# **Day 26 - 22 August 2025**

**Document Name:**Day 26 - hmuvvala@ - Hari Gopal Muvvala

### Task 2: What do you know about Design Workflow?

* Design Workflow = the structured process followed when creating software or a system.
* It ensures requirements → design → implementation → testing → deployment flow smoothly.
* **Typical steps:**
  1. **Requirement Gathering** – understand what needs to be built.
  2. **Analysis & Planning** – feasibility, scope, resources.
  3. **High-Level Design (HLD)** – overall architecture, components, interactions.
  4. **Low-Level Design (LLD)** – detailed class diagrams, database schema, algorithms.
  5. **Implementation** – code according to the design.
  6. **Testing & Validation** – verify against requirements.
  7. **Deployment & Maintenance** – deliver and keep improving.
* Importance: avoids chaos, ensures clarity, reduces rework, makes collaboration easier.
* Debate point: *Agile* favors iterative workflows, while *Waterfall* follows rigid step-by-step. Both are valid depending on project context.

### Task 3: What do you know about Persistent Objects?

* Persistent objects = objects whose state is **stored permanently** (e.g., in a database, file, or storage system) so that data is not lost when the application stops.
* Example: In Java, when you store a Customer object in a database using Hibernate/JPA, it becomes a persistent object.
* Contrast with **Transient objects** – exist only in memory during runtime, lost once program ends.
* Advantages:
  + Data durability (not lost after program ends).
  + Reusability (can load the same object again later).
* Typical mechanisms: ORM frameworks (Hibernate, JPA), serialization, direct DB mapping.
* Debate point:
  + **Pros:** Data consistency, reusability, scalability.
  + **Cons:** Added complexity, performance overhead, requires good mapping design.

### 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)

**Answer: c) Abstract Factory**

Because Invoker, Receiver, and Command are the main elements of the Command pattern, while Abstract Factory belongs to a different design pattern.

### 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.

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

The Invoker does not execute the operation itself; it only triggers the Command object, which then tells the Receiver what to do.

### 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

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

Command objects store requests, making it possible to queue, log, and even reverse actions, hence Undo/Redo support is one of the biggest advantages.

### 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.

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

The Context doesn’t implement the algorithms itself. It just keeps a reference to the chosen Strategy and delegates work to it, making algorithms interchangeable.

### Home Task 4: Package Diagram (Command Pattern)

* **Packages used:**
  + client → contains ClientApp
  + command → contains Command (interface), OpenCommand, SaveCommand
  + receiver → contains DocumentService
  + invoker → contains MenuInvoker
* **Relationships:**
  + OpenCommand, SaveCommand → implement → Command
  + OpenCommand, SaveCommand → use → DocumentService
  + MenuInvoker → has a reference to → Command
  + ClientApp → creates and wires together → Invoker, Commands, and Receiver
* **Purpose:**The Command pattern decouples **sender (Invoker/Client)** from **receiver (DocumentService)** by encapsulating requests as objects. This enables features like **undo/redo, queuing, logging**, and flexible command management.

### Home Task 3: Understanding Command Pattern Components

* **Main components in Command Pattern:**
  1. **Command (Interface/Abstract class):** Declares an interface for executing an operation.
  2. **Concrete Command(s):** Implement the command interface and define the link between a Receiver and an action.
  3. **Receiver:** The object that knows how to perform the operation.
  4. **Invoker:** Asks the command to carry out the request.
  5. **Client:** Creates command objects and associates them with receivers.
* **Key Point:**The pattern encapsulates a request as an object, decoupling the sender (Invoker/Client) from the receiver.

### Task 8

In which mechanism, 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

**Answer:** **c) Static Binding / Early Binding**

* Resolved at compile-time.
* Faster execution.
* Example: method overloading.

### 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

**Answer:** **b) Facade Pattern**

* Hides complexity, shows simple interface.
* Example: DriverManager for DB connections.

### Task 10

Which of the following statements about Persistence is correct?

a) Enforcement of class of an object → objects of different types not interchanged.

b) Property of an object → existence transcends time/space.

c) Property that distinguishes active object from non-active.

d) All of the mentioned

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

* Persistent = data stored permanently (DB, files).
* Exists beyond program execution.
* Opposite: transient (temporary).

### Task 11

Concept in type theory where a single name may denote objects of many different classes (common superclass)?

a) Monomorphism

b) Type Checking

c) Polymorphism

d) Generalization

**Answer:** **c) Polymorphism**

* “Many forms.”
* Same method works differently for different objects.
* Example: draw() → Circle, Rectangle.

### Task 12

Which pattern is used to create a single instance of a class and provide global access?

a) Factory Pattern

b) Singleton Pattern

c) Builder Pattern

d) Prototype Pattern

**Answer:** **b) Singleton Pattern**

* Only one instance allowed.
* Global point of access.
* Example: Runtime.getRuntime().

### Task 13

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

a) Creational

b) Structural

c) Behavioral

d) Concurrency

**Answer:** **b) Structural**

* Converts one interface into another.
* Makes incompatible classes work together.
* Example: InputStreamReader.

### Task 14

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

a) Strategy Pattern

b) Command Pattern

c) Observer Pattern

d) Mediator Pattern

**Answer:** **c) Observer Pattern**

* Subject → Observers.
* Used in event-driven systems.
* Example: GUI Listeners, stock updates.

### Task 15

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

a) Creational

b) Structural

c) Behavioral

d) Architectural

**Answer:** **d) Architectural**

* **Model:** data + business logic.
* **View:** UI.
* **Controller:** handles input, updates model/view.
* Separation of concerns, maintainability.