

2018最新《BAT Java必考面试题集》

2018年10月31日 15:23:52

bug发现与制造

阅读数 233

标签:

2018最新《BAT Java必考面试题集》

Java面试题

更多

版权声明： <https://blog.csdn.net/KamRoseLee/article/details/83584803>

2018最新《BAT Java必考面试题集》

1、面向对象的特征有哪些方面？

答：面向对象的特征主要有以下几个方面：

1) 抽象：抽象是将一类对象的共同特征总结出来构造类的过程，包括数据抽象和行为抽象两方面。抽象只关注对象有哪些属性和行为，并不关注这些行为。

2)继承：继承是从已有类得到继承信息创建新类的过程。提供继承信息的类被称为父类（超类、基类）；得到继承信息的类被称为子类（派生类）。继软件系统有了一定的延续性，同时继承也是封装程序中可变因素的重要手段（如果不能理解请阅读阎宏博士的《Java与模式》或《设计模式精解》中相关的部分）。

3)封装：通常认为封装是把数据和操作数据的方法绑定起来，对数据的访问只能通过已定义的接口。面向对象的本质就是将现实世界描绘成一系列完全对象。我们在类中编写的方法就是对实现细节的一种封装；我们编写一个类就是对数据和数据操作的封装。可以说，封装就是隐藏一切可隐藏的东西，最简单的编程接口（可以想想普通洗衣机和全自动洗衣机的差别，明显全自动洗衣机封装更好因此操作起来更简单；我们现在使用的智能手机也是封装因为几个按键就搞定了所有的事情）。

4)多态性：多态性是指允许不同子类型的对象对同一消息作出不同的响应。简单的说就是用同样的对象引用调用同样的方法但是做了不同的事情。多态的多态性和运行时的多态性。如果将对象的方法视为对象向外界提供的服务，那么运行时的多态性可以解释为：当A系统访问B系统提供的服务时，B系统服务的方式，但一切对A系统来说都是透明的（就像电动剃须刀是A系统，它的供电系统是B系统，B系统可以使用电池供电或者用交流电，甚至还有可A系统只会通过B类对象调用供电的方法，但并不知道供电系统的底层实现是什么，究竟通过何种方式获得了动力）。方法重载（overload）实现的是静态性（也称为前绑定），而方法重写（override）实现的是运行时的多态性（也称为后绑定）。运行时的多态是面向对象最精髓的东西，要实现多态需要方法重写（子类继承父类并重写父类中已有的或抽象的方法）；2. 对象造型（用父类型引用引用子类型对象，这样同样的引用调用同样的方法就会根据不同而表现出不同的行为）。

2、访问修饰符public,private,protected,以及不写（默认）时的区别？

答：区别如下：

作用域	当前类	同包	子类	其他
private	是	否	否	否
default	是	是	否	否
protected	是	是	是	否
public	是	是	是	是

public	✓	✓	✓	✓
--------	---	---	---	---

protected	✓	✓	✓	×
-----------	---	---	---	---

default	✓	✓	×	×
---------	---	---	---	---

private	✓	x	x	x
---------	---	---	---	---

类的成员不写访问修饰时默认为default。默认对于同一个包中的其他类相当于公开（public），对于不是同一个包中的其他类相当于私有（private），（protected）对子类相当于公开，对不是同一包中的没有父子关系的类相当于私有。

3、String 是最基本的数据类型吗?

答：不是。Java中的基本数据类型只有8个：byte、short、int、long、float、double、char、boolean；除了基本类型（primitive type）和枚举类（enumeration type），剩下的都是引用类型（reference type）。

4、float f=3.4;是否正确?

答:不正确。3.4是双精度数,将双精度型(double)赋值给浮点型(float)属于下转型(down-casting,也称为窄化)会造成精度损失,因此float f=(float)3.4;或者写成**float f=3.4F**。

5、short s1 = 1; s1 = s1 + 1;有错吗?short s1 = 1; s1 += 1;有错吗?

答：对于short s1 = 1; s1 = s1 + 1; 由于1是int类型，因此s1+1运算结果也是int型，需要强制转换类型才能赋值给short型。而short s1 = 1; s1 = s1 + 1; 相当于s1 = (short)(s1 + 1); 其中有隐含的强制类型转换。

6、Java 有没有goto?

答: goto 是Java中的保留字, 在目前版本的Java中没有使用。(根据James Gosling (Java之父) 编写的《The Java Programming Language》——出了一个Java关键字列表, 其中有goto和const, 但是这两个是目前无法使用的关键字, 因此有些地方将其称之为保留字, 其实保留字这个词应该有更因为熟悉C语言的程序员都知道, 在系统类库中使用过的有特殊意义的单词或单词的组合都被视为保留字)

7、int 和Integer 有什么区别?

答: Java是一个近乎纯洁的面向对象编程语言, 但是为了编程的方便还是引入不是对象的基本数据类型, 但是为了能够将这些基本数据类型……对象每一个基本数据类型都引入了对应的包装类型 (wrapper class), int的包装类就是Integer, 从JDK 1.5开始引入了自动装箱/拆箱机制, 用户可以

Java 为每个原始类型提供了包装类型:

原始类型: boolean, char, byte, short, int, long, float, double

包装类型: Boolean, Character, Byte, Short, Integer, Long, Float, Double

```
1 package com.lovo;
2
3 public class AutoUnboxingTest {
4
5     public static void main(String[] args) {
6         Integer a = new Integer(3);
7         Integer b = 3;           // 将3自动装箱成Integer类型
8         int c = 3;
9         System.out.println(a == b);    // false 两个引用没有引用同一对象
10        System.out.println(a == c);    // true a自动拆箱成int类型再和c比较
11    }
12 }
```

<https://blog.csdn.net/KamRoseLee>

补充: 最近还遇到一个面试题, 也是和自动装箱和拆箱相关的, 代码如下所示:

```
1 public class Test03 {
2
3     public static void main(String[] args) {
4         Integer f1 = 100, f2 = 100, f3 = 150, f4 = 150;
5
6         System.out.println(f1 == f2);
7         System.out.println(f3 == f4);
8     }
9 }
```

<https://blog.csdn.net/KamRoseLee>

如果不明就里很容易认为两个输出要么都是true要么都是false。首先需要注意的是f1、f2、f3、f4四个变量都是Integer对象, 所以下面的==运算比较引用。装箱的本质是什么呢? 当我们给一个Integer对象赋一个int值的时候, 会调用Integer类的静态方法valueOf, 如果看看valueOf的源代码就知道:

```
1 public static Integer valueOf(int i) {
2     if (i >= IntegerCache.low && i <= IntegerCache.high)
3         return IntegerCache.cache[i + (-IntegerCache.low)];
4     return new Integer(i);
5 }
```

IntegerCache是Integer的内部类, 其代码如下所示:



```

12     private static class IntegerCache {
13         static final int low = -128;
14         static final int high;
15         static final Integer cache[];
16
17         static {
18             // high value may be configured by property
19             int h = 127;
20             String integerCacheHighPropValue =
21                 sun.misc.VM.getSavedProperty("java.lang.Integer.IntegerCache.high");
22             if (integerCacheHighPropValue != null) {
23                 try {
24                     int i = parseInt(integerCacheHighPropValue);
25                     i = Math.max(i, 127);
26                     // Maximum array size is Integer.MAX_VALUE
27                     h = Math.min(i, Integer.MAX_VALUE - (-low) -1);
28                 } catch( NumberFormatException nfe) {
29                     // If the property cannot be parsed into an int, ignore it.
30                 }
31             }
32             high = h;
33
34             cache = new Integer[(high - low) + 1];
35             int j = low;
36             for(int k = 0; k < cache.length; k++)
37                 cache[k] = new Integer(j++);
38
39             // range [-128, 127] must be interned (JLS7 5.1.7)
40             assert IntegerCache.high >= 127;
41         }
42
43         private IntegerCache() {}
44     }

```

<https://blog.csdn.net/KamRoseLee>

简单的说，如果字面量的值在-128到127之间，那么不会new新的Integer对象，而是直接引用常量池中的Integer对象，所以上面的面试题中f1==f2的true，而f3==f4的结果是false。越是貌似简单的面试题其中的玄机就越多，需要面试者有相当深厚的功力。

8、&和&&的区别？

答：&运算符有两种用法：(1)按位与；(2)逻辑与。&&运算符是短路与运算。逻辑与跟短路与的差别是非常巨大的，虽然二者都要求运算符左右两端的true整个表达式的值才是true。&&之所以称为短路运算是因为，如果&&左边的表达式的值是false，右边的表达式会被直接短路掉，不会进行运算。可能都需要用&&而不是&，例如在验证用户登录时判定用户名不是null而且不是空字符串，应当写为：username != null && !username.equals("")。顺序不能交换，更不能用&运算符，因为第一个条件如果不成立，根本不能进行字符串的equals比较，否则会产生NullPointerException异常。注意：逻辑与(&)和短路或运算符(||)的差别也是如此。

补充：如果你熟悉JavaScript，那你可能更能感受到短路运算的强大，想成为JavaScript的高手就先从玩转短路运算开始吧。

9、解释内存中的栈 (stack)、堆(heap)和静态存储区的用法。

答：通常我们定义一个基本数据类型的变量，一个对象的引用，还有就是函数调用的现场保存都使用内存中的栈空间；而通过new关键字和构造器创建空间；程序中的字面量 (literal) 如直接书写的100、“hello”和常量都是放在静态存储区中。栈空间操作最快但是也很小，通常大量的对象都是放在内存包括硬盘上的虚拟内存都可以被当成堆空间来使用。

```
String str = new String( "hello" );
```

上面的语句中str放在栈上，用new创建出来的字符串对象放在堆上，而“hello”这个字面量放在静态存储区。

补充：较新版本的Java中使用了一项叫“逃逸分析”的技术，可以将一些局部对象放在栈上以提升对象的操作性能。

10、Math.round(11.5) 等于多少？Math.round(-11.5)等于多少？

答：Math.round(11.5)的返回值是12，Math.round(-11.5)的返回值是-11。四舍五入的原理是在参数上加0.5然后进行下取整。

11、switch 是否能作用在byte 上，是否能作用在long 上，是否能作用在String上？

答：早期的JDK中，switch (expr) 中，expr可以是byte、short、char、int。从1.5版开始，Java中引入了枚举类型（enum），expr也可以是枚举，开始，还可以是字符串（String）。长整型（long）是不可以的。

12、用最有效率的方法计算2乘以8？

答：2 << 3（左移3位相当于乘以2的3次方，右移3位相当于除以2的3次方）。

补充：我们为编写的类重写hashCode方法时，可能会看到如下所示的代码，其实我们不太理解为什么要使用这样的乘法运算来产生哈希码。这个数是个素数，为什么通常选择31这个数？前两个问题的答案你可以自己百度一下，选择31是因为可以用移位和减法运算来代替乘法，得到更高效的结果。到这里你可能已经想到了：31 * num <==> (num << 5) - num，左移5位相当于乘以2的5次方（32）再减去自身就相当于乘以31。现在都能优化。

```
1 package com.loonstudio;
2
3 public class PhoneNumber {
4     private int areaCode;
5     private String prefix;
6     private String lineNumber;
7
8     @Override
9     public int hashCode() {
10         final int prime = 31;
11         int result = 1;
12         result = prime * result + areaCode;
13         result = prime * result
14             + ((lineNumber == null) ? 0 : lineNumber.hashCode());
15         result = prime * result + ((prefix == null) ? 0 : prefix.hashCode());
16         return result;
17     }
18 }
```

https://blog.csdn.net/KamRoseLee

```
19     @Override
20     public boolean equals(Object obj) {
21         if (this == obj)
22             return true;
23         if (obj == null)
24             return false;
25         if (getClass() != obj.getClass())
26             return false;
27         PhoneNumber other = (PhoneNumber) obj;
28         if (areaCode != other.areaCode)
29             return false;
30         if (lineNumber == null) {
31             if (other.lineNumber != null)
32                 return false;
33         } else if (!lineNumber.equals(other.lineNumber))
34             return false;
35         if (prefix == null) {
36             if (other.prefix != null)
37                 return false;
38         } else if (!prefix.equals(other.prefix))
39             return false;
40         return true;
41     }
42 }
43 }
```

https://blog.csdn.net/KamRoseLee

13、数组有没有length()方法?String 有没有length()方法?

答：数组没有length()方法，有length 的属性。String 有length()方法。JavaScript中，获得字符串的长度是通过length属性得到的，这一点容

14、在Java 中，如何跳出当前的多重嵌套循环？

答：在最外层循环前加一个标记如A，然后用break A;可以跳出多重循环。（Java中支持带标签的break和continue语句，作用有点类似于C和C++中的但是就像要避免使用goto一样，应该避免使用带标签的break和continue，因为它不会让你的程序变得更优雅，很多时候甚至有相反的作用，所以这种知道更好）

15、构造器 (constructor) 是否可被重写 (override) ？

答：构造器不能被继承，因此不能被重写，但可以被重载。

16、两个对象值相同(x.equals(y) == true)，但却可有不同的hash code，这句话对不对？

答：不对，如果两个对象x和y满足x.equals(y) == true，它们的哈希码 (hash code) 应当相同。Java对于equals方法和hashCode方法是规定的一个对象相同 (equals方法返回true)，那么它们的hashCode值一定要相同；(2)如果两个对象的hashCode相同，它们并不一定相同。当然，hashCode值必须做，但是如果你违背了上述原则就会发现在使用容器时，相同的对象可以出现在Set集合中，同时增加新元素的效率会大大下降（对于使用HashMap的哈希码频繁的冲突将会造成存取性能急剧下降）。

补充：关于equals和hashCode方法，很多Java程序都知道，但很多人也就是仅仅知道而已，在Joshua Bloch的大作《Effective Java》（很多软件公司《Effective Java》、《Java编程思想》以及《重构：改善既有代码质量》是Java程序员必看书籍）中是这样介绍equals方法的：首先equals方法必须（x.equals(x)必须返回true）、对称性（x.equals(y)返回true时，y.equals(x)也必须返回true）、传递性（x.equals(y)和y.equals(z)都返回true时，x必须返回true）和一致性（当x和y引用的对象信息没有被修改时，多次调用x.equals(y)应该得到同样的返回值），而且对于任何非null值的引用x，x.equals返回false。实现高质量的equals方法的诀窍包括：1. 使用==操作符检查“参数是否为这个对象的引用”；2. 使用instanceof操作符检查“参数是否为型”；3. 对于类中的关键属性，检查参数传入对象的属性是否与之相匹配；4. 编写完equals方法后，问自己它是否满足对称性、传递性、一致性；5. 总是重写hashCode；6. 不要将equals方法参数中的Object对象替换为其他的类型，在重写时不要忘掉@Override注解。

17、是否可以继承String 类？

答：String 类是final类，不可以被继承。

补充：继承String本身就是一个错误的行为，对String类型最好的重用方式是关联（HAS-A）而不是继承（IS-A）。

18、当一个对象被当作参数传递到一个方法后，此方法可改变这个对象的属性，并可返回变化后的结果，那么这里到底是值传递还是引用传递？

答：是值传递。Java 编程语言只有值传递参数。当一个对象实例作为一个参数被传递到方法中时，参数的值就是对该对象的引用。对象的属性可以在被改变，但对对象的引用是永远不会改变的。C++和C#中可以通过传引用或传输出参数来改变传入的参数值。

补充：Java中没有传引用实在是非常的不方便，这一点在Java 8中仍然没有得到改进，正是如此在Java编写的代码中才会出现大量的Wrapper类（将调用修改的引用置于一个Wrapper类中，再将Wrapper对象传入方法），这样的做法只会让代码变得臃肿，尤其是让从C和C++转型为Java程序员的开忍。

19、String 和StringBuilder、StringBuffer 的区别？

答：Java 平台提供了两种类型的字符串：String和StringBuilder / StringBuffer，它们可以储存和操作字符串。其中String是只读字符串，也就意味着字符串内容是不能被改变的。而StringBuffer和StringBuilder类表示的字符串对象可以直接进行修改。StringBuilder是JDK 1.5中引入的，它和String完全相同，区别在于它是在单线程环境下使用的，因为它的所有方面都没有被synchronized修饰，因此它的效率也比StringBuffer略高。

补充1：有一个面试题问：有没有哪种情况用+做字符串连接比调用StringBuffer / StringBuilder对象的append方法性能更好？如果连接后得到的字符串中是早已存在的，那么用+做字符串连接是优于StringBuffer / StringBuilder的append方法的。

补充2：下面也是一个面试题，问程序的输出，看看自己能不能说出正确答案。



```

1 package com.lovo;
2
3 public class StringEqualTest {
4
5     public static void main(String[] args) {
6         String a = "Programming";
7         String b = new String("Programming");
8         String c = "Program" + "ming";
9
10        System.out.println(a == b);
11        System.out.println(a == c);
12        System.out.println(a.equals(b));
13        System.out.println(a.equals(c));
14        System.out.println(a.intern() == b.intern());
15    }
16 }

```

<https://blog.csdn.net/KamRoseLee>



20、重载 (Overload) 和重

(Override) 的区别。重载的方法能否根据返回类型进行区分？

答：方法的重载和重写都是实现多态的方式，区别在于前者实现的是编译时的多态性，而后者实现的是运行时的多态性。重载发生在一个类中，同名的参数列表（参数类型不同、参数个数不同或者二者都不同）则视为重载；重写发生在子类与父类之间，重写要求子类被重写方法与父类被重写方法类型，比父类被重写方法更好访问，不能比父类被重写方法声明更多的异常（里氏代换原则）。重载对返回类型没有特殊的要求。

补充：华为的面试题中曾经问过这样一个问题：为什么不能根据返回类型来区分重载，说出你的答案吧！😏

21、描述一下JVM 加载class文件的原理机制？

答：JVM 中类的装载是由类加载器（ClassLoader）和它的子类来实现的，Java中的类加载器是一个重要的Java 运行时系统组件，它负责在运行时查件中的类。

补充：

1.由于Java的跨平台性，经过编译的Java源程序并不是一个可执行程序，而是一个或多个类文件。当Java程序需要使用某个类时，JVM会确保这个类已接(验证、准备和解析)和初始化。类的加载是指把类的.class文件中的数据读入到内存中，通常是创建一个字节数组读入.class文件，然后产生与所加载Class对象。加载完成后，Class对象还不完整，所以此时的类还不可用。当类被加载后就进入连接阶段，这一阶段包括验证、准备(为静态变量分配内存的初始值)和解析(将符号引用替换为直接引用)三个步骤。最后JVM对类进行初始化，包括：1如果类存在直接的父类并且这个类还没有被初始化，那么类；2如果类中存在初始化语句，就依次执行这些初始化语句。

2.类的加载是由类加载器完成的，类加载器包括：根加载器（BootStrap）、扩展加载器（Extension）、系统加载器（System）和用户自定义类加载器（java.lang.ClassLoader的子类）。从JDK 1.2开始，类加载过程采取了父亲委托机制(PDM)。PDM更好的保证了Java平台的安全性，在该机制中，JVM Bootstrap是根加载器，其他的加载器都有且仅有一个父类加载器。类的加载首先请求父类加载器加载，父类加载器无能为力时才由其子类加载器自行。会向Java程序提供对Bootstrap的引用。下面是关于几个类加载器的说明：

a)Bootstrap：一般用本地代码实现，负责加载JVM基础核心类库（rt.jar）；

b)Extension：从java.ext.dirs系统属性所指定的目录中加载类库，它的父加载器是Bootstrap；

c)System：又叫应用类加载器，其父类是Extension。它是应用最广泛的类加载器。它从环境变量classpath或者系统属性java.class.path所指定的目录是用户自定义加载器的默认父加载器。

22、char 型变量中能不能存贮一个中文汉字?为什么?

答：char类型可以存储一个中文汉字，因为Java中使用的编码是Unicode（不选择任何特定的编码，直接使用字符在字符集中的编号，这是统一的唯一一个char类型占2个字节（16bit），所以放一个中文是没问题的。

补充：使用Unicode意味着字符在JVM内部和外部有不同的表现形式，在JVM内部都是Unicode，当这个字符被从JVM内部转移到外部时（例如存入文件中），需要进行编码转换。所以Java中有字节流和字符流，以及在字符流和字节流之间进行转换的转换流，如InputStreamReader和OutputStreamWriter。这两个类是字节流和字符流之间的适配器类，承担了编码转换的任务；对于C程序员来说，要完成这样的编码转换恐怕要依赖于union（联合体/共用体）来实现了。

23、抽象类 (abstract class) 和接口 (interface) 有什么异同?

答：抽象类和接口都不能够实例化，但可以定义抽象类和接口类型的引用。一个类如果继承了某个抽象类或者实现了某个接口都需要对其中的抽象方法实现，否则该类仍然需要被声明为抽象类。接口比抽象类更加抽象，因为抽象类中可以定义构造器，可以有抽象方法和具体方法，而接口中不能定义构造



方法全部都是抽象方法。抽象类中的成员可以是private、默认、protected、public的，而接口中的成员全都是public的。抽象类中可以定义成员变量，成员变量实际上都是常量。有抽象方法的类必须被声明为抽象类，而抽象类未必要有抽象方法。

24、静态嵌套类(Static Nested Class)和内部类 (Inner Class) 的不同？

答：Static Nested Class是被声明为静态（static）的内部类，它可以不依赖于外部类实例被实例化。而通常的内部类需要在外部类实例化起来挺诡异的，如下所示。

```
1 package com.lovo;
2
3 /**
4  * 扑克类（一副扑克）
5  * @author 骆昊
6  *
7  */
8 public class Poker {
9     private static String[] suites = {"黑桃", "红桃", "草花", "方块"};
10    private static int[] faces = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13};
11
12    private Card[] cards;
13
14    /**
15     * 构造器
16     *
17     */
18    public Poker() {
19        cards = new Card[52];
20        for(int i = 0; i < suites.length; i++) {
21            for(int j = 0; j < faces.length; j++) {
22                cards[i * 13 + j] = new Card(suites[i], faces[j]);
23            }
24        }
25    }
26
27    /**
28     * 洗牌（随机乱序）
29     *
30     */
31 }
```

<https://blog.csdn.net/KamRoseLee>



4



2 实例



```

31     public void shuffle() {
32         for(int i = 0, len = cards.length; i < len; i++) {
33             int index = (int) (Math.random() * len);
34             Card temp = cards[index];
35             cards[index] = cards[i];
36             cards[i] = temp;
37         }
38     }
39
40     /**
41      * 发牌
42      * @param index 发牌的位置
43      *
44      */
45     public Card deal(int index) {
46         return cards[index];
47     }
48
49     /**
50      * 卡片类 (一张扑克)
51      * [内部类]
52      * @author 骆昊
53      *
54      */
55     public class Card {
56         private String suite;    // 花色
57         private int face;        // 点数
58
59         public Card(String suite, int face) {
60             this.suite = suite;
61             this.face = face;
62         }
63

```

<https://blog.csdn.net/KamRoseLee>

```

64         @Override
65         public String toString() {
66             String faceStr = "";
67             switch(face) {
68                 case 1: faceStr = "A"; break;
69                 case 11: faceStr = "J"; break;
70                 case 12: faceStr = "Q"; break;
71                 case 13: faceStr = "K"; break;
72                 default: faceStr = String.valueOf(face);
73             }
74             return suite + faceStr;
75         }
76     }
77 }

```

<https://blog.csdn.net/KamRoseLee>



4

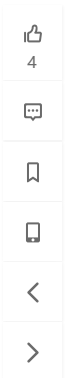



```

1 package com.lovo;
2
3 class PokerTest {
4
5     public static void main(String[] args) {
6         Poker poker = new Poker();
7         poker.shuffle(); // 洗牌
8         Poker.Card c1 = poker.deal(0); // 发第一张牌
9         // 对于非静态内部类Card
10        // 只有通过其外部类Poker对象才能创建Card对象
11        Poker.Card c2 = poker.new Card("红心", 1); // 自己创建一张牌
12
13        System.out.println(c1); // 洗牌后的第一张
14        System.out.println(c2); // 打印: 红心A
15    }
16 }

```

https://blog.csdn.net/KamRoseLee



25、Java 中会存在内存泄漏吗，请简单描述。

答：理论上Java因为有垃圾回收机制（GC）不会存在内存泄露问题（这也是Java被广泛使用于服务器端编程的一个重要原因）；然而在实际开发中，引用但可达的对象，这些对象不能被GC回收也会发生内存泄露。一个例子就是Hibernate的Session（一级缓存）中的对象属于持久态，垃圾回收器是不管的，然而这些对象中可能存在无用的垃圾对象。下面的例子也展示了Java中发生内存泄露的情况：

```

1 package com.lovo;
2
3 import java.util.Arrays;
4 import java.util.EmptyStackException;
5
6 public class MyStack<T> {
7     private T[] elements;
8     private int size = 0;
9
10    private static final int INIT_CAPACITY = 16;
11
12    public MyStack() {
13        elements = (T[]) new Object[INIT_CAPACITY];
14    }
15
16    public void push(T elem) {
17        ensureCapacity();
18        elements[size++] = elem;
19    }
20
21    public T pop() {
22        if(size == 0)
23            throw new EmptyStackException();
24        return elements[--size];
25    }
26
27    private void ensureCapacity() {
28        if(elements.length == size) {
29            elements = Arrays.copyOf(elements, 2 * size + 1);
30        }
31    }
32 }

```

https://blog.csdn.net/KamRoseLee

上面的代码实现了一个栈（先进后出（FILO））结构，乍看之下似乎没有什么明显的问题，它甚至可以通过你编写的各种单元测试。然而其中存在内存泄露的问题，当我们用pop方法弹出栈中的对象时，该对象不会被当作垃圾回收，即使使用栈的程序不再引用这些对象，因为栈内部维护着对对象的引用（obsolete reference）。在支持垃圾回收的语言中，内存泄露是很隐蔽的，这种内存泄露其实就是无意识的对象保持。如果一个对象引用被保持，那么垃圾回收器不会处理这个对象，也不会处理该对象引用的其他对象，即使这样的对象只有少数几个，也可能导致很多的对象被排除在垃圾回收之外，造成重大影响，极端情况下会引发Disk Paging（物理内存与硬盘的虚拟内存交换数据），甚至造成OutOfMemoryError。



26、抽象的（abstract）方法是否可同时是静态的（static），是否可同时是本地方法（native），是否可同时被synchronized修饰？

答：都不能。抽象方法需要子类重写，而静态的方法是无法被重写的，因此二者是矛盾的。本地方法是由本地代码（如C代码）实现的方法，而抽象方法的，也是矛盾的。synchronized和方法的实现细节有关，抽象方法不涉及实现细节，因此也是相互矛盾的。

27、静态变量和实例变量的区别？

答：静态变量是被static修饰符修饰的变量，也称为类变量，它属于类，不属于类的任何一个对象，一个类不管创建多少个对象，静态变量只存在于一个字节；实例变量必须依存于某一实例，需要先创建对象然后通过对象才能访问到它。静态变量可以实现让多个对象共享内存。在Java开发中，静态变量和静态方法会有大量的静态成员。

28、是否可以从一个静态（static）方法内部发出对非静态（non-static）方法的调用？

答：不可以，静态方法只能访问静态成员，因为非静态方法的调用要先创建对象，因此在调用静态方法时可能对象并没有被初始化。

29、如何实现对象克隆？

答：有两种方式：

- 1.实现Cloneable接口并重写Object类中的clone()方法；
- 2.实现Serializable接口，通过对象的序列化和反序列化实现克隆，可以实现真正的深度克隆，代码如下。

```
1 package com.lovo;
2
3 import java.io.ByteArrayInputStream;
4 import java.io.ByteArrayOutputStream;
5 import java.io.ObjectInputStream;
6 import java.io.ObjectOutputStream;
7
8 public class MyUtil {
9
10     private MyUtil() {
11         throw new AssertionError();
12     }
13
14     public static <T> T clone(T obj) throws Exception {
15         ByteArrayOutputStream bout = new ByteArrayOutputStream();
16         ObjectOutputStream oos = new ObjectOutputStream(bout);
17         oos.writeObject(obj);
18
19         ByteArrayInputStream bin = new ByteArrayInputStream(bout.toByteArray());
20         ObjectInputStream ois = new ObjectInputStream(bin);
21         return (T) ois.readObject();
22
23         // 说明：调用ByteArrayInputStream或ByteArrayOutputStream对象的close方法没有任何意义
24         // 这两个基于内存的流只要垃圾回收器清理对象就能够释放资源
25     }
26 }
```

https://blog.csdn.net/KamRoseLee

注意：基于序列化和反序列化实现的克隆不仅仅是深度克隆，更重要的是通过泛型限定，可以检查出要克隆的对象是否支持序列化，这项检查是编译器在运行时抛出异常，这种方案明显优于使用Object类的clone方法克隆对象。

30、GC 是什么？为什么要有GC？

答：GC是垃圾收集的意思，内存处理是编程人员容易出现问题的地方，忘记或者错误的内存回收会导致程序或系统的不稳定甚至崩溃，Java提供的GC监测对象是否超过作用域从而达到自动回收内存的目的，Java语言没有提供释放已分配内存的显示操作方法。Java程序员不用担心内存管理，因为垃圾回收器会自动进行管理。要请求垃圾收集，可以调用下面的方法之一：System.gc() 或Runtime.getRuntime().gc()，但JVM可以屏蔽掉显示的垃圾回收调用。

垃圾回收可以有效的防止内存泄露，有效的使用可以使用的内存。垃圾回收器通常是作为一个单独的低优先级的线程运行，不可预知的情况下或者长时间没有使用的对象进行清除和回收，程序员不能实时的调用垃圾回收器对某个对象或所有对象进行垃圾回收。在Java诞生初期，垃圾回收是它的一个缺点之一，因为服务器端的编程需要有效的防止内存泄露问题，然而时过境迁，如今Java的垃圾回收机制已经成为被诟病的东西。移动智能终端用户对垃圾回收的要求比Android系统有更好的用户体验，其中一个深层次的原因就在于Android系统中垃圾回收的不可预知性。

补充：垃圾回收机制有很多种，包括：分代复制垃圾回收、标记垃圾回收、增量垃圾回收等方式。标准的Java进程既有栈又有堆。栈保存了原始数据，堆保存了要创建的对象。Java平台对堆内存回收和再利用的基本算法被称为标记和清除，但是Java对其进行了改进，采用“分代式垃圾收集”。这种垃圾回收的生命周期将堆内存划分为不同的区域，在垃圾收集过程中，可能会将对象移动到不同区域：

- 伊甸园（Eden）：这是对象最初诞生的区域，并且对大多数对象来说，这里是它们唯一存在过的区域。

- 幸存者乐园 (Survivor)：从伊甸园幸存下来的对象会被挪到这里。
- 终身颐养园 (Tenured)：这是足够老的幸存对象的归宿。年轻代收集 (Minor-GC) 过程是不会触及这个地方的。当年轻代收集不能把对象放进时，就会触发一次完全收集 (Major-GC)，这里可能还会牵扯到压缩，以便为大对象腾出足够的空间。

与垃圾回收相关的JVM参数：

- -Xms / -Xmx --- 堆的初始大小 / 堆的最大大小
- -Xmn --- 堆中年轻代的大小
- -XX:-DisableExplicitGC --- 让System.gc()不产生任何作用
- -XX:+PrintGCDetail --- 打印GC的细节
- -XX:+PrintGCDateStamps --- 打印GC操作的时间戳

31、String s=new String(“xyz”);创建了几个字符串对象？

答：两个对象，一个是静态存储区的"xyz",一个是用new创建在堆上的对象。

32、接口是否可继承 (extends) 接口? 抽象类是否可实现 (implements) 接口? 抽象类是否可继承具体类 (concrete class) ？

答：接口可以继承接口。抽象类可以实现(implements)接口，抽象类可继承具体类，但前提是具体类必须有明确的构造函数。

33、一个“.java”源文件中是否可以包含多个类（不是内部类）？有什么限制？

答：可以，但一个源文件中最多只能有一个公开类（public class）而且文件名必须和公开类的类名完全保持一致。

34、Anonymous Inner Class(匿名内部类)是否可以继承其它类？是否可以实现接口？

答：可以继承其他类或实现其他接口，在Swing编程中常用此方式来实现事件监听和回调。

35、内部类可以引用它的包含类（外部类）的成员吗？有没有什么限制？

答：一个内部类对象可以访问创建它的外部类对象的成员，包括私有成员。

36、Java 中的final关键字有哪些用法？

答：(1)修饰类：表示该类不能被继承；(2)修饰方法：表示方法不能被重写；(3)修饰变量：表示变量只能一次赋值以后值不能被修改（常量）。

37、指出下面程序的运行结果：



```

1  class A{
2
3      static{
4          System.out.print("1");
5      }
6
7      public A(){
8          System.out.print("2");
9      }
10 }
11
12 class B extends A{
13
14     static{
15         System.out.print("a");
16     }
17
18     public B(){
19         System.out.print("b");
20     }
21 }
22
23 public class Hello{
24
25     public static void main(String[] args){
26         A ab = new B();
27         ab = new B();
28     }
29
30 }

```

<https://blog.csdn.net/KamRoseLee>



答：执行结果：1a2b2b。创建对象时构造器的调用顺序是：先初始化静态成员，然后调用父类构造器，再初始化非静态成员，最后调用自身构造器。

38、数据类型之间的转换:

1)如何将字符串转换为基本数据类型?

2)如何将基本数据类型转换为字符串?

答:

1)调用基本数据类型对应的包装类中的方法parseXXX(String)或valueOf(String)即可返回相应基本类型;

2)一种方法是将基本数据类型与空字符串 (" ") 连接 (+) 即可获得其所对应的字符串; 另一种方法是调用String 类中的valueOf(...)方法返回相应字

39、如何实现字符串的反转及替换?

答：方法很多，可以自己写实现也可以使用String或StringBuffer / StringBuilder中的方法。有一道很常见的面试题是用递归实现字符串反转，代码如下

```

1  public static String reverse(String originStr) {
2      if(originStr == null || originStr.length() <= 1)
3          return originStr;
4      return reverse(originStr.substring(1)) + originStr.charAt(0);
5  }

```

40、怎样将GB2312编码的字符串转换为ISO-8859-1编码的字符串?

答：代码如下所示:

String s1 = "你好";

String s2 = new String(s1.getBytes("GB2312"), "ISO-8859-1");

41、日期和时间:

1)如何取得年月日、小时分钟秒?



2)如何取得从1970年1月1日0时0分0秒到现在的毫秒数?

3)如何取得某月的最后一天?

4)如何格式化日期?

答: 操作方法如下所示:

1)创建java.util.Calendar 实例, 调用其get()方法传入不同的参数即可获得参数所对应的值

2)以下方法均可获得该毫秒数:

```
1 Calendar.getInstance().getTimeInMillis();
2 System.currentTimeMillis();
```

3)示例代码如下:

```
1 Calendar time = Calendar.getInstance();
2 time.getActualMaximum(Calendar.DAY_OF_MONTH);
```

4)利用java.text.DateFormat 的子类 (如SimpleDateFormat类) 中的format(Date)方法可将日期格式化。

42、打印昨天的当前时刻。

答:

```
1 public class YesterdayCurrent {
2     public static void main(String[] args){
3         Calendar cal = Calendar.getInstance();
4         cal.add(Calendar.DATE, -1);
5         System.out.println(cal.getTime());
6     }
7 }
```

43、比较一下Java 和JavaScript。

答: JavaScript 与Java是两个公司开发的不同的两个产品。Java 是原Sun 公司推出的面向对象的程序设计语言, 特别适合于互联网应用程序开发; 而Netscape公司的产品, 为了扩展Netscape浏览器的功能而开发的一种可以嵌入Web页面中运行的基于对象和事件驱动的解释性语言, 它的前身是Live Java 的前身是Oak语言。

下面对两种语言间的异同作如下比较:

- 1) 基于对象和面向对象: Java是一种真正的面向对象的语言, 即使是开发简单的程序, 必须设计对象; JavaScript是种脚本语言, 它可以用来制作与用户交互作用的复杂软件。它是一种基于对象 (Object-Based) 和事件驱动 (Event-Driven) 的编程语言。因而它本身提供了非常丰富的内部对象用;
- 2) 解释和编译: Java 的源代码在执行之前, 必须经过编译; JavaScript 是一种解释性编程语言, 其源代码不需经过编译, 由浏览器解释执行;
- 3) 强类型变量和类型弱变量: Java采用强类型变量检查, 即所有变量在编译之前必须作声明; JavaScript中变量声明, 采用其弱类型。即变量在使用前, 而是解释器在运行时检查其数据类型;
- 4) 代码格式不一样。

补充: 上面列出的四点是原来所谓的标准答案中给出的。其实Java和JavaScript最重要的区别是一个是静态语言, 一个是动态语言。目前的编程语言的数式语言和动态语言。在Java中类 (class) 是一等公民, 而JavaScript中函数 (function) 是一等公民。对于这种问题, 在面试时还是用自己的语言回答。😄

44、什么时候用assert?

答: assertion(断言)在软件开发中是一种常用的调试方式, 很多开发语言中都支持这种机制。一般来说, assertion用于保证程序最基本、关键性的检查通常在开发和测试时开启。为了提高性能, 在软件发布后, assertion检查通常是关闭的。在实现中, 断言是一个包含布尔表达式的语句, 规定该表达式为true; 如果表达式计算为false, 那么系统会报告一个AssertionError。

断言用于调试目的:

```
assert(a > 0); // throws an AssertionError if a <= 0
```



4



断言可以有两种形式：

assert Expression1;

assert Expression1 : Expression2 ;

Expression1 应该总是产生一个布尔值。

Expression2 可以是得出一个值的任意表达式；这个值用于生成显示更多调试信息的字符串消息。

断言在默认情况下是禁用的，要在编译时启用断言，需使用source 1.4 标记：

javac -source 1.4 Test.java

要在运行时启用断言，可使用-enableassertions 或者-ea 标记。

要在运行时选择禁用断言，可使用-da 或者-disableassertions 标记。

要在系统类中启用断言，可使用-esa 或者-dsa 标记。还可以在包的基础上启用或者禁用断言。可以在预计正常情况下不会到达的任何位置上放置断言于验证传递给私有方法的参数。不过，断言不应该用于验证传递给公有方法的参数，因为不管是否启用了断言，公有方法都必须检查其参数。不过，办法中，也可以在非公有方法中利用断言测试后置条件。另外，断言不应该以任何方式改变程序的状态。

45、Error 和Exception 有什么区别？

答：Error 表示系统级的错误和程序不必处理的异常，是恢复不是不可能但很困难的情况下的一种严重问题；比如内存溢出，不可能指望程序能处理这Exception 表示需要捕捉或者需要程序进行处理的异常，是一种设计或实现问题；也就是说，它表示如果程序运行正常，从不会发生的情况。

补充：2005年摩托罗拉的面试中曾经问过这么一个问题 “If a process reports a stack overflow run-time error, what’ s the most possible cause 个选项a. lack of memory; b. write on an invalid memory space; c. recursive function calling; d. array index out of boundary. Java程序在运行遇StackOverflowError，这是一个错误无法恢复，只能重新修改代码了，这个面试题的答案是c。如果写了不能迅速收敛的递归，则很有可能引发栈溢下所示：

```
1 package com.lovo;
2
3 public class StackOverflowErrorTest {
4
5     public static void main(String[] args) {
6         main(null);
7     }
8 }
```

https://blog.csdn.net/KamRoseLee

因此，用递归编写程序时一定要牢记两点：1. 递归公式；2. 收敛条件（什么时候就不再递归而是回溯了）。

46、try{}里有一个return语句，那么紧跟在这个try后的finally{}里的code会不会被执行，什么时候被执行，在return前还是后？

答：会执行，在方法返回调用者前执行。Java允许在finally中改变返回值的做法是不好的，因为如果存在finally代码块，try中的return语句不会立马返是记录下返回值待finally代码块执行完毕之后再向调用者返回其值，然后如果在finally中修改了返回值，这会对程序造成很大的困扰，C#中就从语法上样的事。


47、Java 语言如何进行异常处理，关键字：throws、throw、try、catch、finally分别如何使用？

答：Java 通过面向对象的方法进行异常处理，把各种不同的异常进行分类，并提供了良好的接口。在Java 中，每个异常都是一个对象，它是Throwab的实例。当一个方法出现异常后便抛出一个异常对象，该对象中包含有异常信息，调用这个方法可以捕获到这个异常并进行处理。Java 的异常多个关键词来实现的：try、catch、throw、throws和finally。一般情况下是用try来执行一段程序，如果出现异常，系统会抛出（throw）一个异常，这过它的类型来捕捉（catch）它，或最后（finally）由缺省处理器来处理；try用来指定一块预防所有“异常”的程序；catch 子句紧跟在try块后面，用捕捉的“异常”的类型；throw 语句用来明确地抛出一个“异常”；throws用来标明一个成员函数可能抛出的各种“异常”；finally 为确保一段代码“异常”都被执行一段代码；可以在一个成员函数调用的外面写一个try语句，在这个成员函数内部写另一个try语句保护其他代码。每当遇到一个try 语常”的框架就放到栈上面，直到所有的try语句都完成。如果下一级的try语句没有对某种“异常”进行处理，栈就会展开，直到遇到有处理这种“异常”句。

48、运行时异常与受检异常有何异同？

答：异常表示程序运行过程中可能出现的非正常状态，运行时异常表示虚拟机的通常操作中可能遇到的异常，是一种常见运行错误，只要程序运行就不会发生。受检异常跟程序运行的上下文环境有关，即使程序设计无误，仍然可能因使用的问题而引发。Java编译器要求方法必须声明抛出可能发生的受检异常，但是并不要求必须声明抛出未被捕获的运行时常异常。异常和继承一样，是面向对象程序设计中经常被滥用的东西，神作《Effective Java》中对异常下指导原则：


- 不要将异常处理用于正常的控制流（设计良好的API不应该强迫它的调用者为了正常的控制流而使用异常）


4



















- 对可以恢复的情况使用受检异常，对编程错误使用运行时异常
- 避免不必要的使用受检异常（可以通过一些状态检测手段来避免异常的发生）
- 优先使用标准的异常
- 每个方法抛出的异常都要有文档
- 保持异常的原子性
- 不要在catch中忽略掉捕获到的异常



4



49、列出一些你常见的运行时异常？

答：

ArithmeticException（算术异常）

ClassCastException（类转换异常）

IllegalArgumentException（非法参数异常）

IndexOutOfBoundsException（下标越界异常）

NullPointerException（空指针异常）

SecurityException（安全异常）

50、final, finally, finalize 的区别？

答：final：修饰符（关键字）有三种用法：如果一个类被声明为final，意味着它不能再派生出新的子类，即不能被继承，因此它和abstract是反义词。final，可以保证它们在使用中不被改变，被声明为final的变量必须在声明时给定初值，而在以后的引用中只能读取不可修改。被声明为final的方法也用，不能在子类中被重写。finally：通常放在try...catch的后面构造总是执行代码块，这就意味着程序无论正常执行还是发生异常，这里的代码只要JVM执行，可以将释放外部资源的代码写在finally块中。finalize：Object类中定义的方法，Java中允许使用finalize()方法在垃圾收集器将对象从内存中清理必要的清理工作。这个方法是由垃圾收集器在销毁对象时调用的，通过重写finalize()方法可以整理系统资源或者执行其他清理工作。

股王MACD绝密战法曝光，50万资金入市，如今身家过亿！

股管家·顶新



想对作者说点什么

2019新鲜出炉的BAT通关面试题 Java岗

阅读数 353

一面偏架构方面1、介绍一下自己，讲讲项目经历2、你们项目中微服务是怎么划分的，划分粒度怎么确定？3、那在... [博文](#) 来自：[shadow_zed的博客](#)

2018最新BAT-前端必考面试（内附答案）

02-28

2018年BAT的前端面试题汇总，内附答案，pdf版本可直接打印。

下载

2018最新BAT《前端必考面试》.docx

09-20

题目示例如下：1、Doctype作用？严格模式与混杂模式如何区分？它们有何意义？（1）、声明位于文档中的最前面，处于 标签之前。告知浏...

下载

JAVA面试题集-JAVA面试题集

09-16

JAVA面试题集JAVA面试题集JAVA面试题集JAVA面试题集JAVA面试题集JAVA面试题集JAVA面试题集JAVA面试题集JAVA面试题集

下载



车间管理系统

java面试题集

09-07

java面试题集java面试题集java面试题集java面试题集java面试题集java面试题集java面试题集java面试题集java面试题集

下载

硬件工程师面试题集(含答案_很全)

08-20

最全硬件工程师笔试、面试题集，含答案

下载



Java面试题集 收集的

Java面试题集Java面试题集Java面试题集Java面试题集Java面试题集Java面试题集Java面试题集Java面试题集

02-21

下载



4

2018最新BAT java经典必考面试题

2018最新BAT java经典必考面试题，总结常见的知识点，不会后悔看的！

09-12

下载



java面试题集_java面试题集

java面试题集_java面试题集 java面试题集_java面试题集 java面试题集_java面试题集 java面试题集_java面试题集

08-24

下载



虹桥又出一股市怪才！50万资金入市，如今身家过亿！

股管家 · 顶新

JAVA重要的面试题集

JAVA面试题集，JAVA面试题集JAVA面试题集JAVA面试题集JAVA面试题集JAVA面试题集

11-08

下载

最新java面试题集

最新java面试题集

07-15

下载



先知 | 先觉

98篇文章

排名:千里之外

关注



xxs5258

29篇文章

排名:千里之外

关注



weixin_33796205

5821篇文章

排名:千里之外

关注

java 面试 题集 java 面试 题集

java 面试题集 java 面试题集 java 面试题集 java 面试题集 java 面试题集 java 面试题集

02-28

下载

JAVA面试题集宝典

JAVA面试题集 JAVA面试题集 JAVA面试题集 JAVA面试题集 JAVA面试题集

05-18

下载

BAT高级Java面试题70题目含答案

BAT高级Java面试题70题目含答案 BAT高级Java面试题70题目含答案 BAT高级Java面试题70题目含答案

08-17

下载

犹太人的两种神思维：死记七不买三不卖，你将赚到怀疑人生！

股管家 · 顶新

JAVA面试题集-个人网络收集整理。

JAVA面试题集。 JAVA面试题集。 JAVA面试题集。 JAVA面试题集。

11-06

下载

就是java面试题集

java面试题集。java面试题集java面试题集java面试题集

07-19

下载

JAVA面试题集--word格式

JAVA面试题集 JAVA面试题集 JAVA面试题集

02-13

下载

JAVA面试题集

JAVA面试题集JAVA面试题集JAVA面试题集

02-12

下载

Java的面试题集(经典面试题)

Java的面试题集 Java的面试题集 Java的面试题集

10-25

下载

人工智能3.0，多模块自由选择，领取学习图谱，进军AI领域！

程序员转型/入门人工智能领域需具备什么条件？

java 面试题集

java 面试题集 java 面试题集 java 面试题集

03-21

下载



java面试题(2018最新)

SSM1.Spring在SSM起什么作用Spring是一个轻量级框架，也是一个容器，Spring实质上讲就是一个Bean工厂，主...

博文 来自： 冷雨夜

阅读数 4407

2019最新BAT java经典必考面试题

2019最新BAT java经典必考面试题

JAVA面试题集高级篇

JAVA面试题集高级篇JAVA面试题集高级篇JAVA面试题集高级篇JAVA面试题集高级篇JAVA面试题集高级篇

Java陷阱—箩筐——面试题集(附答案)

Java陷阱—箩筐——面试题集 Java陷阱—箩筐——面试题集 Java陷阱—箩筐——面试题集 Java陷阱—箩筐——面试题集

这经典传奇！变态+99999！爆率+99999！卸载算我输！

贪玩游戏 · 顶新

JAVA面试题集(基础篇,高级篇 编程篇)

JAVA面试题集 分为JAVA面试题集基础篇 JAVA面试题集高级篇 JAVA面试题集编程篇

中科软Java面试题集

中科软Java面试题集中科软Java面试题集中科软Java面试题集

前端必考面试

2018年bat 前端必考面试题汇总

BAT前端面试题大全

2018最新BAT《前端必考面试》，包含前端基础知识考核点

大公司的Java面试题集

大公司的Java面试题集 大公司的Java面试题集

人工智能怎么学？对于转型的程序员有什么要求？

从国内的招聘网站看不得不说AI的岗位及薪资较优势，但是程序员转型有什么要求？

JAVA面试题集2.doc

JAVA面试题集2.doc JAVA面试题集2.doc

java面试题集（最新）

Web容器？ 实现J2EE规范中web协议的应用.该协议定义了web程序的运行时环境,包括:并发性,安全性,生命周期管理等等.

2018年最新Java 面试题集

一、第一套面试题1.Mybatis与Ibatis的区别2.Http1与Http2的区别3.SpringMVC的执行流程4.JVM内存溢出具体指... 博文 来自: weixin_33796205...

Java经典面试题集

Java经典面试题集(Java经典面试题集)

大公司的Java面试题集.txt

大公司的Java面试题集.txt大公司的Java面试题集.txt

重磅！6月份PYPL编程语言排行榜Python再次成为第一名，凭什么？

看完Python的就业前景分析，这么火是有原因的！

JAVA面试题收集--下载无需扣分，回帖加1分，欢迎下载，童叟无欺

资源目录如下 | java问题集锦.htm | JAVA面试题最全集.txt | java面试题集 - 《数据库专研》QQ高级群1：8476022；群2：5063844；群3...

JAVA面试题集.doc

JAVA面试题集.doc JAVA面试题集.doc

JAVA面试题集(答案

JAVA面试题集(答案)--基础知识 JAVA面试题集(答案)--基础知识

07-05

下载

12-17

下载

08-21

下载

06-11

下载

08-01

下载

12-15

下载

04-29

下载

10-30

下载

03-24

下载

11-10

下载

阅读数 19

08-10

下载

09-08

下载

04-16

下载

03-24

下载

07-05

下载


4

















JAVA面试题集1.doc

JAVA面试题集1.doc JAVA面试题集1.doc

03-24

下载



4

JAVA面试题集编程篇.pdf

JAVA面试题集编程篇.pdf JAVA面试题集编程篇.pdf JAVA面试题集编程篇.pdf JAVA面试题集编程篇.pdf JAVA面试题集编程篇.pdf

10-25

下载



成都冒菜加盟费多少钱，全程扶持开店！

成都冒菜加盟费多少

2018最新安卓面试大全（含BAT，网易，滴滴）----你面不上BAT的原因：面经宝典，都在这里啦

阅读数 5813

废话不多说，直接进入正题。童鞋们可以扫码右侧二维码，加入微信群，分享你的面试经历哦~ Java篇1.Java中sleep... 博文 来自： 李彬博客专栏

Java面试题全集（上）（中）（下）合集

03-06

Java面试题全集，含（上）（中）（下）合集。原文地址：http://blog.csdn.net/jackfrued/article/details/44931137

下载

2010年Java EE最新面试题集

05-22

Java EE最新面试题集 希望对大家有帮助

下载

Java程序员面试题

03-14

一些Java的面试题：JAVA面试题集基础篇.pdf、JAVA面试题解惑系列.pdf、JAVA面试题集高级篇.pdf、JAVA面试题集编程篇.pdf

下载

java面试题集，看你能回答出几个

12-31

java面试题集，看你能回答出几个 java面试题集，看你能回答出几个

下载

2018 BAT最新《前端必考面试题》 - KamRoseLee的博客 - CSDN博客

版权声明：https://blog.csdn.net/KamRoseLee/article/details/83584481 2018 BAT最新《前端必考面试题》 1、Doctype作用? 严格模式与混杂模式如...

2018最新java面试题(含答案) - weixin_33834910的博客 - CSDN博客

Java_EE面试题集(修正版)

11-18

Java_EE面试题集(修正版)Java_EE面试题集(修正版)Java_EE面试题集(修正版)Java_EE面试题集(修正版)Java_EE面试题集(修正版)Java_EE面试题...

下载

2018最新BAT java经典必考面试题最新版本.docx

09-20

题目示例如下：希望帮到大家 50、final, finally, finalize 的区别? 答：final：修饰符（关键字）有三种用法：如果一个类被声明为final，意味着...

下载

JAVA面试题集-----编程篇

11-12

JAVA面试题集(编程篇).rarJAVA面试题集(编程篇).rarJAVA面试题集(编程篇).rarJAVA面试题集(编程篇).rarJAVA面试题集(编程篇).rarJAVA面试题...

下载

所有java面试题集以及.net 面试题文档 可供参考

05-20

所有java面试题集以及.net 面试题文档 所有java面试题集以及.net 面试题文档 所有java面试题集以及.net 面试题文档 所有java面试题集以及.net...

下载

IBM、SUN等公司的Java面试题集

11-10

IBM、SUN等公司的Java面试题集IBM、SUN等公司的Java面试题集

下载

股神徐翔狱中曝出庄家洗盘规律，牢记这3点，A股就是提款机

股管家 · 顶新

java语言的面试题

09-18

面试题目，面试之前可以看看，Java面试题集-Java面试题,J2EE面试题集

下载

python图片处理类之~PIL.Image模块(ios android icon图标自动生成处理)

阅读数 14万+

1.从pyCharm提示下载PIL包nn http://www.pythonware.com/products/pil/nn nn2.解压后，进入到目录下nnncd ... 博文 来自： 专注于cocos+unit...



搭建单机版的FastDFS服务器

阅读数 1万+

由于FastDFS集群搭建非常复杂，对于初期学习FastDFS来说，搭建个单机版的作为入门更为实际一些。n第一步：搭...

博文 来自: u012453843的专栏



4

jquery/js实现一个网页同时调用多个倒计时(最新的)

阅读数 59万+

jquery/js实现一个网页同时调用多个倒计时(最新的)nn最近需要网页添加多个倒计时. 查阅网络,基本上都是千遍一律...

博文 来自: Websites



异常点/离群点检测算法——LOF

阅读数 8万+

局部异常因子算法-Local Outlier Factor(LOF) 在数据挖掘方面，经常需要在做特征工程和模型训练之前对数据进...

博文 来自: wangyibo0201的...



【深度剖析HMM（附Python代码）】1.前言及隐马尔科夫链HMM的背景

阅读数 3万+

1. 前言nn隐马尔科夫HMM模型是一类重要的机器学习方法，其主要用于序列数据的分析，广泛应用于语音识别、文...

博文 来自: tostq的专栏



微信支付V3微信公众号支付PHP教程(thinkPHP5公众号支付)/JSSDK的使用

阅读数 20万+

扫二维码关注，获取更多技术分享nnn 本文承接之前发布的博客《微信支付V3微信公众号支付PHP教程/thinkPHP5...

博文 来自: Marswill

idea创建springcloud项目图文教程(EurekaServer注册中心)（六）

阅读数 5万+

上一篇： nnidea创建springboot项目图文教程(配置文件)（五） nn nn http://blog.csdn.net/hcmomy/article/d...

博文 来自: hcmomy的博客

linux上安装Docker(非常简单的安装方法)

阅读数 32万+

最近比较有空，大四出来实习几个月了，作为实习狗的我，被叫去研究Docker了，汗汗！ nnDocker的三大核心概念...

博文 来自: 我走小路的博客

webService学习（二）—— 调用自定义对象参数

阅读数 4万+

webService学习（二）—— 调用自定义对象参数nn本文主要内容： rn1、如何通过idea进行webService Client的简...

博文 来自: 止水的专栏

单链表-Python操作

阅读数 1万+

链表是数据结构中最基本常用的，C++语言中单链表是利用指针操作实现的，python作为面向对象编程的，可以使...

博文 来自: 令狐公子的博客

Spring Boot集成持久化Quartz定时任务管理和界面展示

阅读数 8万+

前言本文是对之前的一篇文章Spring+SpringMVC+mybatis+Quartz整合代码部分做的一个修改和补充，其中最大...

博文 来自: 天降风云的博客

机器学习教程 Objective-C培训 交互设计视频教程 颜色模型 设计制作学习

mysql关联查询两次本表 native底部 react extjs glyph 图标 大数据最新教程2018 最新2018区块链价格

Java程序员2018最新面试题,想进BAT企业的必看(含答案解..._CSDN博客

Java程序员2018最新面试题,想进BAT企业的必看(含答案解析)为2019做准备2018年12月27日 16:27:03 weixin_34404393 阅读数:3 以下2018阿里巴巴最...

2018最新java面试题(含答案) - weixin_34216196的博客 - CSDN博客

2018最新《BATJava必考面试题集》1、面向对象的特征有哪些方面?答:面向对象的特征...抽... 博文 来自: KamRoseLee的博客 2018前端面试题总结 03-1...

没有更多推荐了，[返回首页](#)



bug发现与制造

关注

原创 313 粉丝 245 喜欢 2190 评论 57

等级: 访问: 24万+

积分: 7409 排名: 5338

勋章:





中科院在职博士

最新文章

- 网站测试清单
- 输入数据的设计方法和测试用例设计方法
- 浅谈易用性测试及GUI常见的测试要求
- 界面设计的行业标准总结二
- 界面设计的行业标准总结一

博主专栏



测试（开发、自动化）工程师专栏

文章数：19 篇 访问量：103



Windows Server 2012 R2

文章数：12 篇 访问量：58

个人分类

- Windows Server 2012R2服务器... 12篇
- JavaWeb相关 43篇
- window 删除文件提示指定的文件... 1篇
- JavaSE相关学习 17篇
- Java之JDBC的笔记 6篇

展开

归档

- 2019年7月 2篇
- 2019年5月 7篇
- 2019年3月 1篇
- 2019年2月 23篇
- 2019年1月 8篇

展开

热门文章

- Windows Server 2012R2 FTP服务介绍及搭建
阅读数 22050
- Windows Server 2012R2 DNS服务介绍及搭建
阅读数 21083
- Windows Server 2012R2 DHCP服务介绍及搭建
阅读数 14365
- FastDFS 分布式文件系统（部署和运维）
阅读数 13860
- Windows Server 2012R2 组策略



4



最新评论

Nginx负载均衡高可用 (keep...

qq_37242520: 懂原理者得天下!

大数据平台运维之Sqoop

xuejuzhou: 大佬可以给个微信吗

FastDFS 分布式文件系统 (部...

qq1274781334: [reply]u011342403[/reply] 集群的搭建, 最起码要一天, 还要不出错。。

FastDFS 分布式文件系统 (部...

pascallai: 提个小白问题, 网上说了一大堆fastDFS搭建、上传文件的方法, 当存储空间不够如何 ...

界面设计的行业标准总结一

fightsyj: 这个不错



腾讯云
学生服务器套餐
10元/月
• 1核2G • 1M宽带
• 50GB存储
立即购买



CSDN学院



CSDN企业招聘

QQ客服

kefu@csdn.net

客服论坛

400-660-0108

工作时间 8:30-22:00

关于我们 招聘 广告服务 网站地图

百度提供站内搜索 京ICP备19004658号

京公网安备11010502030143

©1999-2019 北京创新乐知网络技术有限公司

网络110报警服务 经营性网站备案信息
北京互联网违法和不良信息举报中心
中国互联网举报中心 家长监护 版权申诉



4

