*Singleton is one of the****Gangs of Four Design Patterns****and comes in the****Creational****Design Pattern category.*

Here we will learn about Singleton design pattern principles, different ways to implement it and best practices to follow along with its usage.

**Singleton Pattern**

* Singleton pattern restricts the instantiation of a class and ensures that only one instance of the class exists per transaction in apex.
* Singleton class must provide a public access point to get the instance of the class for any transaction.
* Singleton pattern can be used for logging and caching related implementation.
* Singleton pattern can also be used with other design patterns like Abstract Factory, Façade, etc.

**Singleton pattern implementation in Apex**

* Private constructor to restrict instantiation of the class from other classes.
* Private static variable of the same class that is the only instance of the class.
* Public static method that returns the instance of the class, this is the public access point for other classes to get the instance of the singleton class.

**Different approaches to implement Singleton pattern**

1. Eager Initialization
2. Lazy Initialization

**Eager Initialization**

* The instance of singleton class is created at the time of class loading.
* This is the easiest method to create a singleton class.

If your singleton class is not using lot of resources, this is the approach to use. Otherwise, we should avoid the instantiation unless client calls the getInstance method.

**Lazy Initialization**

Lazy initialization creates the instance in the public access method if the instance is not created before in the same transaction.

In this pattern, instance will only be instantiated when client will call the getInstance method. This will reduce the effort to load large resources at the time of initialization.