Design Work Flow…

A Design Workflow is simply the structured sequence of steps or activities you follow to go from an initial idea/problem to a finished design (whether that’s software, product, UI/UX, or even an industrial design).

General Stages of a Design Workflow (in software/product development):

1. Requirement Gathering / Problem Understanding
2. Research & Analysis
3. Conceptual Design / Planning
4. Detailed Design
5. Implementation / Development
6. Testing & Evaluation
7. Deployment / Delivery
8. Feedback & Iteration

What do you know Persistent objects:

A **persistent object** is just an object in a program whose information **doesn’t get lost when you close the program**.

* Normally, when a program stops, everything in memory disappears.
* But if you want some objects to **“remember” their data** (like a saved game, a customer record, or a draft document), you make them **persistent** by saving them to a database or file.

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.

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

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

public class DManager {

// ✅ Step 1: Singleton instance

private static DManager instance;

// ✅ Step 2: Thread-safe List

private final List<String> items;

// ✅ Step 3: Private constructor

private DManager() {

items = Collections.synchronizedList(new ArrayList<>());

}

// ✅ Step 4: Thread-safe Singleton accessor

public static synchronized DManager getInstance() {

if (instance == null) {

instance = new DManager();

}

return instance;

}

// ✅ Step 5: List management methods

public synchronized void addItem(String item) {

items.add(item);

}

public synchronized void removeItem(String item) {

items.remove(item);

}

public synchronized List<String> getItems() {

return new ArrayList<>(items); // return a copy for safety

}

}

public class Main {

public static void main(String[] args) {

DManager manager = DManager.getInstance();

manager.addItem("Item 1");

manager.addItem("Item 2");

manager.addItem("Item 3");

System.out.println("List after adding: " + manager.getItems());

manager.removeItem("Item 2");

System.out.println("List after removing Item 2: " + manager.getItems());

}

}

Output:

List after adding: [Item 1, Item 2, Item 3]

List after removing Item 2: [Item 1, Item 3]