SNS COLLEGE OF ENGINEERING

Kurumbapalayam (po), Coimbatore – 641 107

Accredited by NAAC-UGC with 'A' Grade

Approved by AICTE & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

Unit 3

Design Patterns

1. Model-View-Controller (MVC)

Concept:

- **MVC** is a design pattern that separates an application into three distinct parts to improve organization and flexibility.
  - **Model**: Manages data and business logic.
  - **View**: Displays information to users.
  - **Controller**: Takes user input and updates the Model and View accordingly.

Example:

Think of an **online shopping website**:

- **Model**: The database that stores product details, prices, and user orders.
- **View**: The website page that displays the products and cart.
- **Controller**: Handles actions like adding an item to the cart or completing a purchase.

2. Publish-Subscribe (Pub-Sub)

Concept:

- In the **Pub-Sub pattern**, publishers send messages without knowing who receives them, and subscribers receive messages without knowing who sent them.
- A **central mediator (message broker)** helps manage this communication.

Example:

Think of **news subscriptions**:

- A person subscribes to a newspaper.
- Whenever a new edition is released (published), the subscriber gets it automatically.
- The publisher doesn't need to know who all the readers are—it just releases the news.

Software Example:

- YouTube notifications: When a content creator uploads a video, all subscribers receive an alert.

3. Adapter

Concept:

- The **Adapter pattern** acts as a bridge between two incompatible interfaces, allowing them to work together without modifying their internal structures.

Example:

- **Phone Charger Adapter**:
  - Your smartphone has a USB-C charging port, but the charger plug is USB-A.
  - A **USB-C to USB-A adapter** helps connect the charger and the phone.

Software Example:

- Connecting an **old printer** to a modern laptop via a **USB-to-wireless adapter**.

4. Command

Concept:

- The **Command pattern** encapsulates a request as an object.
- It allows you to queue, execute, or undo operations easily.

Example:

- **TV Remote Control**:
  - The buttons on a remote (power, volume, channel change) send **commands** to the TV.
  - The remote doesn't need to know how the TV operates—it just triggers a command.

Software Example:

- **Undo/Redo** feature in Microsoft Word: Every action (typing, deleting, formatting) is stored as a command that can be reversed.

5. Strategy

Concept:

- The **Strategy pattern** allows switching between multiple algorithms or behaviors dynamically.
- Each strategy follows the same interface but has different implementations.

Example:

- **Navigation Apps (Google Maps, Waze, etc.)**:
  - A user can select **Fastest Route, Shortest Distance, or Avoid Highways** as different navigation strategies.
  - The app applies the selected strategy to provide a suitable route.

Software Example:

- **Payment options in an online store** (Credit Card, PayPal, Cash on Delivery).

6. Observer

Concept:

- The **Observer pattern** establishes a relationship where one object (Subject) notifies multiple dependent objects (Observers) when there is a change.

Example:

- **Stock Market Alerts**:
  - Investors subscribe to stock prices.
  - When stock prices change, all subscribed investors receive an alert.

Software Example:

- **Facebook or Instagram notifications**:
  - When someone you follow posts a new update, you get a notification.

7. Proxy

Concept:

- A **Proxy** acts as an intermediary between a user and a resource to control access, enhance security, or improve performance.

Example:

- **Internet Proxy Server**:
  - In an office, employees access the internet through a proxy server that **filters** and **monitors** traffic.

Software Example:

- **VPN (Virtual Private Network)**:
  - A VPN acts as a proxy by encrypting internet traffic and routing it through a secure server.

8. Facade

Concept:

- The **Facade pattern** provides a simple interface to a complex system by hiding its internal workings.
- It makes the system easier to use by offering a **single entry point** for multiple functionalities.

Example:

- **Car Dashboard**:
  - Instead of directly controlling each car component (engine, brakes, lights), the driver interacts with a **simple dashboard** that does everything behind the scenes.

Software Example:

- **Online Banking App**:
  - Instead of handling database queries, security checks, and payment processing separately, the app provides a **single interface** to handle all banking operations.