Design Pattern: is a general repeatable solution for the common problems that occur in software design. A design pattern not a particular implementation. It is an idea and description or template for how to solve a problem that can be used in many different situations. The pattern typically shows relationships and interactions between classes or objects.

Types of design pattern:

1. Creational design patterns
2. Structural design patterns
3. Behavioral design patterns

Creational design patterns:

These design patterns are all about class instantiation. This pattern can be further divided into class-creation patterns and object-creational patterns. While class-creation patterns use inheritance effectively in the instantiation process, object-creation patterns use delegation effectively to get the job done.

Types:

1. Abstract Factory

Creates an instance of several families of classes

1. Builder

Separates object construction from its representation

1. Factory Method

Creates an instance of several derived classes

1. Object Pool

Avoid expensive acquisition and release of resources by recycling objects that are no longer in use

1. Prototype

A fully initialized instance to be copied or cloned

1. Singleton

A class of which only a single instance can exist

Structural design patterns

These design patterns are all about Class and Object composition. Structural class-creation patterns use inheritance to compose interfaces. Structural object-patterns define ways to compose objects to obtain new functionality.

Types:

1. Adapter

Match interfaces of different classes

1. Bridge

Separates an object’s interface from its implementation

1. Composite

A tree structure of simple and composite objects

1. Decorator

Add responsibilities to objects dynamically

1. Facade

A single class that represents an entire subsystem

1. Flyweight

A fine-grained instance used for efficient sharing

1. Private Class Data

Restricts accessor/mutator access

1. Proxy

An object representing another object

Behavioral design patterns:

These design patterns are all about Class's objects communication. Behavioral patterns are those patterns that are most specifically concerned with communication between objects.

Types:

1. Chain of responsibility

A way of passing a request between a chain of objects

1. Command

Encapsulate a command request as an object

1. Interpreter

A way to include language elements in a program

1. Iterator

Sequentially access the elements of a collection

1. Mediator

Defines simplified communication between classes

1. Memento

Capture and restore an object's internal state

1. Null Object

Designed to act as a default value of an object

1. Observer

A way of notifying change to a number of classes

1. State

Alter an object's behavior when its state changes

1. Strategy

Encapsulates an algorithm inside a class

1. Template method

Defer the exact steps of an algorithm to a subclass

1. Visitor

Defines a new operation to a class without change

Benefits of Design pattern:

1. Foresee and rectify future problems easily.
2. Helps in maintaining binary compatibility with subsequent releases.
3. Just by following [SOLID Principles](http://www.vishalchovatiya.com/single-responsibility-principle-in-cpp-solid-as-a-rock/) helps greatly in agile or adaptive software development.
4. The solution facilitates the development of highly [cohesive](https://en.wikipedia.org/wiki/Cohesion_(computer_science)) modules with minimal coupling. Thus, increasing extensibility and reusability.
5. There are some patterns like Facade, Proxy, etc which encapsulates the complexity in itself to provide easy and intuitive interface to the client. Thus, making the overall system easier to understand and reduce learning curve.
6. Design Patterns make communication between designers and developers more crystal and precise. A developer can immediately picture the high-level design in their heads when they refer to the name of the pattern used to solve a particular issue when discussing software design.