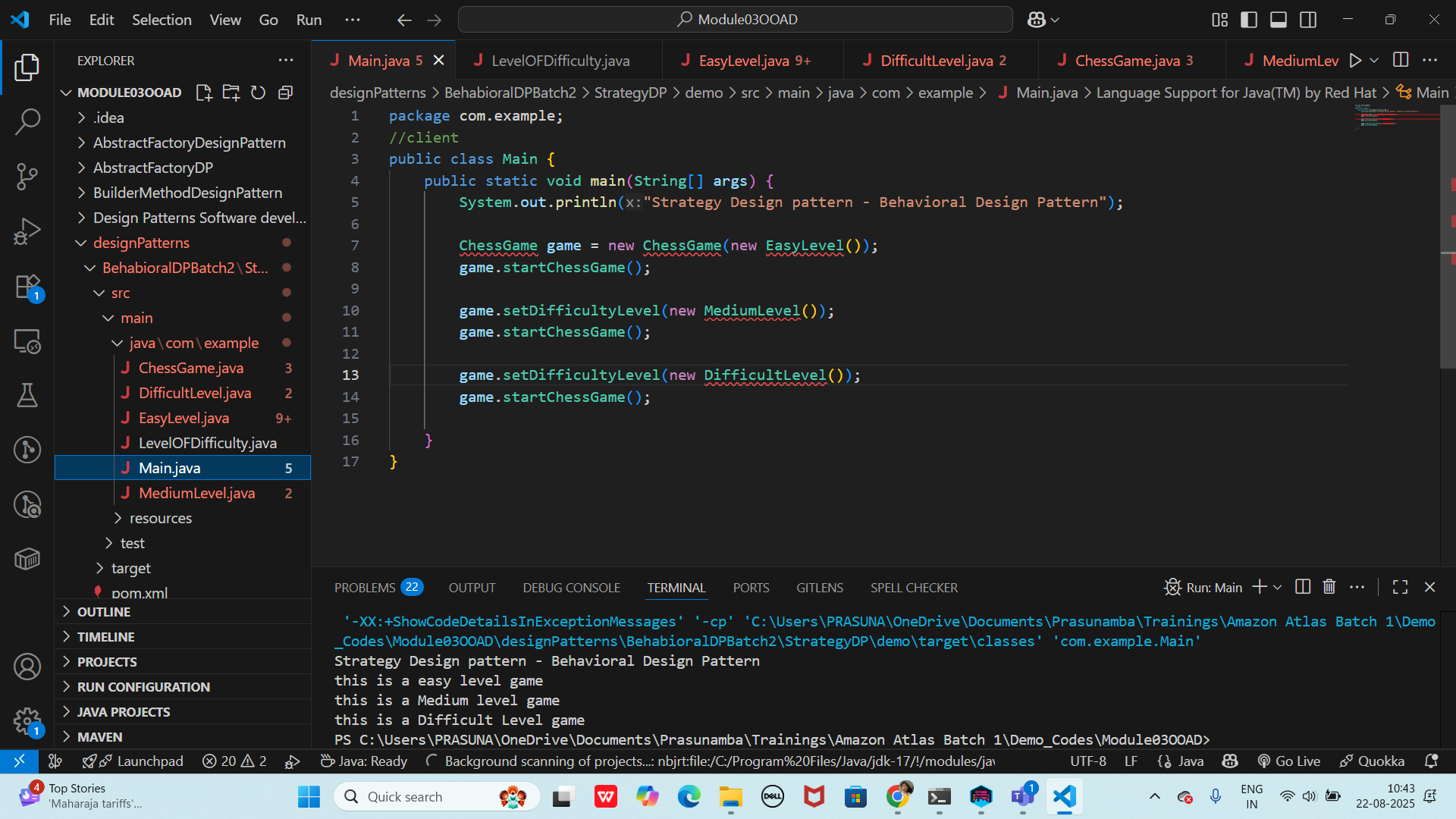
Day 26 - 22nd Aug 2025

Strategy Method Design pattern



Strategy Method Design Pattern code is in

Code for reference —> OOAD .. StrategyMethodDP\_2

Code for reference link

<https://drive.google.com/drive/folders/1LwhNov1s1-vHzF9GPAObLSnP9kAvipmw?usp=sharing>

— 10.44 to 10.47

Pub Sub

Home task:

Code is in Code for reference → ooad → pubsubpattern\_2 folder..

Rating Qns:

Task 2:

Answer: A design workflow is the structured sequence of steps taken to transform

requirements into design artifacts. It includes requirements gathering, analysis, system

architecture, modeling (UML, diagrams), pattern application, prototyping, validation, and

documentation.

Task 3:

Answer: Persistent objects are objects whose state outlives the process that created them,

stored in databases or files. Debate includes persistence mechanisms (ORM, serialization),

advantages (durability, recovery), and trade-offs (performance, coupling).

Task 4:

Which of the following components is not typically part of the Command pattern?

a) Invoker

b) Receiver

c) Abstract Factory

d) Command (interface/abstract class)

Task 5:

What role does the Invoker play in the Command pattern?

a) It knows how to perform the operations associated with a request.

b) It encapsulates the request as an object.

c) It asks the command to carry out the request.

d) It defines the interface for executing an operation.

Task 6:

A key benefit of using the Command pattern is its ability to support:

a) Lazy initialization

b) Undo/Redo functionality

c) Singleton instance creation

d) Compile-time polymorphism

Task 7:

In the Strategy pattern, what role does the "Context" play?

A. It defines the interface for the algorithms.

B. It implements a specific algorithm.

C. It maintains a reference to a Strategy object and delegates the task to it.

D. It creates the Concrete Strategy objects.

Task 8:

1. In which of the following mechanisms, types of all variables and expressions are fixed at compilation time.

a) Strong Typing

b) Weak Typing

c) Static Binding/ early binding

d) Dynamic Binding/ late binding

Task 9:

In which pattern does a class represent the functionality of another class, providing a simplified interface to a complex subsystem?

a) Decorator Pattern

b) Facade Pattern

c) Proxy Pattern

d) Composite Pattern

Task 10:

Which of the following statements about Persistence is correct?

a) It is the enforcement of the class of an object, such that objects of different types may not be interchanged, or at the most they may be interchanged only in very restricted ways.

b) It is the property of an object through which its existence transcends time and/or space.

c) It is the property that distinguishes an active object from one that is not active.

d) All of the mentioned

Task 11:

What is that concept in type theory in which a single name may denote objects of many different classes that are related by some common super class referred to \_\_\_\_\_\_

a) Monomorphism

b) Type Checking

c) Polymorphism

d) Generalization

Task 12:

Which of the following patterns is used to create a single instance of a class and provide a global point of access to it?

a) Factory Pattern

b) Singleton Pattern

c) Builder Pattern

d) Prototype Pattern

Task 13:

The Adapter pattern is a type of \_\_\_\_\_\_ pattern.

a) Creational

b) Structural

c) Behavioral

d) Concurrency

Task 14:

Which design pattern defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically?

a) Strategy Pattern

b) Command Pattern

c) Observer Pattern

d) Mediator Pattern

Task 15:

The Model-View-Controller (MVC) is an example of a \_\_\_\_\_\_ pattern.

a) Creational

b) Structural

c) Behavioral

d) Architectural

Task 16:

Clas name DManager

Add , remove , retrieve.. Methods (list)

→ Declare private static variable ..- type Dmanager - make sure it holds singleton instance

→ create private constructor to prevent direct instantiation of class

→ create instance..

→ throw an illegalStaticException , create **singleton** instance if not created .. use getInstance()

→ public static synchronized method – getInstance()

→ return existing instance..

→ create a new instance if doesnot exist..

In the thread safe list management:

→ declare a private variable

→ initialize the list inside the constructor using new..

→ also implement public sync methods

→ addItem

→ removeitem

→ list

**Answer**:

import java.util.ArrayList;

import java.util.List;

public class DManager {

private static DManager instance;

private final List&lt;String&gt; list;

private DManager() {

list = new ArrayList&lt;&gt;();

}

public static synchronized DManager getInstance() {

if (instance == null) {

instance = new DManager();

}

return instance;

}

public synchronized void addItem(String item) {

list.add(item);

}

public synchronized void removeItem(String item) {

list.remove(item);

}

public synchronized List&lt;String&gt; retrieveItems() {

return new ArrayList&lt;&gt;(list);

}

}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Home Tasks:

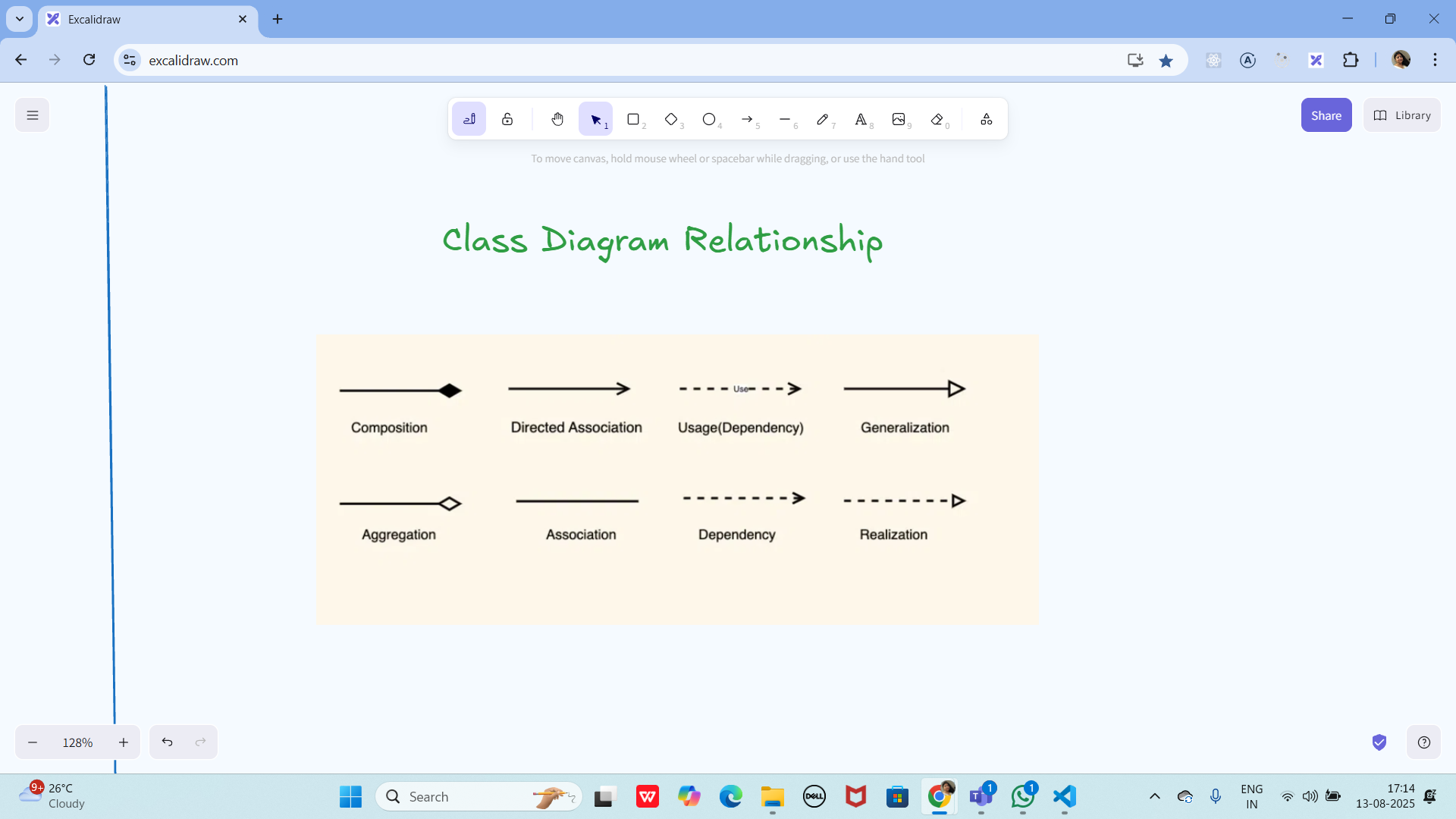
Task 1 :

Pub Subs code – complete it

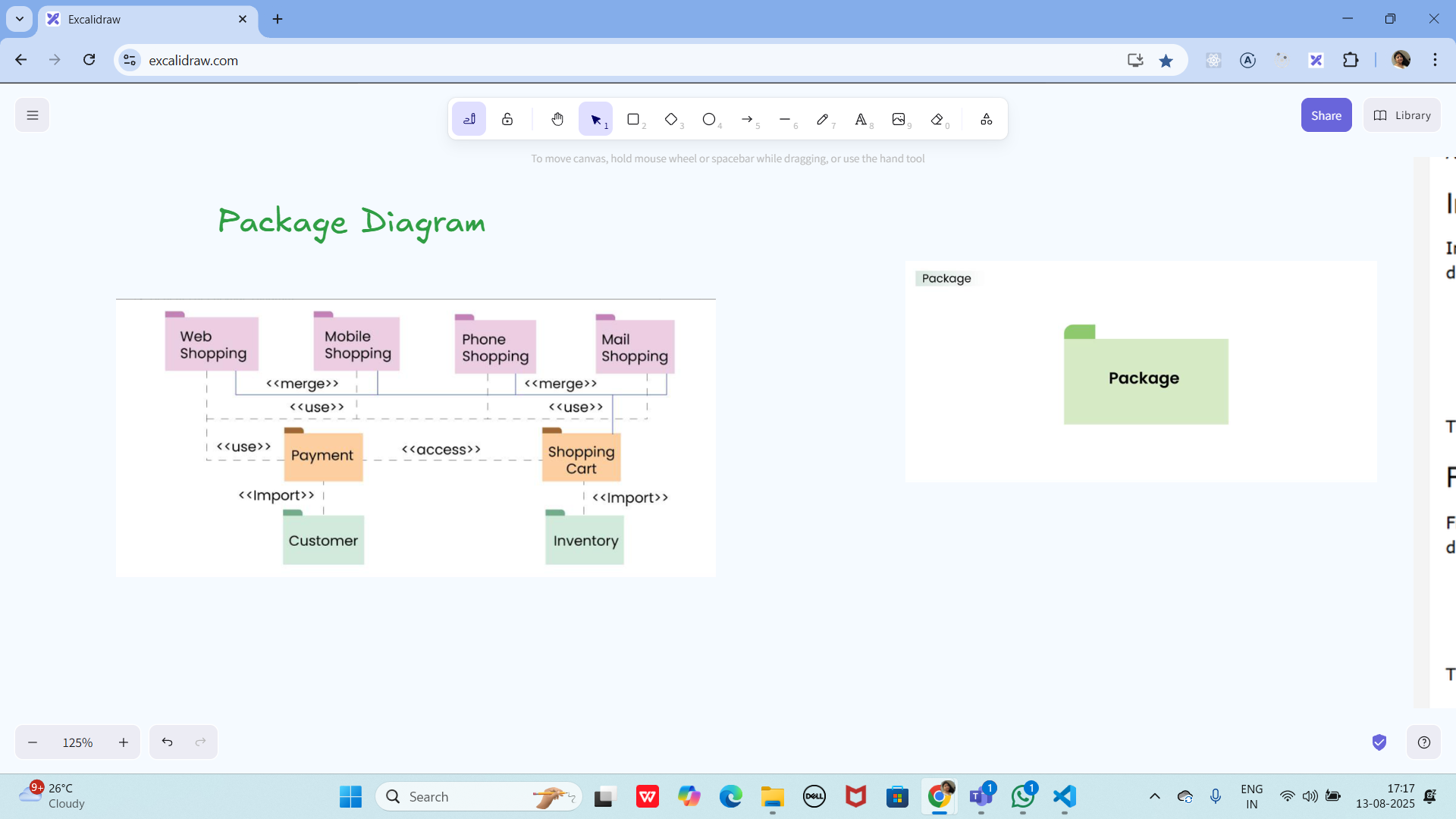
Task 2:

Practice singleton and Strategy design patterns …

Task 03:



Task 04:



Create a package diagram representation of Command Design Pattern code..

┌──────────────────────────────────────────────────────────────────────────────┐

│ com.example.texteditor │

├──────────────────────────────────────────────────────────────────────────────┤

│ │

│ ┌─────────────────────────────────────┐ ┌─────────────────────────────────┐ │

│ │ commands │ │ invokers │ │

│ ├─────────────────────────────────────┤ ├─────────────────────────────────┤ │

│ │ │ │ │ │

│ │ ┌─────────────────────────────┐ │ │ ┌─────────────────────────┐ │ │

│ │ │ <<interface>> │ │ │ │ MenuInvoker │ │ │

│ │ │ Command │ │ │ ├─────────────────────────┤ │ │

│ │ ├─────────────────────────────┤ │ │ │ - commands: List │ │ │

│ │ │ + execute(): void │ │ │ │ - current: int │ │ │

│ │ │ + undo(): void │ │ │ ├─────────────────────────┤ │ │

│ │ └─────────────────────────────┘ │ │ │ + setCommand() │ │ │

│ │ ▲ │ │ │ + executeCommand() │ │ │

│ │ │ │ │ │ + undoCommand() │ │ │

│ │ ┌───────────┴───────────┐ │ │ └─────────────────────────┘ │ │

│ │ │ │ │ │ │ │

│ │ ▼ ▼ │ │ ┌─────────────────────────┐ │ │

│ │ ┌─────────────────┐ ┌──────────────┐│ │ │ ToolbarInvoker │ │ │

│ │ │ CopyCommand │ │ PasteCommand ││ │ ├─────────────────────────┤ │ │

│ │ ├─────────────────┤ ├──────────────┤│ │ │ - buttonCommands: Map │ │ │

│ │ │ - receiver │ │ - receiver ││ │ ├─────────────────────────┤ │ │

│ │ ├─────────────────┤ ├──────────────┤│ │ │ + addButton() │ │ │

│ │ │ + execute() │ │ + execute() ││ │ │ + onButtonClick() │ │ │

│ │ │ + undo() │ │ + undo() ││ │ └─────────────────────────┘ │ │

│ │ └─────────────────┘ └──────────────┘│ │ │ │

│ │ │ └─────────────────────────────────┘ │

│ │ ┌─────────────────┐ ┌──────────────┐│ │

│ │ │ CutCommand │ │ UndoCommand ││ ┌─────────────────────────────────┐ │

│ │ ├─────────────────┤ ├──────────────┤│ │ receivers │ │

│ │ │ - receiver │ │ - receiver ││ ├─────────────────────────────────┤ │

│ │ ├─────────────────┤ ├──────────────┤│ │ │ │

│ │ │ + execute() │ │ + execute() ││ │ ┌─────────────────────────┐ │ │

│ │ │ + undo() │ │ + undo() ││ │ │ TextEditor │ │ │

│ │ └─────────────────┘ └──────────────┘│ │ ├─────────────────────────┤ │ │

│ │ │ │ │ - content: StringBuilder│ │ │

│ │ ┌─────────────────────────────────┐ │ │ │ - clipboard: String │ │ │

│ │ │ MacroCommand │ │ │ │ - history: Stack │ │ │

│ │ ├─────────────────────────────────┤ │ │ ├─────────────────────────┤ │ │

│ │ │ - commands: List<Command> │ │ │ │ + copy(start, end) │ │ │

│ │ ├─────────────────────────────────┤ │ │ │ + paste(position) │ │ │

│ │ │ + addCommand() │ │ │ │ + cut(start, end) │ │ │

│ │ │ + execute() │ │ │ │ + undo() │ │ │

│ │ │ + undo() │ │ │ │ + getContent() │ │ │

│ │ └─────────────────────────────────┘ │ │ └─────────────────────────┘ │ │

│ └─────────────────────────────────────┘ │ │ │

│ │ ┌─────────────────────────┐ │ │

│ ┌─────────────────────────────────────┐ │ │ FileManager │ │ │

│ │ ui │ │ ├─────────────────────────┤ │ │

│ ├─────────────────────────────────────┤ │ │ - currentFile: File │ │ │

│ │ │ │ ├─────────────────────────┤ │ │

│ │ ┌─────────────────────────────┐ │ │ │ + save(content) │ │ │

│ │ │ EditorWindow │ │ │ │ + load(): String │ │ │

│ │ ├─────────────────────────────┤ │ │ │ + saveAs(file, content) │ │ │

│ │ │ - menuBar: MenuInvoker │ │ │ └─────────────────────────┘ │ │

│ │ │ - toolbar: ToolbarInvoker │ │ └─────────────────────────────────┘ │

│ │ │ - textArea: JTextArea │ │ │

│ │ ├─────────────────────────────┤ │ ┌─────────────────────────────────┐ │

│ │ │ + initializeCommands() │ │ │ client │ │

│ │ │ + setupUI() │ │ ├─────────────────────────────────┤ │

│ │ │ + main() │ │ │ │ │

│ │ └─────────────────────────────┘ │ │ ┌─────────────────────────┐ │ │

│ └─────────────────────────────────────┘ │ │ Application │ │ │

│ │ ├─────────────────────────┤ │ │

└───────────────────────────────────────────│ │ + main(args) │ │ │

│ │ + initializeEditor() │ │ │

│ │ + configureCommands() │ │ │

│ └─────────────────────────┘ │ │

└─────────────────────────────────┘ │

│

└──────────────────────────────────────────────────────────────────────────────┘

**Package Dependencies Diagram**

┌─────────────────────────────────────────────────────────────────────┐

│ Package Dependencies │

└─────────────────────────────────────────────────────────────────────┘

┌─────────────────┐

│ client │

└─────────┬───────┘

│

▼ depends on

┌─────────────────┐

│ ui │

└─────────┬───────┘

│

┌──────────────┼──────────────┐

▼ ▼ ▼ depends on

┌─────────────────┐ ┌─────────────┐ ┌─────────────────┐

│ commands │ │ invokers │ │ receivers │

└─────────────────┘ └─────────────┘ └─────────────────┘

▲ │ ▲

│ │ │

└──────────────┼──────────────┘

▼ depends on

(uses)

**Detailed Package Structure with Import Dependencies**

Package: com.example.texteditor.commands

├── Command.java (interface)

├── CopyCommand.java

├── PasteCommand.java

├── CutCommand.java

├── UndoCommand.java

└── MacroCommand.java

Imports:

→ com.example.texteditor.receivers.\*

─────────────────────────────────────

Package: com.example.texteditor.receivers

├── TextEditor.java

└── FileManager.java

Imports: (none - core business logic)

─────────────────────────────────────

Package: com.example.texteditor.invokers

├── MenuInvoker.java

└── ToolbarInvoker.java

Imports:

→ com.example.texteditor.commands.Command

─────────────────────────────────────

Package: com.example.texteditor.ui

└── EditorWindow.java

Imports:

→ com.example.texteditor.commands.\*

→ com.example.texteditor.invokers.\*

→ com.example.texteditor.receivers.\*

─────────────────────────────────────

Package: com.example.texteditor.client

└── Application.java

Imports:

→ com.example.texteditor.ui.\*

→ com.example.texteditor.commands.\*

→ com.example.texteditor.receivers.\*

→ com.example.texteditor.invokers.\*

**Package Interaction Flow**

┌─────────────────────────────────────────────────────────────────────┐

│ Command Pattern Flow │

└─────────────────────────────────────────────────────────────────────┘

Client Package

│

│ 1. Creates

▼

┌─────────────┐ 2. Creates ┌─────────────┐

│ Application │ ────────────────► │ EditorWindow│

└─────────────┘ └─────────────┘

│ │

│ 3. Instantiates │ 4. Sets up

▼ ▼

┌─────────────┐ ┌─────────────┐

│ Receivers │ │ Invokers │

│ Package │ │ Package │

└─────────────┘ └─────────────┘

▲ │

│ │ 5. Uses

│ 6. Calls methods ▼

│ ┌─────────────┐

└──────────────────────────│ Commands │

│ Package │

└─────────────┘

Create uml diagrams to represent different kinds of relation ships

**UML Diagrams: Different Types of Relationships**

**1. Association Relationship**

**"Uses" or "Has a" relationship**

┌─────────────────┐ ┌─────────────────┐

│ Student │◆─────────◇│ Course │

├─────────────────┤ enrolls ├─────────────────┤

│ - studentId │ 1..\* │ - courseId │

│ - name │ │ - courseName │

│ - email │ │ - credits │

├─────────────────┤ ├─────────────────┤

│ + enroll() │ │ + addStudent() │

│ + study() │ │ + getDetails() │

└─────────────────┘ └─────────────────┘

**2. Aggregation Relationship**

**"Has-a" relationship (Weak ownership)**

┌─────────────────┐ ┌─────────────────┐

│ Department │◇─────────◆│ Employee │

├─────────────────┤ contains ├─────────────────┤

│ - deptId │ 1..\* │ - empId │

│ - deptName │ │ - name │

│ - location │ │ - position │

├─────────────────┤ ├─────────────────┤

│ + addEmployee() │ │ + work() │

│ + getEmployees()│ │ + getDetails() │

└─────────────────┘ └─────────────────┘

**3. Composition Relationship**

**"Part-of" relationship (Strong ownership)**

┌─────────────────┐ ┌─────────────────┐

│ House │◆─────────◇│ Room │

├─────────────────┤ contains ├─────────────────┤

│ - address │ 1..\* │ - roomNumber │

│ - owner │ │ - size │

│ - yearBuilt │ │ - type │

├─────────────────┤ ├─────────────────┤

│ + addRoom() │ │ + furnish() │

│ + sellHouse() │ │ + clean() │

└─────────────────┘ └─────────────────┘

**4. Inheritance (Generalization)**

**"Is-a" relationship**

┌─────────────────┐

│ Vehicle │

├─────────────────┤

│ - make │

│ - model │

│ - year │

├─────────────────┤

│ + start() │

│ + stop() │

└─────────┬───────┘

│

┌─────────────┴─────────────┐

▲ ▲

┌─────────────────┐ ┌─────────────────┐

│ Car │ │ Motorcycle │

├─────────────────┤ ├─────────────────┤

│ - doors │ │ - engineSize │

│ - fuelType │ │ - hasWindshield │

├─────────────────┤ ├─────────────────┤

│ + openTrunk() │ │ + wheelie() │

│ + lockDoors() │ │ + leanInTurn() │

└─────────────────┘ └─────────────────┘

**5. Realization/Implementation**

**Interface implementation**

┌─────────────────┐

│ <<interface>> │

│ Flyable │

├─────────────────┤

│ + fly() │

│ + land() │

└─────────┬───────┘

│

┌─────────┴───────┐

⋯ ⋯ (implements)

┌─────────────────┐ ┌─────────────────┐

│ Airplane │ │ Bird │

├─────────────────┤ ├─────────────────┤

│ - wingspan │ │ - species │

│ - maxAltitude │ │ - migrationPath │

├─────────────────┤ ├─────────────────┤

│ + fly() │ │ + fly() │

│ + land() │ │ + land() │

│ + refuel() │ │ + buildNest() │

└─────────────────┘ └─────────────────┘

**6. Dependency Relationship**

**"Uses" relationship (temporary)**

┌─────────────────┐ ┌─────────────────┐

│ Printer │ │ Document │

├─────────────────┤ ├─────────────────┤

│ - model │- - - - - -│ - content │

│ - status │ uses │ - format │

├─────────────────┤ ├─────────────────┤

│ + print(doc) │ │ + getContent() │

│ + getStatus() │ │ + format() │

└─────────────────┘ └─────────────────┘

**7. Complete Example: Library Management System**

┌─────────────────┐

│ Person │

├─────────────────┤

│ - name │

│ - address │

│ - phone │

├─────────────────┤

│ + getDetails() │

└─────────┬───────┘

│

┌─────────┴───────┐

▲ ▲

┌─────────────────┐ ┌─────────────────┐

│ Librarian │ │ Member │

├─────────────────┤ ├─────────────────┤

│ - employeeId │ │ - memberId │

│ - department │ │ - membershipDate│

├─────────────────┤ ├─────────────────┤

│ + issueBook() │ │ + borrowBook() │

│ + returnBook() │ │ + returnBook() │

└─────────────────┘ └─────────┬───────┘

│

┌─────────┴───────┐

│ borrows │

│ 0..\* │

▼

┌─────────────────┐

│ Book │

├─────────────────┤

│ - isbn │

│ - title │

│ - author │

│ - publishYear │

├─────────────────┤

│ + getInfo() │

│ + isAvailable() │

└─────────┬───────┘

│

│ belongs to

│ 1..\*

▼

┌─────────────────┐

│ Library │

├─────────────────┤

│ - name │

│ - address │

│ - capacity │

├─────────────────┤

│ + addBook() │

│ + removeBook() │

└─────────────────┘

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Info Box

Excalidraw updated at 9.41am

<https://excalidraw.com/#json=BzWUiyDQJSrM3H4fnMZmA,8ku7hxm4MdwGtTl8l9xBAg>

Excalidraw updated at 12.57

<https://excalidraw.com/#json=wKUoOyPWlHWl88tFfcZLN,2peOceri73zO1fANtEIRXg>

Exaclidraw updated at 13.20

<https://excalidraw.com/#json=FvOSdghWg2edjx8IdkC_6,Jmc2IB9qmnTVmDtKPUgvZA>

Excalidraw updated at 15.38

<https://excalidraw.com/#json=ducQ4gsj0ERH6C5R1FeAX,3jVSsU8c1C5rxj9f9KMtBw>

Excalidraw updated at 16.01

<https://excalidraw.com/#json=203GR_BNJIcTIEDfwxLgl,H0yHSbcWxSqo1IJs35Cmfw>

For MCQ practice

<https://mcqmate.com/topic/object-oriented-software-engineering>

Mcqmate:

Git hub.. Link update sheet

<https://docs.google.com/spreadsheets/d/1y1753RPMqal4WxHqvdBuV4jqh5bMFWr_N4tcavWOmH8/edit?usp=sharing>

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%