

35从零开始学Java35之析构方法又是咋回事？

前言

配套开源项目资料

一. 析构方法

1. 概念
2. 作用
3. 特点

二. 基本使用

1. finalize简介

2. 代码案例

2.1 Counter计数器

2.2 CounterTest测试类

四. 结语

五. 今日作业

1. 第一题

作者：孙玉昌，昵称【**一一哥**】，另外【**壹壹哥**】也是我哦

千锋教育高级教研员、CSDN博客专家、万粉博主、阿里云专家博主、掘金优质作者

前言

在上一篇文章中，**壹哥**给大家详细地介绍了构造方法的使用、特点等内容。我们知道，构造方法用于创建和初始化类对象。也就是说，构造方法负责”生出“一个类对象，并可以在对象出生时进行必要的操作。那么有的同学就好奇了，既然有人负责对象的”出生“，那有没有人负责对象的”死亡“呢？如果你能这样思考，说明你很爱钻研，是个做java程序员的好苗子。

java中还真有专门负责对象”死亡“的方法！这种方法就叫做”析构方法“！

那么接下来在今天的文章中，**壹哥**会给大家简单介绍一下析构方法。之所以是简单的介绍，主要是因为该方法相对来说，并没有那么重要。而且即使我们实现了，也不一定就很靠谱，所以我们暂作

了解即可。

-----前戏已做完，精彩即开始-----

全文大约【1600】字，不说废话，只讲可以让你学到技术、明白原理的纯干货！本文带有丰富的案例及配图视频，让你更好地理解 and 运用文中的技术概念，并可以给你带来具有足够启迪的思考.....

配套开源项目资料

Github:

GitHub – SunLtd/LearnJava

Contribute to SunLtd/LearnJava development by creating an account on GitHub.

GitHub

Gitee:



一一哥/从零开始学Java

从零开始学Java系列 稀土掘金专栏地址: <https://juejin.cn/column/7175082165548351546> CSDN专...

Gitee

一. 析构方法

1. 概念

我们现在已经知道，构造方法负责创建一个Java的类对象，并可以对该对象进行初始化。与此相对应的，其实还有一个方法，可以负责对象的销毁，这个负责对象销毁的方法，就叫做析构方法。在Java中，有一个专门的析构方法，即finalize()方法！

2. 作用

finalize()析构方法负责回收Java对象所占用的内存，该方法一般是在对象被垃圾收集器回收之前调用。通常我们会在finalize()方法中，指定对象销毁时要执行的操作，比如关闭对象打开的文件、

IO流、释放内存资源等清理垃圾碎片的工作。

3. 特点

finalize()析构方法具有以下这些特点：

- 垃圾回收器是否会执行finalize方法，以及何时执行该方法，是不确定的；
- finalize()方法有可能会使对象复活，恢复到可触及的状态；
- 垃圾回收器执行finalize()方法时，如果出现异常，垃圾回收器不会报告异常，程序会继续正常运行。

二. 基本使用

1. finalize简介

在大多数情况下，Java的内存和垃圾回收都是由JVM的GC机制来自动完成。如果我们想手动实现，就可以使用finalize()方法，但该方法的执行与否是不确定的。也就是说，即使我们调用了finalize()方法，JVM也不一定就会立刻执行垃圾回收操作，这个取决于当前系统的内存占用情况。

另外finalize()是一个被protected关键词修饰的方法，可以确保该方法不会被该类以外的代码调用。在每个Java类中都有finalize()方法，我们可以复写当前类中的finalize()方法。

```
1  @Override
2  protected void finalize(){
3      // 在这里终结代码
4  }
```

Java | 复制代码

2. 代码案例

接下来壹哥设计一个代码案例，来给大家讲讲析构方法的基本使用。

2.1 Counter计数器

我们先来定义一个Counter计数器类，设计一个静态的计数器变量，通过方法调用改变count的值。

```
Java | 复制代码

1  /**
2   * @author 一一哥Sun
3   * QQ: 2312119590
4   * CSDN、掘金、知乎找我哦
5   */
6  public class Counter {
7
8      private static int count = 0; // 计数器变量
9
10     public Counter() {
11         // 构造方法
12         this.count++; // 创建实例时增加值
13     }
14
15     public int getCount() {
16         // 获取计数器的值
17         return this.count;
18     }
19
20     @Override
21     protected void finalize() {
22         // 析构方法
23         this.count--; // 实例销毁时减少值
24         System.out.println("对象销毁");
25     }
26
27 }
```

以上案例中，static、@override等关键字，有些同学可能还不明白是怎么回事！先别急，后面我们都会学到，这里请大家先照着代码练习一下。

2.2 CounterTest测试类

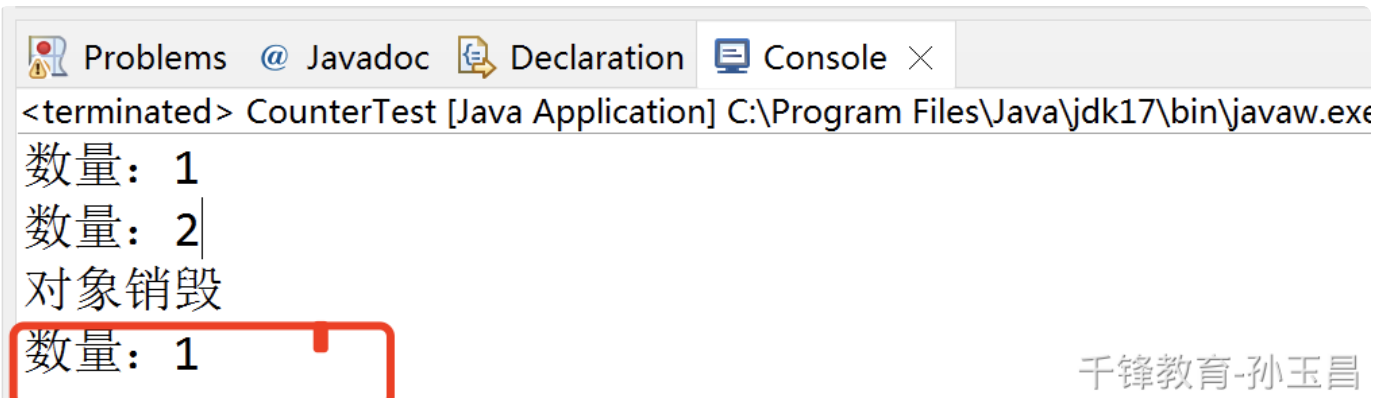
接着我们再创建第2个类，对Counter里的finalize()进行测试。

```

1  /**
2   * @author 一一哥Sun
3   * QQ: 2312119590
4   * CSDN、掘金、知乎找我哦
5   */
6  public class CounterTest {
7
8      public static void main(String[] args) {
9          Counter cnt1 = new Counter(); // 创建第一个实例对象
10         System.out.println("数量: " + cnt1.getCount()); // 输出1
11
12         Counter cnt2 = new Counter(); // 创建第二个实例对象
13         System.out.println("数量: " + cnt2.getCount()); // 输出2
14         cnt2 = null; // 销毁实例2
15
16         try {
17             System.gc(); // 手动调用清理内存的gc()方法
18             Thread.currentThread().sleep(1000); // 延时1000毫秒
19             System.out.println("数量: " + cnt1.getCount()); // 再次输出1, 说
明有个对象被销毁了
20         } catch (InterruptedException e) {
21             e.printStackTrace();
22         }
23     }
24
25 }

```

因为`finalize()`方法具有不确定性，也就是说，该方法到底会不会起作用是不一定的！所以我们在程序中可以调用 `System.gc()`方法或者 `Runtime.gc()`方法，来显式地提醒垃圾回收器尽快去执行垃圾回收操作。最终执行效果如下图所示：



```

Problems  @ Javadoc  Declaration  Console ×
<terminated> CounterTest [Java Application] C:\Program Files\Java\jdk17\bin\javaw.exe
数量: 1
数量: 2
对象销毁
数量: 1

```

四. 结语

至此，壹哥就把析构方法的功能及使用给大家介绍完毕了，现在你学会了吗？对我们这些初学者来说，我们暂时知道finalize析构方法的作用和基本用法即可。等到后面，壹哥会再给大家专门讲解Java的垃圾回收机制，那时候我们再单独分析。如果你单独学习觉得有困难，可以加入壹哥的学习交流群哦。

五. 今日作业

1. 第一题

设计一个类，定义几个属性，在析构方法中回收该类对象，观察属性值的变化。