**Design Pattern**

* Design Patterns are nothing but documented and tested solutions for recurring problems in a given context.
* They are reusable solutions to the problems that as a developer we encounter in our day to day programming.
* One thing you need to remember is that the design patterns are for projects and projects are not for patterns.
* Many developers are enforcing the design pattern into their project which makes the project messy. So, use only when required.

**Types**

* [**Creational Design Pattern**](https://dotnettutorials.net/lesson/creational-design-pattern/) (object creation and initialization)
* [**Structural Design Pattern**](https://dotnettutorials.net/lesson/structural-design-pattern/) (Structural Changes of class and interfaces, and the relationship between classes)
* [**Behavioural HYPERLINK "https://dotnettutorials.net/lesson/behavioral-design-pattern/" Design Pattern**](https://dotnettutorials.net/lesson/behavioral-design-pattern/) (Communication Between Objects)
* **Creational Design Pattern**
* Creational Design Patterns are concerned with the way in which objects are created.
* They reduce complexities and instability by creating objects in a controlled manner.
* The Creational Design Pattern helps us to centralize the object creation logic and depending upon the condition, it will create, initialize, and returns the appropriate object to the client.
* There are 6 Creational Design Patterns –
* Singleton DP
* Factory DP
* Abstract Factory DP
* Fluent Interface DP
* Builder DP
* Prototype DP

1) **Singleton DP** - A class must ensure that only a single instance of a class should be created and can be used by all other classes.

There are 2 types of singleton design patterns -

1. Early Instantiation: the creation of an instance at load time.
2. Lazy Instantiation: the creation of instances when required.

Advantages -

1. Saves memory because the object is not created at each request. Only a single instance is used again and again.

Usage -

1. It is used in multi-threaded, d/b applications, logging, caching, thread pools and configuration settings, etc.

We can create a singleton class by -

1. Static member - It gets memory only once because of static, it contains the instance of the singleton class.
2. Private constructor - It will prevent instantiation of the singleton class from outside class.
3. Static factory method- This provides the global point of access to the singleton object and returns the instance to the caller.

