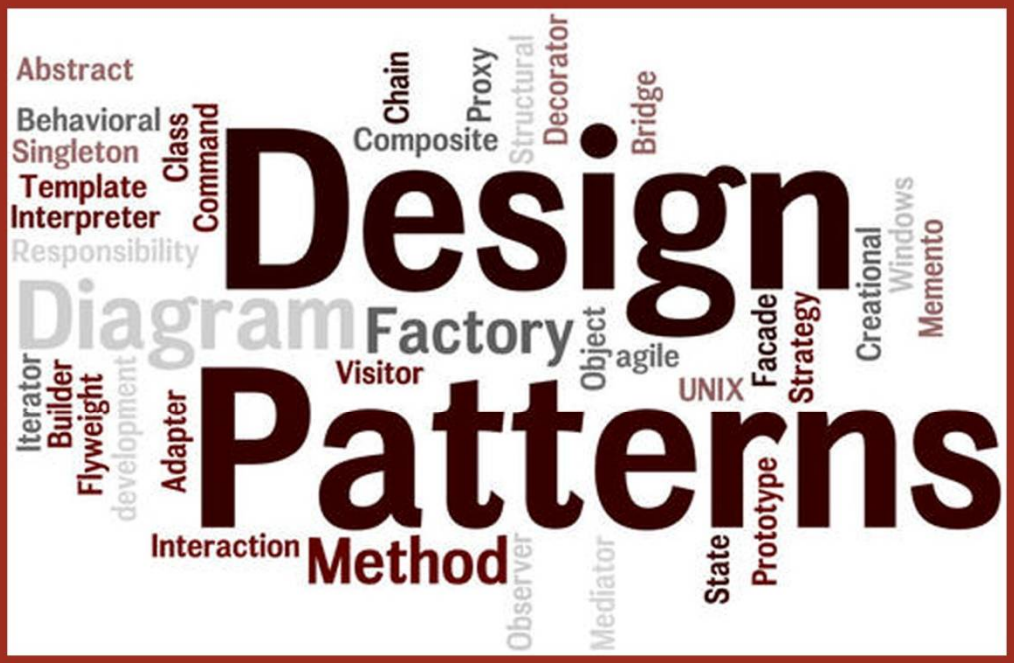
JAVA means DURGA SOFT

Design Patterns

Core Level Design Patterns

2. Singleton Java Class



India's No.1 Software Training Institute

DURGASOFT

www.durgasoft.com Ph: 9246212143 ,8096969696

Singleton Java Class

Problem: Creating multiple objects for java class with same data is wastage of memory.

Solution: Use Singleton java class, which says create only one object for java class and use it for multiple times on each JVM in order to minimize the memory wastage and to increase the performance.

Def: Singleton java class is java class, which allows us to create only one object per **JVM.**

Sometimes it's important to have only one instance for a class. For example, in a system there should be only one window manager (or only a file system or only a print spooler). Usually singletons are used for centralized management of internal or external resources and they provide a global point of access to themselves.

The singleton java class is used to encapsulate the creation of an object in order to maintain control over it. This not only ensures only one object is created, but also allows **lazy instantiation** i.e. the instantiation of object can be delayed until it is actually needed. This is especially beneficial if the constructor needs to perform a costly operation, such as accessing a remote database.

Intent:

- ✓ Ensure that only one instance of a class is created.
- ✓ Provide a global point of access to the object.

Note: For a normal java class if programmer or container is creating only one object even though that class allows to create multiple objects then that java class is not singleton java class. According to this, then a java class of servlet program is not singleton java class. It is a normal java class for which servlet container creates only one object.

Applicability

- The singleton pattern should be used when there must be exactly one instance of a class, and when it must be accessible to clients from a global access point.
- If multiple applications of a project that are running from a single JVM wants to work with objects of java class having same data then it is recommended to make that java class as singleton java class. So that only one object will be allowed to create for that class and we can use that object for multiple tiles in multiple applications.

Ex:

I. Most of the JDBC driver classes are given as singleton java classes for better utilization.

II. In struts environment (1.x), we see **ActionServlet** as singleton java class.

III. In log4j environment, the **Logger** class is given as singleton java class.

Rules To Develop Singleton Java Class:

1. Take private static reference variable of the current class as instance variable.
2. To prevent instantiating the class more than once, take private constructor so that an instance can be created only from within the static method of the class.
3. Develop static factory method, which will return an instance of **current class**, or ***null***, if the class has already been instantiated.

4. Override clone() and return the above created object to suppress the object creation through cloning process.

Sample Code:

```
// SingletonPatternTest.java
class Demo
{
	private static Demo instance;
	private Demo()
	{
		System.out.println("Demo : 0-arg  Con");
	}
	//static factory method
	public static  Demo getInstance()
	{
		// Singleton Logic
		if (instance == null)
			instance = new Demo();
		return instance;
	}
	//override Object class clone()
	public Object clone()
	{
		return  instance;
	}
	//normal method
```

```
        public void called()
        {
                System.out.println("The normal method will be called");
        }
};

public class SingletonPatternTest
{
        public static void main(String arg[]) throws Exception
        {
                System.out.println();
                Demo d1=Demo.getInstance();
                System.out.println("Hashcode of d1 Object: "+d1.hashCode());
                d1.called();
                System.out.println();

                Demo  d2=d1.getInstance();
                System.out.println("Hashcode of d2 Object: "+d2.hashCode());
                d2.called();
                System.out.println();

                Demo d3=(Demo)d1.clone();
                System.out.println("Hashcode of d3 Object: "+d3.hashCode());
                d3.called();
        }//main
}//class
```

Output:

```
D:\Java\Design Patterns\Programs\Singleton\Main>javac SingletonPatternTest.java

D:\Java\Design Patterns\Programs\Singleton\Main>java SingletonPatternTest

Demo : 0-arg  Con
Hashcode of d1 Object: 1671711
The normal method will be called

Hashcode of d2 Object: 1671711
The normal method will be called

Hashcode of d3 Object: 1671711
The normal method will be called
```