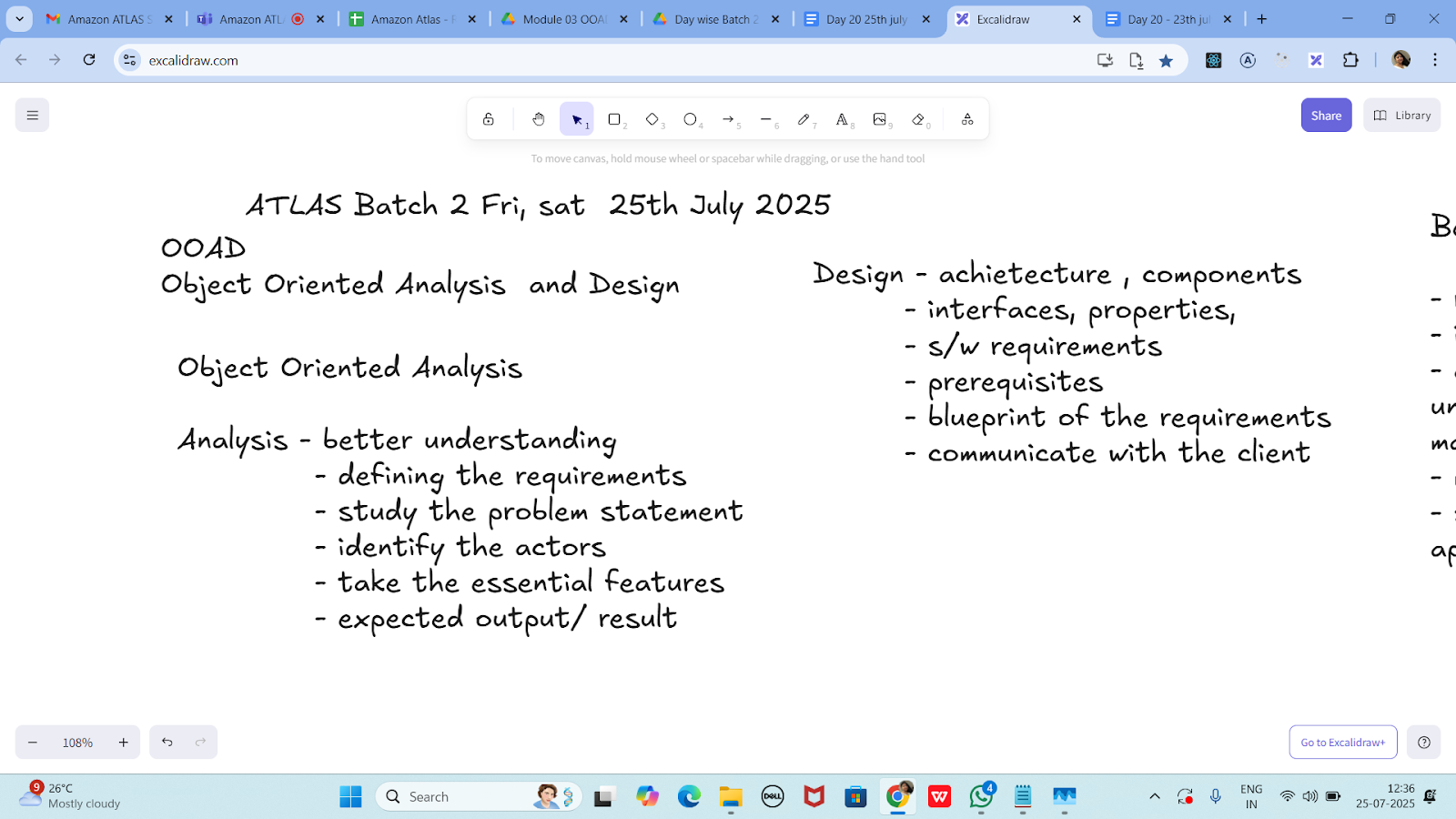
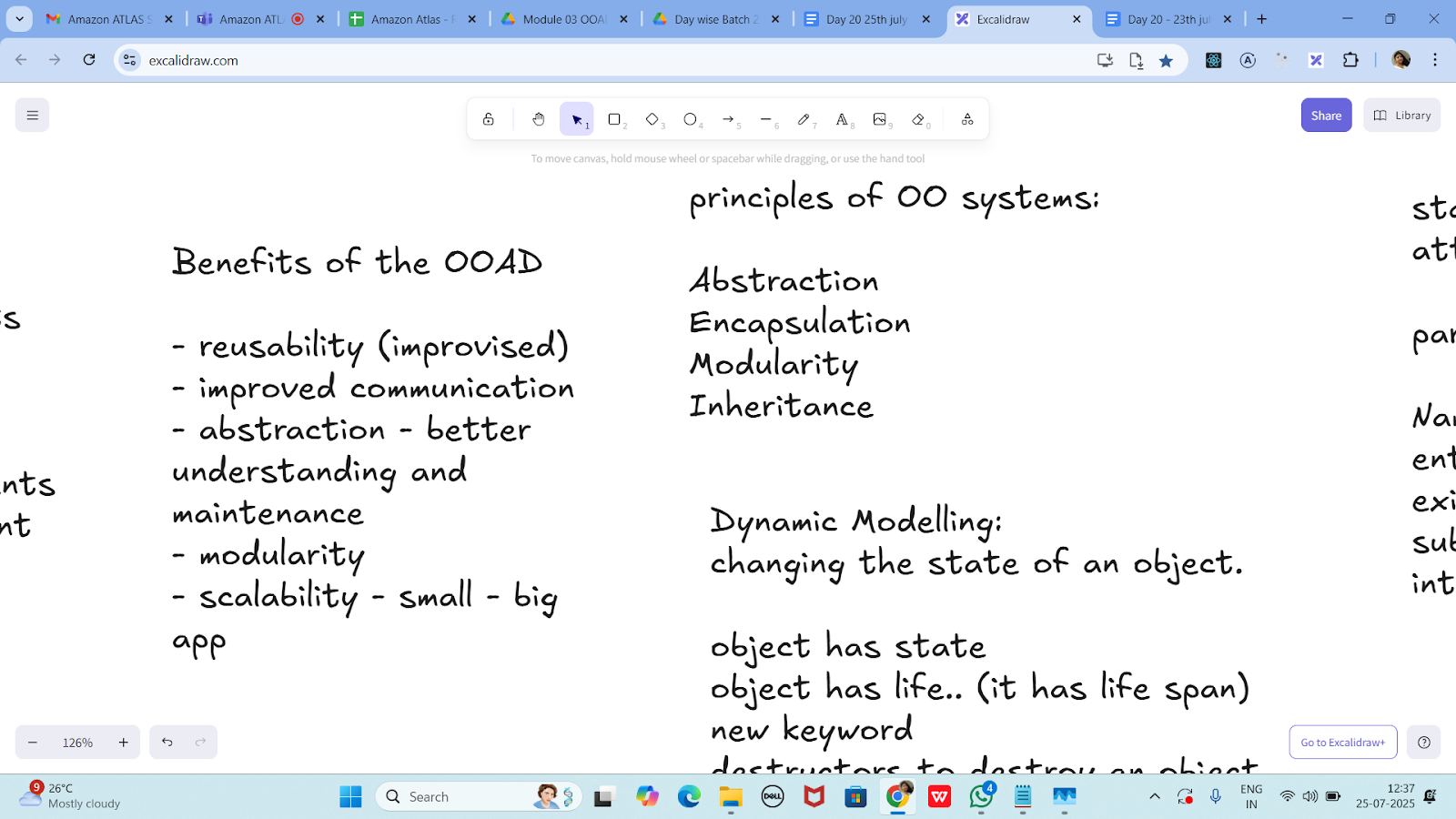
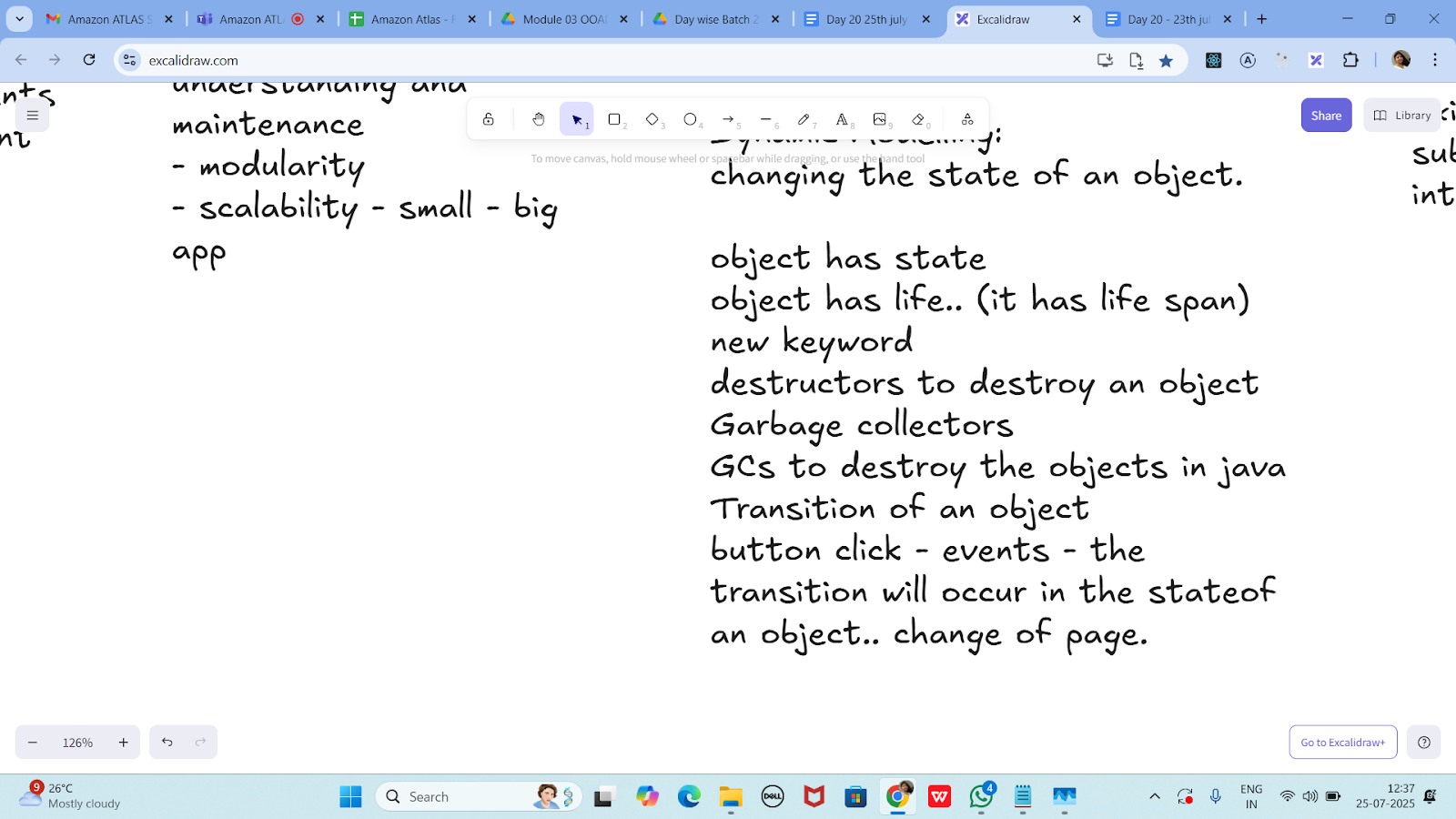
Day 20 25th july 2025

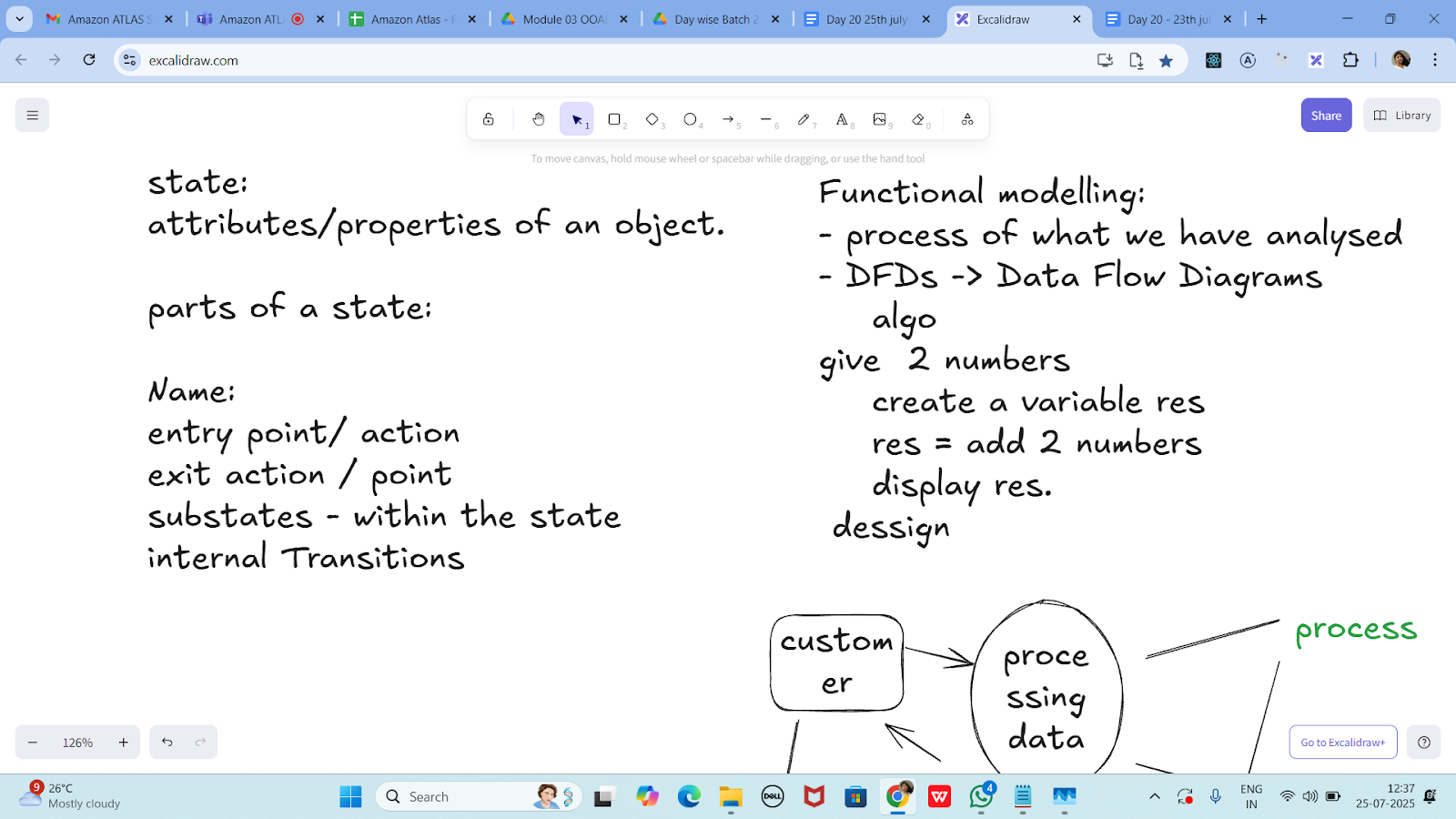
OOAD

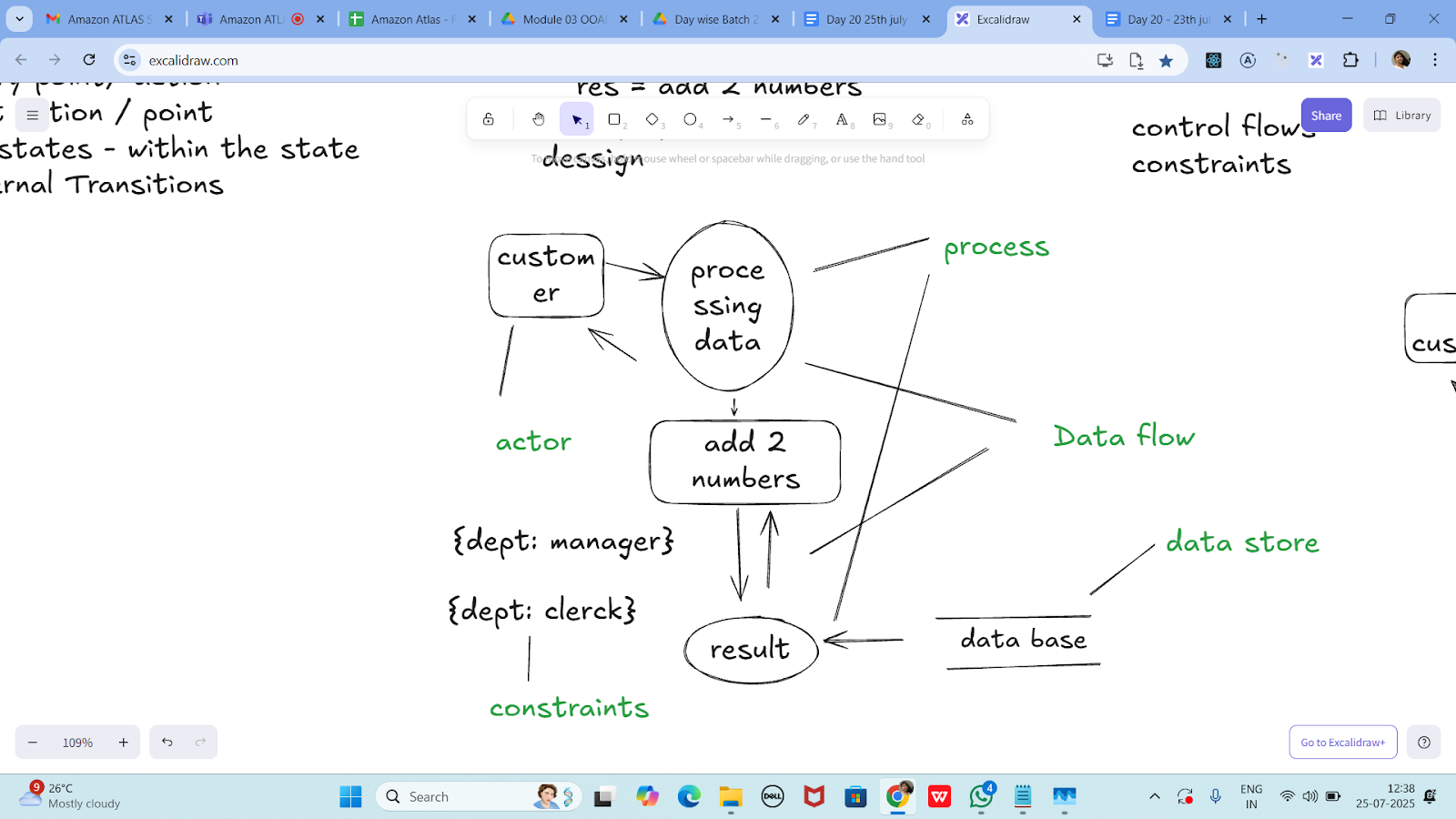
Object Oriented paradigm, Design Patterns, Design Workflow, Refactoring

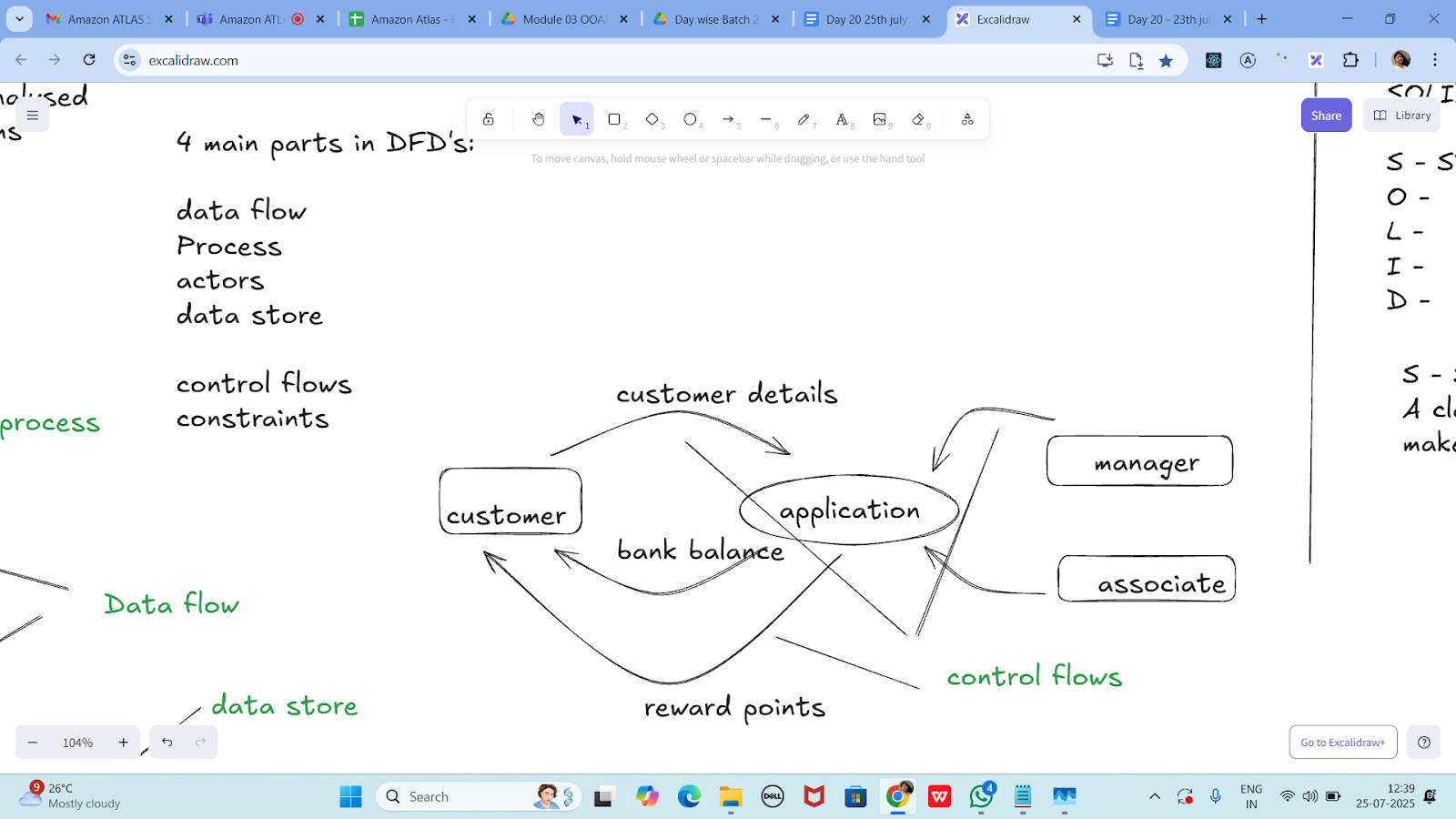


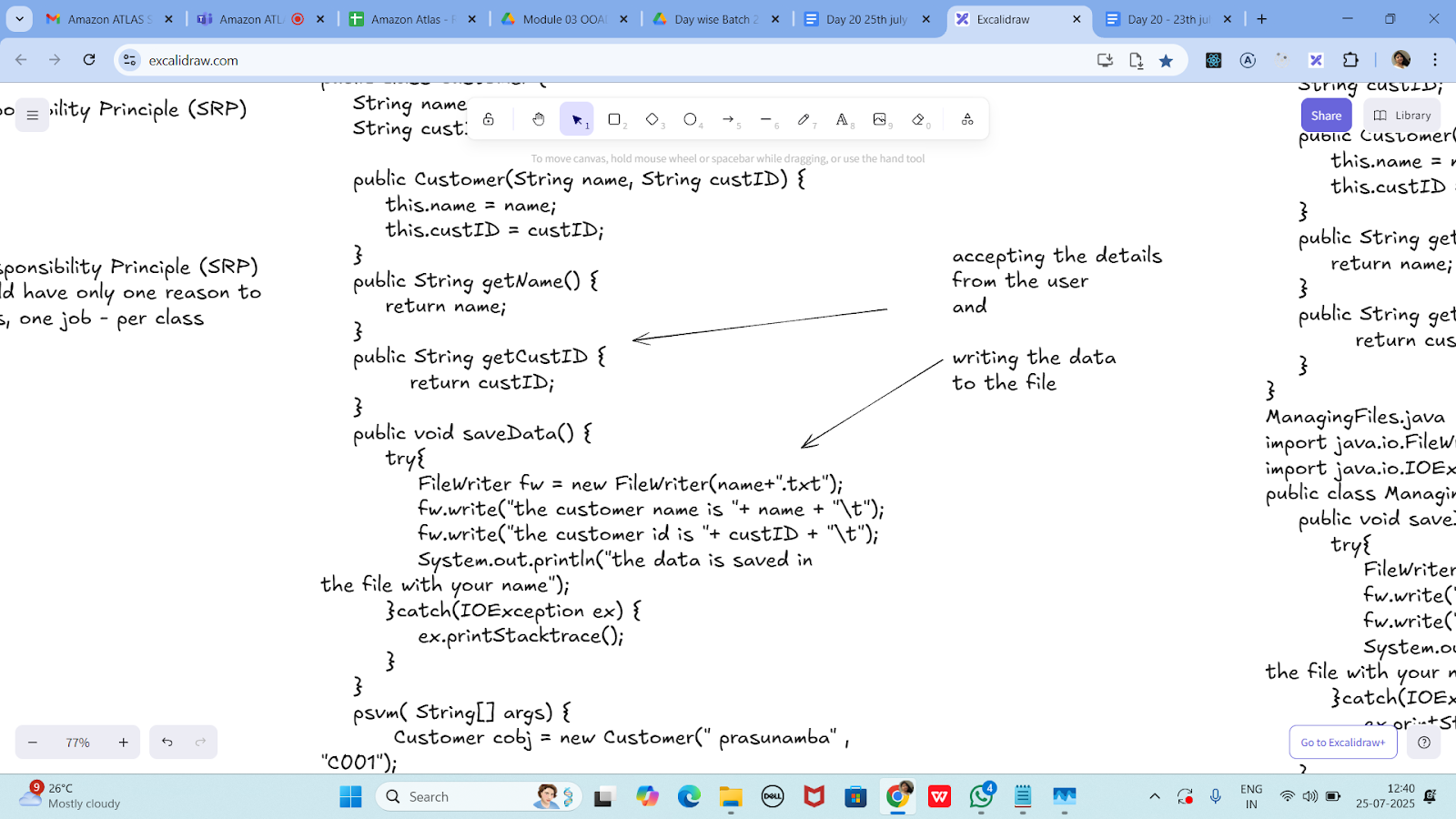












Without SRP:

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();

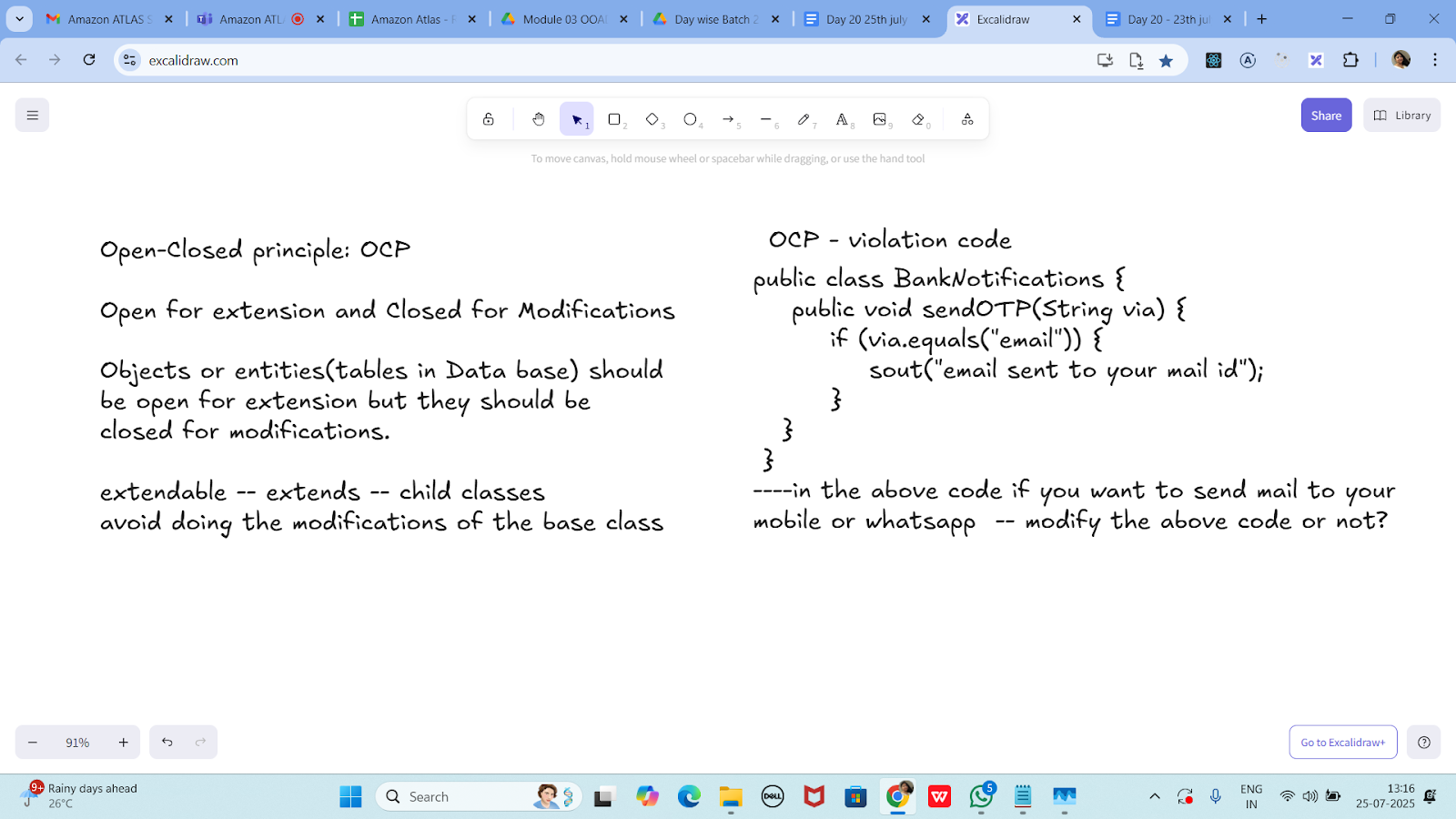
        mobj.saveData();

    }

}

—---------------------------------------------------------------------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

}

public class task03 {

    public static void main(String[] args) {

        Book book = new Book("Java Book", "Me", 200);

        BookFormatter formatter = new BookFormatter();

        PriceCalculator calculator = new PriceCalculator();

        System.out.println(formatter.formatTitle(book));

        System.out.println(formatter.formatBookInfo(book));

        System.out.println("Discounted Price: $" +

            calculator.calculateDiscountedPrice(book, 0.2));

    }

}

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 getTitle() {

        return title;

    }

    public String getAuthor() {

        return author;

    }

    public double getPrice() {

        return price;

    }

}

class BookFormatter {

    public String formatTitle(Book book) {

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

    }

    public String formatBookInfo(Book book) {

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

               "\nAuthor: " + book.getAuthor() +

               "\nPrice: $" + book.getPrice();

    }

}

class PriceCalculator {

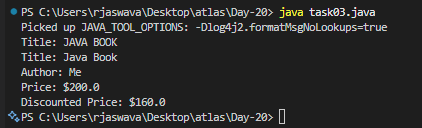
    public double calculateDiscountedPrice(Book book, double discountPercentage) {

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

    }

}

Output:



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.

class task04 {

    public static void main(String[] args) {

        Employee employee = new Employee("Jaswanth", "jaswanth@gmail.com", 50000.0);

        PdfReportGenerator reportGenerator = new PdfReportGenerator();

        EmailService emailService = new EmailService();

        reportGenerator.generatePdfReport(employee);

        emailService.sendEmail(employee);

    }

}

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;

    }

    public String getName() {

        return name;

    }

    public String getEmail() {

        return email;

    }

    public double getSalary() {

        return salary;

    }

}

class PdfReportGenerator {

    public void generatePdfReport(Employee employee) {

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

    }

}

class EmailService {

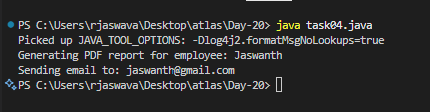
    public void sendEmail(Employee employee) {

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

    }

}

Output:



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();

  }

}

class Shape {

    public double area() {

        return 0.0;

    }

}

class Square extends Shape {

    int height;

    public Square(int height) {

        this.height = height;

    }

    @Override

    public double area() {

        return height \* height;

    }

}

class Circle extends Shape {

    int r;

    public Circle(int r) {

        this.r = r;

    }

    @Override

    public double area() {

        return Math.PI \* r \* r;

    }

}

class OpenClosedExample {

    public double compareArea(Shape shape1, Shape shape2) {

        return shape1.area() - shape2.area();

    }

}

class task05 {

    public static void main(String[] args) {

        Square square1 = new Square(5);

        Square square2 = new Square(4);

        Circle circle1 = new Circle(3);

        OpenClosedExample example = new OpenClosedExample();

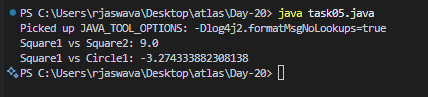
        System.out.println("Square1 vs Square2: " + example.compareArea(square1, square2));

        System.out.println("Square1 vs Circle1: " + example.compareArea(square1, circle1));

    }

}

Output:



Task 06:

Can you guys create diagrams for structural diagrams…

**CLASS DIAGRAM**

id: int

name: String

email: String

getId()

getName()

getEmail()

setName()

setEmail()

COMPONENT DIAGRAM Example

Web Browser

HTTP

Web Server

API Database

COMPOSITE STRUCTURE DIAGRAM Example

Computer

CPU RAM

GPU HDD

PACKAGE DIAGRAM Example

com.app

com.app.ui

com.app.util

com.app.data

DEPLOYMENT DIAGRAM Example

Client PC

Web Browser

HTTPS

Web Server

Apache/Nginx

Database Server

MySQL