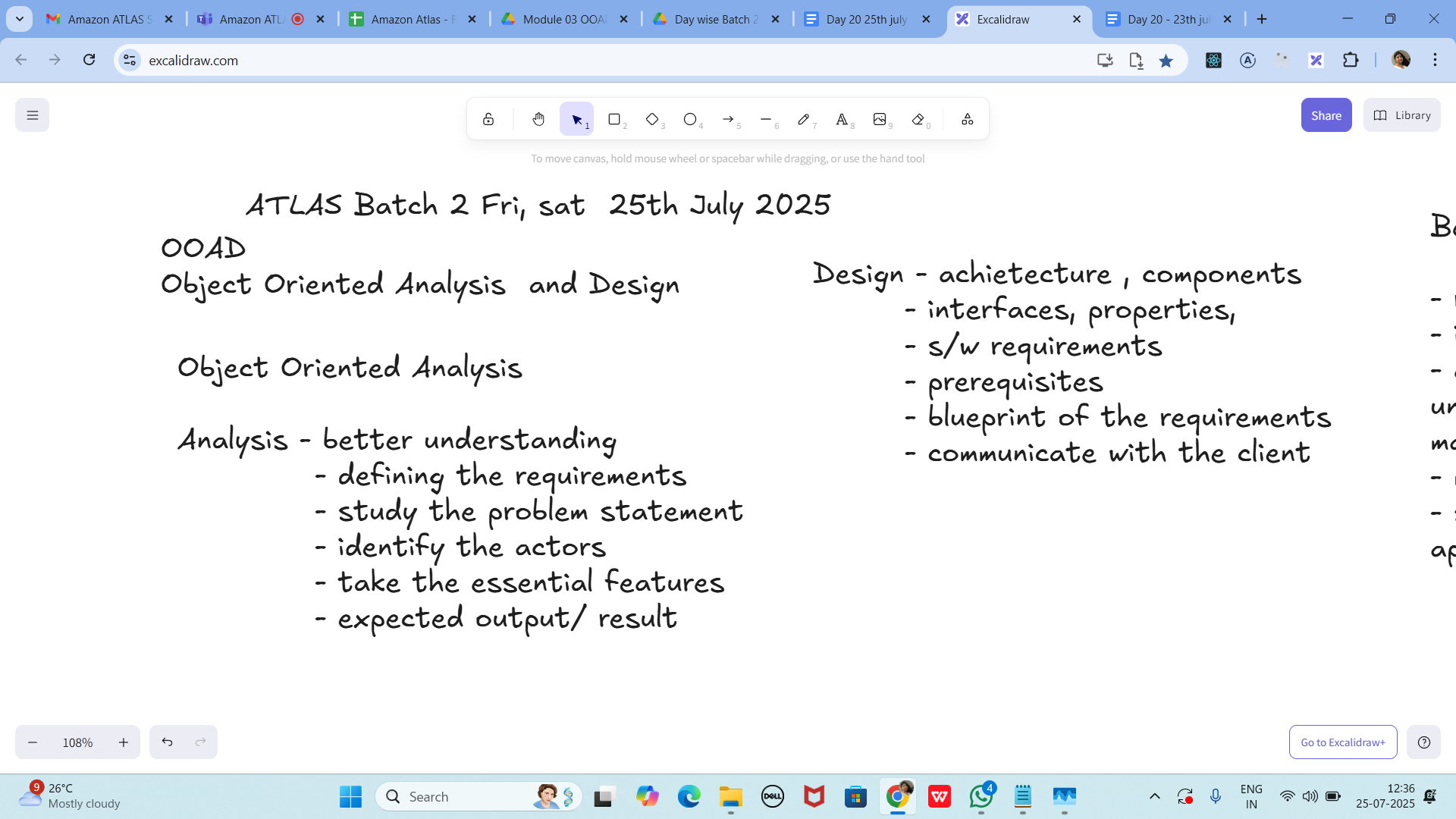
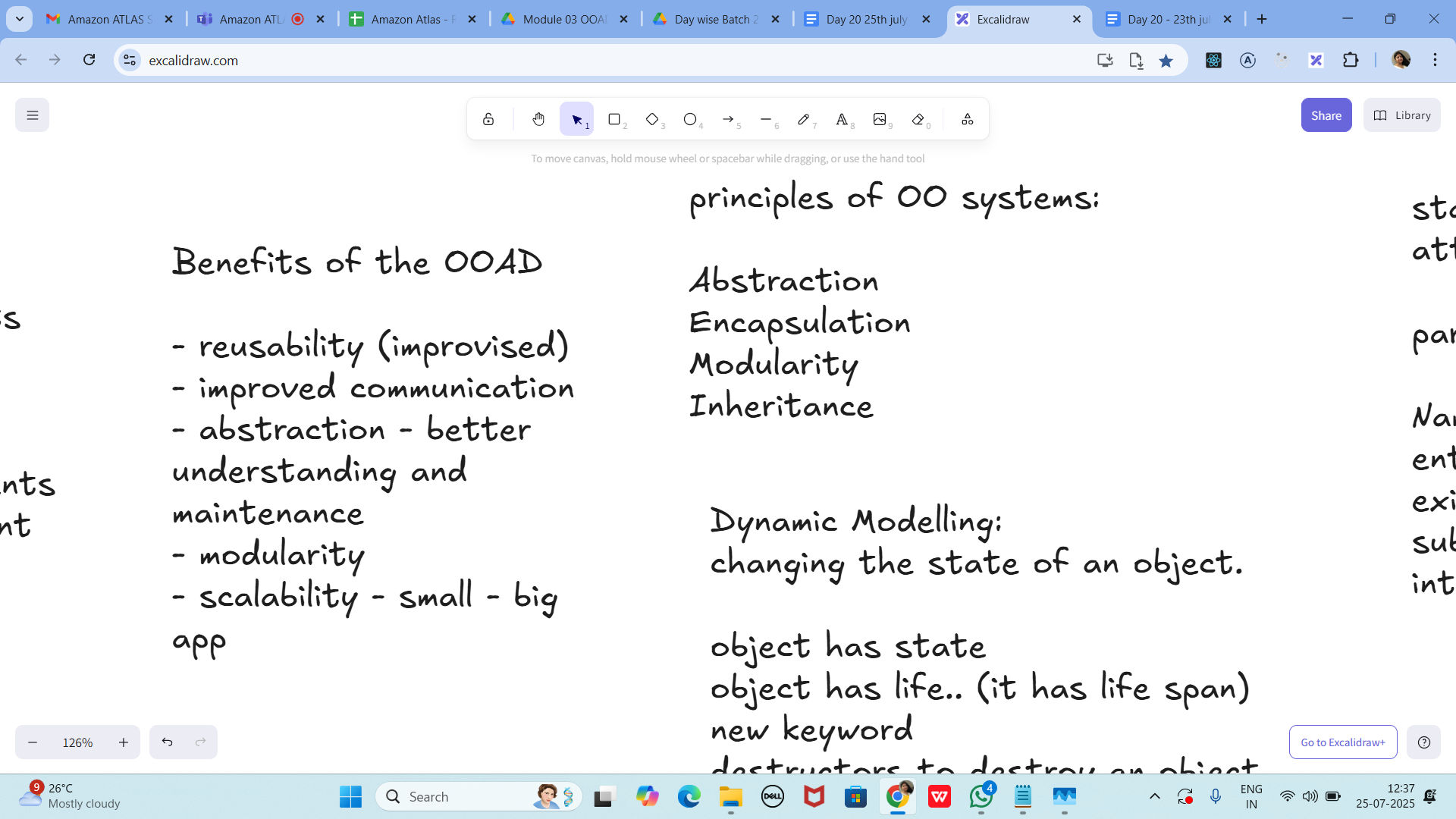
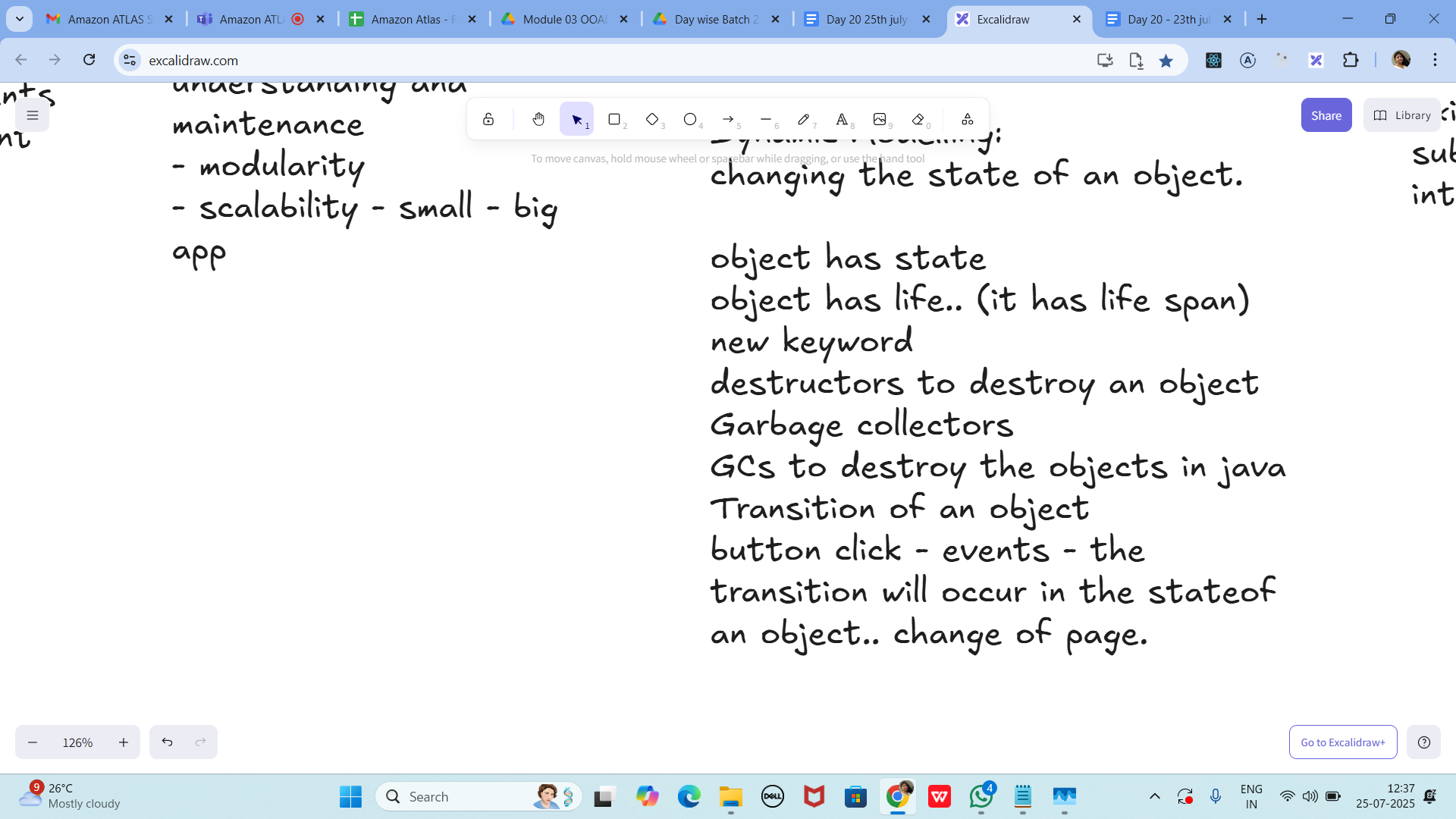
Day 20 25th july 2025

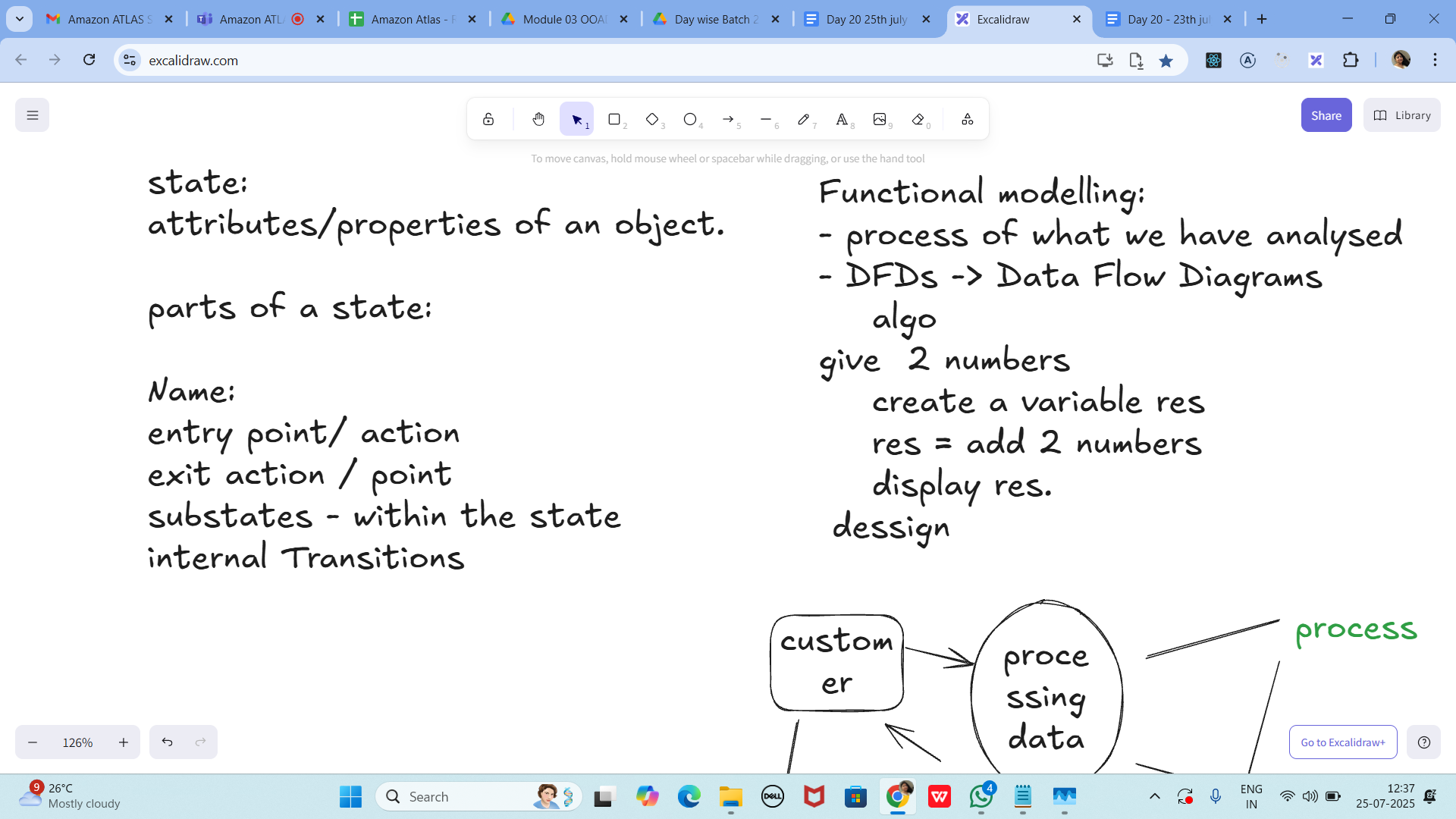
OOAD

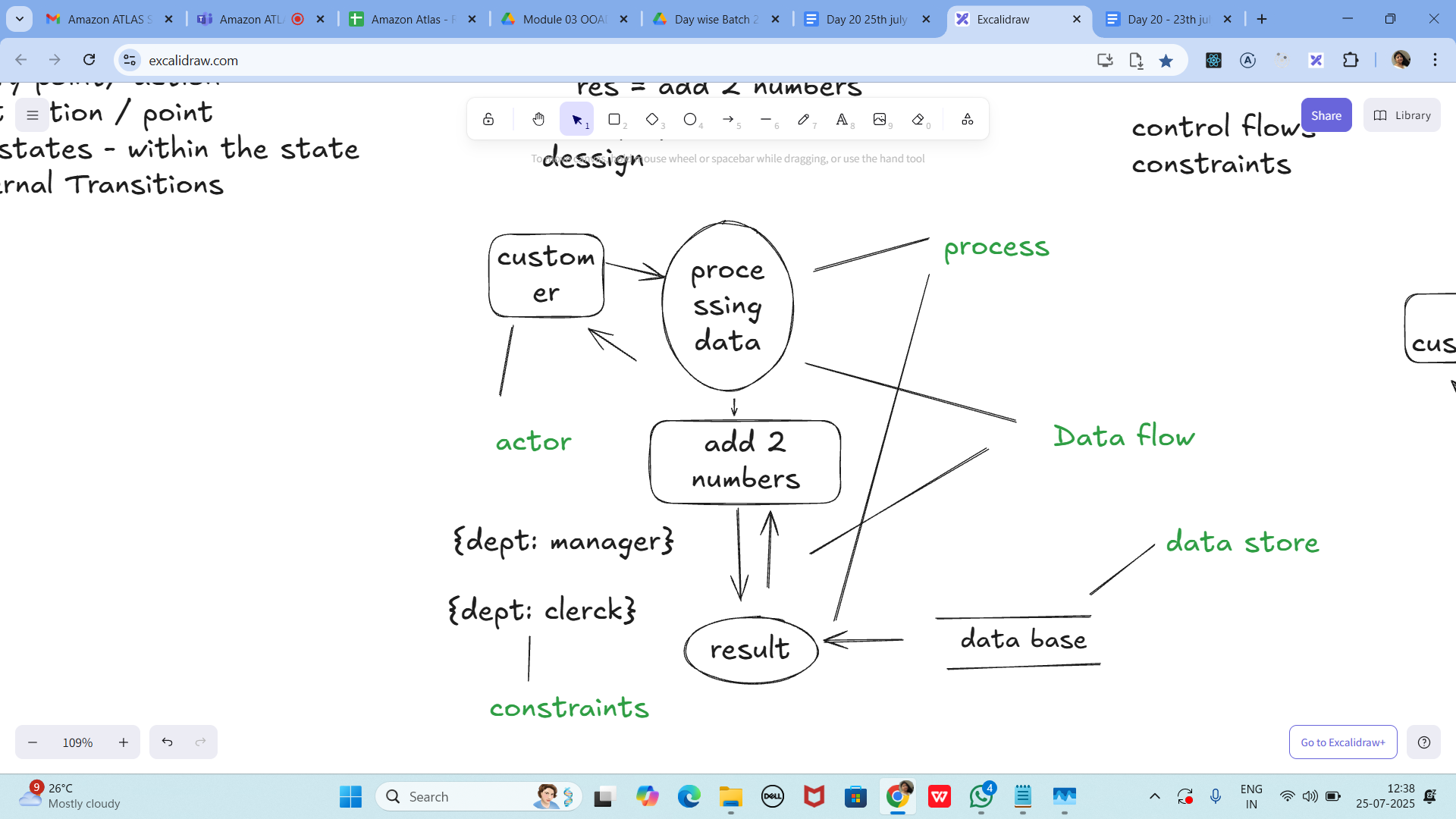
Object Oriented paradigm, Design Patterns, Design Workflow, Refactoring

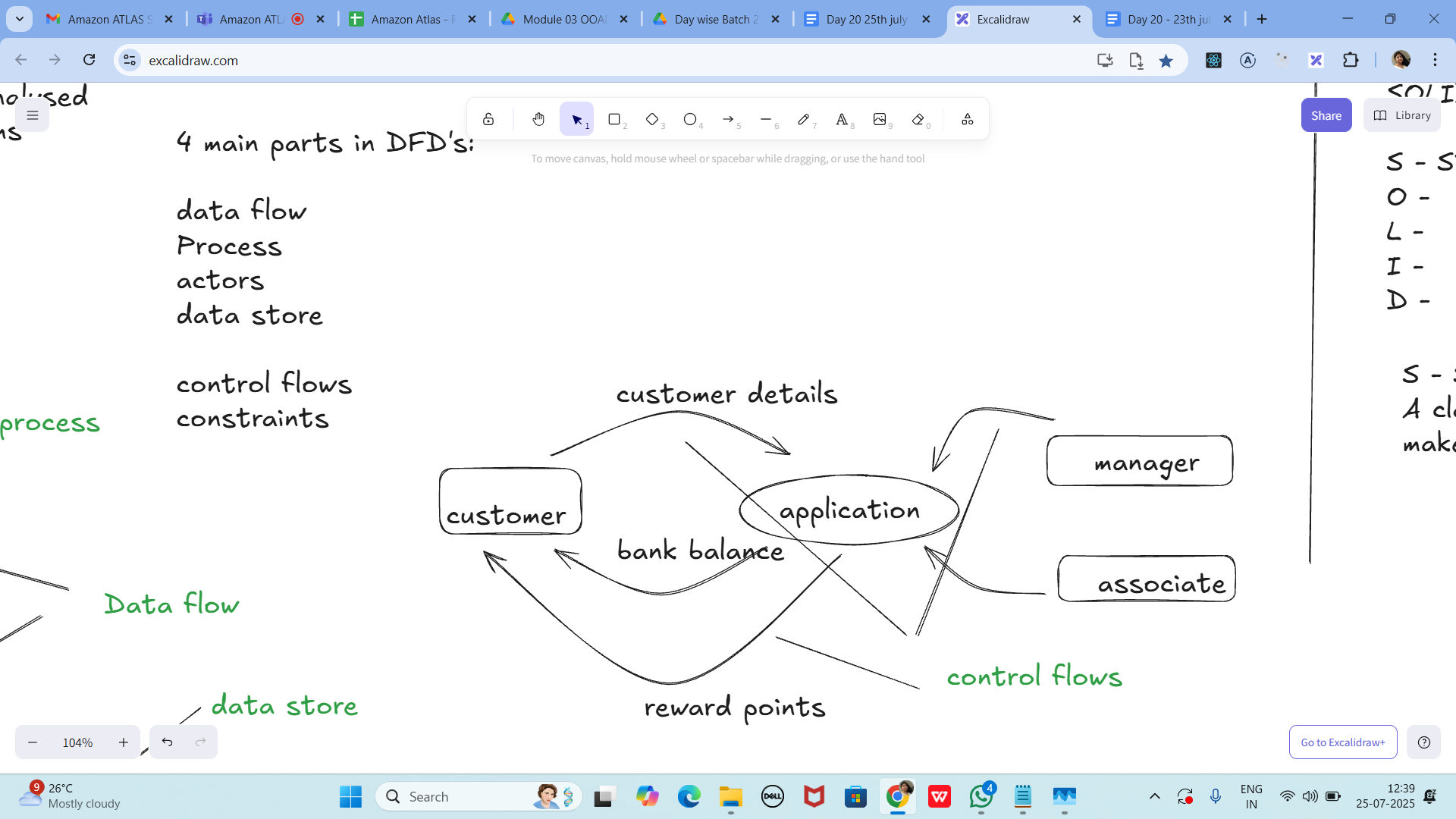


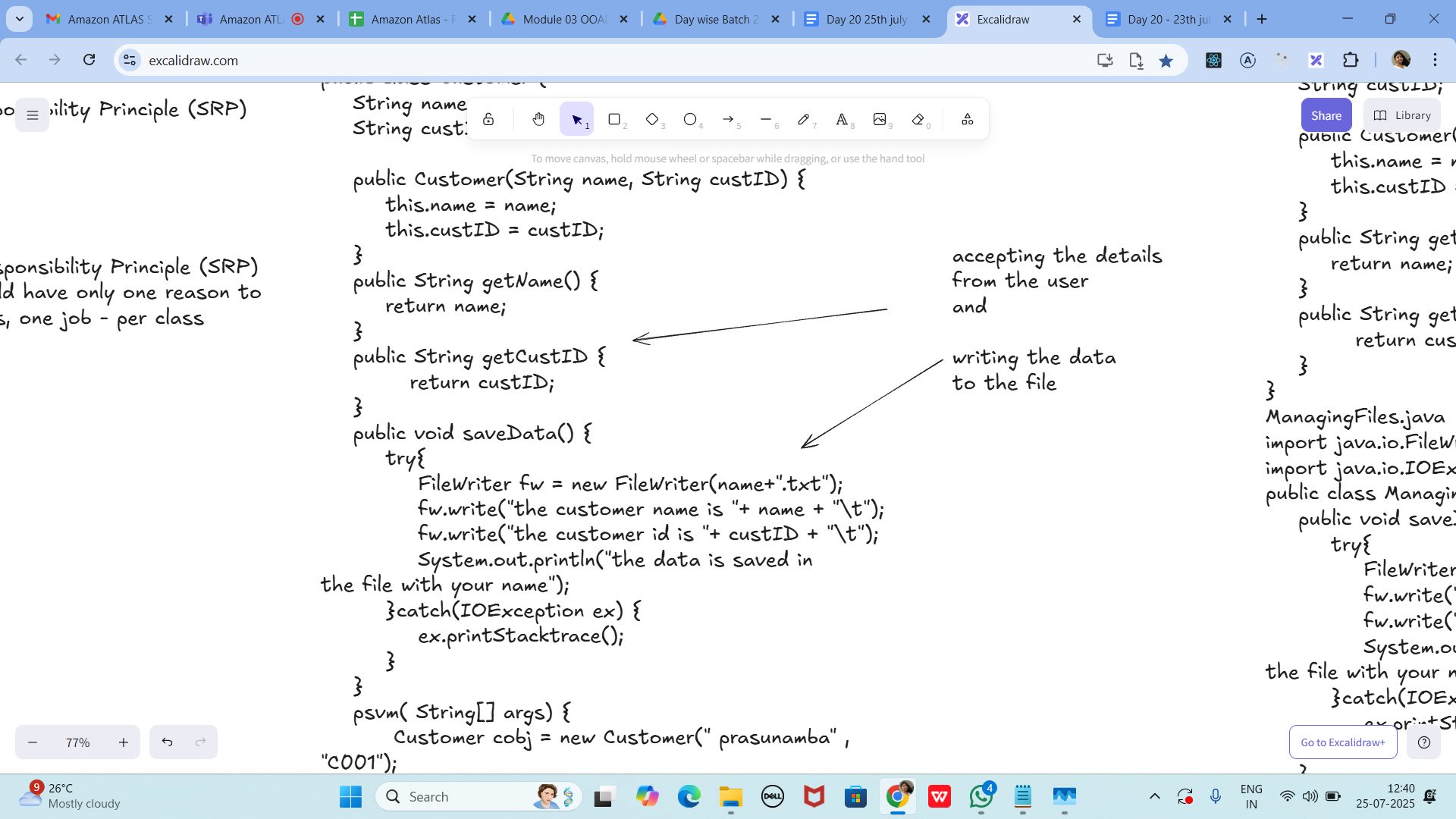












Without SRP:

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();

}

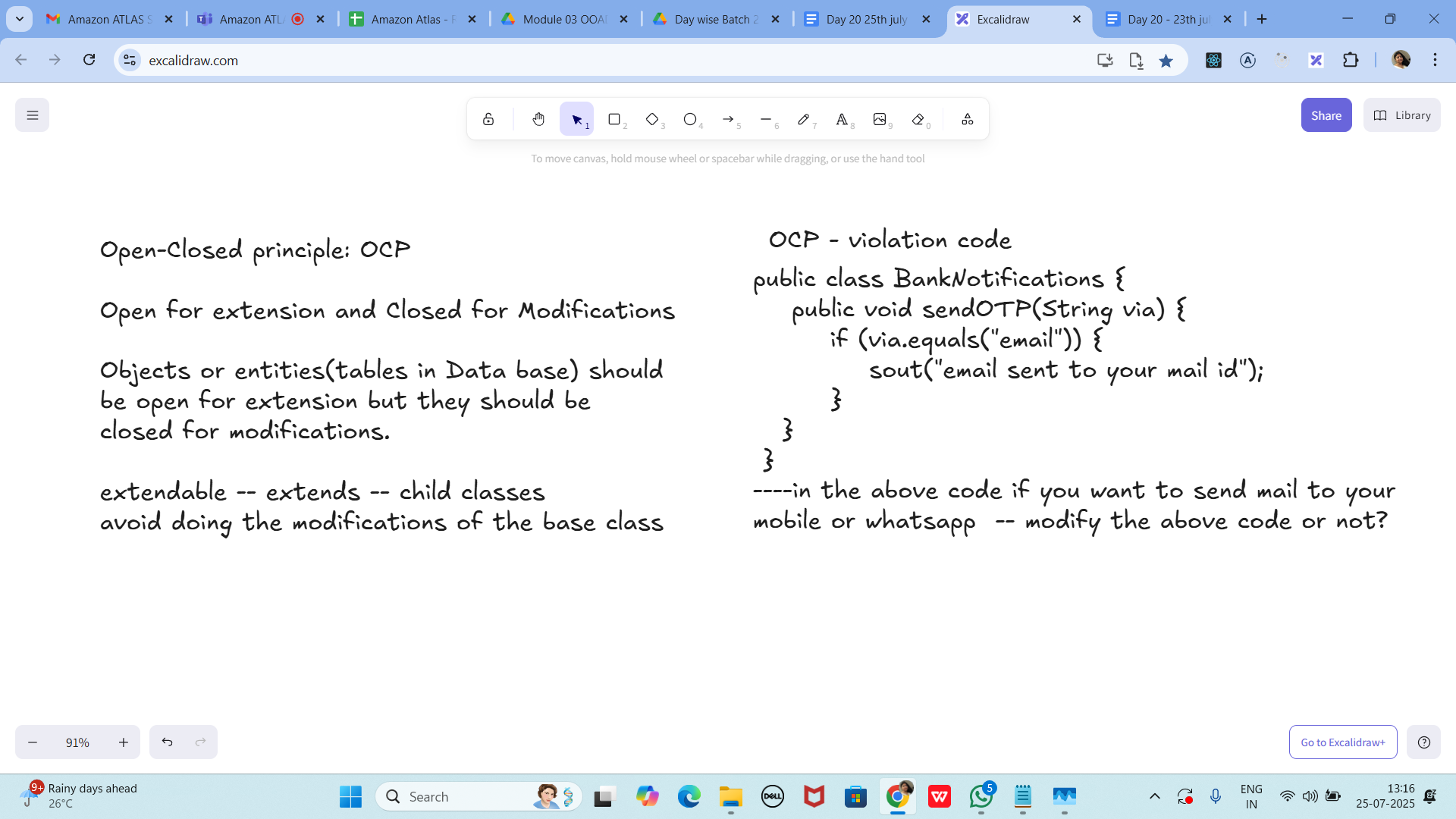
}

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

Answer: Fix by passing Customer to saveData(...), making fields private, using try-with-

resources, and correct main/printStackTrace()—see code above.

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..

public interface BankNotification { void sendOTP(String msg, String recipient); }

class EmailNotify implements BankNotification {

public void sendOTP(String msg, String recipient){

System.out.println(&quot;Email to &quot;+recipient+&quot;: &quot;+msg);

}

}

class MobileNotify implements BankNotification {

public void sendOTP(String msg, String recipient){

System.out.println(&quot;SMS to &quot;+recipient+&quot;: &quot;+msg);

}

}

class WhatsappNotify implements BankNotification {

public void sendOTP(String msg, String recipient){

System.out.println(&quot;WhatsApp to &quot;+recipient+&quot;: &quot;+msg);

}

}

// Add PhysicalMailNotify by implementing BankNotification (no changes elsewhere).

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

}

// srp implementation

public class BookDetails {

private String title;

private String author;

private double price;

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

this.title = title;

this.author = author;

this.price = price;

}

// ... getters and setters for book details

}

public class BookFormatter {

public String formatTitle(String title) {

return &quot;Title: &quot; + title.toUpperCase();

}

}

public class PriceCalculator {

public double calculateDiscountedPrice(double originalPrice, double discountPercentage) {

return originalPrice \* (1 - discountPercentage);

}

}

Answer:

public class BookDetails {

private final String title, author; private final double price;

public BookDetails(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; }

}

public class BookFormatter {

public String formatTitle(BookDetails b){ return &quot;Title: &quot; + b.getTitle().toUpperCase(); }

}

public class PriceCalculator {

public double calculateDiscountedPrice(BookDetails b, double discountFraction){

return b.getPrice() \* (1 - discountFraction);

}

}

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.

// 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

}

}

Answer:

public class Employee {

private final String name, email; private final double salary;

public Employee(String n,String e,double s){ name=n; email=e; salary=s; }

public String getName(){ return name; } public String getEmail(){ return email; } public double

getSalary(){ return salary; }

}

interface ReportGenerator { byte[] generate(Employee e); }

class PdfReportGenerator implements ReportGenerator {

public byte[] generate(Employee e){ return (&quot;REPORT: &quot;+e.getName()).getBytes(); }

}

interface EmailSender { void send(String to, String subject, String body, byte[] attachment); }

class SmtpEmailSender implements EmailSender {

public void send(String to,String subject,String body,byte[] attachment){

System.out.println(&quot;Email sent to &quot;+to+&quot; : &quot;+subject);

}

}

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();

}

}

Square1 vs Square2: -24.0

Circle1 vs Circle2: -21.991148575128552

Square1 vs Circle1: -3.274333882308138

UML:

UML - Unified modeling Language

plantUML:

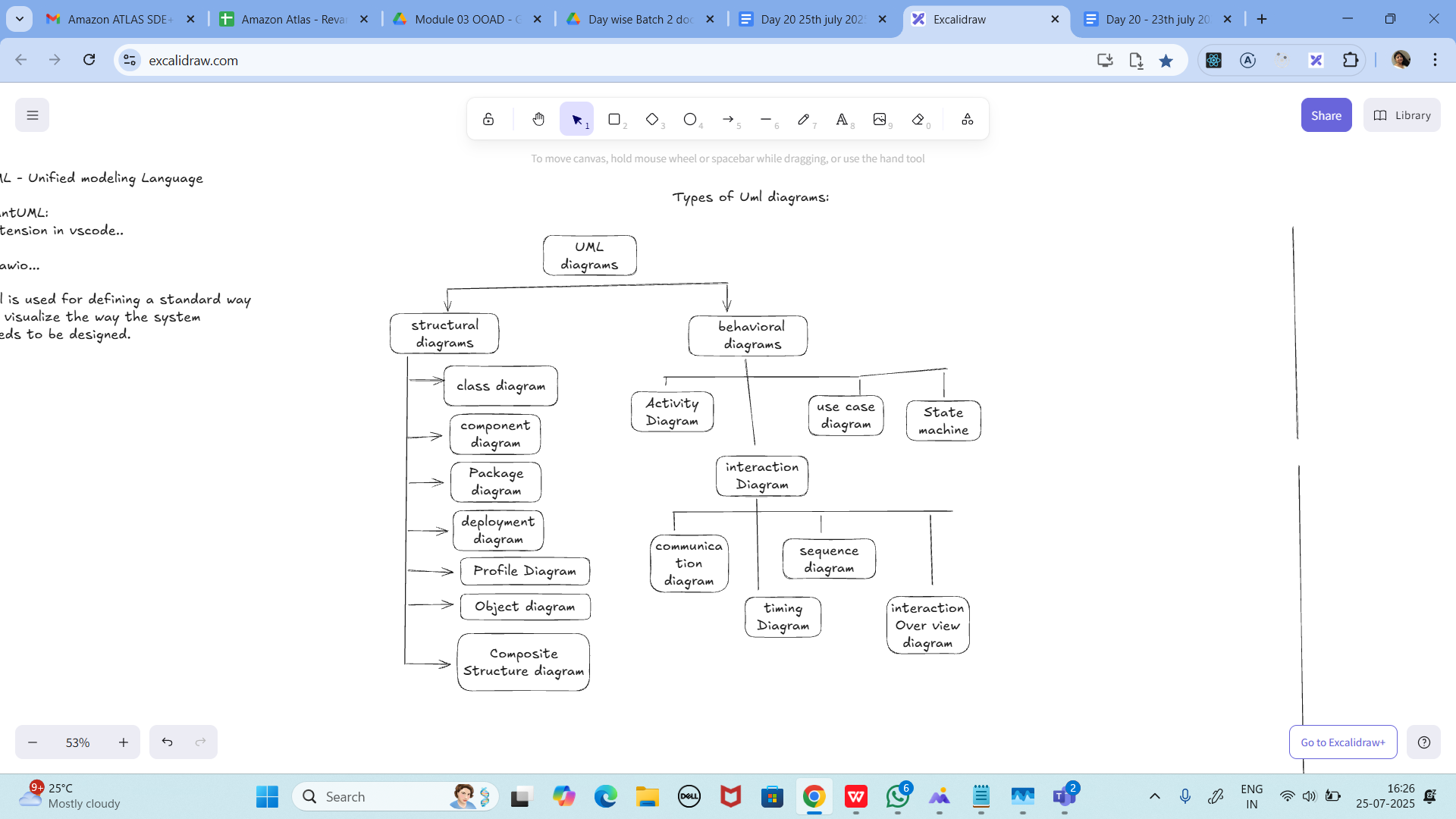
extension in vscode..

Drawio...

uml is used for defining a standard way

to visualize the way the system

needs to be designed.



Answer: Use Class/Component/Deployment for structural; Use Case/Sequence/Activity/State

for behavioural.

Task 06:

Can you guys create diagrams for structural diagrams…

Here are the structural diagrams (Class Diagrams) for both implementations:

1. \*\*Book Implementation Class Diagram\*\*:

```

+-------------------+

| Book |

+-------------------+

| - title: String |

| - author: String |

| - price: double |

+-------------------+

| + getTitle() |

| + getAuthor() |

| + getPrice() |

+-------------------+

▲

|

| uses

|

+-------------------+ +--------------------+

| BookFormatter | | PriceCalculator |

+-------------------+ +--------------------+

| + formatTitle() | | + calculateDiscount|

| + formatAuthor() | | edPrice() |

| + formatDetails() | | |

+-------------------+ +--------------------+

▲ ▲

| |

| uses |

| |

+-------------------+

| BookStore |

+-------------------+

| - formatter |

| - calculator |

+-------------------+

| + displayDetails()|

| + applyDiscount() |

+-------------------+

```

2. \*\*Shape Implementation Class Diagram\*\*:

```

+-------------------+

| <<interface>> |

| Shape |

+-------------------+

| + area() |

+-------------------+

▲

|

implements

|

+-------------------+ +-------------------+

| Square | | Circle |

+-------------------+ +-------------------+

| - height: double | | - radius: double |

+-------------------+ +-------------------+

| + area() | | + area() |

+-------------------+ +-------------------+

▲ ▲

| |

| uses |

| |

+-------------------+

| AreaComparator |

+-------------------+

| + compareAreas() |

| + getLarger() |

| + sortByArea() |

+-------------------+

```

Sequence Diagram for Book Implementation:

```

Actor -> BookStore: displayDetails(book)

BookStore -> BookFormatter: formatDetails(book)

BookFormatter -> Book: getTitle()

Book --> BookFormatter: title

BookFormatter -> Book: getAuthor()

Book --> BookFormatter: author

BookFormatter -> Book: getPrice()

Book --> BookFormatter: price

BookFormatter --> BookStore: formatted string

BookStore --> Actor: display result

Actor -> BookStore: applyDiscount(book)

BookStore -> PriceCalculator: calculateDiscountedPrice(book)

PriceCalculator -> Book: getPrice()

Book --> PriceCalculator: price

PriceCalculator --> BookStore: discounted price

BookStore --> Actor: display result

```

Component Diagram:

```

+------------------+

| Client |

+------------------+

|

v

+------------------+

| BookStore |

+------------------+

| |

v v

+---------+ +---------+

|Formatter| |Calculator|

+---------+ +---------+

| |

v v

+------------------+

| Book |

+------------------+

```

Package Diagram:

```

+----------------------+

| com.bookstore |

+----------------------+

| + Book |

| + BookFormatter |

| + PriceCalculator |

| + BookStore |

| + Main |

+----------------------+

```

These diagrams help visualize:

1. Class relationships and dependencies

2. Interface implementations

3. Method calls and data flow

4. Component organization

5. Package structure

Key points shown in diagrams:

- Clear separation of responsibilities

- Dependencies between classes

- Interface implementations

- Data flow between components

- System organization

Benefits of using these diagrams:

1. Better understanding of system architecture

2. Easy to communicate design to team members

3. Helps identify potential issues

4. Makes maintenance easier

5. Guides implementation

You can create these diagrams using tools like:

- PlantUML

- Draw.io

- Lucidchart

- Visual Paradigm

- Enterprise Architect

Home Task 1

Wap to create a class Student.. With 3 methods

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

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

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

Answer:

public class Student {

private final String id, name, course;

private final int[] marks;

private final double baseFee;

public Student(String id,String name,String course,int[] marks,double baseFee){

this.id=id; this.name=name; this.course=course; this.marks=marks; this.baseFee=baseFee;

}

public void registrationDetails(){

System.out.println(&quot;ID: &quot;+id+&quot;, Name: &quot;+name+&quot;, Course: &quot;+course);

}

public double marksCalc(){

if(marks==null||marks.length==0) return 0;

int sum=0; for(int m:marks) sum+=m;

double avg = (double)sum/marks.length;

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

return avg;

}

public double feesCalc(){

double avg = marksCalc();

double fee = (avg&gt;=85)? baseFee\*0.8 : baseFee;

System.out.println(&quot;Payable Fee: &quot;+fee);

return fee;

}

}

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

**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>

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

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