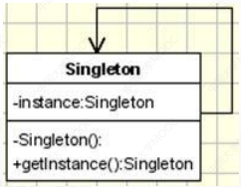
单例模式

» 单例模式从某种程度而言属于“非创建”模式，但是依然将其和其他创建型设计模式放在一起

» 目的:

- 保证每一个类只有一个对应的实例被创建
- 提供一个指向这一单一实例的全局指针

单例模式



单例模式

» 参与者

- 参与此种模式的类或对象

» Singleton

- 定义了让用户访问其唯一实例的实例化操作
- 获取实例方法是一个类操作
- 负责创造和维护唯一实例

单例模式1

```
class Singleton
{
        private static Singleton instance;

        private Singleton()
        {
        System.out.println("Singleton(): Initializing Instance");
        }
```

单例模式1

```
public static Singleton getInstance(){
        if (instance == null) {
                synchronized(Singleton.class){
                        if (instance == null){
                        System.out.println("getInstance(): First time
getInstance was invoked!");
                        instance = new Singleton();   }
                                }
                        }

                        return instance;
        }
}
```

单例模式2

```
class Singleton
{
        private static Singleton instance = new Singleton();

        private Singleton()
        {
                System.out.println("Singleton(): Initializing
Instance");
        }
```

单例模式2

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

        public void doSomething()
        {
                System.out.println("doSomething(): Singleton
does something!");
        }
}
```

单例模式3

```
public class Singleton {
  // Private constructor prevents instantiation from other classes
  private Singleton() {}
 private static class SingletonHolder {
    private static final Singleton INSTANCE = new Singleton();
  }
 public static Singleton getInstance() {
    return SingletonHolder.INSTANCE;
  }
}
```