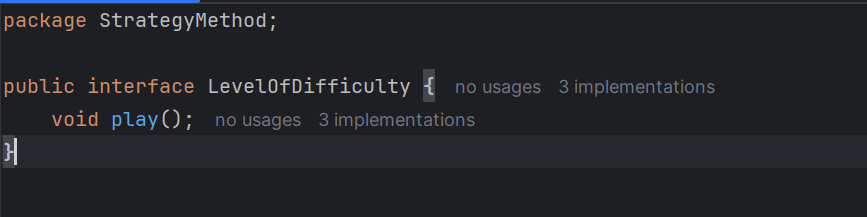
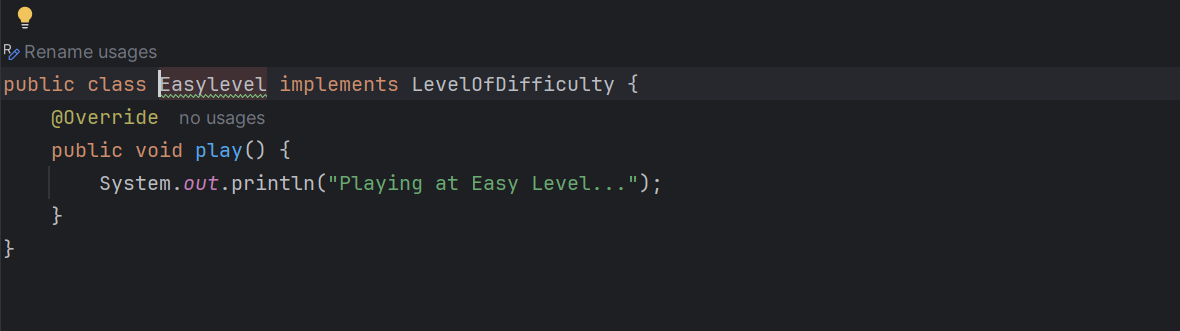
**Day 26 – August 22nd**

**TASK 1 - STRATEGY METHOD DESIGN PATTERN**

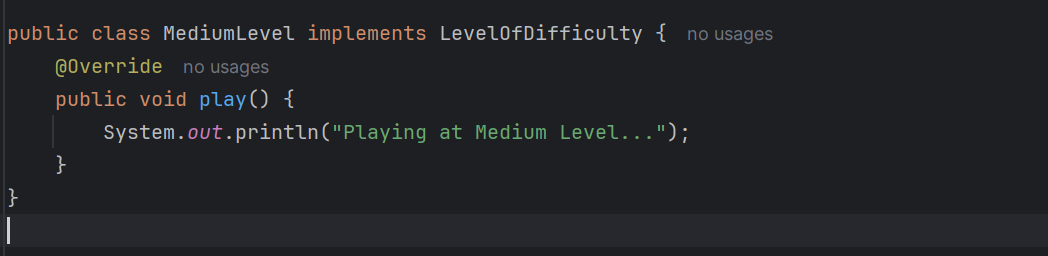
**LevelOfDifficulty.java**

****

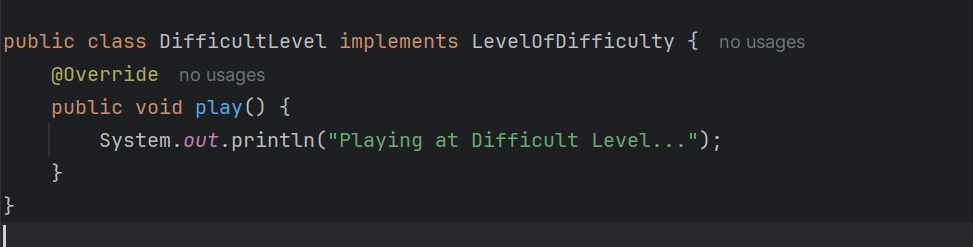
**EasyLevel.java**

****

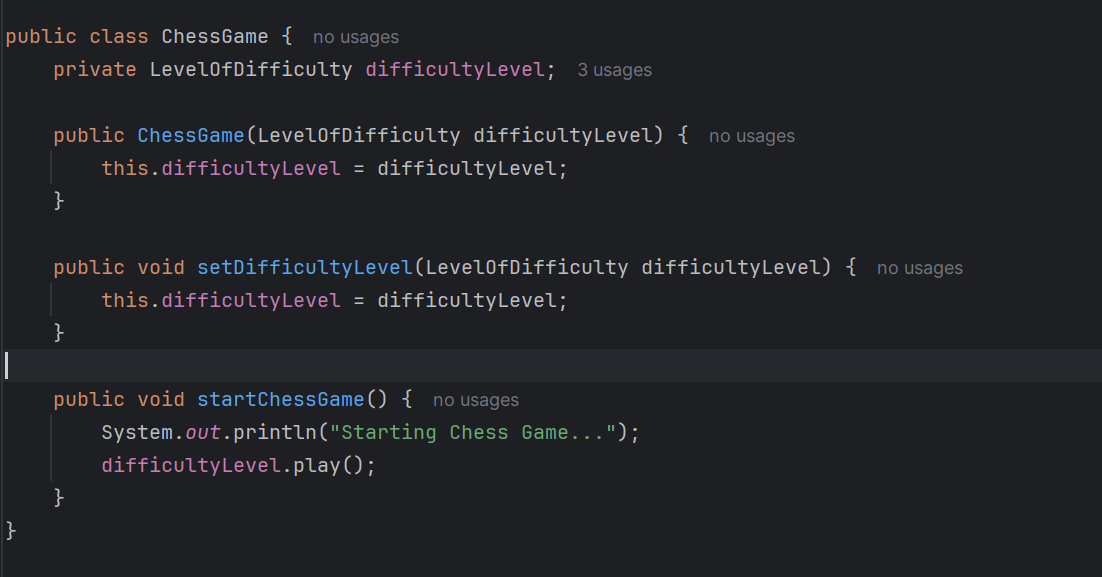
**MediumLevel.java**

****

**DifficultLevel.java**

****

**ChessGame.java**

****

**Main.java**

****

**OUTPUT**

Strategy Design pattern - Behavioral Design Pattern

Starting Chess Game...

Playing at Easy Level...

Starting Chess Game...

Playing at Medium Level...

Starting Chess Game...

Playing at Difficult Level...

**TASK 2**

**DESIGN WORKFLOW**

Design Workflow means the step-by-step process followed to design and implement software. It ensures that the software is planned, structured, and developed systematically before writing actual code.

1. **Requirement Analysis**
   * Understand what the software should do.
   * Example: Gather user requirements, business goals.
2. **System Design (High-Level Design)**
   * Decide architecture
   * Define modules and their interactions.
   * Create UML diagrams like:
     + Class Diagram (for OOP structure)

**Detailed Design (Low-Level Design)**

* + Define data structures, algorithms, and class responsibilities.
  + Plan APIs, function signatures, database schema.

1. **Implementation (Coding)**
   * Convert design into code
   * Example: Implement classes, interfaces.
2. **Testing**
   * Unit Testing, Integration Testing, System Testing.
3. **Deployment**
   * Move the application to production.
4. **Maintenance & Iteration**
   * Fix bugs, improve performance, add features.

**TASK 3**

**PERSISTENT OBJECTS**

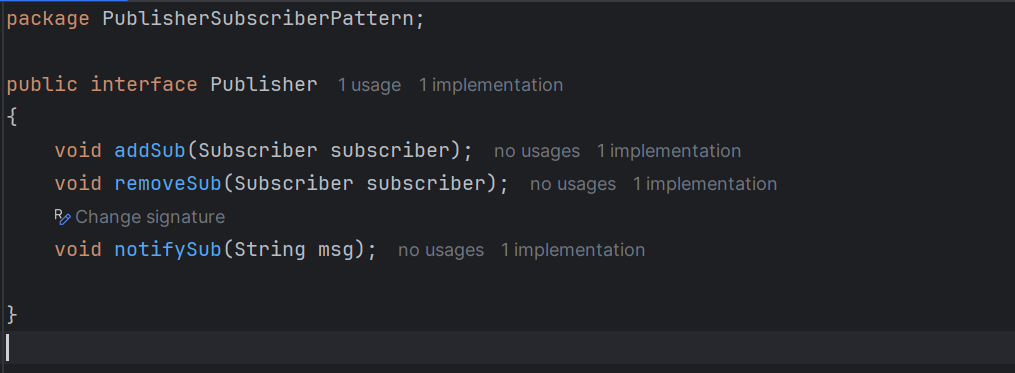
It is an object whose state persists beyond the execution of the program. Its data is stored in non-volatile storage (database, file system) so it can be retrieved and reconstructed later.

Associated with Data base session, can be auto-saved.

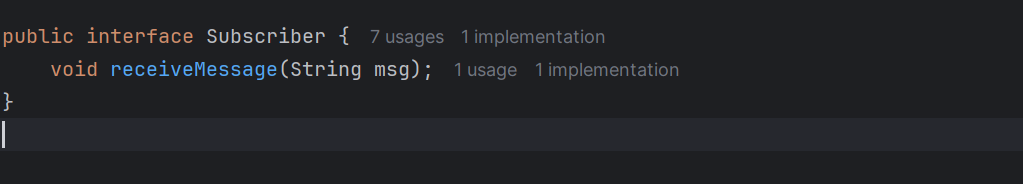
**HOME TASK -1**

**PUB/SUB PATTERN**

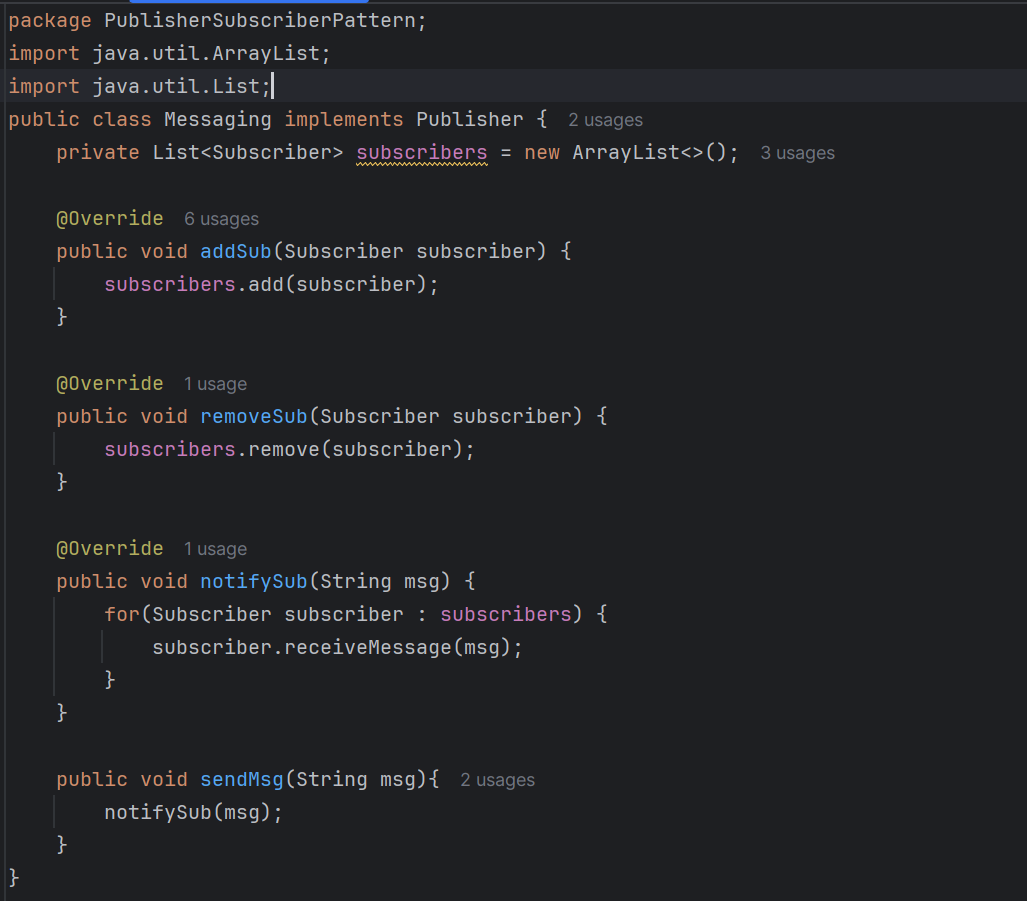
**Publisher.java**

****

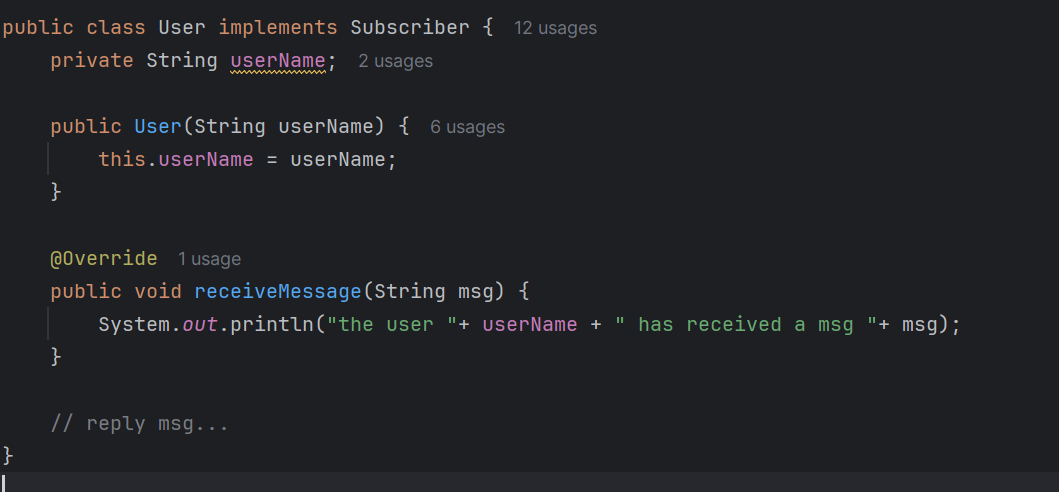
**Subscriber**

****

**Messaging.java**

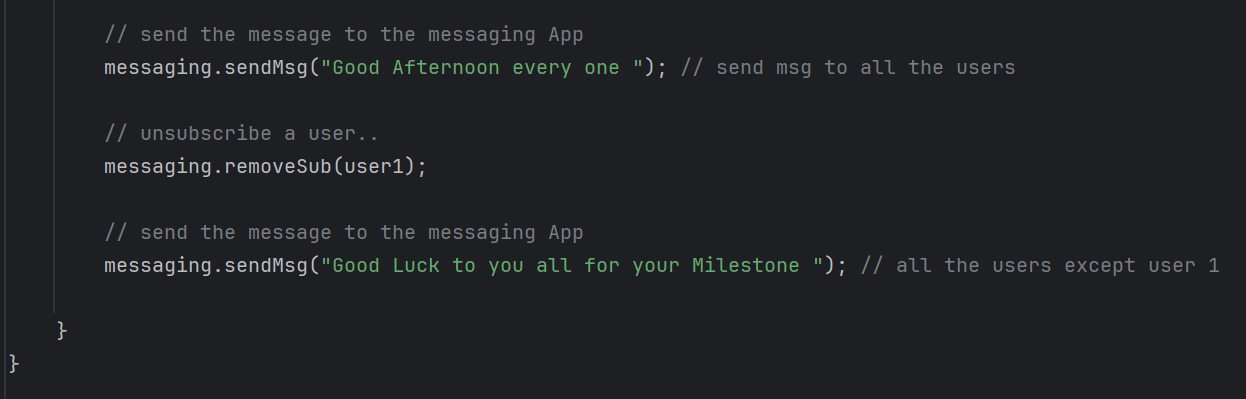
****

**User.java**

****

**Main.java**

****

****

**OUTPUT**

Pub sub Pattern

the user Sharun has received a msg Good Afternoon every one

the user Kumar has received a msg Good Afternoon every one

the user Harsha has received a msg Good Afternoon every one

the user Teddy has received a msg Good Afternoon every one

the user Arun Kumar has received a msg Good Afternoon every one

the user Arun has received a msg Good Afternoon every one

the user Kumar has received a msg Good Luck to you all for your Milestone

the user Harsha has received a msg Good Luck to you all for your Milestone

the user Teddy has received a msg Good Luck to you all for your Milestone

the user Arun Kumar has received a msg Good Luck to you all for your Milestone

the user Arun has received a msg Good Luck to you all for your Milestone

**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)

**Ans:** c) Abstract Factory

**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.

**Ans:** c) It asks the command to carry out the request.

**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

**Ans:** b) Undo/Redo functionality

**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.

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

**TASK 8**

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

**Ans:** c) Static Binding/ early 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

**Ans:** b) Facade 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

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

**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

**Ans:** c) Polymorphism

**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

**Ans:** b) Singleton Pattern

**TASK 13**

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

a) Creational

b) Structural

c) Behavioral

d) Concurrency

**Ans:** b) Structural

**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

**Ans:** c) Observer Pattern

**TASK 15**

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

a) Creational

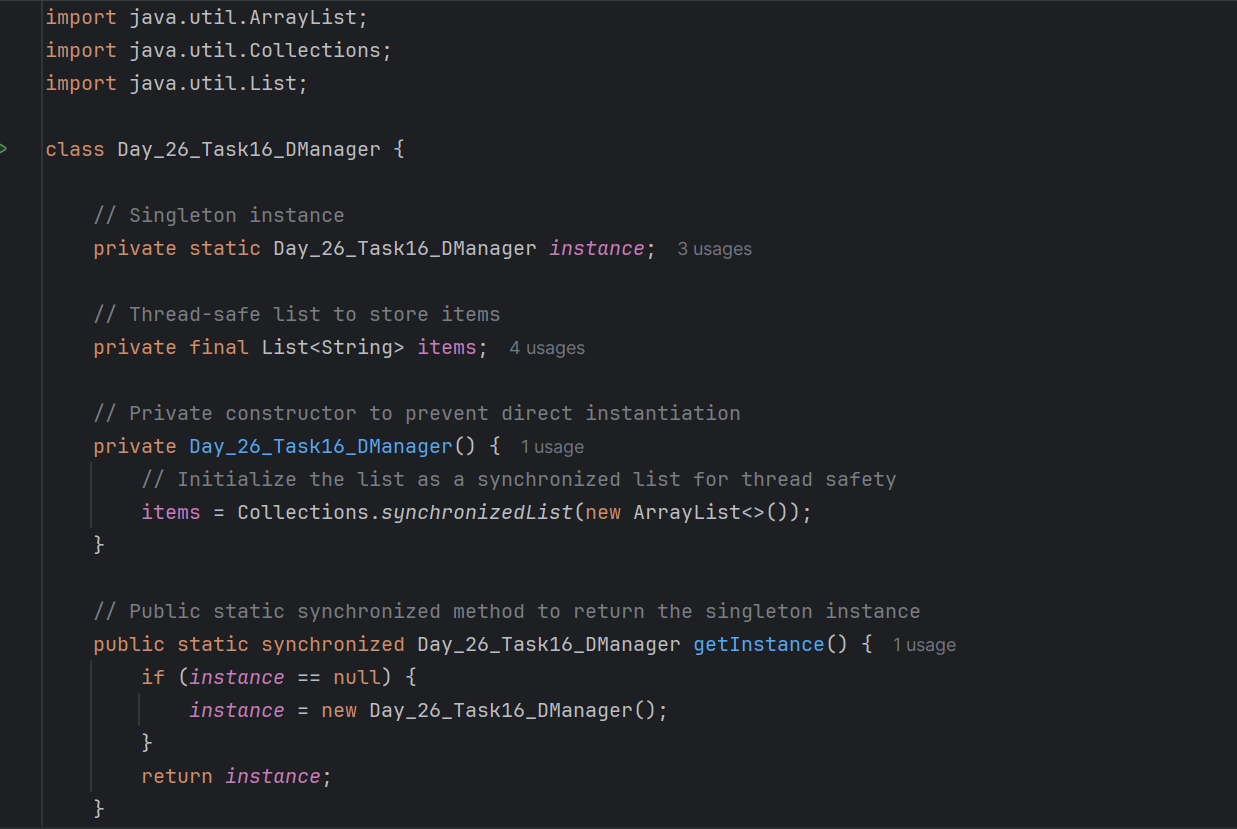
b) Structural

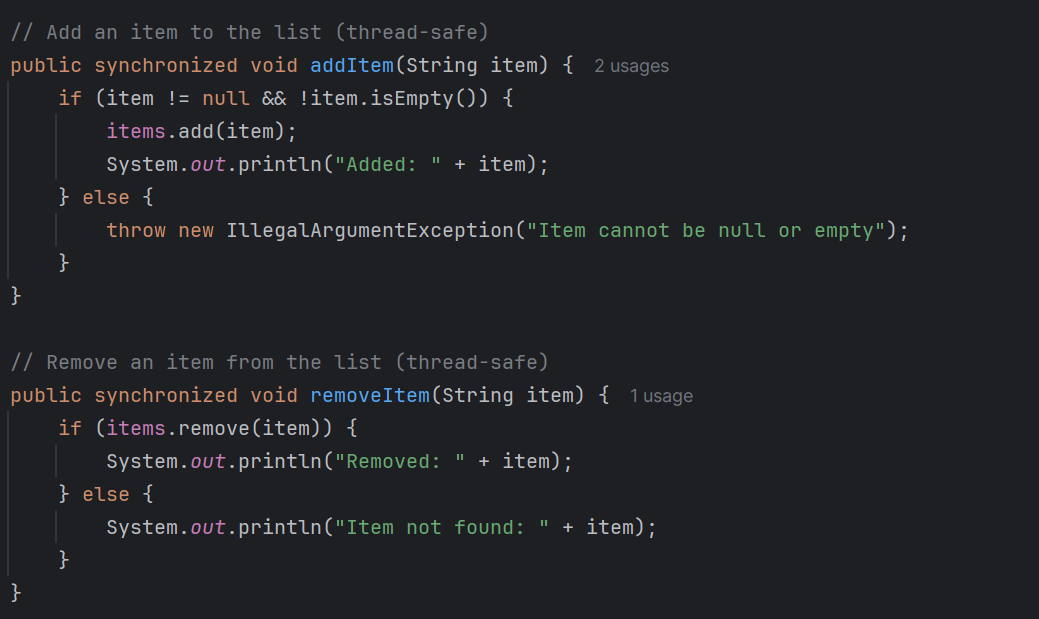
c) Behavioral

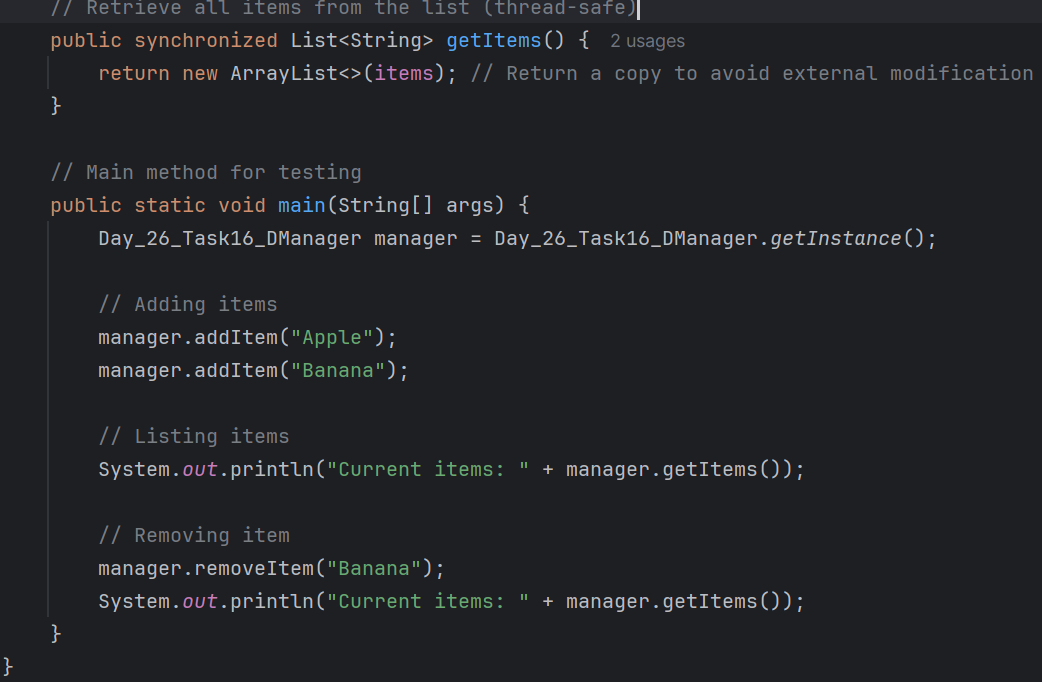
d) Architectural

**Ans:** d) Architectural

**TASK 16**

****

****

****

**Output:**

Added: Apple

Added: Banana

Current items: [Apple, Banana]

Removed: Banana

Current items: [Apple]