第9章

单例模式

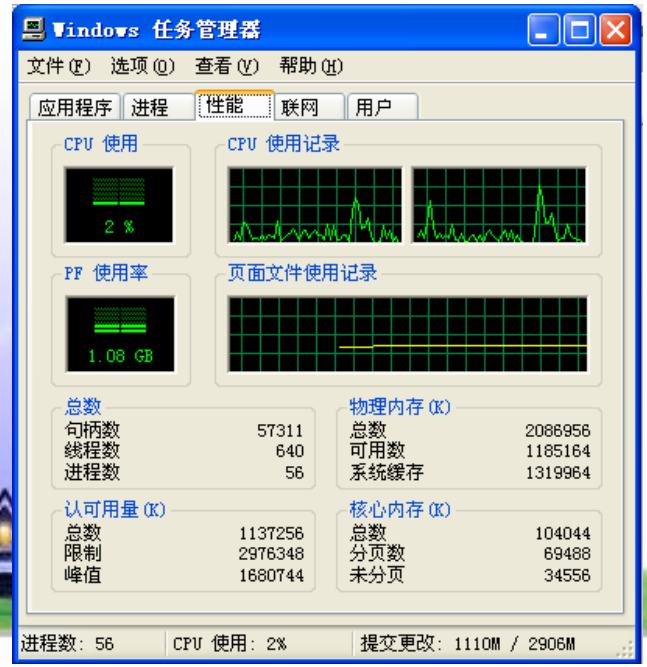
主讲教师：程细柱　韶关学院计算机系
本书主编：刘　伟　　清华大学出版社

本章教学内容

- 单例模式
  - ✓ 模式动机与定义
  - ✓ 模式结构与分析
  - ✓ 模式实例与解析
  - ✓ 模式效果与应用
  - ✓ 模式扩展

单例模式

- 模式动机
  - ✓ 对于系统中的某些类来说，只有一个实例很重要，例如，一个系统中可以存在多个打印任务，但是只能有一个正在工作的任务；一个系统只能有一个窗口管理器或文件系统；一个系统只能有一个计时工具或ID（序号）生成器。



单例模式

- 模式动机
  - ✓ 如何保证一个类只有一个实例并且这个实例易于被访问呢？定义一个全局变量可以确保对象随时都可以被访问，但不能防止我们实例化多个对象。
  - ✓ 一个更好的解决办法是让类自身负责保存它的唯一实例。这个类可以保证没有其他实例被创建，并且它可以提供一个访问该实例的方法。这就是单例模式的模式动机。

单例模式

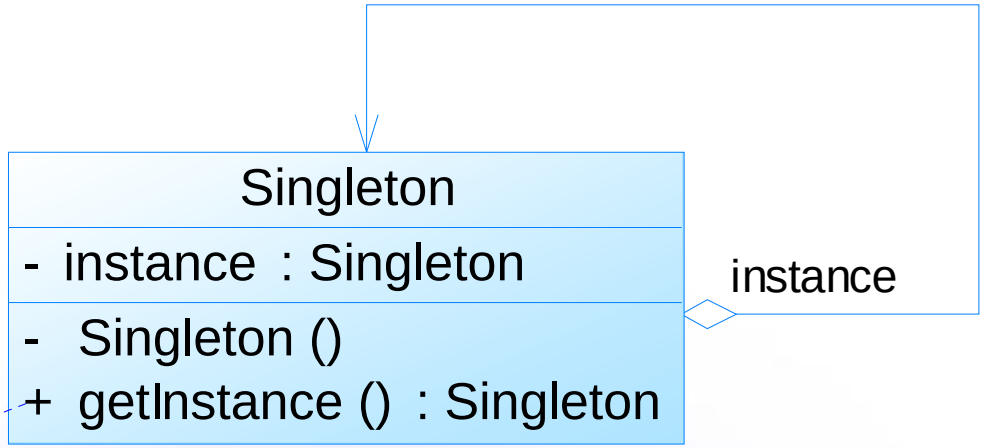
- 模式定义
  - ✓ 单例模式(Singleton Pattern)：单例模式确保某一个类只有一个实例，而且自行实例化并向整个系统提供这个实例，这个类称为单例类，它提供全局访问的方法。
  - ✓ 单例模式的要点有三个：一是某个类只能有一个实例；二是它必须自行创建这个实例；三是它必须自行向整个系统提供这个实例。单例模式是一种对象创建型模式。单例模式又名单件模式或单态模式。

单例模式

- 模式定义
  - ✓ **Singleton Pattern:** Ensure a class has **only one instance** and **provide a global point of access to it**.
  - ✓ Frequency of use: **medium high**

单例模式

- 模式结构




```
if(instance==null)
        instance=new Singleton();
return instance;
```

单例模式

- **模式结构**
  - ✓单例模式包含如下角色:
    - Singleton: 单例

单例模式

- 模式分析
  - ✓ 单例模式的目的是保证一个类仅有一个实例，并提供一个访问它的全局访问点。单例模式包含的角色只有一个，就是单例类——Singleton。单例类拥有一个私有构造函数，确保用户无法通过new关键字直接实例化它。除此之外，该模式中包含一个静态私有成员变量与静态公有的工厂方法，该工厂方法负责检验实例的存在性并实例化自己，然后存储在静态成员变量中，以确保只有一个实例被创建。

单例模式

- 模式分析
  - ✓ 单例模式的实现代码如下所示:

```
public class Singleton
{
    private static Singleton instance=null;  //静态私有成员变量
    //私有构造函数
    private Singleton(){}
      //静态公有工厂方法，返回唯一实例
    public static Singleton getInstance() {
        if(instance==null)
            instance=new Singleton();
        return instance;
    }
}
```

单例模式

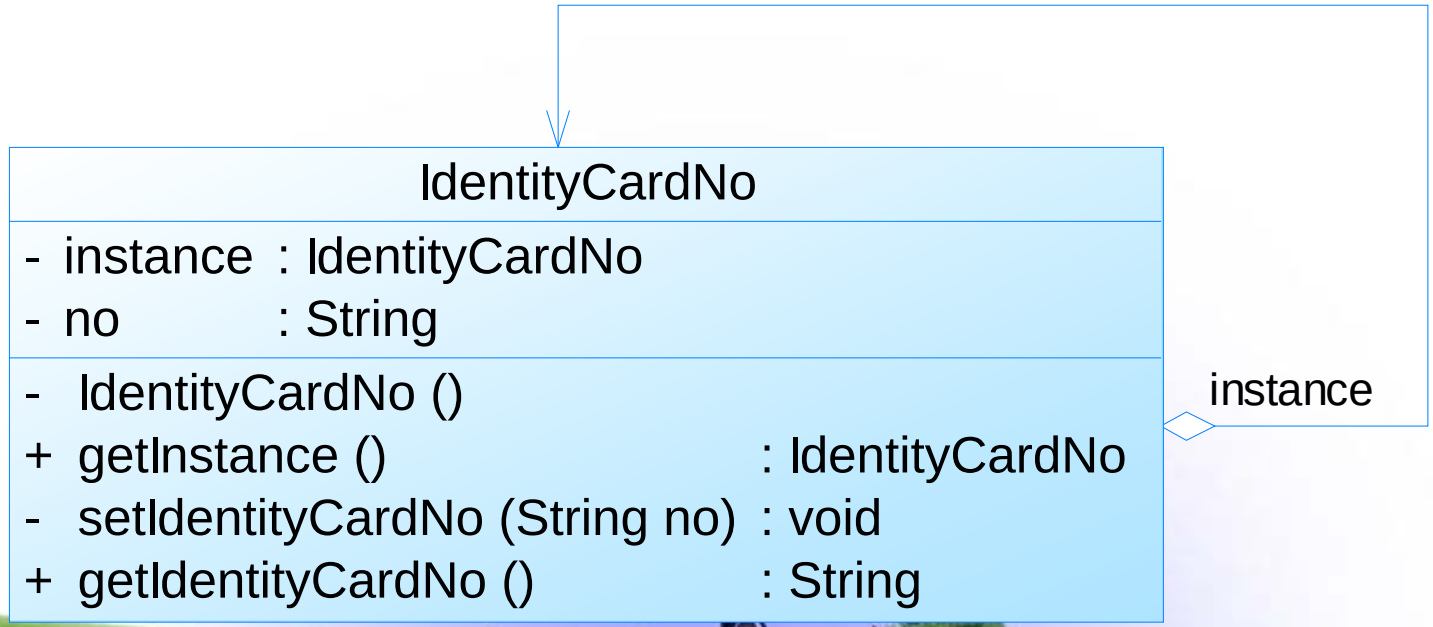
- **模式分析**
  - ✓在单例模式的实现过程中，需要注意如下三点：
    - 单例类的构造函数为私有；
    - 提供一个自身的静态私有成员变量；
    - 提供一个公有的静态工厂方法。

单例模式

- **单例模式实例与解析**
  - ✓ 实例一：身份证号码
    - 在现实生活中，居民身份证号码具有唯一性，同一个人不允许有多个身份证号码，第一次申请身份证时将给居民分配一个身份证号码，如果之后因为遗失等原因补办时，还是使用原来的身份证号码，不会产生新的号码。现使用单例模式模拟该场景。

单例模式

- 单例模式实例与解析
  - ✓实例一：身份证号码



单例模式

- **单例模式实例与解析**
  - ✓ 实例一：身份证号码
    - 参考代码 **(Chapter 09 Singleton\sample01)**

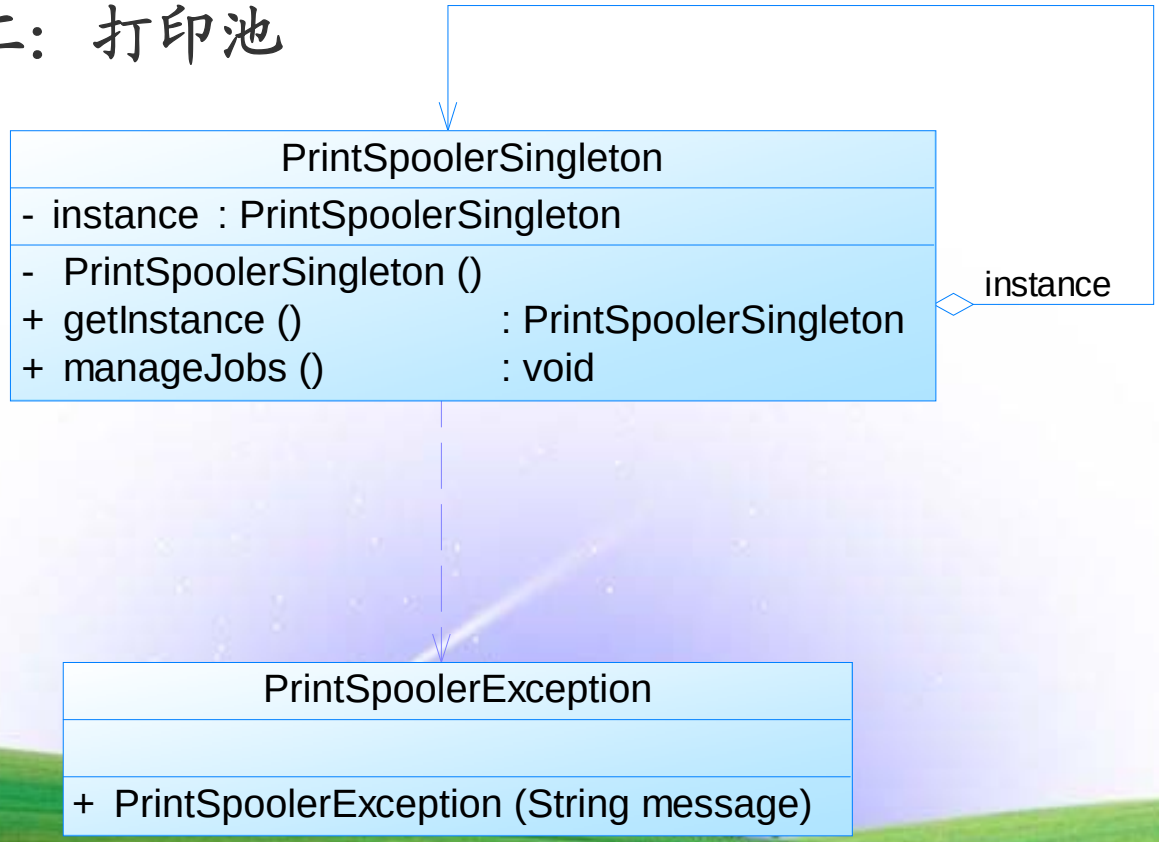
演示……

单例模式

- **单例模式实例与解析**
  - ✓ 实例二：打印池
    - 在操作系统中，打印池(Print Spooler)是一个用于管理打印任务的应用程序，通过打印池用户可以删除、中止或者改变打印任务的优先级，在一个系统中只允许运行一个打印池对象，如果重复创建打印池则抛出异常。现使用单例模式来模拟实现打印池的设计。

单例模式

- 单例模式实例与解析
  - ✓ 实例二：打印池



单例模式

- **单例模式实例与解析**
  - ✓ 实例二：打印池
    - 参考代码 (Chapter 09 Singleton\sample02)

演示……

单例模式

- **模式优缺点**
  - ✓ 单例模式的优点
    - 提供了对唯一实例的受控访问。因为单例类封装了它的唯一实例，所以它可以严格控制客户怎样以及何时访问它，并为设计及开发团队提供了共享的概念。
    - 由于在系统内存中只存在一个对象，因此可以节约系统资源，对于一些需要频繁创建和销毁的对象，单例模式无疑可以提高系统的性能。
    - 允许可变数目的实例。我们可以基于单例模式进行扩展，使用与单例控制相似的方法来获得指定个数的对象实例。

单例模式

- **模式优缺点**
  - ✓ 单例模式的缺点
    - 由于单例模式中没有抽象层，因此单例类的扩展有很大的困难。
    - 单例类的职责过重，在一定程度上违背了“单一职责原则”。因为单例类既充当了工厂角色，提供了工厂方法，同时又充当了产品角色，包含一些业务方法，将产品的创建和产品的本身的功能融合到一起。
    - 滥用单例将带来一些负面问题，如为了节省资源将数据库连接池对象设计为单例类，可能会导致共享连接池对象的程序过多而出现连接池溢出；现在很多面向对象语言(如Java、C#)的运行环境都提供了自动垃圾回收的技术，因此，如果实例化的对象长时间不被利用，系统会认为它是垃圾，会自动销毁并回收资源，下次利用时又将重新实例化，这将导致对象状态的丢失。

单例模式

- **模式适用环境**
  - ✓ 在以下情况下可以使用单例模式:
    - 系统只需要一个实例对象，如系统要求提供一个唯一的序列号生成器，或者需要考虑资源消耗太大而只允许创建一个对象。
    - 客户调用类的单个实例只允许使用一个公共访问点，除了该公共访问点，不能通过其他途径访问该实例。
    - 在一个系统中要求一个类只有一个实例时才应当使用单例模式。反过来，如果一个类可以有几个实例共存，就需要对单例模式进行改进，使之成为多例模式。

单例模式

- 模式应用
  - ✓ (1) java.lang.Runtime类

```
public class Runtime {
    private static Runtime currentRuntime =
new Runtime();
    public static Runtime getRuntime() {
        return currentRuntime;
    }
    private Runtime() {}
    ......
}
```

单例模式

- 模式应用
  - ✓(2) 一个具有自动编号主键的表可以有多个用户同时使用，但数据库中只能有一个地方分配下一个主键编号，否则会出现主键重复，因此该主键编号生成器必须具备唯一性，可以通过单例模式来实现。

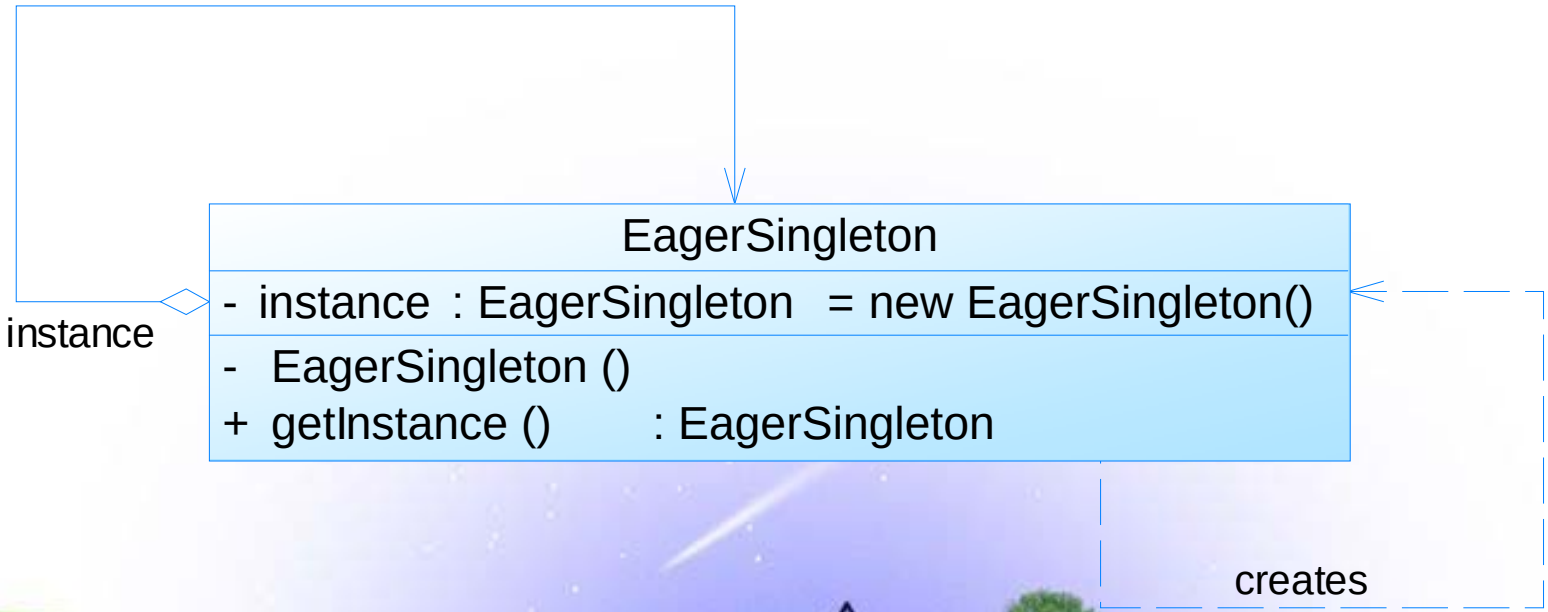
单例模式

- 模式应用
  - ✓(3) 默认情况下，Spring会通过单例模式创建bean实例:

```
<bean id="date" class="java.util.Date"
   scope="singleton"/>
```

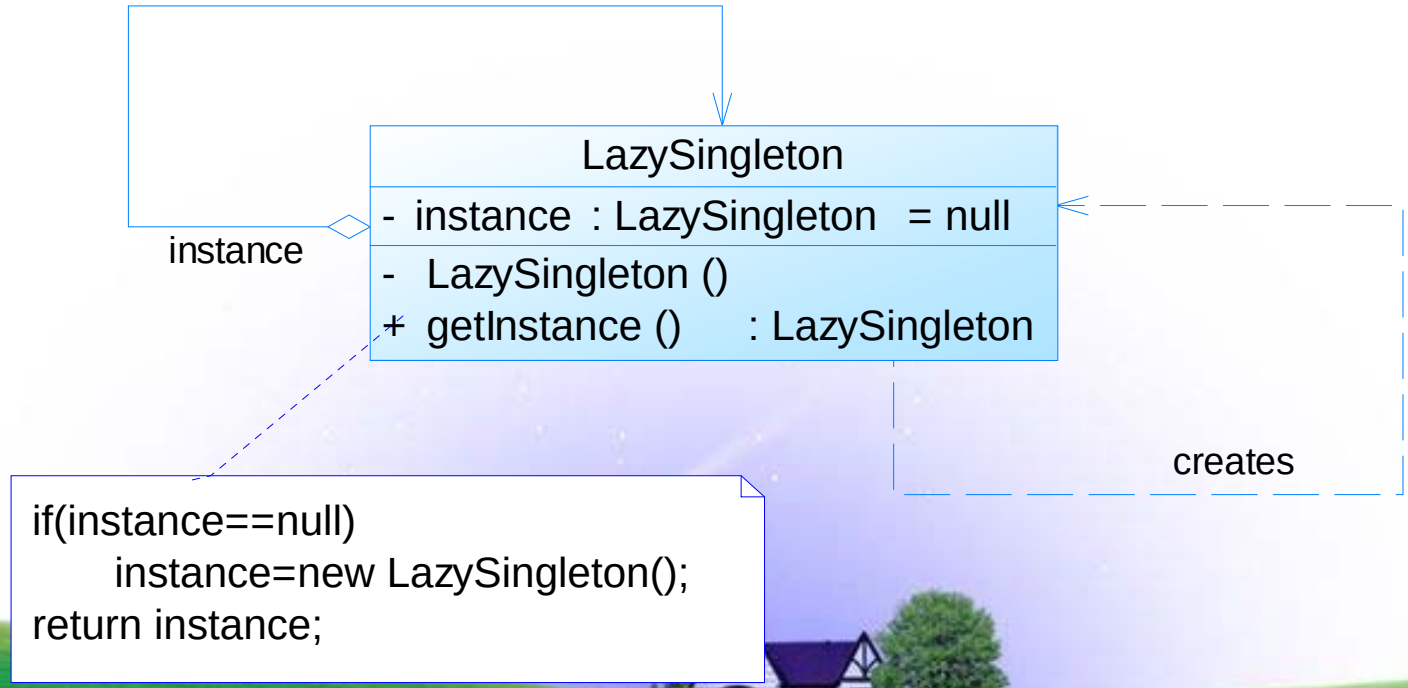
单例模式

- **模式扩展**
  - ✓ 饿汉式单例类：在自己被加载时就将自己实例化。



单例模式

- 模式扩展
  - ✓ 懒汉式单例类：在自己第一次被引用时将自己实例化。



单例模式

- **模式扩展**
  - ✓饿汉式单例与懒汉式单例类比较
    - 饿汉式单例类在自己被加载时就将自己实例化。单从资源利用效率角度来讲，这个比懒汉式单例类稍差些。从速度和反应时间角度来讲，则比懒汉式单例类稍好些。
    - 懒汉式单例类在实例化时，必须处理好在多个线程同时首次引用此类时的访问限制问题，特别是当单例类作为资源控制器，在实例化时必然涉及资源初始化，而资源初始化很有可能耗费大量时间，这意味着出现多线程同时首次引用此类的机率变得较大，需要通过同步化机制进行控制。

本章小结

- 单例模式确保某一个类只有一个实例，而且自行实例化并向整个系统提供这个实例，这个类称为单例类，它提供全局访问的方法。单例模式的要点有三个：一是某个类只能有一个实例；二是它必须自行创建这个实例；三是它必须自行向整个系统提供这个实例。单例模式是一种对象创建型模式。
- 单例模式只包含一个单例角色：在单例类的内部实现只生成一个实例，同时它提供一个静态的工厂方法，让客户可以使用它的唯一实例；为了防止在外部对其实例化，将其构造函数设计为私有。

本章小结

- 单例模式的目的是保证一个类仅有一个实例，并提供一个访问它的全局访问点。单例类拥有一个私有构造函数，确保用户无法通过new关键字直接实例化它。除此之外，该模式中包含一个静态私有成员变量与静态公有的工厂方法。该工厂方法负责检验实例的存在性并实例化自己，然后存储在静态成员变量中，以确保只有一个实例被创建。
- 单例模式的主要优点在于提供了对唯一实例的受控访问并可以节约系统资源；其主要缺点在于因为缺少抽象层而难以扩展，且单例类职责过重。
- 单例模式适用情况包括：系统只需要一个实例对象；客户调用类的单个实例只允许使用一个公共访问点。

END

Thanks!