**DESIGN PATTERNS**

Design patterns are typical solutions to commonly occurring problems in software design. They are like pre-made blueprints that you can customize to solve a recurring design problem in your code.

There are three types of Design patterns.

Design patterns:

* Creational Design Patterns
* Structural Design Patterns
* Behavioral Design Patterns

**Creational Design Patterns:** Creational design patterns provide various object creation mechanisms, which increase flexibility and reuse of existing code.

Below are the types of Creational design patterns:

**Factory method:**

The factory method provides a way to produce objects in a superclass while allowing subclasses to change the kind of objects that are created.

For example Consider a scenario like where we have to produce a product of two types like a charger of two types (for two kinds of mobiles i.e., an apple mobile and an Android mobile).

Another example is when providing a bank account we have to provide that account based on the type (i.e., Savings and Current)

The main advantage of this pattern is that when we want to scale the types of product further, this pattern provides an easy way to scale the new types into the code.

**Abstract Factory Method:**

Abstract Factory is a creational design pattern that lets you produce families of related objects without specifying their concrete classes. It is nothing but the abstraction over the Factory method.’

As a factory method allows us to create a product of one or more than one type, Abstract factory allows us to create a factory of factories that produces one or more products.

Abstract Factory pattern is useful for creating multiple factories of related objects so that any specific object can be created at runtime from any of the concrete factory classes.

For example, in a food application we are going to provide veg items and non veg items for the users. In Veg, we are going to have food items like Veg-Pizza, veg-burger, etc as well as in non veg we are going to have non-veg pizza, non-veg burger,etc. For this scenario we can implement the Abstract factory method by implementing an abstract factory over veg and non veg and by creating abstract classes for the items like pizza, burger, etc.

**Builder Method:**

The builder method allows us to create a complex object in a step by step procedure with correct sequence.

Here a complex object will be created by going through a number of stages rather than creation in a single step.

For Example if we have to search for a mobile or laptop in an ecommerce website we have to select some features of that product. One user wants to search with one or two features, another user wants to search with 5 features then this method plays a key role.

**Prototype Method:**

Prototype method allows us to hide the complexity of creating clones of the existing objects. It helps us to clone new objects with the help of existing objects, not by building them from scratch. We are going to implement a clone method for the cloning in which the deep cloning took place.

For example, When we have to clone a product which is in demand we have to produce the items in very large amounts then if we build them from scratch it will take a lot of time . To save time, we can use prototype methods to clone the existing objects easily.

**Singleton Method:**

It allows us to restrict the instantiation of one class to one object only.

For example, Sometimes we have to provide a single instance of a particular class to all the users of our application then this singleton method helps us to provide us this feature.

If we are conducting an auction then the bidding price for a particular item should be always available as a centralized single instance for all the participants then we can use the Singleton method.