=========

###

 demo

```
public class Singleton {
    //
    private Singleton() {}
    //
    private static final Singleton instance = new Singleton();

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

 `getInstance()` `Singleton`

****

****

```

```
public class Singleton {
    //
    private Singleton() {}
    //
    private static Singleton instance = null;

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

`if` `instance` `Singleton`

###

 new

```
public class Singleton {
    //
    private Singleton() {}
    private static Singleton instance = null;

    public static Singleton getInstance() {
        //
        if (instance == null) {
            //
            synchronized (Singleton.class) {
                //
                if (instance == null) {
                    instance = new Singleton();
                }
            }
        }
        return instance;
    }
}
```

`synchronized``Singleton`

A B B new

###

`instance = new Singleton();`

* a.
* b.
* c. `instance`

a b c

a c b `instance` `return instance;`

[](http://cxyroad.com/ "https://mp.weixin.qq.com/s/CrXYubsQ32DuIDxtnVqFrQ")
`volatile`

```
public class Singleton {
    //
    private Singleton() {}
    private volatile static Singleton instance = null;

    public static Singleton getInstance() {
        //
        if (instance == null) {
            //
            synchronized (Singleton.class) {
                //
                if (instance == null) {
                    instance = new Singleton();
                }
```

```
        }
    }
    return instance;
    }
}
```

###

****


```
public class Singleton {
    //
    private Singleton() {}

    private static class SingletonHolder {
        //
        private static final Singleton INSTANCE = new Singleton();
    }

    public static Singleton getInstance() {
        //
        return SingletonHolder.INSTANCE;
    }
}
```

`getInstance()`

* `Singleton`
* `true`
* `newInstance`

```
    // 
    Constructor constructor = Singleton.class.getDeclaredConstructor();
    // 
    constructor.setAccessible(true);
    // 
    Singleton singleton1 = (Singleton)constructor.newInstance();
    Singleton singleton2 = (Singleton)constructor.newInstance();
    // 
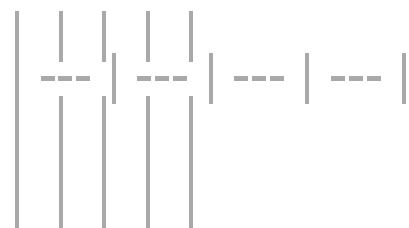    System.out.println(singleton1.equals(singleton2));
```

`false`

****


```
public enum SingletonEnum {
    INSTANCE;
}
```


"****"

"”****Java

: https://juejin.cn/post/7358011504966877236