```
// Eager initialization
class Singleton {
    private static Singleton obj= new Singleton;


    private Singleton() {}

    public static Singleton getInstance()
    {
                return obj;
    }
}
```

```
// Lazy initialization

class Singleton {
    private static Singleton obj;

    private Singleton() {}

    public static Singleton getInstance()
    {
        if (obj == null)
            obj = new Singleton();
        return obj;
    }
}
```

```
// Thread Synchronized Block

class Singleton {
    private static Singleton obj;
    private Singleton() {}

    // Only one thread can execute this at a time
    public static synchronized Singleton getInstance()
    {
        if (obj == null)
            obj = new Singleton();
        return obj;
    }
}
```

```
// Double Checked Locking

class Singleton {
    private static volatile Singleton obj = null;
    private Singleton() {}

    public static Singleton getInstance()
    {
        if (obj == null) {
            // To make thread safe
            synchronized (Singleton.class)
            {

                if (obj == null)
                    obj = new Singleton();
            }
        }
        return obj;
    }
}
```

```java
//  Bill pugh Solution

public class Singleton {

    private Singleton() {
    System.out.println("Instance created");
    }

      private static class SingletonInner{

      private static final Singleton INSTANCE=new Singleton();
    }
    public static Singleton getInstance()
    {
       return SingletonInner.INSTANCE;
    }
}
```