

Java 基础方面:

1、作用域 **public,private,protected**,以及不写时的区别

答: 区别如下:

| 作用域 | 当前类 | 同一 package | 子孙类 | 其他 package |
|-----------|-----|------------|-----|------------|
| public | √ | √ | √ | √ |
| protected | √ | √ | √ | × |
| friendly | √ | √ | × | × |
| private | √ | × | × | × |

不写时默认为 friendly

2、**Anonymous Inner Class (匿名内部类)** 是否可以 **extends(继承)**其它类, 是否可以 **implements(实现)interface(接口)**

答: 匿名的内部类是没有名字的内部类。不能 extends(继承) 其它类, 但一个内部类可以作为一个接口, 由另一个内部类实现

3、**Static Nested Class** 和 **Inner Class** 的不同

答: Nested Class (一般是 C++ 的说法), Inner Class (一般是 JAVA 的说法)。Java 内部类与 C++ 嵌套类最大的不同就在于是否有指向外部的引用上。注: 静态内部类 (Inner Class) 意味着 1 创建一个 static 内部类的对象, 不需要一个外部类对象, 2 不能从一个 static 内部类的一个对象访问一个外部类对象

4、**&**和**&&**的区别

答: &是位运算符, 表示按位与运算, &&是逻辑运算符, 表示逻辑与 (and)

5、**Collection** 和 **Collections** 的区别

答: Collection 是集合类的上级接口, 继承与他的接口主要有 Set 和 List.

Collections 是针对集合类的一个帮助类, 他提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作



更多关注 Java 大后端公众号

6、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。在实现中,**assertion** 就是在程序中的一条语句,它对一个 **boolean** 表达式进行检查,一个正确程序必须保证这个 **boolean** 表达式的值为 **true**;如果该值为 **false**,说明程序已经处于不正确的状态下,系统将给出警告或退出。一般来说,**assertion** 用于保证程序最基本、关键的正确性。**assertion** 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

7、**String s = new String("xyz");**创建了几个 **String Object**

答: 两个,一个字符对象,一个字符对象引用对象

8、**Math.round(11.5)**等於多少? **Math.round(-11.5)**等於多少

答: **Math.round(11.5)**==12;**Math.round(-11.5)**==-11;**round** 方法返回与参数最接近的长整数,参数加 1/2 后求其 **floor**

9、**short s1 = 1; s1 = s1 + 1;**有什么错? **short s1 = 1; s1 += 1;**有什么错

答: **short s1 = 1; s1 = s1 + 1;** (**s1+1** 运算结果是 **int** 型,需要强制转换类型) **short s1 = 1; s1 += 1;** (可以正确编译)

10、Java 有没有 **goto**

答: java 中的保留字,现在没有在 java 中使用

11、数组有没有 **length()**这个方法? **String** 有没有 **length()**这个方法

答: 数组没有 **length()** 这个方法,有 **length** 的属性。**String** 有 **length()** 这个方法

12、**Overload** 和 **Override** 的区别。**Overloaded** 的方法是否可以改变返回值的类型

答: 方法的重写 **Overriding** 和重载 **Overloading** 是 Java 多态性的不同表现。重写 **Overriding** 是父类与子类之间多态性的一种表现,重载 **Overloading** 是一个类中多态性的一种表现。如果在子类中定义某方法与其父类有相同的名称和参数,我们说该方法被重写 (**Overriding**)。子类的对象使用这个方法时,将调用子类中的定义,对它而言,父类中的定义如同被“屏蔽”了。如果在一个类中定义了多个同名的方法,它们或有不同的参数个数或有不同的参数类型,则称为方法的



更多关注 Java 大后端公众号

重载(Overloading)。Overloaded 的方法是可以改变返回值的类型

13、Set 里的元素是不能重复的，那么用什么方法来区分重复与否呢？是用 == 还是 equals ()？它们有何区别

答：Set 里的元素是不能重复的，那么用 iterator() 方法来区分重复与否。equals() 是判读两个 Set 是否相等

equals() 和 == 方法决定引用值是否指向同一对象 equals() 在类中被覆盖，为的是当两个分离的对象的内容和类型相配的话，返回真值

14、给我一个你最常见到的 runtime exception

答：常见的运行时异常有如下这些 ArithmeticException, ArrayStoreException, BufferOverflowException, BufferUnderflowException, CannotRedoException, CannotUndoException, ClassCastException, CMMException, ConcurrentModificationException, DOMException, EmptyStackException, IllegalArgumentException, IllegalMonitorStateException, IllegalPathStateException, IllegalStateException, ImagingOpException, IndexOutOfBoundsException, MissingResourceException, NegativeArraySizeException, NoSuchElementException, NullPointerException, ProfileDataException, ProviderException, RasterFormatException, SecurityException, SystemException, UndeclaredThrowableException, UnmodifiableSetException, UnsupportedOperationException

15、error 和 exception 有什么区别

答：error 表示恢复不是不可能但很困难的情况下的一种严重问题。比如说内存溢出。不可能指望程序能处理这样的情况

exception 表示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

16、List, Set, Map 是否继承自 Collection 接口

答：List, Set 是, Map 不是

17、abstract class 和 interface 有什么区别



更多关注 Java 大后端公众号

答：声明方法的存在而不去实现它的类被叫做抽象类（**abstract class**），它用于要创建一个体现某些基本行为的类，并为该类声明方法，但不能在该类中实现该类的情况。不能创建 **abstract** 类的实例。然而可以创建一个变量，其类型是一个抽象类，并让它指向具体子类的一个实例。不能有抽象构造函数或抽象静态方法。**Abstract** 类的子类为它们父类中的所有抽象方法提供实现，否则它们也是抽象类为。取而代之，在子类中实现该方法。知道其行为的其它类可以在类中实现这些方法

接口（**interface**）是抽象类的变体。在接口中，所有方法都是抽象的。多继承性可通过实现这样的接口而获得。接口中的所有方法都是抽象的，没有一个有程序体。接口只可以定义 **static final** 成员变量。接口的实现与子类相似，除了该实现类不能从接口定义中继承行为。当类实现特殊接口时，它定义（即将程序体给予）所有这种接口的方法。然后，它可以在实现了该接口的类的任何对象上调用接口的方法。由于有抽象类，它允许使用接口名作为引用变量的类型。通常的动态联编将生效。引用可以转换到接口类型或从接口类型转换，**instanceof** 运算符可以用来决定某对象的类是否实现了接口

18、abstract 的 method 是否可同时是 static,是否可同时是 native, 是否可同时是 synchronized

答：都不能

19、接口是否可继承接口？抽象类是否可实现(implements)接口？抽象类是否可继承实体类(concrete class)

答：接口可以继承接口。抽象类可以实现(implements)接口，抽象类是否可继承实体类，但前提是实体类必须有明确的构造函数

20、构造器 Constructor 是否可被 override

答：构造器 Constructor 不能被继承，因此不能重写 Overriding，但可以被重载 Overloading

21、是否可以继承 String 类

答：String 类是 final 类故不可以继承



更多关注 Java 后端公众号

22、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 return 前执行

23、用最有效率的方法算出 2 乘以 8 等於几

答：2 << 3

24、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，有相同的 hash code

25、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。对象的内容可以在被调用的方法中改变，但对象的引用是永远不会改变的

26、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：witch (expr1) 中，expr1 是一个整数表达式。因此传递给 switch 和 case 语句的参数应该是 int、short、char 或者 byte。long,string 都不能作用于 switch

27、ArrayList 和 Vector 的区别,HashMap 和 Hashtable 的区别

答：就 ArrayList 与 Vector 主要从二方面来说。

一.同步性:Vector 是线程安全的，也就是说同步的，而 ArrayList 是线程程序不安全的，不是同步的

二.数据增长:当需要增长时,Vector 默认增长为原来一倍，而 ArrayList 却是原来的一半

就 HashMap 与 Hashtable 主要从三方面来说。

一.历史原因:Hashtable 是基于陈旧的 Dictionary 类的，HashMap 是 Java 1.2 引进的 Map 接口的一个实现

二.同步性:Hashtable 是线程安全的，也就是说同步的，而 HashMap 是线程程序不安全的，



更多关注 Java 大后端公众号

不是同步的

三.值: 只有 HashMap 可以让你将空值作为一个表的条目的 key 或 value

28、char 型变量中能不能存贮一个中文汉字?为什么?

答: 是能够定义成为一个中文的, 因为 java 中以 unicode 编码, 一个 char 占 16 个字节, 所以放一个中文是没问题的

29、GC 是什么? 为什么要有 GC

答: GC 是垃圾收集的意思 (Garbage Collection), 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 功能可以自动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。

30、float 型 float f=3.4 是否正确?

答: 不正确。精度不准确, 应该用强制类型转换, 如下所示: float f=(float)3.4

31、介绍 JAVA 中的 Collection Framework(包括如何写自己的数据结构)?

答: Collection Framework 如下:

Collection

└List

| └LinkedList

| └ArrayList

| └Vector

| └Stack

└Set

Map

└Hashtable

└HashMap

└WeakHashMap

Collection 是最基本的集合接口, 一个 Collection 代表一组 Object, 即 Collection 的元素 (Elements)

Map 提供 key 到 value 的映射



更多关注 Java 大后端公众号

32、抽象类与接口？

答：抽象类与接口都用于抽象，但是抽象类(JAVA 中)可以有自己的部分实现，而接口则完全是一个标识(同时有多重继承的功能)。

JAVA 类实现序列化的方法是实现 `java.io.Serializable` 接口

Collection 框架中实现比较要实现 `Comparable` 接口和 `Comparator` 接口

33、STRING 与 STRINGBUFFER 的区别。

答：STRING 的长度是不可变的，STRINGBUFFER 的长度是可变的。如果你对字符串中的内容经常进行操作，特别是内容要修改时，那么使用 `StringBuffer`，如果最后需要 `String`，那么使用 `StringBuffer` 的 `toString()` 方法

34、谈谈 final, finally, finalize 的区别

答：final—修饰符（关键字）如果一个类被声明为 final，意味着它不能再派生出新的子类，不能作为父类被继承。因此一个类不能既被声明为 `abstract` 的，又被声明为 `final` 的。将变量或方法声明为 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 的方法也同样只能使用，不能重载
finally—再异常处理时提供 finally 块来执行任何清除操作。如果抛出一个异常，那么相匹配的 `catch` 子句就会执行，然后控制就会进入 finally 块（如果有的话）

finalize—方法名。Java 技术允许使用 `finalize()` 方法在垃圾收集器将对象从内存中清除出去之前做必要的清理工作。这个方法是由垃圾收集器在确定这个对象没有被引用时对这个对象调用的。它是在 `Object` 类中定义的，因此所有的类都继承了它。子类覆盖 `finalize()` 方法以整理系统资源或者执行其他清理工作。`finalize()` 方法是在垃圾收集器删除对象之前对这个对象调用的

35、面向对象的特征有哪些方面

答：主要有以下四方面：

1.抽象：

抽象就是忽略一个主题中与当前目标无关的那些方面，以便更充分地注意与当前目标有关的方面。抽象并不打算了解全部问题，而只是选择其中的一部分，暂时不用部分细节。抽象包括两个方面，一是过程抽象，二是数据抽象。

2.继承：

继承是一种联结类的层次模型，并且允许和鼓励类的重用，它提供了一种明确表述共性的方法。对象的一个新类可以从现有的类中派生，这个过程称为类继承。新类继承了原始类的特性，新类称为原始类的派生类（子类），而原始类称为新类的基类（父类）。派生类可以从它的基类那里继承方法和实例变量，并且类可以修改或增加新的方法使之更适合特殊的需要。

3.封装：

封装是把过程和数据包围起来，对数据的访问只能通过已定义的界面。面向对象计算始于这个基本概念，即现实世界可以被描绘成一系列完全自治、封装的对象，这些对象通过一个受保护的接口访问其他对象。

4. 多态性：

多态性是指允许不同类的对象对同一消息作出响应。多态性包括参数化多态性和包含多态性。多



更多关注 Java 后端公众号

态性语言具有灵活、抽象、行为共享、代码共享的优势，很好的解决了应用程序函数同名问题。

36、String 是最基本的数据类型吗

答：基本数据类型包括 byte、int、char、long、float、double、boolean 和 short。

java.lang.String 类是 final 类型的，因此不可以继承这个类、不能修改这个类。为了提高效率节省空间，我们应该用 StringBuffer 类

37、int 和 Integer 有什么区别

答：Java 提供两种不同的类型：引用类型和原始类型（或内置类型）。Int 是 java 的原始数据类型，Integer 是 java 为 int 提供的封装类。Java 为每个原始类型提供了封装类。

原始类型封装类, booleanBoolean, charCharacter, byteByte, shortShort, intInteger, longLong, floatFloat, doubleDouble

引用类型和原始类型的行为完全不同，并且它们具有不同的语义。引用类型和原始类型具有不同的特征和用法，它们包括：大小和速度问题，这种类型以哪种类型的数据结构存储，当引用类型和原始类型用作某个类的实例数据时所指定的缺省值。对象引用实例变量的缺省值为 null，而原始类型实例变量的缺省值与它们的类型有关

38、运行时异常与一般异常有何异同

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。java 编译器要求方法必须声明抛出可能发生的非运行时异常，但是并不要求必须声明抛出未被捕获的运行时异常。

39、说出 ArrayList, Vector, LinkedList 的存储性能和特性

答：ArrayList 和 Vector 都是使用数组方式存储数据，此数组元素数大于实际存储的数据以便增加和插入元素，它们都允许直接按序号索引元素，但是插入元素要涉及数组元素移动等内存操作，所以索引数据快而插入数据慢，Vector 由于使用了 synchronized 方法（线程安全），通常性能上较 ArrayList 差，而 LinkedList 使用双向链表实现存储，按序号索引数据需要进行前向或后向遍历，但是插入数据时只需要记录本项的前后项即可，所以插入速度较快。

40、HashMap 和 Hashtable 的区别

答：HashMap 是 Hashtable 的轻量级实现（非线程安全的实现），他们都完成了 Map 接口，主要区别在于 HashMap 允许空（null）键值（key），由于非线程安全，效率上可能高于 Hashtable。HashMap 允许将 null 作为一个 entry 的 key 或者 value，而 Hashtable 不允许。

HashMap 把 Hashtable 的 contains 方法去掉了，改成 containsvalue 和 containskey。因为 contains 方法容易让人引起误解。

Hashtable 继承自 Dictionary 类，而 HashMap 是 Java1.2 引进的 Map interface 的一个实现。最大的不同是，Hashtable 的方法是 Synchronize 的，而 HashMap 不是，在多个线程访问 Hashtable 时，不需要自己为它的方法实现同步，而 HashMap 就必须为之提供外同步。

Hashtable 和 HashMap 采用的 hash/rehash 算法都大概一样，所以性能不会有很大的差异。

41、heap 和 stack 有什么区别

答：栈是一种线形集合，其添加和删除元素的操作应在同一段完成。栈按照后进先出的方式进行处理。堆是栈的一个组成元素

42、Java 的接口和 C++ 的虚类的相同和不同处



更多关注 Java 大后端公众号

答：由于 Java 不支持多继承，而有可能某个类或对象要使用分别在几个类或对象里面的方法或属性，现有的单继承机制就不能满足要求。与继承相比，接口有更高的灵活性，因为接口中没有任何实现代码。当一个类实现了接口以后，该类要实现接口里面所有的方法和属性，并且接口里面的属性在默认状态下面都是 `public static`，所有方法默认情况下是 `public`。一个类可以实现多个接口。

43、Java 中的异常处理机制的简单原理和应用

答：当 JAVA 程序违反了 JAVA 的语义规则时，JAVA 虚拟机就会将发生的错误表示为一个异常。违反语义规则包括 2 种情况。一种是 JAVA 类库内置的语义检查。例如数组下标越界，会引发 `IndexOutOfBoundsException`；访问 `null` 的对象时会引发 `NullPointerException`。另一种情况就是 JAVA 允许程序员扩展这种语义检查，程序员可以创建自己的异常，并自由选择何时用 `throw` 关键字引发异常。所有的异常都是 `java.lang.Throwable` 的子类。

43、垃圾回收的优点和原理。并考虑 2 种回收机制

答：Java 语言中一个显著的特点就是引入了垃圾回收机制，使 `c++` 程序员最头疼的内存管理的问题迎刃而解，它使得 Java 程序员在编写程序的时候不再需要考虑内存管理。由于有个垃圾回收机制，Java 中的对象不再有“作用域”的概念，只有对象的引用才有“作用域”。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级别的线程运行，不可预知的情况下对内存堆中已经死亡的或者长时间没有使用的对象进行清楚和回收，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。回收机制有分代复制垃圾回收和标记垃圾回收，增量垃圾回收。

44、你所知道的集合类都有哪些？主要方法？

答：最常用的集合类是 `List` 和 `Map`。`List` 的具体实现包括 `ArrayList` 和 `Vector`，它们是可变大小的列表，比较适合构建、存储和操作任何类型对象的元素列表。`List` 适用于按数值索引访问元素的情形。

`Map` 提供了一个更通用的元素存储方法。`Map` 集合类用于存储元素对（称作“键”和“值”），其中每个键映射到一个值。

45、描述一下 JVM 加载 class 文件的原理机制？

答：JVM 中类的装载是由 `ClassLoader` 和它的子类来实现的，Java `ClassLoader` 是一个重要的 Java 运行时系统组件。它负责在运行时查找和装入类文件的类。

46、排序都有哪几种方法？请列举

答：排序的方法有：插入排序（直接插入排序、希尔排序），交换排序（冒泡排序、快速排序），选择排序（直接选择排序、堆排序），归并排序，分配排序（箱排序、基数排序）
快速排序的伪代码。

// 使用快速排序方法对 `a[0 :n- 1]` 排序

从 `a[0 :n- 1]` 中选择一个元素作为 `m i d d l e`，该元素为支点

把余下的元素分割为两段 `left` 和 `right`，使得 `l e f t` 中的元素都小于等于支点，而 `right` 中的元素都大于等于支点

递归地使用快速排序方法对 `left` 进行排序

递归地使用快速排序方法对 `right` 进行排序



更多关注 Java 大后端公众号

所得结果为 l e f t + m i d d l e + r i g h t

47、JAVA 语言如何进行异常处理，关键字：throws,throw,try,catch,finally 分别代表什么意义？在 try 块中可以抛出异常吗？

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 Throwable 类或其它子类的实例。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 的异常处理是通过 5 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（throws）一个异常，这时候你可以通过它的类型来捕捉（catch）它，或最后（finally）由缺省处理器来处理。用 try 来指定一块预防所有“异常”的程序。紧跟在 try 程序后面，应包含一个 catch 子句来指定你想要捕捉的“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 为确保一段代码不管发生什么“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 语句，“异常”的框架就放到堆栈上面，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，堆栈就会展开，直到遇到有处理这种“异常”的 try 语句。

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

答：可以。必须只有一个类名与文件名相同。

49、java 中有几种类型的流？JDK 为每种类型的流提供了一些抽象类以供继承，请说出他们分别是哪些类？

答：字节流，字符流。字节流继承于 InputStream OutputStream，字符流继承于 InputStreamReader OutputStreamWriter。在 java.io 包中还有许多其他的流，主要是为了提高性能和使用方便。

50、java 中会存在内存泄漏吗，请简单描述。

答：会。自己实现堆载的数据结构时有可能会出现内存泄露，可参看 effective java.

51、java 中实现多态的机制是什么？

答：方法的重写 Overriding 和重载 Overloading 是 Java 多态性的不同表现。重写 Overriding 是父类与子类之间多态性的一种表现，重载 Overloading 是一个类中多态性的一种表现。

52、垃圾回收器的基本原理是什么？垃圾回收器可以马上回收内存吗？有什么办法主动通知虚拟机进行垃圾回收

答：对于 GC 来说，当程序员创建对象时，GC 就开始监控这个对象的地址、大小以及使用情况。通常，GC 采用有向图的方式记录和管理堆(heap)中的所有对象。通过这种方式确定哪些对象是“可达的”，哪些对象是“不可达的”。当 GC 确定一些对象为“不可达”时，GC 就有责任回收这些内存空间。可以。程序员可以手动执行 System.gc()，通知 GC 运行，但是 Java 语言规范并不保证 GC 一定会执行。

53、静态变量和实例变量的区别？



更多关注 Java 大后端公众号

答: `static i = 10; //常量` `class A a;` `a.i =10; //可变`

54、什么是 java 序列化, 如何实现 java 序列化?

答: 序列化就是一种用来处理对象流的机制, 所谓对象流也就是将对象的内容进行流化。可以对流化后的对象进行读写操作, 也可将流化后的对象传输于网络之间。序列化是为了解决在对对象流进行读写操作时所引发的问题。

序列化的实现: 将需要被序列化的类实现 `Serializable` 接口, 该接口没有需要实现的方法, `implements Serializable` 只是为了标注该对象是可被序列化的, 然后使用一个输出流 (如: `FileOutputStream`) 来构造一个 `ObjectOutputStream` (对象流) 对象, 接着, 使用 `ObjectOutputStream` 对象的 `writeObject(Object obj)` 方法就可以将参数为 `obj` 的对象写出 (即保存其状态), 要恢复的话则用输入流。

55、是否可以从一个 static 方法内部发出对非 static 方法的调用?

答: 不可以, 如果其中包含对象的 `method()`; 不能保证对象初始化。

56、写 clone() 方法时, 通常都有一行代码, 是什么?

答: `Clone` 有缺省行为, `super.clone()`; 他负责产生正确大小的空间, 并逐位复制。

57、在 JAVA 中, 如何跳出当前的多重嵌套循环?

答: 用 `break`; `return` 方法。

58、List、Map、Set 三个接口, 存取元素时, 各有什么特点?

答: `List` 以特定次序来持有元素, 可有重复元素。`Set` 无法拥有重复元素, 内部排序。`Map` 保存 key-value 值, value 可多值。

59、说出一些常用的类, 包, 接口, 请各举 5 个

答: 常用的类: `BufferedReader` `BufferedWriter` `FileReader` `FileWritter` `String` `Integer`

常用的包: `java.lang` `java.awt` `java.io` `java.util` `java.sql`

常用的接口: `Remote` `List` `Map` `Document` `NodeList`



更多关注 Java 大后端公众号