JAVA WEB编程基础

第5讲：面向对象编程思想—封装

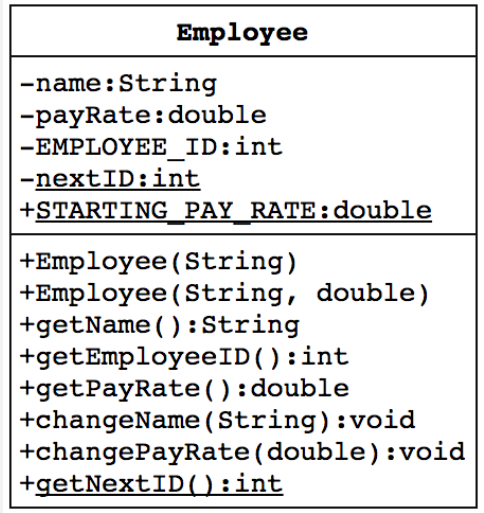
主讲人：康育哲

本讲内容

- static
  - 静态成员（static member）
  - 初始化代码块（initializer block）
  - 类加载器（class loader）
- final
  - final的意义
  - final vs abstract
- package & import
  - 导入类
  - 导入类成员
  - 创建包
  - 存档（JAR）

STATIC

- 静态成员
  - 一个类的内部声明的static成员（变量/方法）属于这个类，被所有实例共享，故又称类成员
    - 没有用static声明的成员属于实例，每个实例拥有一份，又称实例成员
  - 静态成员的引用：ClassName.variableName或ClassName.methodName()
    - 实例成员的引用：anInstanceName.variableName或anInstanceName.methodName()
  - 静态成员的作用：可作为全局变量
  - UML标记：类名加下划线

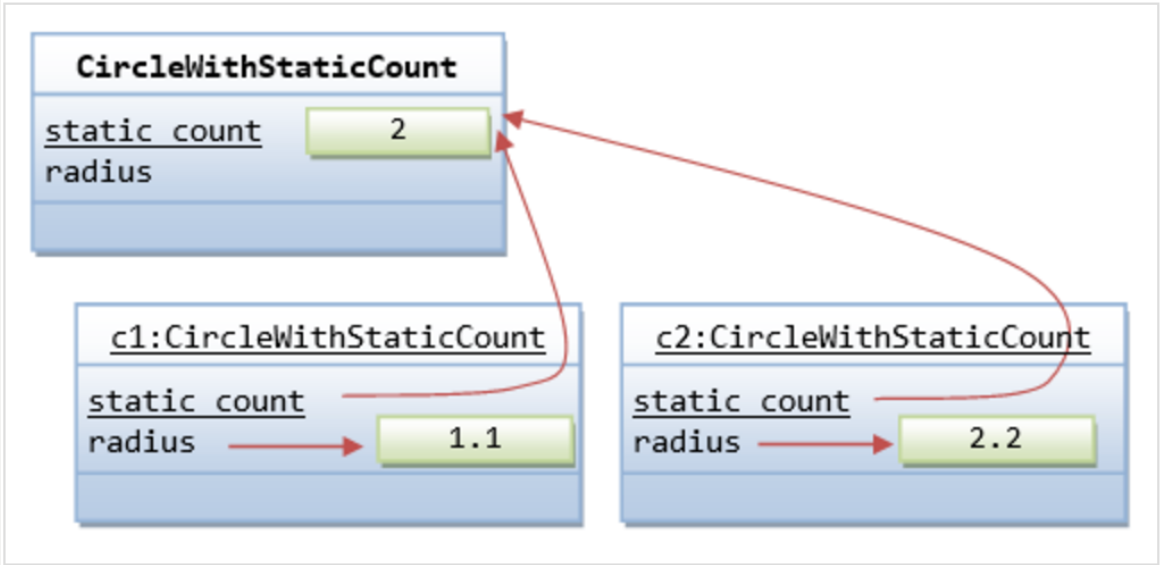


STATIC

- 静态成员
  - 示例：假设需要统计类Circle共创建了多少个实例，该怎么做?
    - 问题：使用实例成员不管用
    - 原因：每个实例存一份属于自己的成员变量count
    - 解决方案：使用static成员变量，所有实例可共享

```
public class Circle {
   public int count = 0;  // To count the number of instances created.
                          // Set to public to simplify access.
   private double radius;
   public Circle(double radius) {
      this.radius = radius;
      ++count;
   }
}
```

```
public class TestCircle {
   public static void main(String[] args) {
      Circle c1 = new Circle(1.1);
      System.out.println(c1.count);  // Output: 1
      Circle c2 = new Circle(2.2);
      System.out.println(c2.count);  // Output: 1
      Circle c3 = new Circle(3.3);
      System.out.println(c3.count);  // Output: 1
   }
}
```



STATIC

CircleWithStaticCount.java

```
1  public class CircleWithStaticCount {
2     public static int count = 0;  // A static variable to count the number of instances created
3                                   // shared by all the instances
4                                   // Set to public to simplify access
5     private double radius; // An instance variable for each circle to maintain its own radius
6     public CircleWithStaticCount(double radius) {
7        this.radius = radius;
8        ++count;            // one more instance created
9     }
10 }
```

A Test Driver

```
1  public class TestCircleWithStaticCount {
2     public static void main(String[] args) {
3        CircleWithStaticCount c1 = new CircleWithStaticCount(1.1);
4        System.out.println(c1.count);   // 1
5        System.out.println(CircleWithStaticCount.count);  // Can access static variable via classname too
6        CircleWithStaticCount c2 = new CircleWithStaticCount(2.2);
7        System.out.println(CircleWithStaticCount.count);  // 2
8        System.out.println(c1.count);  // 2
9        System.out.println(c2.count);  // 2
10       CircleWithStaticCount c3 = new CircleWithStaticCount(3.3);
11       System.out.println(CircleWithStaticCount.count);  // 3
12       System.out.println(c1.count);  // 3
13       System.out.println(c2.count);  // 3
14       System.out.println(c3.count);  // 3
15    }
16 }
```

STATIC

- 初始化代码块
  - 写法：代码块前添加static标识
  - 只在JVM加载类的时候执行一次

```
public class Foo {
   static int number;       // a static variable
   static {                 // a static initializer block - run only once when the class is loaded
      number = 88;
      System.out.println("loading class...");
   }
   .......
}
```

STATIC

- 类加载器（class loader）
  - JVM有一个全局对象—l类加载器（类java.lang.ClassLoader的唯一实例），负责在Java程序启动的时候把所有的类定义加载到内存
  - JVM加载类的时候完成所有静态成员的初始化
  - 初始化执行顺序：按代码出现的顺序

```
public class Hello {
   private static int number1 = 11;  // explicit initializer
   static {                          // static initializer
      number1 = 99;
      number2 = 88;
   }
   private static int number2 = 22;  // explicit initializer

   public static void main(String[] args) {
      System.out.println("number1 is " + number1);  // 99
      System.out.println("number2 is " + number2);  // 22
   }
}
```

FINAL

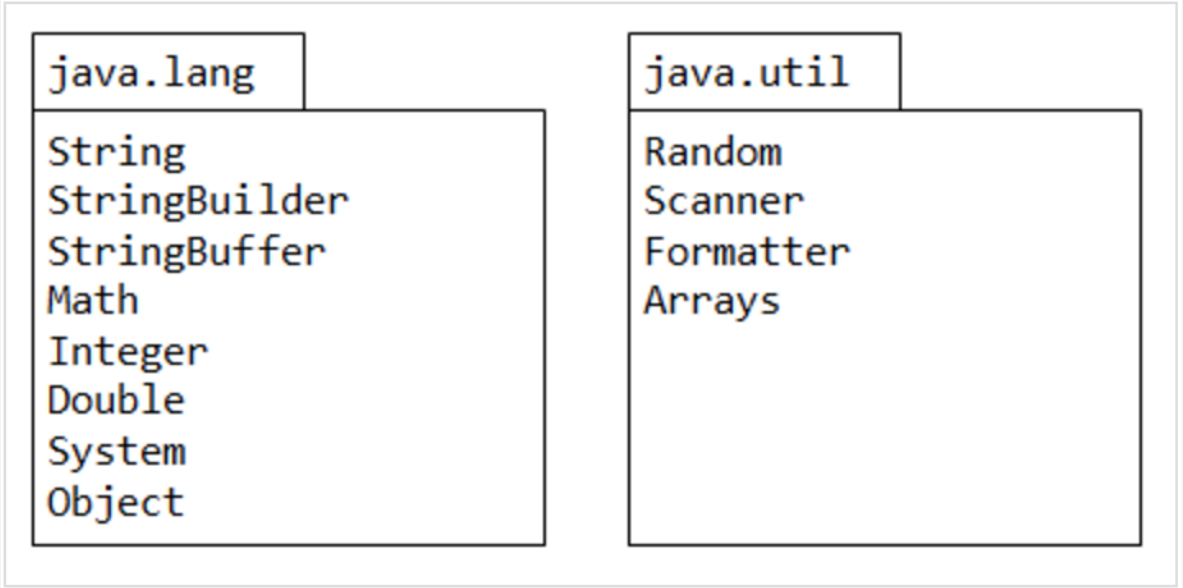
- final class
  - 不可以被继承
- final method
  - 不可以被重写
- final variable
  - 不可以被修改
  - 原生类型加final：值不可以改变
  - 对象类型加final：引用的对象不可以改变，但对象内部的数据可以改变
  - static final：静态成员常量，一般用作全局常量

FINAL

- 命名规范：名词或名词词组，单词之间用_隔开
  - 例：MIN_WIDTH, MAX_VALUE, PI, RED
- final vs abstract
  - final class：不可继承
  - abstract class：必须继承
  - final method：不可重写
  - abstract method：必须重写

PACKAGE & IMPORT

- 包（package）
  - 概念：相当于程序库（library），是一系列类、接口以及一系列相关定义的集合
  - 作用
    - 组织代码
    - 解决命名冲突
    - 控制访问
    - 发布类库
  - 命名规范
    - 项目名+组织域名，倒序，用.隔开
    - 例：com.zeta.project
  - 目录结构
    - 包名中每一个.代表一级目录
    - 例：com.zeta.project.Hello => src/com/zeta/project/Hello.java



PACKAGE & IMPORT

- 导入类
  - 在文件头部添加import语句
    - 导入某个类：import java.util.Scanner;
    - 导入全部类：import java.util.*;
  - 在代码中引用已导入的类，可以只用类名
    - Scanner in = new Scanner(System.in);
  - 在代码中引用没有导入的类，必须使用全名
    - java.util.Scanner in = new java.util.Scanner(System.in);
  - 同一个包中的类自动导入，不需要声明import

PACKAGE & IMPORT

- 导入类成员
  - 在文件头部添加import static语句

```
import static packagename.classname.staticVariableName;
import static packagename.classname.staticMethodName;
import static packagename.classname.*;   // wildcard * denotes all static variables/methods of the class
```

- 类的访问控制
  - public class：所有外部类可见
  - protected class：本包内部的所有类可见
  - private class：只有当前文件内部可见

PACKAGE & IMPORT

- 创建包
  - 在文件头部添加package语句
    - package com.zeta.test;
  - 匿名包
    - 没有声明package的文件会被归为匿名包
    - 匿名包的内容不能被导入到其它包，故不推荐使用
- 存档（JAR）
  - 把一个Java包打包压缩成单个文件，便于发布和导入