大家好，很高兴我们可以继续学习交流Java高频面试题。本小节是Java基础篇章的第三小节，主要讲述**Java中的Exception与Error，JIT编译器以及值传递与引用传递**的知识点。熟练掌握本小节的知识点，可以帮助大家更好的复习与掌握Java面试中的基础类题目，对我们的实际开发也会有很大的帮助。

（1）Java中的Exception和Error有什么区别?

答： Exception和Error的主要区别可以概括如下。

- **Exception**是程序正常运行中**预料到可能会出现**的错误，并且应该被捕获并进行相应的处理，是一种**异常**现象
- **Error**是正常情况下不可能发生的错误，**Error会导致JVM处于一种不可恢复的状态**，不需要捕获处理，比如说OutOfMemoryError

解析：

Exception又分为了**运行时异常和编译时异常**。

编译时异常（受检异常）表示当前调用的方法体内部抛出了一个异常，所以编译器检测到这段代码在运行时可能会出异常，所以要求我们必须对异常进行相应的处理，可以捕获异常或者抛给上层调用方。

运行时异常（非受检异常）表示在运行时出现的异常，常见的运行时异常包括：空指针异常，数组越界异常，数字转换异常以及算术异常等。

前边说到了异常Exception应该被捕获，我们可以使用try – catch – finally 来处理异常，并且使得程序恢复正常。

那么我们捕获异常应该遵循哪些原则呢?

- 尽可能捕获比较详细的异常，而不是使用Exception一起捕获。
- 当本模块不知道捕获之后该怎么处理异常时，可以将其抛给上层模块。上层模块拥有更多的业务逻辑，可以进行更好的处理。
- 捕获异常后至少应该有日志记录，方便之后的排查。
- 不要使用一个很大的try – catch包住整段代码，不利于问题的排查。

这些都是笔者血淋淋的教训，捕获到异常，却看不出是哪里抛出的，这才是绝望。然后，我们**再来看一个容易混淆的异常与错误知识点：**

NoClassDefFoundError 和 ClassNotFoundException 有什么区别?

答：从名字中，我们可以看出前者是一个错误，后者是一个异常。我们先来看下JDK中对**ClassNotFoundException**异常的阐述：

```
 * Thrown when an application tries to load in a class through its
 * string name using:
 * <ul>
 * <li>The <code>forName</code> method in class <code>Class</code>.
 * <li>The <code>findSystemClass</code> method in class
 *     <code>ClassLoader</code> .
 * <li>The <code>loadClass</code> method in class <code>ClassLoader</code>.
 * </ul>
 * <p>
 * but no definition for the class with the specified name could be found.
 *
```


大概意思就是在说，当我们使用例如Class.forName方法来动态的加载该类的时候，传入了一个类名，但是其并没有在类路径中被找到的时候，就会报ClassNotFoundException异常。**出现这种情况，一般都是类名字传入有误导致的。**

我们再来看下JDK中对该错误**NoClassDefFoundError**的阐述：

```
]/**
 * Thrown if the Java Virtual Machine or a <code>ClassLoader</code> instance
 * tries to load in the definition of a class (as part of a normal method call
 * or as part of creating a new instance using the <code>new</code> expression)
 * and no definition of the class could be found.
 * <p>
 * The searched-for class definition existed when the currently
 * executing class was compiled, but the definition can no longer be
 * found.
 *
```


大概意思是这样的，如果JVM或者ClassLoader实例尝试加载（可以通过正常的方法调用，也可能是使用new来创建新的对象）类的时候却找不到类的定义。但是**要查找的类在编译的时候是存在的，运行的时候却找不到了**。这个时候就会导致NoClassDefFoundError。出现这种情况，**一般是由于打包的时候漏掉了部分类或者Jar包被篡改已经损坏。**

（2）JIT编译器有了解吗？

答：前面我们谈到了Java是一种先编译，后解释执行的语言。那么我们就来说下何为JIT编译器吧。

JIT编译器全名叫**Just In Time Compile** 也就是即时编译器，把经常运行的代码**作为"热点代码"编译成与本地平台相关的机器码**，并进行各种层次的优化。JIT编译除了具有缓存的功能外，还会对代码做各种优化，包括**逃逸分析**、锁消除、 锁膨胀、方法内联、空值检查消除、类型检测消除以及公共子表达式消除等。

解析：

JIT编译器属于Java基础中的比较有深度的题目了，**回答出来算是一个亮点了**。既然说到了JIT编译器，我们来看下JIT对代码优化使用到的逃逸分析技术吧。

逃逸分析：

逃逸分析的基本行为就是分析对象动态作用域，当一个对象在方法中被定义后，它可能被外部方法所引用，例如作为调用参数传递到其他地方中，称为**方法逃逸**。JIT编译器的优化包括如下：

- **同布省略：**也就是锁消除，当JIT编译器判断不会产生并发问题，那么会将同步synchronized去掉
- **标量替换**

31

我们先来解释下标量和聚合量的基本概念。

- **标量（Scalar）**是指一个无法再分解成更小的数据的数据。Java中的原始数据类型就是标量。
- **聚合量（Aggregate）**是还可以分解的数据。Java中的对象就是聚合量，因为他可以分解成其他聚合量和标量。

在JIT阶段，如果经过逃逸分析，发现一个对象不会被外界访问的话，那么经过JIT优化，就会把这个对象拆解成若干个其中包含的若干个成员变量来代替。这个过程就是标量替换。标量替换的好处就是对象可以不在堆内存进行分配，为栈上分配提供了良好的基础。

那么逃逸分析技术存在哪些缺点呢?

技术不是特别成熟，分析的过程也很耗时，如果没有一个对象是不逃逸的，那么就得不偿失了。

（3）Java中的值传递和引用传递可以解释下吗?

答：值传递和引用传递的解释可以概括如下。

- **值传递，**意味着传递了对象的一个副本，即使副本被改变，也不会影响源对象。
- **引用传递，**意味着传递的并不是实际的对象，而是对象的引用。因此，外部对引用对象的改变会反映到所有的对象上。

解析：

我们先来看一个值传递的例子：

```
public class Test {
    public static void main(String[] args)  {
        int x=0;
        change(x);
        System.out.println(x);
    }
    static void change(int i){
        i=7;
    }
}
```

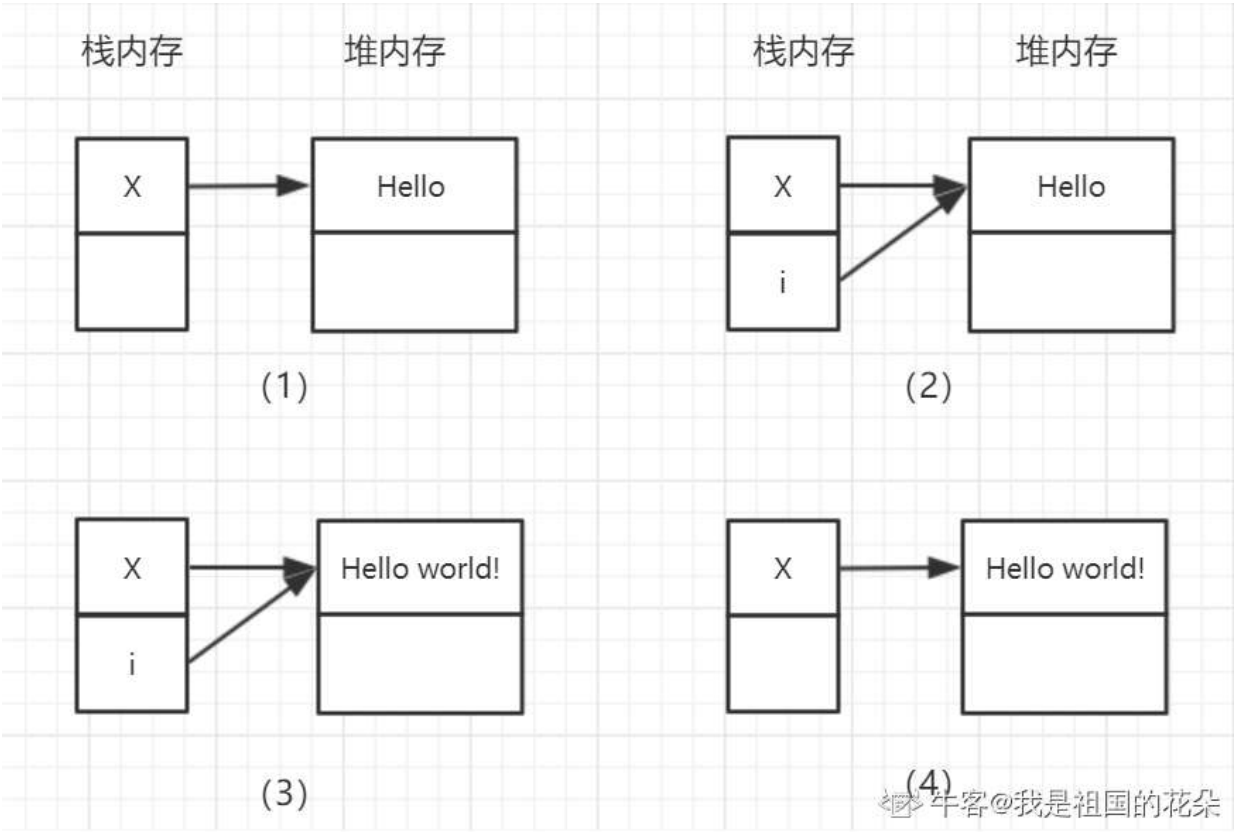
毫无疑问，上边的代码会输出0。因为如果参数是基本数据类型，那么是属于值传递的范畴，传递的其实是源对象的一个copy副本，不会影响源对象的值。

我们再来分析一个引用传递的例子：

```
public class Test {
    public static void main(String[] args)  {
        StringBuffer x = new StringBuffer("Hello");
        change(x);
        System.out.println(x);
    }
```

```
    static void change(StringBuffer i) {
        i.append(" world!");
    }
}
```

通过运行程序，**输出为Hello world！**接下来我们通过图片来分析下程序执行过程种的内存变化吧。

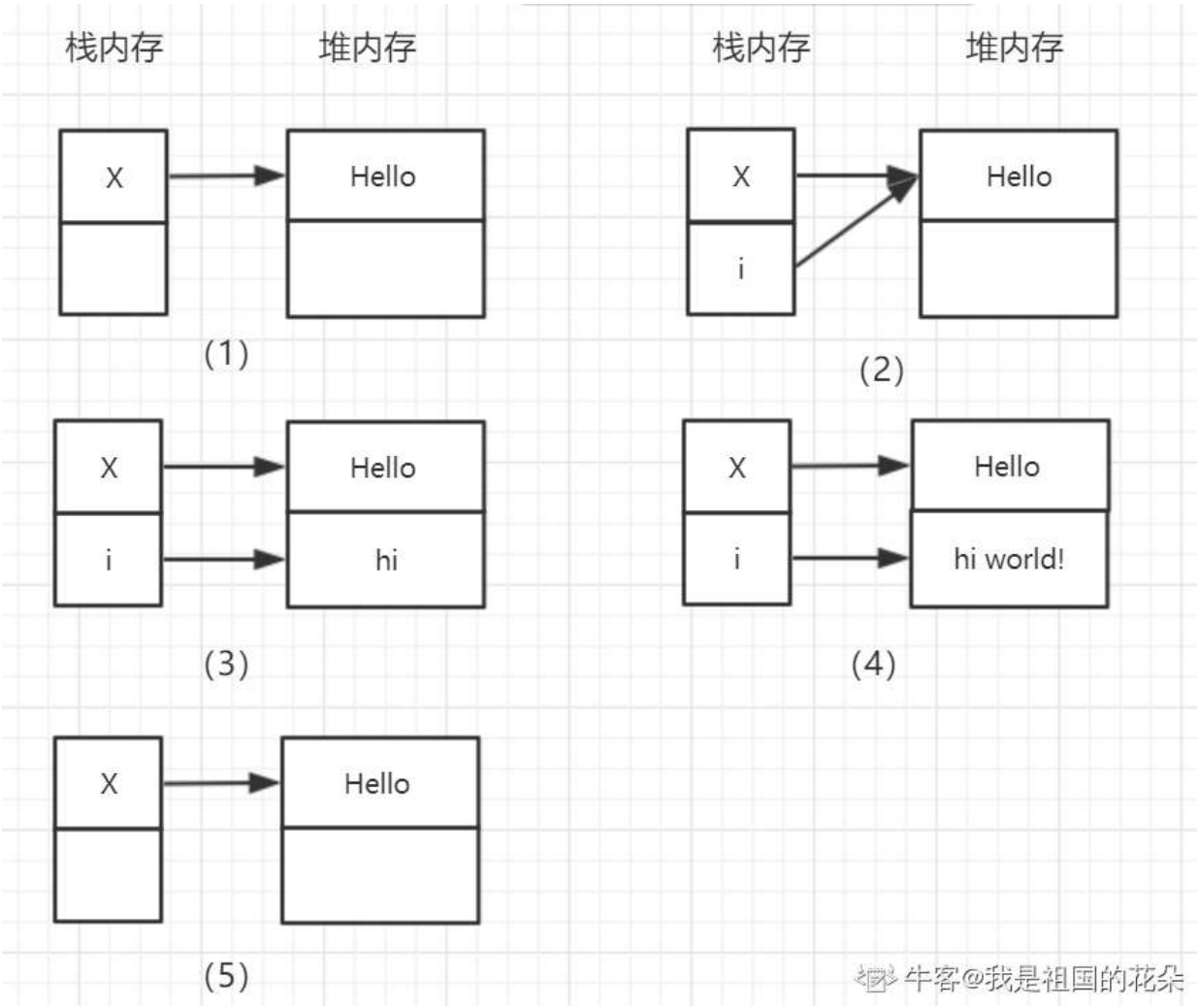


由图中我们可以看出**x和i指向了同样的内存地址**，那么i.append操作将直接修改了内存地址里边的值，所以当方法结束，**局部变量i消失**，先前变量x所指向的内存值已经发生了变化，**所以输出为Hello world!**

接着，我们修改下change方法，代码如下所示：

```
public class Test {
    public static void main(String[] args)  {
        StringBuffer x = new StringBuffer("Hello");
        change2(x);
        System.out.println(x);
    }
    static void change2(StringBuffer i) {
        i = new StringBuffer("hi");
        i.append(" world!");
    }
}
```

先给出答案，上边Demo的输出为Hello，我们依然来画图分析内存变化。



由图中我们可以看出来，**在函数change2中将引用变量i重新指向了堆内存中另一块区域**，下边都是对另一块区域进行修改，**所以输出是Hello。**

最后，我们继续升级该题目代码如下：

```
public class Test {
    public static void main(String[] args)  {
        StringBuffer sb = new StringBuffer("Hello ");
        System.out.println("Before change, sb = " + sb);
        changeData(sb);
        System.out.println("After change, sb = " + sb);
    }
    public static void changeData(StringBuffer strBuf) {
        StringBuffer sb2 = new StringBuffer("Hi, I am ");
        strBuf = sb2;
        sb2.append("World!");
    }
}
```

那么这个题目的输出是什么呢？相信大家都可以给出正确的答案。

请大家将这个题目的答案以及内存分析过程评论在本节文章的下边，我会在下一小节中给出参考答案分析哦~

（4）Java中的其余经典基础面试题目：

限于文章篇幅，我将**《Java基础 - 必知必会》**三个小节中没有讲解到的**常见基础面试题目**罗列在此，大家有想要交流的可以直接在评论区留言。

- StringBuffer与StringBuilder的区别?

- Java中的泛型的理解
- Java序列化与反序列化的过程
- equals和hashCode方法的关系?
- Java和C++的区别有哪些?
- 静态与非静态的区别?
- Java中equals方法和==的区别?

31

总结:

限于篇幅，笔者尽量将最常见的面试题目与大家进行了交流与分享。有的题目可能解析比较简单，仅仅起到了一个抛砖引玉的作用，因为每一个知识点都可能需要阐述一篇文章。从下一节开始，我们开始交流学习**Java基础中的三大集合，这几乎是面试中的必考知识点**，在难度和深度上有所加深，希望我们可以一起进步。

限于作者水平，文章中难免会有不妥之处。大家在学习过程中遇到我没有表达清楚或者表述有误的地方，欢迎随时在文章下边指出，我会及时关注，随时改正。另外，大家有任何话题都可以在下边留言，我们一起交流探讨。

讨论 评论

八辈渔民 1#

看完了，期待您的再次更新

发表于 2019-11-21 19:20:36 赞(0) 回复(1)

我是祖国的花朵 N 作者 ： 正在加急更新中，争取早日更新完毕。（慢主要是因为写完需要等牛客给出反馈意见哈~）

2019-11-23 23:17:04 赞(0) 回复(0)

请输入你的观点

回复

一只没有感情的鸽子 2#

打卡

发表于 2019-11-25 17:10:16 赞(0) 回复(0)

μarine 3#

剩下的应该只有Hello了

发表于 2019-11-26 20:51:16 赞(1) 回复(1)

我是祖国的花朵 N 作者 ： 赞

2019-12-01 23:01:34 赞(0) 回复(0)

请输入你的观点

回复

菜鸡真的不配有offer 4#

打卡

发表于 2019-11-28 08:34:12 赞(0) 回复(0)

31

人见人爱的OFFER收割机 5#

其余经典面试题也可以给下答案吗？谢谢

发表于 2019-11-28 22:47:17 赞(0) 回复(1)

我是祖国的花朵 作者： 你好，这里边的每一个知识点都可以阐述很多，如果我仅仅给出几句话，无法帮助大家真正理解该知识点，所以我建议大家自行学习。当然，如果你在哪个知识点中遇到了困难，欢迎留言，我们一起交流学习。

2019-12-01 23:00:46 赞(0) 回复(0)

请输入你的观点

回复

还年轻多学习 6#

其余经典面试题也可以给下答案吗？谢谢

发表于 2019-12-01 18:48:07 赞(2) 回复(1)

我是祖国的花朵 作者： 你好，这里边的每一个知识点都可以阐述很多，如果我仅仅给出几句话，无法帮助大家真正理解该知识点，所以我建议大家自行学习。当然，如果你在哪个知识点中遇到了困难，欢迎留言，我们一起交流学习。

2019-12-01 23:01:00 赞(0) 回复(0)

请输入你的观点

回复

还年轻多学习 7#

特别是泛型

发表于 2019-12-01 18:48:17 赞(1) 回复(2)

泽阳Alex： 泛型就是参数化类型，在不创建新的数据类型情况下，通过泛型控制具体不同类型的形参。泛型最常用的场景就是在集合中，能够简化开发，并且能够保证代码质量。泛型是在编译期间有效，在运行阶段就会去泛型化，也就是将泛型信息抹掉，这也是不支持泛型数组的原因。

2020-02-28 21:37:23 赞(2) 回复(0)

Windranger： 1. 通过泛型的语法定义，编译器可以在编译期提供一定的类型安全检查，过滤掉大部分因为类型不符而导致的运行时异常。

2. 泛型可以让程序代码的可读性更高，并且由于本身只是一个语法糖，所以对于 JVM 运行时的性能是没有任何影响的。

2020-04-29 10:34:48 赞(0) 回复(0)

请输入你的观点

31

回复

夏洛克201904141035199 8#

所以Java中是只有值传递了

发表于 2019-12-03 20:06:21 赞(0) 回复(0)

sunny_day 9#

可以更新下堆内存，栈内存，常量池的知识点吗

发表于 2019-12-16 00:30:32 赞(0) 回复(2)

我是祖国的花朵 作者 ： 你好，可以将你对这几个内存区域具体的疑惑点说出来，可以一起看下。

2019-12-16 21:05:34 赞(0) 回复(0)

sunny_day 回复 我是祖国的花朵 作者 ： 其实也不是疑惑，就是老是记不住这些，比如说堆内存和栈内存各存放什么，为什么存放这些东西，这样做有原因的还是说跟知识点一样，背下来就行了

2019-12-17 18:37:08 赞(2) 回复(0)

请输入你的观点

回复

我爱吃面条 10#

D:\JAVA\bin\java.exe -javaagent:D:\IDEA2018\ideaIU-2018.3.5.win\lib\idea_rt.jar=63045:D:\IDEA2018\ideaIU-2018.3.5.win\bin -Dfile.encoding=UTF-8 -classpath D:\JAVA\jre\lib\charsets.jar;D:\JAVA\jre\lib\deploy.jar;D:\JAVA\jre\lib\ext\access-bridge-64.jar;D:\JAVA\jre\lib\ext\cldrdata.jar;D:\JAVA\jre\lib\ext\dnsns.jar;D:\JAVA\jre\lib\ext\jaccess.jar;D:\JAVA\jre\lib\ext\jfxrt.jar;D:\JAVA\jre\lib\ext\localedata.jar;D:\JAVA\jre\lib\ext\nashorn.jar;D:\JAVA\jre\lib\ext\sunec.jar;D:\JAVA\jre\lib\ext\sunjce_provider.jar;D:\JAVA\jre\lib\ext\sunmscapi.jar;D:\JAVA\jre\lib\ext\sunpkcs11.jar;D:\JAVA\jre\lib\ext\zipfs.jar;D:\JAVA\jre\lib\javaws.jar;D:\JAVA\jre\lib\jce.jar;D:\JAVA\jre\lib\jfr.jar;D:\JAVA\jre\lib\jfxswt.jar;D:\JAVA\jre\lib\jsse.jar;D:\JAVA\jre\lib\management-agent.jar;D:\JAVA\jre\lib\plugin.jar;D:\JAVA\jre\lib\resources.jar;D:\JAVA\jre\lib\rt.jar;F:\DSA\out\production\DSA com.bjsxt.test09.Test
Before change, sb = Hello
After change, sb = Hello

Process finished with exit code 0

发表于 2019-12-28 09:45:59 赞(0) 回复(0)

Meloddy 11#

1.StringBuffer和StringBuilder的区别主要是一个是线程安全，一个非线程安全；其中StringBuffer属于非线程安全，StringBuilder属于线程安全，并且StringBuffer的运行效率要比StringBuilder的运行效率要高

31

发表于 2020-01-03 18:40:52 赞(0) 回复(5)

我是祖国的花朵 作者 ： 同学，这块知识点你记忆刚好相反。StringBuffer是线程安全的，可以按照safe安全后缀fe来，记忆StringBuffer是线程安全的。

2020-01-03 22:37:38 赞(3) 回复(0)

Meloddy 回复 我是祖国的花朵 作者 ： 嗷，对对对，谢谢谢谢

2020-01-03 22:38:15 赞(0) 回复(0)

我是祖国的花朵 作者 回复 Meloddy ： 加油~

2020-01-03 22:44:09 赞(0) 回复(0)

Deer77 ： 应该是StringBuilder比StringBufer效率要高把?

2020-02-21 12:01:52 赞(0) 回复(0)

sky爱吃青菜 ： 加了buffer（加成）的是线程安全的，游戏里不是有什么蓝buffer 红buffer 吗?

2020-06-20 07:35:14 赞(0) 回复(0)

请输入你的观点

回复

Meloddy 12#

3.将对象转换为字节序列存储到硬盘上的过程叫做对象的序列化；反序列化则是把硬盘上的字节序列读取恢复成一个对象的过程

发表于 2020-01-03 18:46:02 赞(1) 回复(2)

L201911221558754 ： 步骤一：创建一个对象输出流，它可以包装一个其它类型的目标输出流，如文件输出流：

ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("D:\\object.out"));

步骤二：通过对象输出流的writeObject()方法写对象：

oos.writeObject(new User("xuliugen", "123456", "male"));

2020-04-14 15:02:55 赞(0) 回复(0)

L201911221558754 ： 同学，作者不是问过程吗

2020-04-14 15:03:13 赞(0) 回复(0)

请输入你的观点

回复

Meloddy 13#

4.hashcode方法存在主要是提高判断对象是否相等的效率，当对象equals相等时，它们的hashcode一定相等，但是当对象的hashcode相等时，两个对象不一定是相等的。再比较对象是否相等时，先比较通过hashcode进行比较，然后进行equals比较。（仅供参考，欢迎补充）

发表于 2020-01-03 18:54:41 赞(2) 回复(0)

31

Meloddy 14#

5.Java和C++的区别主要在于，Java是一门面向对象的编程语言，C++是一门面向过程的编程语言。举个例子：你现在需要一辆汽车，Java的操作应该是将现成的轮子，发动机，车门，车的框架等各个组件如何组装起来；C++的操作则更像，首先，你需要造一个轮子，然后造车门，然后造发动机等这样的操作流程。Java面向对象编程的思想具有低耦合的性质，但是效率上是不如C++的。还有一点，Java的GC是程序员的福音，不需要程序员手动的清除无用对象。（仅供参考，欢迎补充）

发表于 2020-01-03 19:02:46 赞(0) 回复(6)

皮皮陆： JAVA和C++都是面向对象的吧，C是面向过程的，，

2020-01-07 16:31:22 赞(0) 回复(0)

Meloddy 回复 皮皮陆： 多谢指正😆😆确实是的

2020-01-07 16:33:33 赞(0) 回复(0)

皮皮陆： 加油加油！！一起学习！

2020-01-07 17:29:08 赞(0) 回复(0)

Meloddy 回复 皮皮陆： 加油☺

2020-01-07 17:29:41 赞(0) 回复(0)

冲啊猪猪： 主要是指针和内存自动管理

2020-02-23 15:26:17 赞(0) 回复(0)

sky爱吃青菜： 1.指针
2.垃圾自动回收机制
3.是否可以多继承

2020-06-20 10:53:55 赞(0) 回复(0)

请输入你的观点

回复

Meloddy 15#

6.静态所修饰的在类中是唯一的，static可以修饰成员变量、方法、代码块，从内存角度来看，静态成员变量存放在常量池中，并且是在该类在加载时就会被放入常量池，而非静态成员变量是存放在堆中，局部变量存放在栈中；从运行时期来看，在代码被编译成.class文件时，由static修饰的代码块、成员变量、方***

被放入内存的方法区中，如果在静态方法中调用了非静态的成员变量、局部变量或是非静态的方法，则会出现错误。（仅供参考，欢迎补充）

发表于 2020-01-03 19:16:42 赞(0) 回复(1)

sky爱吃青菜： 静态成员变量就是在方法区中，不是存放在常量池中吧。

2020-06-20 11:10:55 赞(0) 回复(0)

请输入你的观点

回复

31

Meloddy 16#

7.Java中==用于比较Java对象的引用是否相同，equals则是用于比较Java对象的内容是否相同

发表于 2020-01-03 19:23:53 赞(2) 回复(4)

周黑字： 没重写的话equals就是调用的== 重写之后的话才是比较内容

2020-01-08 23:28:33 赞(1) 回复(0)

Meloddy 回复 周黑字： 多谢补充！！！

2020-01-09 10:50:23 赞(0) 回复(0)

if不赶due 回复 周黑字： 兄弟说的对

2020-02-26 12:41:08 赞(0) 回复(0)

牛客215551098号： == 在比较数值的时候比较的是值,==在比较对象的时候比较的是地址值,equals没有重写时和==相同的,重写了就比较内容是否相同

2020-04-15 22:34:03 赞(1) 回复(0)

请输入你的观点

回复

我是祖国的花朵 作者 17#

Meloddy同学的总结很棒，除了第一个StringBuffer和StringBuilder。StringBuffer是线程安全的，可以按照safe后缀fe来记忆哪个是线程安全的。

发表于 2020-01-03 22:43:49 赞(2) 回复(0)

牧水s 18#

打卡

发表于 2020-01-11 15:38:38 赞(0) 回复(0)

ZQ.stu 19#

static, final, finalize这些好像没有提及?
作者可不可以把其他常见的题目列一个目录出来以求全?

发表于 2020-02-15 16:22:22 赞(0) 回复(2)

我是祖国的花朵 作者 ： 你好，static已经在评论区解析。至于final和finalize是两个完全没有任何联系的关键字。 31

2020-02-15 16:25:22 赞(0) 回复(0)

L201911221558754 ： final:可以用来修饰类、方法、成员变量。修饰类表示该类不能被继承，修饰方法表示不能被重写，修饰成员变量表示成员变量的值不能被修改。

finally:一般配合try catch使用，用来保证程序不管是正常执行或者是抛出异常或者中途return,都会执行finally中的代码。

finalized:这是配合垃圾回收(简称GC)，垃圾回收之前都会执行该方法。

2020-04-14 15:44:53 赞(0) 回复(0)

请输入你的观点

回复

小源20190118151956 20#

打卡

发表于 2020-02-16 12:42:29 赞(0) 回复(0)

首页 上一页 1 2 下一页 尾页