链滴

【Java 复习】关于 static 关键字

作者：c3gen

原文链接：https://ld246.com/article/1483608845724

来源网站：链滴

最近在复习回顾Java的学习，然后就变复习，边整理总结。

static 关键字 很常见，想想用法无非是两种情况：

1.为某特定数据类型或对象分配单一的存储空间。

2.实现某个方法或属性与类而不是对象关联在一起

这两种 都是为了避开 与对象的关联。

所以简言之，static是跟类有关的，跟对象没关系，下面再详细的说：

具体的代码来说 ， static主要有4中使用情况：成员变量、成员方法、代码块和内部类

1）成员变量

成员变量分为 静态变量 和 实例变量。

很好的理解就是只要加了static ，

那么这个变量就属于类，他跟对象是没有关系的。只要静态变量所在的类被加载，这个静态变量就会分配空间，因此就可以被使用了。对静态变量的引用有两种方式，分别是“类.静态变量"和”对象.静变量"

实例变量属于对象，只有对象被创建后，实例变量才会被分配内存空间，才能被使用，它在内存中存多个复制，只有用“对象.实例变量”的方式来引用。

2）成员方法

static方法是类的方法，不需要创建对象就可以被调用，而非static方法是对象的方法，只有对象被创出来后才可以被使用。

static方法中不能使用this和super关键字，不能调用非static方法，只能访问所属类的静态成员变量成员方法，因为当static方法被调用时，这个类的对象可能还没被创建，即使已经被创建了，也无法定调用哪个对象的方法。同理，static方法也不能访问非static类型的变量。

关于static修饰的方法，最常用的就是单例设计模式，一开始我们在学设计模式的时候就遇到。

```
public class SingleTest {
    // 创建对象的变量
    private static SingleTest install = null;

    // 私有化构造函数
    private SingleTest() {
        System.out.println("123456");
    }

    // 创建对象的方法
    public static SingleTest getInstance() {
        if (install == null) {
            install = new SingleTest();
        }
        return install;
    }
```

```
}
```

这样 其他的类要想访问SingleTest 就只能通过 SingleTest.getInstance();对外暴露的方法来访问这对象。

3）代码块

static代码块在类中是独立于成员变量和成员函数的代码块的，所以这代码块只会被执行一次。

测试代码如下。

```
public class Test {
    // 静态变量
    private static int a = 0;
    static {
        a = 10;
        System.out.println("静态代码块 a =" + a);
    }

    {
        a = 20;
        System.out.println("非静态代码块  a =" + a);
    }

    public Test() {
        System.out.println("无参构造方法在执行a=" + a);
    }

    public Test(String string) {
        System.out.println("带参构造方法中的值 string = " + string);
        System.out.println("带参构造方法中的值 a =" + a);
    }

    public static void main(String[ ] args) {
        Test test = null;
        System.out.println("----------------------");
        test = new Test();
        Test test2 = new Test("带参");
    }

}
```

4）内部类

内部类，按照名称来说就是类里面的类，static关键字可以修饰内部类，但是不能修饰外部类。

静态static内部类和非静态的内部类的区别如下：

a) 内部静态类不需要有指向外部类的引用。但非静态内部类需要持有对外部类的引用。

b) 非静态内部类能够访问外部类的静态和非静态成员。静态类不能访问外部类的非静态成员。他只能问外部类的静态成员。

c) 一个非静态内部类不能脱离外部类实体被创建，一个非静态内部类可以访问外部类的数据和方法，为他就在外部类里面。

```
/**
 *
 *
 * Module:
 *
 * Outer.java
 *
 * @author c3gen
 * @since JDK 1.8
 * @version 1.0
 * @description: 测试内部类
 */
public class Outer {
    private static String str = " test  class ";

    public static class NestedStaticClass {
        // 静态内部类只能访问外部类的静态成员
        public void privatePrintMsg() {
            // 试着将str改成非静态的，这将导致编译错误
            System.out.println("静态内部类: " + str);
        }
    }

    // 非静态内部类
    public class InnerClass {
        // 不管是静态方法还是非静态方法都可以在非静态内部类中访问
        public void publicPrintMsg() {
            System.out.println("非静态内部类: " + str);
        }
    }

    public static void main(String[ ] args) {
        // 创建静态内部类的实例
        Outer.NestedStaticClass nsc = new Outer.NestedStaticClass();
        // 创建静态内部类的非静态方法
        nsc.privatePrintMsg();
        // 为了创建非静态内部类，我们需要外部类的实例
        Outer outer = new Outer();
        Outer.InnerClass inner = outer.new InnerClass();
        // 调用非静态内部类的非静态方法
        inner.publicPrintMsg();
        // 我们也可以结合以上步骤，一步创建的内部类实例
        Outer.InnerClass innerObject = new Outer().new InnerClass();
        // 同样我们现在可以调用内部类方法
        innerObject.publicPrintMsg();

    }
}
```

结果：

静态内部类: test class

非静态内部类: test class

非静态内部类: test class

通过结果 我们可以分析 ： 非静态内部类是附属在外部类对象上的，需要先实例化一个外部类的对象通过外部类对象才能实例化非静态内部类；而静态内部类可以看做是直接附属在外部类上的，这个静代表附属体是外部类，而不是外部类实例；外部类在进程中是唯一的，而静态内部类不需要唯一，可生成多个实例。