# Design Patterns

Design pattern is a general reusable solution to a commonly occurring problem in software design.

Design patterns help making new software quickly and efficiently.

Less effort in making a software if we use already made design pattern.

Based on principles of object-oriented programming,

Are independent of application domain.

### **Observer Design Pattern (Behavioural Pattern)**

The *observer* pattern defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.

The object which is being watched is called the *subject*. The objects which are watching the state changes are called *observers* or *listeners*.

Example: WhatsApp group notifications.

### **Factory Design Pattern (Creational Pattern)**

Factory pattern is one of the most used design patterns in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface.

Example: A factory creates bottles; the client decides whether to make it Coke or Fanta or etc.

### **Singleton Design Pattern (Creational Pattern)**

Ensure a class has only one instance and provide a global point of access to it.

### **Adapter Design Pattern**

The adapter pattern is a design pattern that translates one interface for a class into a compatible interface.

### **façade Design Pattern (Structural Design Pattern)**

A façade is an object that provides a simplified interface to a larger body of code, such as class library. 🡪 Credit Card, Login.

### **Iterator Design Pattern**

Iterator pattern is very commonly used design pattern in Java and .Net programming environment. This pattern is used to get a way to access the elements of a collection object in sequential manner without any need to know its underlying representation.

*Single Responsibility Principle*. You can clean up the client code and the collections by extracting bulky traversal algorithms into separate classes.

You can iterate over the same collection in parallel because each iterator object contains its own iteration state.

For the same reason, you can delay an iteration and continue it when needed.

### **Template Design Pattern**

Template method design pattern is to define an algorithm as a skeleton of operations and leave the details to be implemented by the child classes. The overall structure and sequence of the algorithm is preserved by the parent class.  
Template means Preset format like HTML templates which has a fixed preset format. Similarly in the template method pattern, we have a preset structure method called template method which consists of steps. This steps can be an abstract method which will be implemented by its subclasses.

This behavioral design pattern is one of the easiest to understand and implement. This design pattern is used popularly in framework development. This helps to avoid code duplication also.

**Template Method** is a behavioral design pattern that defines the skeleton of an algorithm in the superclass but lets subclasses override specific steps of the algorithm without changing its structure.