INTRODUDCTION TO PATTERN DESIGN

In 1994, four authors Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides published a book titled **Design Patterns - Elements of Reusable Object-Oriented Software** which initiated the concept of Design Pattern in Software development.

These authors are collectively known as **Gang of Four (GOF)**. According to these authors design patterns are primarily based on the following principles of object orientated design.

- Program to an interface not an implementation
- Favor object composition over inheritance

Design patterns provide a standard terminology and are specific to particular scenario. For example, a singleton design pattern signifies use of single object so all developers familiar with single design pattern will make use of single object and they can tell each other that program is following a singleton pattern.

Types of Design Patterns

As per the design pattern reference book **Design Patterns - Elements of Reusable Object-Oriented Software** , there are 23 design patterns which can be classified in three categories: Creational, Structural and Behavioral patterns. We'll also discuss another category of design pattern: J2EE design patterns.

S.N.	Pattern & Description
1	**Creational Patterns** These design patterns provide a way to create objects while hiding the creation logic, rather than instantiating objects directly using new operator. This gives program more flexibility in deciding which objects need to be created for a given use case.

2	**Structural Patterns** These design patterns concern class and object composition. Concept of inheritance is used to compose interfaces and define ways to compose objects to obtain new functionalities.
3	**Behavioral Patterns** These design patterns are specifically concerned with communication between objects.
4	**J2EE Patterns** These design patterns are specifically concerned with the presentation tier. These patterns are identified by Sun Java Center.

Creational Design Patterns

Creational patterns often used in place of direct instantiation with constructors. They make the creation process more adaptable and dynamic. In particular, they can provide a great deal of flexibility about which objects are created, how those objects are created, and how they are initialized.

Singleton

When an application wants to have one and only one instance of any class per JVM, in all possible scenarios without any exceptional condition.

Factory

This is most suitable where there is some complex object creation steps are involved. To ensure that these steps are centralized and not exposed to composing classes, factory pattern should be used.

Abstract factory

Whenever you need another level of abstraction over a group of factories, you should consider using abstract factory pattern.

Structural Design Patterns

These design patterns show you how to glue different pieces of a system together in a flexible and extensible fashion. Structural patterns help you guarantee that when one of the parts changes, the entire structure does not need to change.

Adapter

Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.

Decorator

This is used to add additional features or behaviors to a particular instance of a class, while not modifying the other instances of same class.

Behavioral Design Patterns

A behavioral pattern abstracts an action you want to take from the object or class that takes the action. By changing the object or class, you can change the algorithm used, the objects affected, or the behavior, while still retaining the same basic interface for client classes.

Command

Command pattern is a behavioral design pattern which is useful to abstract business logic into discrete actions which we call commands. This command object helps in loose coupling between two classes where one class (invoker) shall call a method on other class (receiver) to perform a business operation.

Visitor

When you want a hierarchy of objects to modify their behavior but without modifying their source code.

Memento

Memento design pattern provides ability to capture(save) an object's state and then restore back this captured state when required by the system.

State

State Design Pattern allows the behavior of an object to vary based on its state. I.e. whenever the object’s state changes, its behavior changes as per its new state. To the observer it appears as if the object has changed its class.