**Design Patterns**

**Date: 05/08/2018**

Design patterns are some of the software solutions which are used to solve the problems faced by software developers. These solutions are obtained by trial and error methods by many software developers over a period of time. Design patterns provides us industry standard approach to solve recurring problems and by using them sensibly we can save our time and energy.

Design patterns started around in 1994, There was a book by four authors who are popularly called Gang of Four (GOF). They published a book titled Design Patterns - Elements of Reusable Object-Oriented Software. This book actually initiated the principles of design patterns in software development.

**Usage of Design Patterns:**

These design patterns have two main usages in software development.

Common platform for developers:

Using design patterns, a common platform will be created for all the developers by providing a standard terminology and specific to particular scenario. When developers are aware of design patterns they will know the outcome of the particular class or project. By this developers would be able to understand the code easily.

Best Practices:

Design patterns provide the best solutions to certain problems which are faced during software development and learning design patterns helps unexperienced developers to learn software design in a faster way.

**Types of Design patterns:**

There are three types of design patterns which are classified basing on objects and their behaviors. They are:

1. Creational Patterns
2. Structural Patterns
3. Behavioral Patterns

**Creational Patterns:**

Creational Patterns deals with creating objects by hiding the creational logic, rather than instantiating objects. By using these design patterns, we can have a flexibility of deciding which objects need to be created by the given use case.

We can implement creational patterns in following ways:

1. Factory Pattern
2. Singleton Pattern
3. Abstract Factory Pattern
4. Proxy Pattern
5. Builder Pattern

**Factory Pattern:**

Factory pattern is one of the most popular design pattern in java. Here we can create an object without exposing the logic to client. He can refer to a newly created object using a common interface.

Examples of Factory pattern:

1. Java.util.Calendar.ResourceBundle and NumberFormat getInstance() methods uses Factory Pattern.
2. Valueof() method in Wrapper Classes.

Advantages of Factory Pattern:

1. This method provides approach to interface rather than implementation.
2. Factory pattern makes the code more robust, less coupled and easy to extend.
3. Factory pattern provides abstraction between implementation and client classes through inheritance.

**Builder Pattern:**

This pattern quotes to “Construct a complex object using simple objects by step by step approach”.

When an object cannot be created in one single step this pattern is used.

Example:

1. A veg burger object will a complex object of Burger, wrapping and Item obejects.

Advantages:

1. It provides clear separation between the construction and representation of object.
2. Change in the internal representation of objects is easy.
3. It provides better control over construction process.

**Structural Patterns:**

This kind of design pattern deals with composition of objects. Concept of inheritance is used to compose interfaces and provide ways for new functionalities.

Structural Pattern involves:

1. Composite
2. Adapter
3. Proxy
4. Fly weight
5. Façade
6. Bridge
7. Decorator

**Facade Design Pattern:**

This pattern hides the complexities present in a system. Client can directly access the system easily as all the complexities are hidden. We add an interface to the existing system to hide its complexities.

Advantages:

1. All the complexities and method calls are collected and are represented in one façade call. Hence client would only one call.
2. Changes in the subsystem will have no effect upon the client.
3. Easier to use.
4. More structured environment.

Disadvantages:

1. If there are any methods connecting to façade layer are changed, The façade layer also faces change.

**Adapter Design Pattern:**

Adapter pattern acts as a bridge between two incompatible interfaces. Here a single class is responsible to join the functionalities.

Eg : Memory Card and Laptop.. A memory card reader is used to join both.

Advantages:

1. Achieves Reusability and responsibility.
2. Client class can be simple.

Disadvantages:

1. Ad everything needs an adapter they will be a slight overhead on the work.

**Behavioral Design Patterns:**

These design patterns primarily concern on the communication between objects.

Behavioral Design patterns are:

1. Template Method.
2. Mediator Method.
3. Chain of Responsibility.
4. Observer Method.
5. Strategy.
6. Command.
7. State.
8. Visitor.
9. Iterator.
10. Interpreter.
11. Memento.

**Command Pattern:**

In this pattern we will have an interface which has an execute method, the request is wrapped as an object and it is sent to the invoker object. The invoker object receives the request and send it to those methods which can handle that request.

Advantages:

1. Extensions to new command can be done without modifying the existing code.
2. It helps to execute the commands in a sequence following the queue system.
3. We can also roll back to the command class in case of modifications etc.

Disadvantages:

1. Many number of classes are created, this may make the developer confuse about the project.
2. Number of classes will be increased as every individual command is a class.

**Mediator Pattern:**

This pattern helps to reduce the complexity in the communication of the objects. This generally creates a mediator class to handle the communication between different classes.

Example: Chatroom.

Advantages:

1. Simplicity.
2. Loosely Coupled.
3. Object can be replaced without effecting the structure of the project.

Disadvantages:

1. Mediator class should be structured properly.