JAVA - Singleton class

A java class can only be called as Singleton when it is only giving one instance.

Singleton class can have multiple references, but only one object/ instance creation.

To make java class as Singleton below are the essential points to be considered.

- 1. The class should have Static Reference (Refer below code)
- 2. The class should have Private constructor (Cannot be initialized by others)
- 3. The class should have a Public static method which will return instance.

There two way of an instance creates in singleton classes.

- Load time instance creation (eager/early)
- Whenever required instance creation (demanded/Lazy)

It comes under the singleton design pattern. Please check out below code for more understanding.

package design pattern;

public class Singleton{

private static Singleton singleton;//new Singleton (); (If we create object here then it is eager/early initialization)

```
private Singleton(){
    System.out.println("I am constructor");
}
```

```
public static /*synchronized*/ Singleton getInstance(){ //You can use
synchronization on method/ block level
if(singleton==null){
```

```
synchronized(Singleton.class){
          if(singleton==null){
            System.out.println(Thread.currentThread().getName());
            singleton = new Singleton(); //(That is demanded/Lazy
initialization)
          }
       }
     }
     return singleton;
  }
  public static void main(String[] args){
     Thread task1 = new Thread(new Runnable() {
       @Override
       public void run() {
          System.out.println("Frist Thread>>>>"+Singleton.getInstance());
       }
    },"Frist");
     Thread task2 = new Thread(new Runnable() {
       @Override
       public void run() {
          System.out.println("Second Thread>>>>"+Singleton.getInstance());
       }
     },"Second");
     task1.start();
     task2.start();
  }
}
```

Output:

Frist

I am constructor

Frist Thread>>>>designPattern.Singleton@2ae9093c

Second Thread>>>designPattern.Singleton@2ae9093c

Please add on comments and like this lesson, if it gave you some revision or refreshment on Singleton class concepts.