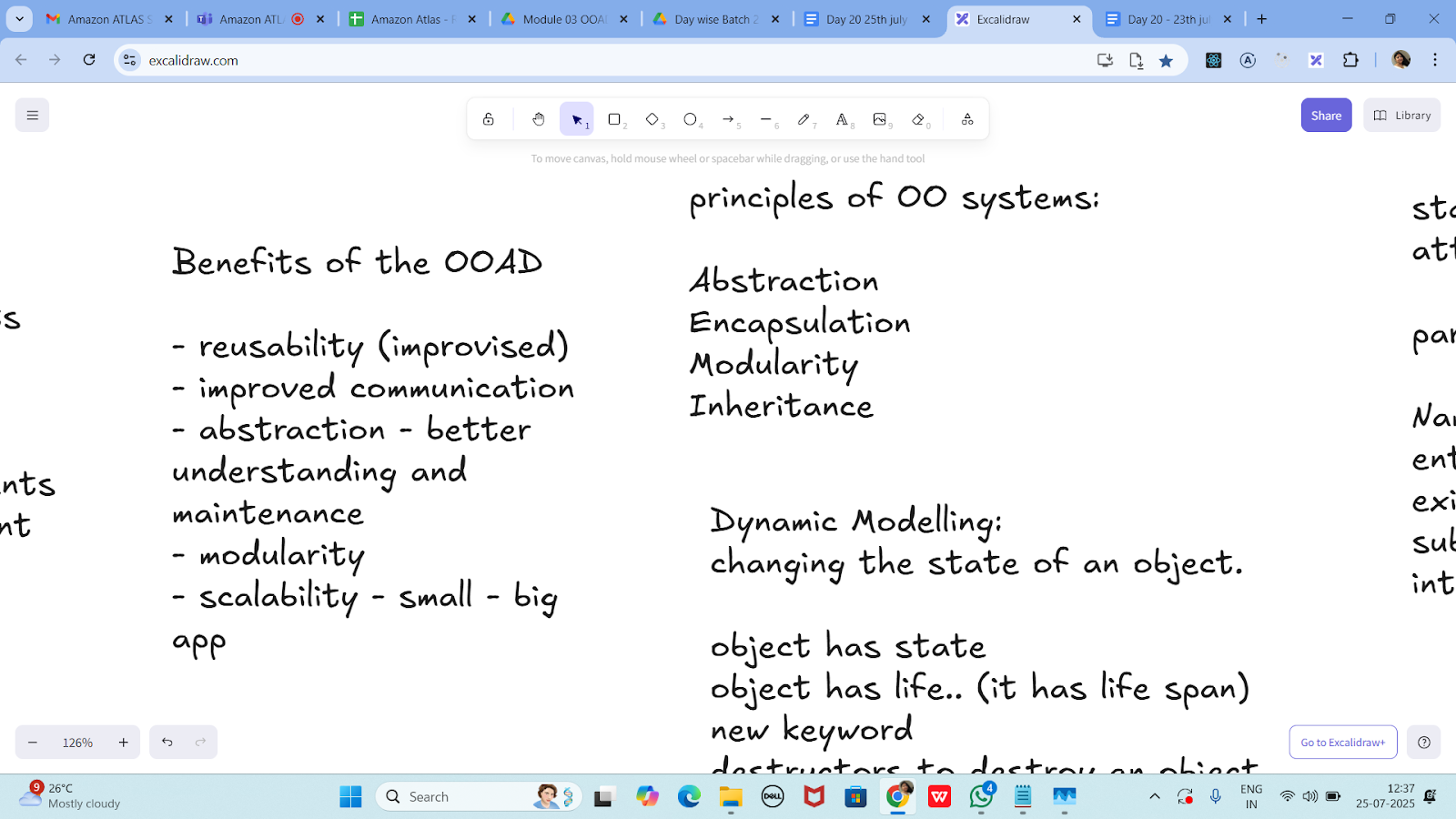
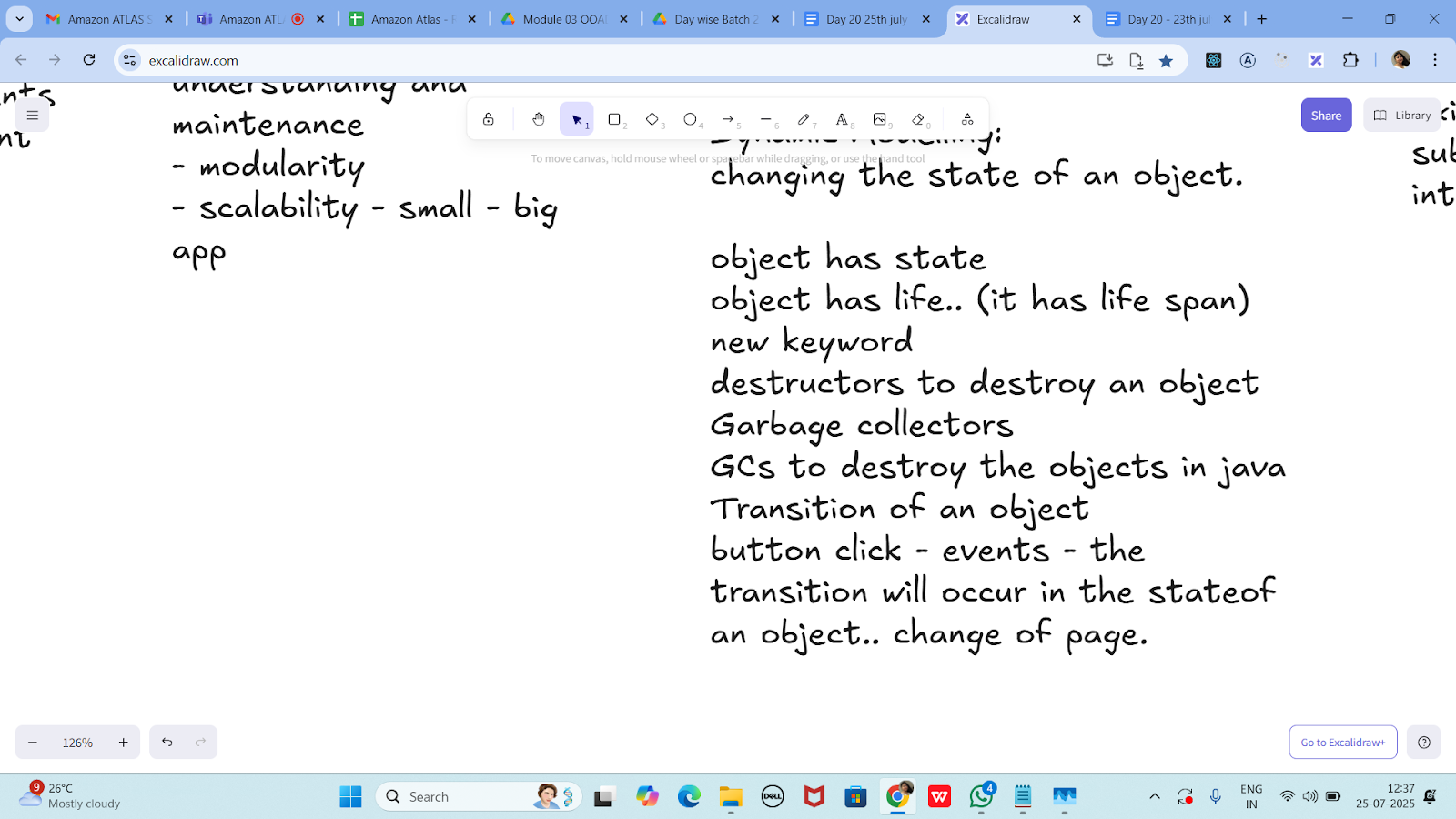
Day 20 25th july 2025

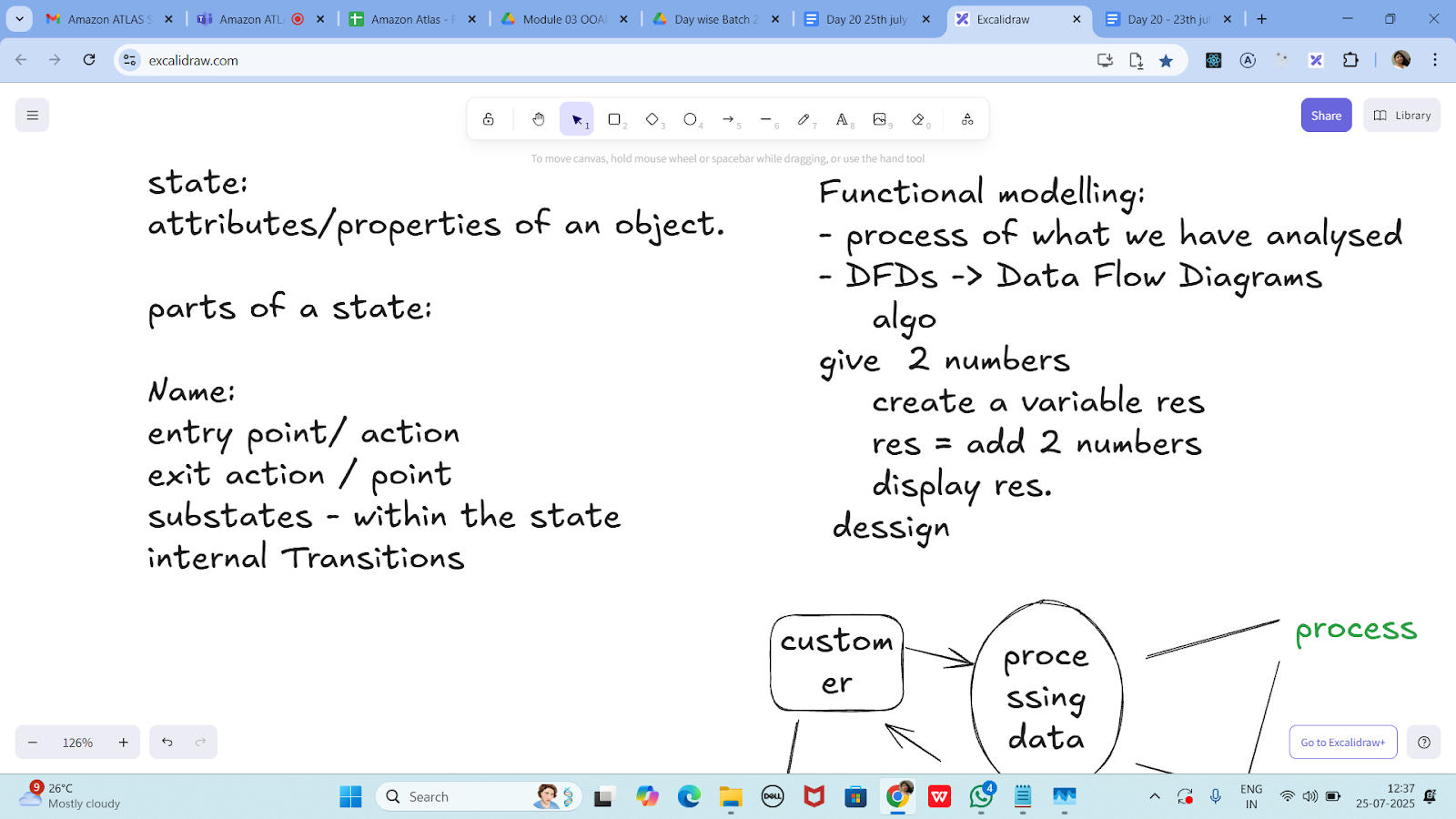
OOAD

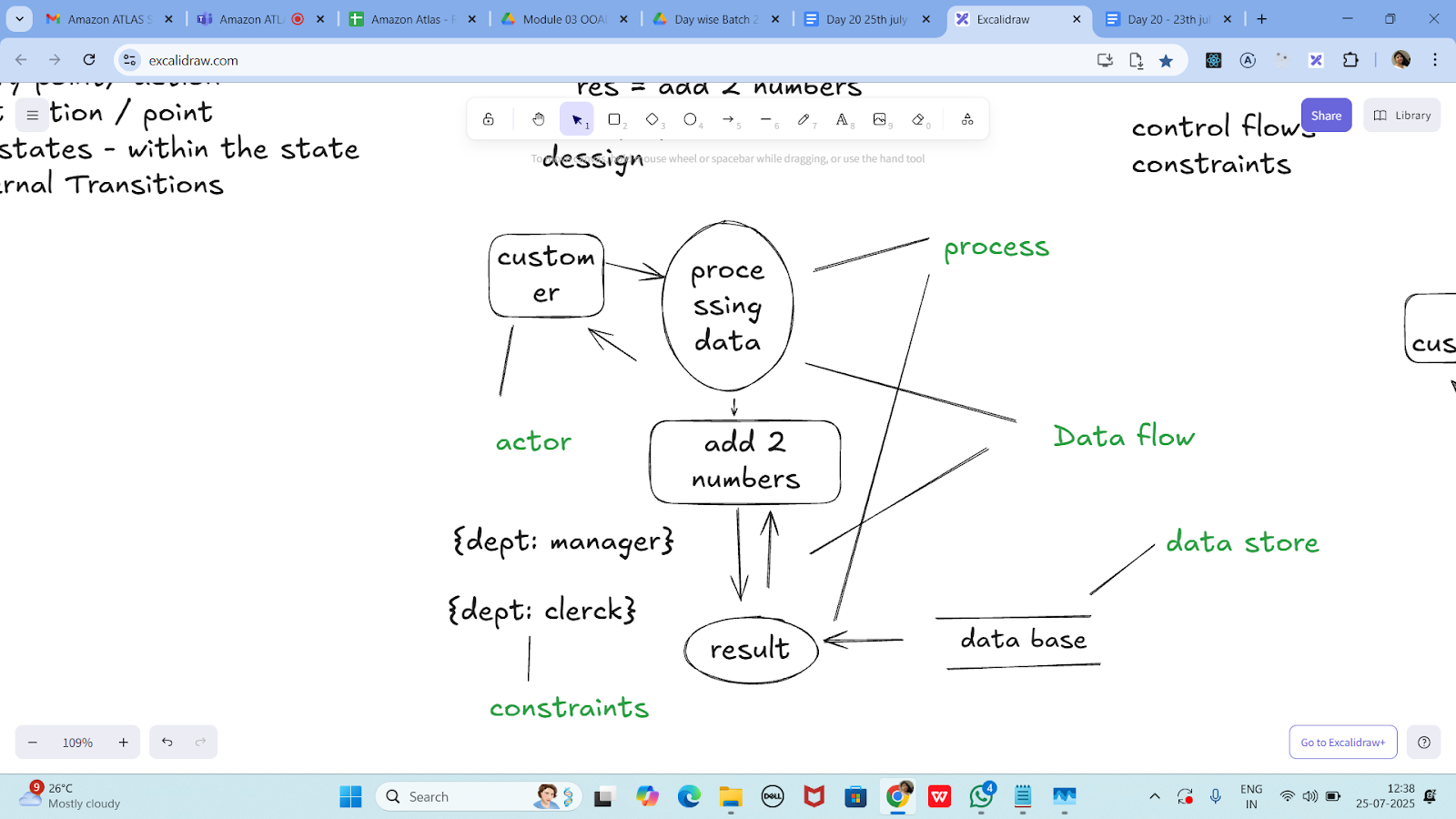
Object Oriented paradigm, Design Patterns, Design Workflow, Refactoring

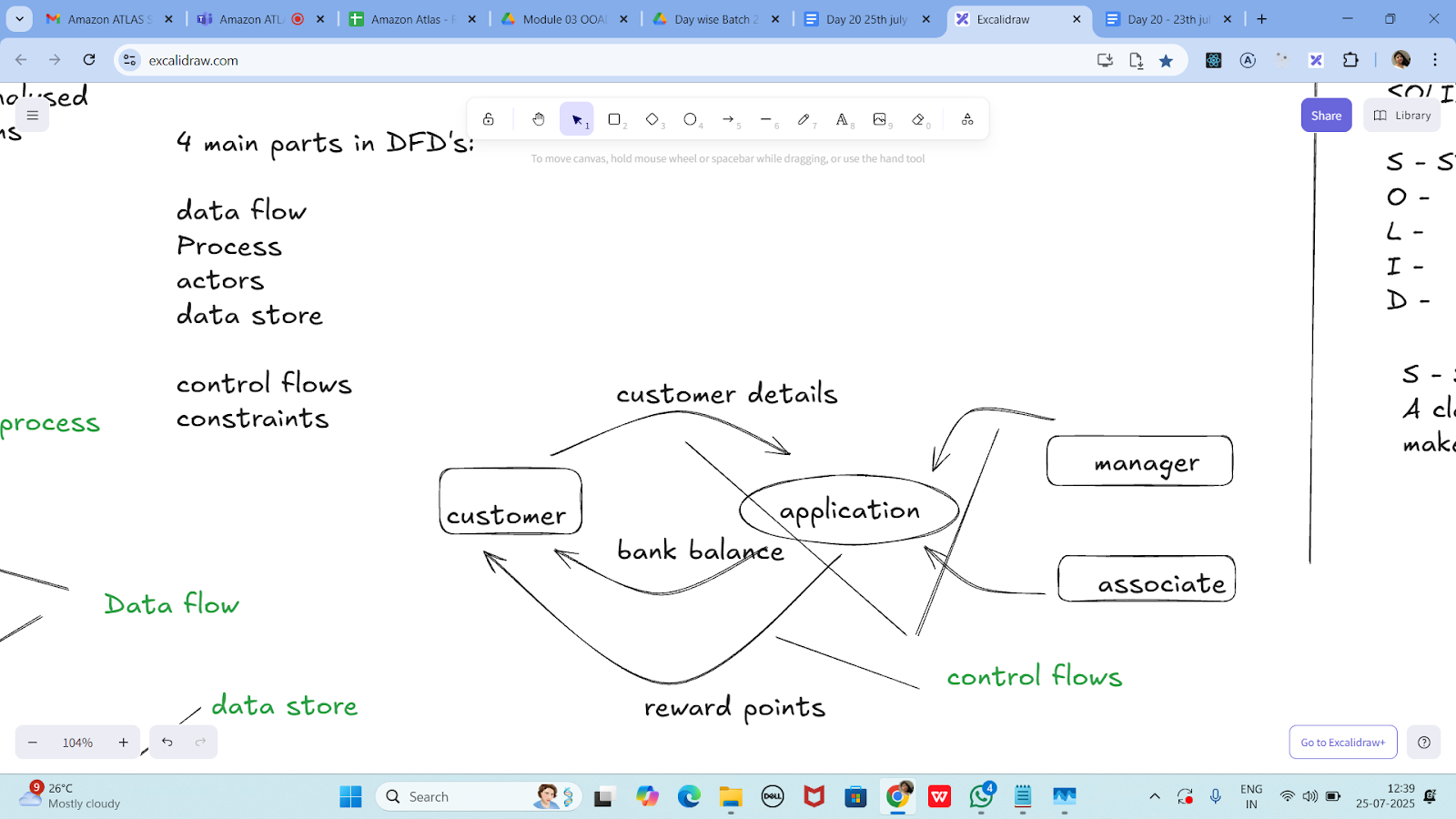


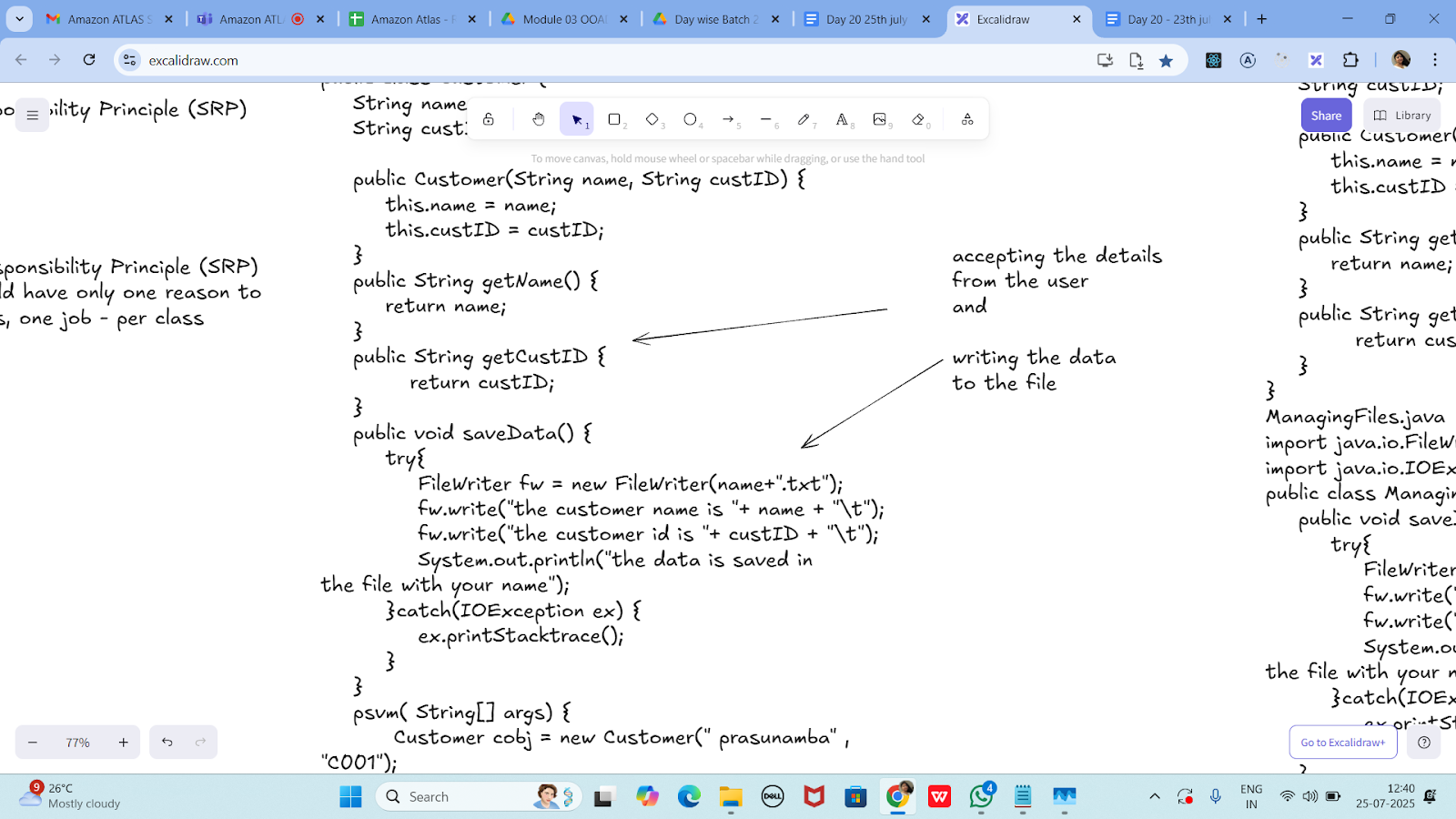












**Without SRP:**

**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();

}

}

import java.io.FileWriter;

import java.io.IOException;

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;

}

// ❌ This method violates SRP - Customer class should not handle file saving logic

public void saveData() {

try {

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

fw.write("Customer Name: " + name + "\n");

fw.write("Customer ID: " + custID + "\n");

fw.close();

System.*out*.println("Data saved in file named after the customer.");

} catch (IOException ex) {

ex.printStackTrace();

}

}

public static void main(String[] args) {

Customer cobj = new Customer("Pratheesh", "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();

}

}

import java.io.FileWriter;

import java.io.IOException;

// Class that holds Customer2 data — single responsibility

class Customer2 {

private String name;

private String custID;

public Customer2(String name, String custID) {

this.name = name;

this.custID = custID;

}

public String getName() {

return name;

}

public String getCustID() {

return custID;

}

}

// Class that handles file operations — single responsibility

class ManagingFiles {

public void saveData(Customer2 Customer2) {

try {

FileWriter fw = new FileWriter(Customer2.getName() + ".txt");

fw.write("The Customer2 name is: " + Customer2.getName() + "\n");

fw.write("The Customer2 ID is: " + Customer2.getCustID() + "\n");

fw.close();

System.*out*.println("Data saved to file with the Customer2's name.");

} catch (IOException ex) {

ex.printStackTrace();

}

}

}

// Public class for execution — only one public class allowed

public class SRP\_Imple {

public static void main(String[] args) {

Customer2 cobj = new Customer2("Prasunamba", "C001");

ManagingFiles mobj = new ManagingFiles();

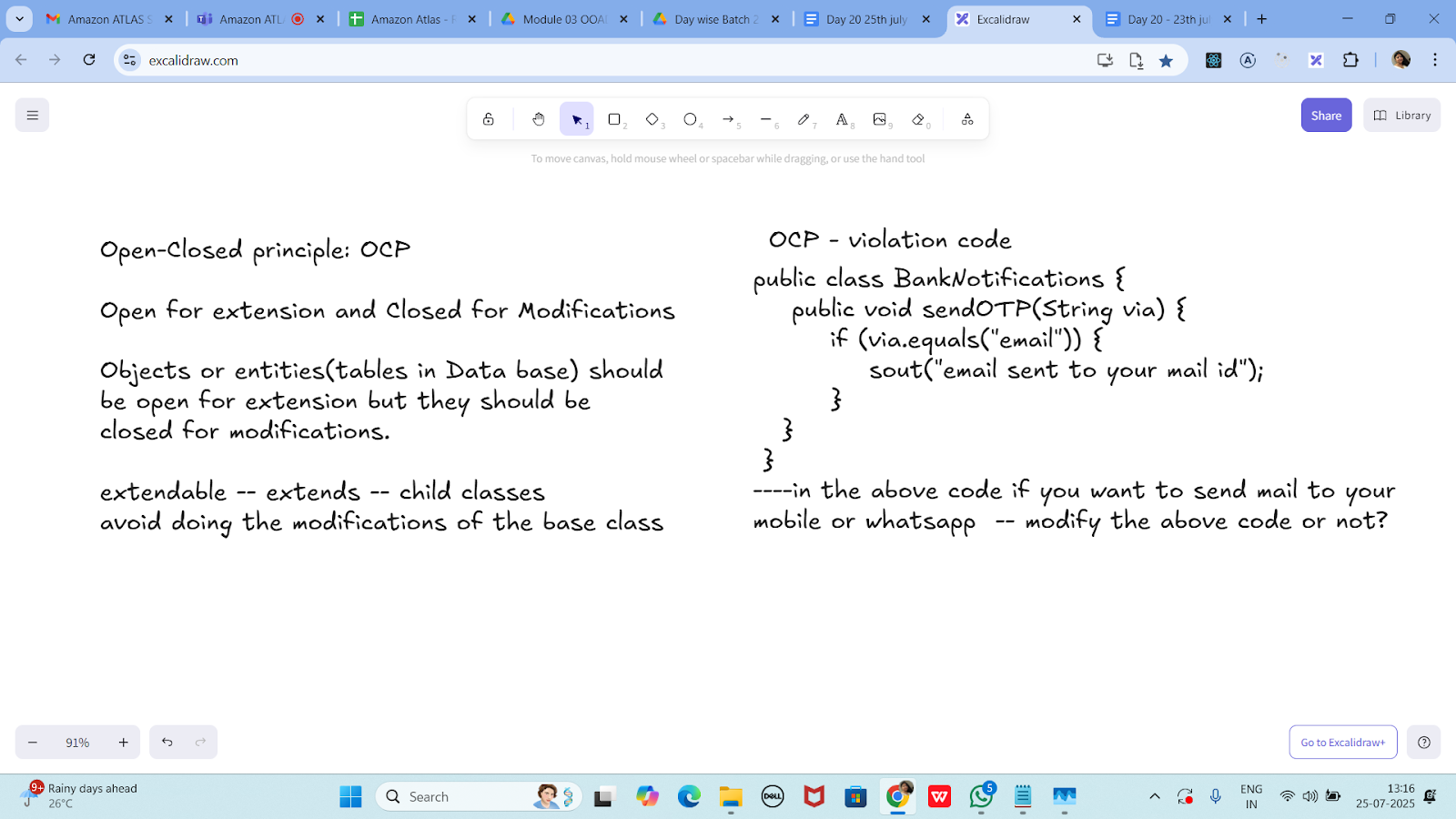
mobj.saveData(cobj);

}

}

—---------------------------------------------------------------------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?

public class BankNotifications {

public void sendOTP(String via) {

if (via.equals("email")) {

System.*out*.println("Email sent to your mail ID");

} else if (via.equals("sms")) {

System.*out*.println("OTP sent via SMS");

} else if (via.equals("whatsapp")) {

System.*out*.println("OTP sent via WhatsApp");

}

}

public static void main(String[] args) {

BankNotifications notify = new BankNotifications();

notify.sendOTP("email");

notify.sendOTP("sms");

notify.sendOTP("whatsapp");

}

}

**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..

// Interface for OTP notifications

interface BankNotifications2 {

void sendOTP();

}

// Email notification implementation

class EmailNotify implements BankNotifications2 {

public void sendOTP() {

System.*out*.println("OTP sent to your email.");

}

}

// SMS notification implementation

class MobileNotify implements BankNotifications2 {

public void sendOTP() {

System.*out*.println("OTP sent to your mobile.");

}

}

// WhatsApp notification implementation

class WhatsappNotify implements BankNotifications2 {

public void sendOTP() {

System.*out*.println("OTP sent to your WhatsApp.");

}

}

// Physical letter notification (extending system without modifying existing classes)

class PhysicalNotify implements BankNotifications2 {

public void sendOTP() {

System.*out*.println("OTP sent via physical mail.");

}

}

// Main class to use any notification

public class OCP\_Example {

public static void main(String[] args) {

BankNotifications2 notify1 = new EmailNotify();

BankNotifications2 notify2 = new MobileNotify();

BankNotifications2 notify3 = new WhatsappNotify();

BankNotifications2 notify4 = new PhysicalNotify();

notify1.sendOTP();

notify2.sendOTP();

notify3.sendOTP();

notify4.sendOTP();

}

}

**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

}

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;

}

// Presentation logic mixed in — violates SRP

public String getFormattedTitle() {

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

}

// Business logic mixed in — violates SRP

public double calculateDiscountedPrice(double discountPercentage) {

return price \* (1 - discountPercentage);

}

public static void main(String[] args) {

Book book = new Book("Clean Code", "Robert C. Martin", 500.0);

System.*out*.println(book.getFormattedTitle());

double discountedPrice = book.calculateDiscountedPrice(0.10); // 10% discount

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

}

}

**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.

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;

}

// Core responsibility: Employee data

public String getDetails() {

return "Name: " + name + ", Email: " + email + ", Salary: ₹" + salary;

}

// Additional responsibility: Report generation — violates SRP

public void generatePdfReport() {

System.*out*.println("Generating PDF report for " + name);

// Imagine actual PDF code here

}

// Additional responsibility: Sending email — violates SRP

public void sendEmail() {

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

// Imagine actual email code here

}

public static void main(String[] args) {

Employee emp = new Employee("Alice", "alice@example.com", 60000);

System.*out*.println(emp.getDetails());

emp.generatePdfReport(); // Not Employee's responsibility

emp.sendEmail(); // Not Employee's responsibility

}

}

2.55 to 3pm

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

}

}

**class Employee2 {**

**private String name;**

**private String email;**

**private double salary;**

**public Employee2(String name, String email, double salary) {**

**this.name = name;**

**this.email = email;**

**this.salary = salary;**

**}**

**public String getName() {**

**return name;**

**}**

**public String getEmail() {**

**return email;**

**}**

**public double getSalary() {**

**return salary;**

**}**

**public String getDetails() {**

**return "Name: " + name + ", Email: " + email + ", Salary: ₹" + salary;**

**}**

**}**

**// Single Responsibility: Generating reports**

**class ReportGenerator {**

**public void generatePdfReport(Employee2 Employee2) {**

**System.*out*.println("Generating PDF report for " + Employee2.getName());**

**// Simulated report generation**

**}**

**}**

**// Single Responsibility: Sending emails**

**class EmailSender {**

**public void sendEmail(String recipient, String message) {**

**System.*out*.println("Sending email to " + recipient);**

**System.*out*.println("Message: " + message);**

**// Simulated email send**

**}**

**}**

**// Main class for testing**

**public class SRPExample {**

**public static void main(String[] args) {**

**Employee2 emp = new Employee2("John Doe", "john@example.com", 75000);**

**ReportGenerator reportGen = new ReportGenerator();**

**EmailSender emailSender = new EmailSender();**

**System.*out*.println(emp.getDetails());**

**reportGen.generatePdfReport(emp);**

**emailSender.sendEmail(emp.getEmail(), "Your PDF report is ready.");**

**}**

**}**

**Task 05:**

**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();

}

}

3.00 to 3.05

class Square {

int height;

Square(int height) {

this.height = height;

}

public int area() {

return height \* height;

}

}

class Circle {

int r;

Circle(int r) {

this.r = r;

}

public int area() {

return (int)(Math.*PI* \* r \* r); // casting to int for simplicity

}

}

public class OpenOpenExample {

// Not scalable — violates OCP

public int compareArea(Square a, Square b) {

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

}

public int compareArea(Circle x, Circle y) {

return x.area() - y.area();

}

public static void main(String[] args) {

Square s1 = new Square(4);

Square s2 = new Square(6);

Circle c1 = new Circle(3);

Circle c2 = new Circle(5);

OpenOpenExample example = new OpenOpenExample();

System.*out*.println("Square area difference: " + example.compareArea(s1, s2));

System.*out*.println("Circle area difference: " + example.compareArea(c1, c2));

}

}

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

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();

}

}

interface Shape {

int area();

}

class Circle2 implements Shape {

int r;

Circle2(int r) {

this.r = r;

}

public int area() {

return (int)(Math.*PI* \* r \* r); // Cast to int for simplicity

}

}

class Square2 implements Shape {

int height;

Square2(int height) {

this.height = height;

}

public int area() {

return height \* height;

}

}

public class OpenClosedExample {

public int compareArea(Shape a, Shape b) {

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

}

public static void main(String[] args) {

Shape s1 = new Square2(5);

Shape s2 = new Circle2(3);

OpenClosedExample example = new OpenClosedExample();

System.*out*.println("Area difference: " + example.compareArea(s1, s2));

}

}

UML:

UML - Unified modeling Language

plantUML:

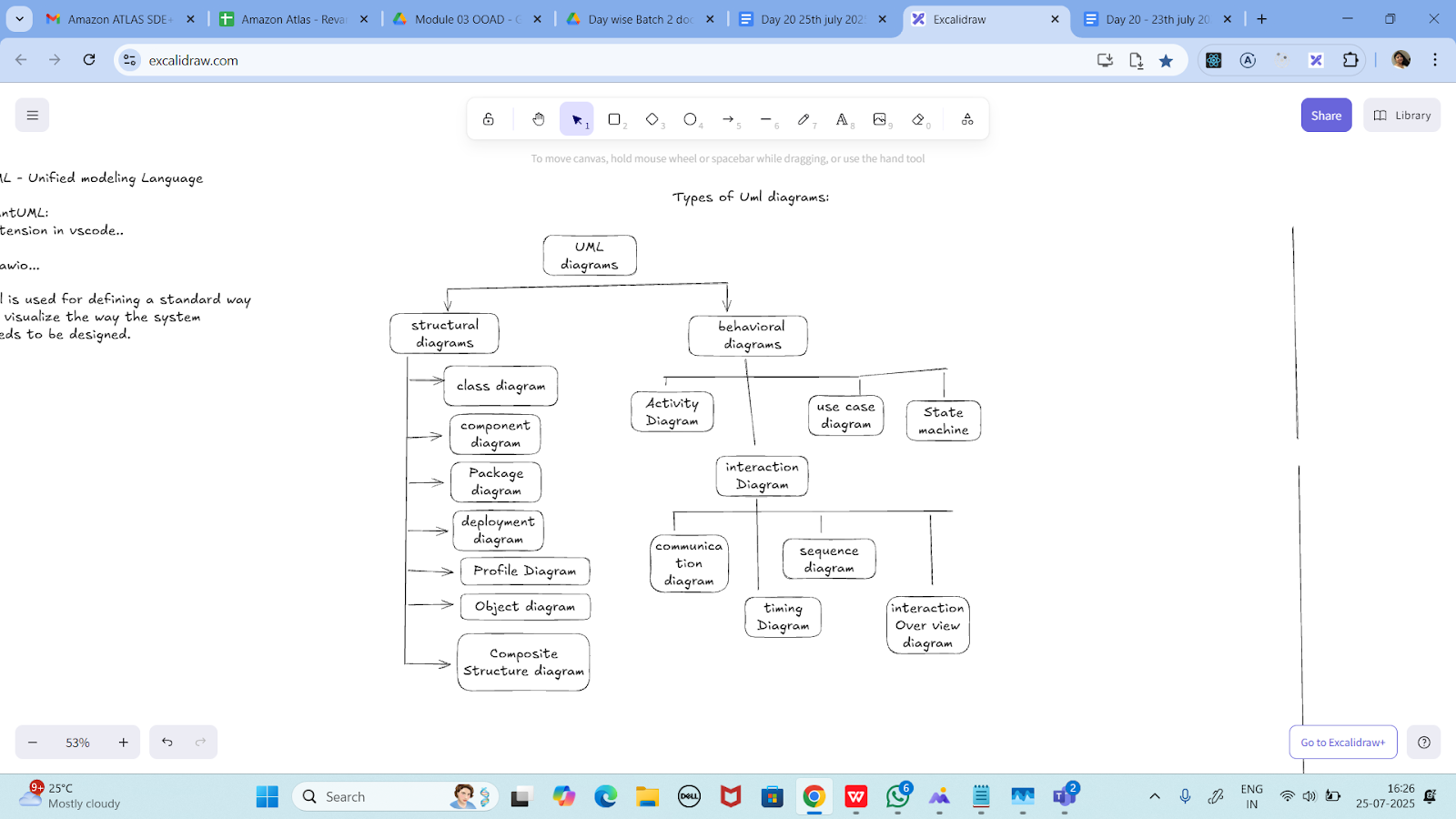
extension in vscode..

Drawio...

uml is used for defining a standard way

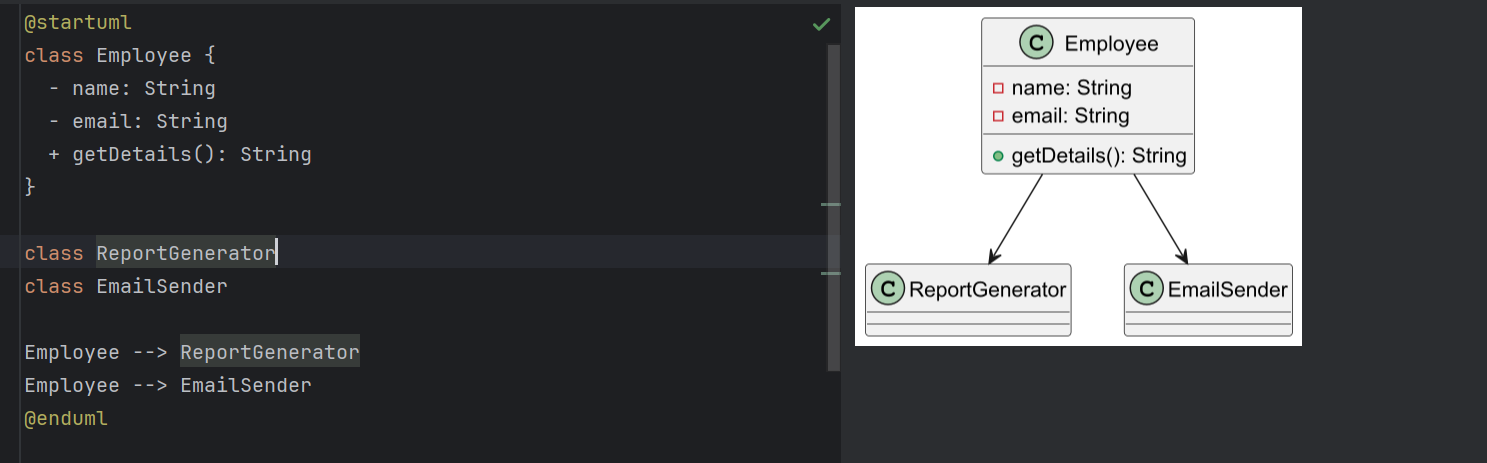
to visualize the way the system

needs to be designed.



Task 06:

Can you guys create diagrams for structural diagrams…



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

**Info box:**

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

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

The above link updated// plz click below link

<https://excalidraw.com/#json=KJCtzEJJ4apOcoW0CEq-e,PrVjs6qYCOMlOnp_oVqgfg>

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

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