**JDK 和 JRE 有什么区别？\*\***

JDK：java 开发工具包

JRE：java 运行环境

JRE包含核心类库和JVM，而JDK包含编译、运行等开发工具和JRE。

如果需要运行java程序，只需安装JRE，如果需要编写java 程序，需要安装JDK。

**类的加载顺序**

(1) 父类静态变量，父类静态代码块（只执行一次）

(2) 子类静态变量，子类静态代码块（只执行一次）

(3) 父类非静态变量，父类非静态代码块

(4) 父类构造函数

(5) 子类非静态变量，子类非静态代码块

(6) 子类构造函数

**关键字**

true、false、null都不是关键字

**int和Integer的区别**

1、Integer是包装类，int是基本数据类型

2、Integer 存储的是对象引用，int 存储的是数据值

3、Integer的默认值是null，int的默认值是0。

4. Integer 用 equals 来比较，int 用 == 比较

**静态方法不能被重写**

static方法不能被子类覆写，在子类中定义了和父类完全相同的static方法，则父类的static方法被隐藏，Son.staticmethod()或new Son().staticmethod()都是调用的子类的static方法，如果是Father.staticmethod()或者Father f = new Son(); f.staticmethod()调用的都是父类的static方法。

**Java有哪些数据类型\*\***

基本数据类型有8种：byte、short、int、long、float、double、char、boolean，引用数据类型有类(class)、接口(interface)、数组([])

**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**

switch(expr)中，expr可以是byte、short、int、enum 类型、char、String

用最有效率的方法计算 2 乘以 8

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）。

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

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。

表达式的自动类型转换

在表达式中，小范围类型的变量会自动转换成当前较大范围的类型再运算

byte 、short、char 🡪 int 🡪 long 🡪 float 🡪 double

表达式的最终结果类型由表达式中的最高类型决定

在表达式中，byte、short、char 是直接转换成int类型参与运算的

同级之间转换需要强转，高级向低级转换也需要强转

float f=3.4;是否正确

不正确。3.4是double类型，将double赋值给float属于向下转型，会造成精度损失，因此需要强制类型转换float f =(float)3.4; 或者写成 float f =3.4F;。

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

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 += 1;可以正确编译，因为 s1+= 1;相当于 s1 = (short(s1 + 1);其中有隐含的强制类型转换。

== 和 equals 的区别是什么？\*\*

==对于基本类型，比较的是值，对于引用类型，比较的是内存地址。而 equals 默认比较的是内存地址，String、Integer 等类重写了 equals 方法，比较的是内容。

final 在 java 中有什么作用？\*\*

final 修饰的类叫最终类，该类不能被继承。

final 修饰的方法不能被重写。

final 修饰的变量叫常量，常量必须初始化，值不能被修改。

final finally finalize区别\*\*

final 修饰的类叫最终类，该类不能被继承。final 修饰的方法不能被重写。

final 修饰的变量叫常量，常量必须初始化，值不能被修改。

finally表示不管是否出现异常，该代码块都会执行

finalize，是Java中的一个方法名，垃圾回收器准备释放内存时，会先调用finalize()

访问修饰符public,private,protected,以及缺省（默认）时的区别？\*\*

**private：只能在本类中访问。**

**缺省：本类，同一个包下的类中。**

**protected：本类，同一个包下的类中，其他包下的子类**

**public：任何地方都可以**

&和&&的区别？

**&&具有短路效果，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。**

this关键字的用法\*\*

this是指向对象本身的一个指针。

形参与成员变量重名，用this来区分

可以引用本类的构造函数

super关键字的用法\*\*

super是指向自己父类对象的一个指针

子类中的成员变量或方法与父类中的成员变量或方法同名时，用super来区分

可以引用父类的构造函数

用super.xxx来引用父类的成员。

static关键字的作用？\*\*

**static修饰成员变量表示该成员变量在内存中只存储一份，可以被共享访问、修改。**

static注意事项\*\*

**静态方法只能访问静态的成员，不可以直接访问实例成员。**

**实例方法可以访问静态的成员，也可以访问实例成员。**

**静态方法中是不可以出现this关键字的。static不能修饰局部变量。**

break ,continue ,return 的区别及作用

**break 跳出循环，不再执行循环**

**continue 跳出本次循环，继续执行下次循环**

**return 程序返回，不再执行下面的代码**

什么是面向对象？ \*\*

对比面向过程，是两种不同的处理问题的角度。

面向过程更注重事情的每一个步骤及顺序，面向对象更注重事情有哪些参与者（对象）、及各自需要做什么。

面向对象的四大特性？\*\*

**封装：隐藏对象的属性和实现细节，仅对外公开接口。**

**继承：子类继承父类的特征和行为。子类可以有父类的属性，方法，也可以对父类进行扩展。**

**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 ,抽象只关注对象的哪些属性和行为,并不关注这些行为的细节是什么。(普通类)**

**多态：同类型的对象，执行同一个行为，会表现出不同的行为特征。**

Java中实现多态的机制是什么？\*\*

**前提：有继承或者实现关系、有父类引用指向子类对象、有方法重写。**

**多态中成员访问特点：**

**变量调用：编译看左边，运行也看左边。**

**方法调用：编译看左边，运行看右边。**

静态方法：编译和运行都看左边

接口和抽象类有什么区别？\*\*

实现：抽象类的子类使用 extends 来继承；接口的实现类使用 implements 来实现。

构造函数：抽象类可以有构造函数；接口不能有。

实现数量：类只能继承一个抽象类，可以实现很多个接口。

访问修饰符：抽象类中的方法可以是任意访问修饰符，接口中的方法默认使用 public修饰。

变量类型：抽象类中的成员变量可以是各种类型的，而接口中的成员变量只能是public static final类型的。

应用：

拥有一些方法并且想让它们中的一些有默认实现，使用抽象类。

部分方法有默认实现，用抽象类

如果想实现多重继承，那么必须使用接口。由于Java不支持多继承，子类不能够继承多个类，但可以实现多个接口。因此就可以使用接口来解决它。

普通类和抽象类有哪些区别？

普通类不能包含抽象方法，抽象类可以包含抽象方法。

普通类可以直接实例化，抽象类不能直接实例化。

抽象类必须要有抽象方法吗？

抽象类中不一定有抽象方法，有抽象方法的类一定是抽象类

final和abstract是什么关系？

互斥关系

abstract定义的抽象类作为模板让子类继承，final定义的类不能被继承。

abstract定义的方法让子类重写，final定义的方法子类不能重写。

创建一个对象用什么关键字？对象实例与对象引用有何不同？

new关键字，new创建对象实例放在堆内存中，对象引用指向对象实例，对象引用存放在栈内存中。一个对象引用可以指向0个或1个对象，一个对象可以有n个引用指向它

成员变量与局部变量的区别有哪些

成员变量：方法外部，类内部定义的变量

局部变量：类的方法中的变量。

作用域：成员变量针对整个类有效，局部变量只在方法内有效。

生命周期：成员变量随着对象的创建而存在，随着对象的消失而消失，存储在堆内存中。局部变量在方法被调用时存在，方法结束后就消失，存储在栈内存中。

初始值：成员变量有默认初始值，局部变量没有默认初始值，使用前必须赋值。

使用原则

在使用变量时需要遵循的原则为：就近原则

首先在局部范围找，有就使用；接着在成员位置找。

在Java中定义一个不做事且没有参数的构造方法的作用

**子类构造方法如果没有用super()来调用父类特定的构造方法，则会默认调用父类中的无参构造方法。**

**因此，如果父类中只定义了有参数的构造方法，而在子类的构造方法中又没有用super()来调用特定的构造方法，则编译时发生错误，因为Java程序在父类中找不到无参构造方法可供执行。**

**解决方法：在父类中加一个不做事且没有参数的构造方法。**

一个类的构造方法的作用是什么？若一个类没有声明构造方法，改程序能正确执行吗？为什么？

主要作用是完成对象的初始化工作。可以执行。因为一个类即使没有声明构造方法也会有默认的不带参数的构造方法。

内部类

普通类（外部类）：只能用public、default（不写）、abstract、final修饰。

成员内部类：定义在外部类的成员位置上，没有static修饰，修饰类成员的public、protected、default、private等关键字都能使用。可理解为外部类的成员

静态内部类：定义在外部类的成员位置上，使用static修饰，特点与外部类一样

局部内部类：定义在方法内部的类，不能用上述关键词来修饰。

匿名内部类：用于创建一个类的实例，而这个类只需要使用一次

构造方法有哪些特性？

**名字与类名相同。**

**没有返回值，但不能用void声明构造函数。**

**生成类的对象时自动执行，无需调用。**

构造器（constructor）是否可被重写（override）？

**构造器不能被继承，因此不能被重写，但可以被重载。**

静态变量和实例变量区别\*\*

**静态变量：属于类的，在内存中只存储一份，可以被共享访问、修改。**

**实例变量：属于实例对象的，会为每个对象分配成员变量内存空间。**

静态方法和实例方法区别

**调用静态方法，可以使用"类名.方法名"或者"对象名.方法名"两种方式，可以无需创建对象**

**调用实例方法只能使用"对象名.方法名"的方式**

**静态方法只能访问静态的成员，不可以直接访问实例成员，实例方法则无此限制**

什么是方法的返回值？返回值的作用是什么？

方法的返回值是指获取到的某个方法体中的代码执行后产生的结果。返回值的作用是接收结果。

重载和重写的区别

重载：发生在同一个类中，方法名必须相同，参数类型不同、个数不同、顺序不同。(方法名参数列表相同，不构成重载，编译器报错)

重写：发生在父子类中，方法名、参数列表必须相同，子类方法的访问修饰符范围大于等于父类；如果父类方法访问修饰符为private则子类不能重写该方法。

值传递和引用传递有什么区别

值传递是指基本数据类型在方法中的传递，引用传递是指引用数据类型在方法中的传递。

值传递是传递变量的值，并不会改变方法外变量的值；引用传递是传递对象的地址，会改变对象本身的值；

创建对象的几种方式

//1.通过new申请对象空间（堆内存）

Person person = new Person();

//2. 运用反射手段,通过Class类中的newInstance()方法

Class clazz = Person.class;

Person p = (Person) clazz.newInstance();

//3.通过Object类中的clone()方法

//类对象，实现实现Cloneable接口

//不会执行构造方法

//设计模式 ------>Prototype原型模式

//4.通过对象的反序列

FileInputStream fileInputStream = new FileInputStream("path");

ObjectInputStream ois = new ObjectInputStream(fileInputStream);

Person person1 = (Person) ois.readObject();

JDK 中常用的包有哪些

java.lang：这个是系统的基础类；

java.io：这里面是所有输入输出有关的类，比如文件操作等；

java.nio：为了完善 io 包中的功能，提高 io 包中性能而写的一个新包；

java.net：这里面是与网络有关的类；

java.util：这个是系统辅助类，特别是集合类；

java.sql：这个是数据库操作的类。

Files的常用方法都有哪些？

Files.exists()：检测文件路径是否存在。

Files.createFile()：创建文件。

Files.createDirectory()：创建文件夹。

Files.delete()：删除一个文件或目录。

Files.copy()：复制文件。

Files.move()：移动文件。

Files.size()：查看文件个数。

Files.read()：读取文件。

Files.write()：写入文件。

java 中操作字符串都有哪些类？它们之间有什么区别？

操作字符串的类有：String、StringBuffer、StringBuilder。

String 声明的是不可变的对象，每次操作都会生成新的 String 对象，然后将指针指向新的 String 对象，而 StringBuffer、StringBuilder 可以在原有对象的基础上进行操作，经常改变字符串内容时最好不要使用 String。

StringBuffer 是线程安全的，而 StringBuilder 是非线程安全的，在单线程下使用 StringBuilder，多线程下使用 StringBuffer。

String字符串内容不可变，引用可变

String str="i"与 String str=new String("i")一样吗？

内存的分配方式不一样。String str="i"的方式，java 虚拟机会将其分配到常量池中；而 String str=new String("i") 则会被分配到堆内存中。

什么是字符串常量池？

字符串常量池位于方法区中，用来存储字符串常量，避免开辟多块空间存储相同的字符串，在创建字符串时 JVM 会首先检查字符串常量池，如果该字符串已经存在池中，则返回它的引用，如果不存在，则实例化一个字符串放到池中，并返回其引用。

是否可以继承String类？

**String 类是final类，不可以被继承。**

char型变量中能否能不能存储一个中文汉字，为什么？

**char可以存储一个中文汉字，因为Java中使用的编码是Unicode，一个char 类型占2个字节（16 比特），所以放一个中文是没问题的。**

String 类的常用方法都有那些？

indexOf()：返回指定字符的索引。

charAt()：返回指定索引处的字符。

replace()：字符串替换。

trim()：去除字符串两端空白。

split()：分割字符串，返回一个分割后的字符串数组。

getBytes()：返回字符串的 byte 类型数组。

length()：返回字符串长度。

toLowerCase()：将字符串转成小写字母。

toUpperCase()：将字符串转成大写字符。

substring()：截取字符串。

equals()：字符串比较。

toCharArray()

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

如果常量池中存在“xyz”就只会创建一个对象，如果没有就会创建两个！

自动装箱与拆箱

装箱：将基本类型用它们对应的引用类型包装起来；

拆箱：将包装类型转换为基本数据类型；

集合和数组的区别

数组是固定长度的；集合是可变长度的。

数组可以存储基本数据类型，也可以存储引用数据类型；集合只能存储引用数据类型。

ArrayList 和 LinkedList 的区别是什么？

**ArrayList底层是基于数组实现的，查询快，增删慢。支持随机访问**

**LinkedList底层基于双向循环链表实现的，查询慢，增删快。顺序访问。LinkedList 比 ArrayList 更占内存，因为 LinkedList 的节点除了存储数据， 还存储了两个引用**

常用的集合类有哪些？

Map接口和Collection接口是所有集合框架的父接口：

Collection接口的子接口包括：Set接口和List接口

**Map接口的实现类主要有：HashMap、TreeMap、Hashtable、ConcurrentHashMap以及Properties等**

Set接口的实现类主要有：HashSet、TreeSet、LinkedHashSet等

List接口的实现类主要有：ArrayList、LinkedList、Stack以及Vector等

集合框架底层数据结构

Collection

1. List

Arraylist： Object数组

Vector： Object数组

LinkedList： 双向循环链表

1. Set

HashSet（无序，唯一）：基于 HashMap 实现的，底层采用 HashMap 来保存元素

LinkedHashSet： LinkedHashSet 继承与 HashSet，并且其内部是通过 LinkedHashMap来实现的。有点类似于我们之前说的LinkedHashMap 其内部是基于 Hashmap 实现一样，不过还是有一点点区别的。

TreeSet（有序，唯一）： 红黑树(自平衡的排序二叉树。)

1. Map

HashMap：JDK1.8之前HashMap由数组+链表组成的，数组是HashMap的主体，链表则是主要为了解决哈希冲突而存在的（“拉链法”解决冲突）.JDK1.8以后在解决哈希冲突时有了较大的变化，当链表长度大于阈值（默认为8）时，将链表转化为红黑树，以减少搜索时间

LinkedHashMap：LinkedHashMap 继承自 HashMap，所以它的底层仍然是基于拉链式散列结构即由数组和链表或红黑树组成。另外，LinkedHashMap 在上面结构的基础上，增加了一条双向链表，使得上面的结构可以保持键值对的插入顺序。同时通过对链表进行相应的操作， 实现了访问顺序相关逻辑。LinkedHashMap按照插入顺序排序

HashTable： 数组+链表组成的，数组是 HashMap 的主体，链表则是主要为了解决哈希冲突而存在的

TreeMap： 红黑树（自平衡的排序二叉树），可以自然排序

Collection 和 Collections 有什么区别？

Collection 是一个集合接口，它的子接口有List与Set。

Collections是集合类的一个工具类，提供了一系列静态方法，用于对集合中元素进行排序、搜索等。

List、Set、Map 之间的区别是什么？

List:有序(存储和取出的元素顺序一致)、可重复、有索引(可以通过索引操作元素), 可以存入多个null元素。List 实现类有 ArrayList、LinkedList 和 Vector

Set:无序、不重复、无索引, 只允许存入一个null元素，Set 的实现类有 HashSet、TreeSet 和 LinkedHashSet

Map: Map 集合中存储的是键值对，键不能重复，值可以重复。Map 的实现类有 HashMap、TreeMap 和 LinkedHashMap

哪些集合类是线程安全的？

vector：就比arraylist多了个同步化机制（线程安全），因为效率较低，现在已经不太建议使用。

statck：堆栈类，先进后出。

hashtable：就比hashmap多了个线程安全。

迭代器 Iterator 是什么？

Iterator是可以遍历集合的对象，目的是屏蔽不同数据集合的特点，统一遍历集合的接口

Iterator 怎么使用？有什么特点？

List list = new ArrayList<>();

Iterator it = list. iterator();

while(it. hasNext()){ String obj = it. next(); System. out. println(obj);}

Iterator 的特点是只能单向遍历

for从头往后遍历删除有啥问题

for-each循环：不能直接删除元素，会抛出并发修改异常ConcurrentModificationException

普通for循环：可能导致索引错位和性能问题。

Iterator：是遍历和删除元素的最佳方式，既安全又高效。

如何边遍历边移除 Collection 中的元素？

使用 Iterator.remove() 方法，如下：

Iterator<Integer> it = list.iterator();

while(it.hasNext()){\*// do something\* it.remove();}

Iterator 和 ListIterator 有什么区别？

Iterator 可以遍历List和Set，ListIterator 只能遍历 List。

Iterator 只能单向遍历，ListIterator 可以双向遍历（向前/后遍历）。

遍历一个 List 有哪些不同的方式？每种方法的实现原理是什么？Java 中 List 遍历的最佳实践是什么？

遍历方式有以下几种：

1. for 循环遍历，基于计数器。

2. 迭代器遍历，Iterator。Iterator是可以遍历集合的对象，目的是屏蔽不同数据集合的特点，统一遍历集合的接口

3. foreach 循环遍历。foreach 内部也是采用了 Iterator 的方式实现，缺点是不能在遍历过程中操作数据集合，例如删除、替换。

最佳实践：建议用 Iterator 或 foreach 遍历。

如何实现数组和 List 之间的转换？

**List转换成为数组：调用ArrayList的toArray方法。**

**数组转换成为List：调用Arrays的asList方法。**

多线程场景下如何使用 ArrayList？

**ArrayList 不是线程安全的，如果遇到多线程场景，可以通过 Collections 的synchronizedList方法将其转换成线程安全的容器后再使用。例如像下面这样： List<String> synchronizedList = Collections.synchronizedList(list);**

**synchronizedList.add("aaa");**

**synchronizedList.add("bbb");**

**for(int i=0;i<synchronizedList.size();i++){ System.out.println(synchronizedList.get(i));**

**}**

说一下 HashSet 的实现原理？

**HashSet 是基于 HashMap 实现的，HashSet的值存放于HashMap的key上**

说一下 HashMap的实现原理？\*\*

**HashMap由数组+链表组成，元素以内部节点Node节点存在。**

**插入一个key时，计算key的hash值，二次hash然后对数组长度取模，对应到数组下标。**

**如果没有产生hash冲突，则创建Node存入数组。**

**如果产生hash冲突，遍历该位置上链表中的所有数据，通过equals方法和传入的key比较，如果key相同则取代该元素，否则，判断链表高度并插入链表，链表高度达到8，数组长度达到64则转为红黑树，长度低于6则将红黑树转回链表。（数据碰撞的时候遍历该位置上链表中的所有数据，并且通过equals()方法来比对每个数据的key。如果key相同的话，会将链表上该位置的数据进行覆盖。）**

1.7 插入操作使用头插法

1.8 插入操作使用尾插法

HashMap是怎么解决哈希冲突的？

1. **使用链地址法来链接拥有相同hash值的数据；**
2. 再哈希法：即发生冲突时，由其他的函数再计算一次哈希值。

什么是哈希冲突？

**两个不同的输入值，根据同一散列函数计算出相同的散列值，我们就把它叫做碰撞（哈希碰撞）。**

Hash结构存在无碰撞吗

**当键（Key）的数量超过哈希表的桶（Bucket）数量时，必然存在不同键被映射到同一位置的情况，导致碰撞。**

Hashmap什么api线程不安全

**put、remove、clear、resize，数据不一致、死循环、丢失更新**

hash 函数的作用？

**hash方法实际是让key.hashCode()与key.hashCode()>>>16进行异或操作，高16bit补0，一个数和0异或不变，所以作用就是：高16bit不变，低16bit和高16bit做了一个异或，目的是减少碰撞**

HashMap 的长度为什么是2的幂次方

为了能让 HashMap 存取高效，尽量较少碰撞，也就是要尽量把数据分配均匀

如何决定使用 HashMap 还是 TreeMap？

TreeMap可以存储有序的key

HashMap 、HashTable、ConcurrentHashMap 的区别

HashMap键值对可以为null，线程不安全

ConcurrentHashMap键值对不允许为null，线程安全，性能优于HashTable，底层采用分段的数组+链表实现。synchronized只锁定当前链表或红黑二叉树的首节点

HashTable键值对不允许为null，线程安全。

ConcurrentHashMap和Hashtable的区别

ConcurrentHashMap只会锁定map的某个部分，而Hashtable则会锁定整个map

ConcurrentHashMap 底层具体实现知道吗？实现原理是什么？

ConcurrentHashMap采用数组+链表+红黑树来存放元素。当链表长度超过阈值（默认 8）且ConcurrentHashMap的容量大于或等于阈值（默认64）,链表会转化为红黑树；当树中的节点数小于阈值（默认为 6）时，又会退化为链表.

**1. CAS：插入数据时，若桶为空，通过CAS操作直接写入，避免加锁。**

**2. synchronized锁桶头节点：若桶非空，仅对当前桶的头节点加锁，锁粒度更细。**

throw 和 throws 的区别是什么？

Java的异常处理包括声明异常和拋出异常，使用throws在方法上声明要拋出的异常，使用throw在方法内拋出异常对象。

throw 关键字用在方法内部，只能用于抛出一种异常，用来抛出方法或代码块中的异常，受查异常和非受查异常都可以被抛出。

throws 关键字用在方法声明上，可以抛出多个异常，用来标识该方法可能抛出的异常列表。一个方法用throws 标识了可能抛出的异常列表，调用该方法的方法中必须包含可处理异常的代码，否则也要在方法签名中用 throws 关键字声明相应的异常。

常见的 RuntimeException 有哪些？

ClassCastException(类转换异常)

IndexOutOfBoundsException(数组越界)

NullPointerException(空指针)

异常

Throwable有两个子类：Exception（异常）和Error（错误）

**Error（错误）：程序无法处理的错误**

**Exception（异常）：程序可以处理的异常**

**运行时异常：都是RuntimeException类及其子类异常**

**非运行时异常：必须捕获或者抛出，否则编译不通过。如IOException,ClassNotFoundException等。**

Java异常关键字

**try – 用于监听可能抛出异常的代码。（用于监听。将要被监听的代码(可能抛出异常的代码)放在try语句块之内，当try语句块内发生异常时，异常就被抛出。）**

**catch ，用来捕获异常。**

finally表示不管是否出现异常，该代码块都会执行

**throw – 用于抛出异常**

**throws – 用在方法签名中，用于声明该方法可能抛出的异常。**

错误有哪些

OutOfMemorryError，StackOverFlowError

类定义错误（NoClassDefFoundError）

运行时异常和一般异常(受检异常)区别是什么？

运行时异常包括 RuntimeException 类及其子类，表示 JVM 在运行期间可能出现的异常。

受检异常是Exception 中除 RuntimeException 及其子类之外的异常。 Java 编译器会检查受检异常

Error 和 Exception 区别是什么？

Error类型的错误发生，通常应用程序会被终止，仅靠应用程序本身无法恢复；

Exception类的错误是可以在应用程序中进行捕获并处理的

try-catch-finally 中，如果 catch 中 return 了，finally 还会执行吗？

会执行，在 return 前执行。finally是在try、catch中的return后面的表达式运算后执行的

try-catch-finally 中哪个部分可以省略？

catch 可以省略

如何确定捕获顺序(多个catch)

[Java中有多个异常, 如何确定捕获顺序(多个catch),先从上到下执行,判断异常的大小,如果包含捕到异常,就进入这个catch,后面的就不再执行](https://www.cnblogs.com/andy-lehhaxm/p/9528877.html)

深拷贝和浅拷贝区别是什么？

**浅拷贝是增加了一个指针指向原有对象（clone方法是一个浅拷贝）**

**深拷贝是增加了一个指针并且创建了一个新的对象（序列化该对象，然后反序列化回来）**

Java 8的接口新增了哪些特性？

**默认方法: 类似之前写的普通实例方法,必须用default修饰, 需要用接口的实现类的对象来调用**

**静态方法: 必须用static修饰，用接口名来调用**

**私有方法：必须使用private修饰，只能在本类中被默认方法或者私有方法访问(从JDK 1.9才开始有的。)**

同步和异步

**同步代表需要等到前一个任务执行完毕之后，才能进行下一个任务。**

**异步代表不同的任务之间并不会相互等待，在执行任务A的时候，也可以同时执行任务B**

**需要等待结果返回，才能继续运行就是同步**

**不需要等待结果返回，就能继续运行就是异步**

优先队列和普通队列的区别

**在队列中，执行先进先出规则，而在优先级队列中，根据优先级删除值。首先删除具有最高优先级的元素。**

JDK新特性

私有接口方法

增强的switch分支判断

double精确到几位小数

15位

int的范围

-2^31~2^31-1

引用数据类型有哪些

引用数据类型包括：类、接口、数组、枚举、注解，字符串。

枚举的好处

增加代码的可读性。限定用户输入的值。保证安全性,有效性。

.Maven作用域

Compile：编译、运行和测试，

Provided：编译、测试

Runtime：运行、测试

Test：测试

Comparable和Comparator的区别

Comparable是一个内部比较器，重写compareTo()方法。

Comparator是一个外部比较器，实现Comparator接口，重写compare()方法，参数是两个需要对比的对象

二者同时存在，Comparator优先级高

Comparable接口位于java.lang包下，Comparator接口位于java.util包下

Comparator实际应用广

集合扩容

需要将原来数组中的数据复制到新数组里。新的容量为旧的容量的1.5倍。ArrayList初始化大小为10，扩容因子为1.5

hashmap扩容

HashMap初始化大小为16，扩容因子为0.75，当hashmap中的元素个数超过数组大小 \* 扩容因子loadFactor(0.75)时，就会进行数组扩容，loadFactor的默认值为0.75，也就是说，默认情况下，数组大小为16，那么当hashmap中元素个数超过16\*0.75=12的时候，把数组的大小扩展为原来的2倍。2\*16=32，即扩大一倍。HashMap 的容量必须是2的N次方

NGINX

（多台服务器时统一暴露，负载均衡）

[java8](https://so.csdn.net/so/search?q=java8&spm=1001.2101.3001.7020)新特性

1、Lambda:表达式- - - - - Lambda允许把函数作为一个方法的参数

2、方法引用- - - - - 可以直接引用已有的java类或对象(实例)的方法或构造器

3、默认方法- - - - - 接口中一个实现的方法

4、Stream Api - - - - - 新添加的stream API(java.util.stream)，把真正的函数式编程风格引入到Java中

5、Optional类 - - - - - 已经成为Java8类库的一部分，解决空指针异常

6、日期时间 API - - - - - 将日期时间的操作类集中到java.time类中

怎么防止超卖

数据库表添加版本号字段，每次更新数据都判断版本号

幂等性方案

业务表唯一索引、乐观锁版本号，分布式锁（审核接口，审核完成后会去调用退款的api接口，然后还插入多个表的数据）常见的时间复杂度量级有：

时间复杂度

算法中循环执行的次数就是算法的时间复杂度

常数阶$O(1)$

线性阶$O(n)$

平方阶$O(n^2)$

立方阶$O(n^3)$

对数阶$O(logn)$

线性对数阶$O(nlogn)$

指数阶$O(2^n)$

设计模式6大原则讲一下

开闭原则：对增加开放，对修改关闭。

单一职责原则：一个类只负责一项职责，避免一个类过度的把一些“功能”耦合在一起

接口隔离原则：接口细化，接口中的方法一定要少，切忌接口臃肿化。

迪米特原则：降低类之间的耦合。一个对象应该对其他实体依赖越少越好，核心在于类间的解耦，只有弱耦合之后类的复用率才会提高。

依赖倒置原则：面向接口编程。每个类尽量提供接口或抽象类。变量的声明类型尽量是接口或抽象类。接口就是一种抽象，只要不修改接口声明，大家可以放心大胆调用，至于接口的内部实现则无需关心，可以随便重构

里氏替换原则：子类尽量不要去重写父类的方法

Java都有哪些锁

公平锁，指多个线程按照申请锁的顺序获得锁。

非公平锁

可重入锁，同一个线程可无限次地进入同一把锁的不同代码

独享锁是指该锁一次只能被一个线程所持有，

共享锁可以被多个线程所持有

读写锁，读写锁的特点是：读读不互斥、读写互斥、写写互斥。

悲观锁，假定并发冲突会发生，通过在访问数据前加锁（如synchronized或ReentrantLock）确保独占操作的并发控制机制，适用于高竞争或强一致性场景。

乐观锁，一种通过版本号或CAS机制在数据更新时检测冲突的无锁并发控制方法，适用于低竞争的高并发场景。

自旋锁，线程不会阻塞，采用循环的方式去尝试获取锁，当循环条件被其他线程改变时，才能进入临界区。这样的好处是减少线程上下文切换的消耗，缺点是会消耗CPU

生产问题排查，举例说明（CPU飙升）

生产环境有相关监控，监控cpu、内存等信息，当过高时就会发送告警邮件

比如内存使用过高导致cpu满载

1. 通过top命令查看cpu使用率高的进程的pid
2. 通过ps -mp命令查看这个pid下的线程占用cpu情况
3. 用jstack + pid和tid查看线程的运行状态，可见信息很齐全，有包名，类名，方法名，当前运行的代码行

Java如何进行高效的数组拷贝?

Arrays.copyOf或 System.arraycopy，是自己new数组, 然后for循环复制效率的两倍左右

为什么快，因为它们是native方法

java中值传递和引用传递的区别是什么

值传递会创建副本，而引用传递不创建副本。引用传递，在函数中对参数所进行的修改，将影响到实际参数。

Java是编译型语言还是解释型语言

编译型的。因为所有的Java代码都需要经过javac编译为.class文件。

解释型。.class文件不能直接运行在操作系统上，还需要经过jvm解释为二进制代码

osi模型的七个层次

osi模型的七个层次：物理层、数据链路层、网络层、传输层、会话层、表示层和应用层

物理链路、网络传输、会话表示应用

HTTP协议属于应用层

TCP三次握手

首先A向B发SYN（同步请求），然后B回复SYN+ACK（同步请求应答），最后A回复ACK确认，这样TCP的一次连接（三次握手）的过程就建立了！三次握手保证双方都具有接受和发送数据的能力

单例如何实现线程安全：加synchronized

equals 和 hashCode 的区别

hashCode()：计算出对象实例的哈希码，并返回哈希码，又称为散列函数。equals()：比较两个对象的内存地址或者内容是否相等

两者通常需要一起重写，以确保一致性。

hashCode 与 equals (重要)

1.hashCode是为了提高在散列结构存储中查找的效率，在线性表中没有作用。

2.equals和hashCode需要同时覆盖。

3.若两个对象equals返回true，则hashCode有必要也返回相同的int数。

4.若两个对象equals返回false，则hashCode不一定返回不同的int数,但为不相等的对象生成不同hashCode值可以提高哈希表的性能。

5.若两个对象hashCode返回相同int数，则equals不一定返回true。

6.若两个对象hashCode返回不同int数，则equals一定返回false。

7.同一对象在执行期间若已经存储在集合中，则不能修改影响hashCode值的相关信息，否则会导致内存泄露问题。

两个对象equals相等，则它们的hashcode必须相等，反之则不一定。

Java两大规则。

规则一：如果 equals 比较对象的内容相等，那么 HashCode 一定相等。（根据官方文档得出）

规则二：如果 equals 比较对象的内容不相等，那么 HashCode 可能相等，也可能不相等。（根据官方文档得出）

为什么重写equals方法，还必须要重写hashcode方法

如果重写了equals方法，而没有重写hashcode方法，会出现equals相等的对象，hashcode不相等的情况，重写hashcode方法就是为了避免这种情况的出现。

TCP

TCP（传输控制协议，Transmission Control Protocol）是一种面向连接的、可靠的、基于字节流的传输层通信协议。以下是其核心特点：

1. 面向连接：通信前需通过“三次握手”建立连接，通信结束后通过“四次挥手”断开连接。

2.可靠性：通过确认机制、重传机制、校验和等方式确保数据准确无误地传输

5. 全双工通信：允许双方同时发送和接收数据。

UDP

无连接、不可靠传输、效率高、可能丢包或乱序、适用于视频流

在分布式系统中，多个服务实例同时运行定时任务时，可能会出现重复执行的问题。为了保证同一时间只有一个定时任务在执行

1. 分布式锁

使用分布式锁可以确保同一时间只有一个实例能够获取锁并执行任务。常见的分布式锁实现有：Redis 分布式锁，使用 Redis 的 `SETNX` 命令实现锁。

使用Redisson，内置了分布式锁的自动续期功能。

2. 数据库唯一约束

通过数据库的唯一约束来确保同一时间只有一个实例能够执行任务。可以在数据库中创建一个表，用于记录任务执行状态。

#### 示例：

```sql

CREATE TABLE scheduled\_tasks (

task\_name VARCHAR(255) PRIMARY KEY,

locked\_by VARCHAR(255),

locked\_at TIMESTAMP

);

在任务执行前，尝试插入一条记录，如果插入成功，则执行任务；如果插入失败（由于唯一约束），则说明其他实例已经在执行任务。

事务不关闭会有什么问题

事务必须显式关闭（提交或回滚），否则会导致资源泄露、数据不一致和性能问题

cleancode整洁之道

1. 命名规范

2. 方法精简，50行以内，一个方法只做一件事

3. 注释精简，用代码表达意图，而非依赖注释

4. 代码格式规范

5. 错误处理，优先处理异常，而非忽略

6. 面向对象，遵循SOLID原则

7. DRY原则（Don't Repeat Yourself），消除重复

8. 类的规模应尽量小，高内聚低耦合

9. 代码坏味道识别，长参数列表，魔法数字，过度设计(避免不必要的模式)

内存泄漏场景

1. 静态集合（如`static HashMap`、`List`）长期持有对象引用，导致对象无法释放，解决：使用弱引用
2. 未关闭资源（如文件、数据库连接），导致内存和资源占用，未调用`close()`释放资源（如`InputStream`、`Connection`），
3. 线程池中复用线程时，未清理ThreadLocal数据，导致旧数据残留。解决：使用`ThreadLocal.remove()`及时清理数据。

JVM内建的指令

1. jps：列出当前系统中所有的 Java 进程及其主类名和 JVM 参数。

2. jstack：生成 Java 进程的线程堆栈快照。

3. jmap：生成 Java 进程的内存映射信息或堆转储文件（Heap Dump）。

4. jstat：实时监控 JVM 的统计信息，如垃圾回收、类加载、编译情况等。

5. jhat：分析堆转储文件（Heap Dump），提供 Web 界面查看内存中的对象信息。

1.什么是反射？反射的作用是什么？

反射是 Java 在运行时动态获取类的信息（类名、方法、属性、构造器等）并操作类对象的能力。

作用：动态加载类、实现通用框架（如 Spring 的依赖注入）、调用私有方法/属性等。

应用场景：框架设计

2. 如何获取一个类的 Class 对象？

- `类名.class`（如 `String.class`）。

- `对象.getClass()`（如 `"abc".getClass()`）。

- `Class.forName("全限定类名")`（如 `Class.forName("java.lang.String")`）。

3. 反射能否访问私有方法或属性？如何实现？

- 可以。通过 `setAccessible(true)` 关闭安全检查：

Field field = clazz.getDeclaredField("privateField");

field.setAccessible(true);

Object value = field.get(obj);

4. 反射的优缺点是什么？

- 优点：灵活性强，支持动态编程。

- 缺点：性能较低（安全检查开销）、破坏封装性、代码可维护性差。

5. 动态代理的实现方式？

基于接口：`java.lang.reflect.Proxy` + `InvocationHandler`。

基于类：第三方库（如 CGLib）。

6. Class 类中的 `getMethod` 和`getDeclaredMethod` 区别？

`getMethod`：获取类及其父类的 public 方法。

`getDeclaredMethod`：获取类自身声明的所有方法（包括私有），不包含父类。

1. 字节流和字符流的区别？

- 字节流（`InputStream`/`OutputStream`）：以字节为单位操作，适合所有类型文件（如图片、视频）。

- 字符流（`Reader`/`Writer`）：以字符为单位，自动处理编码，适合文本文件。

2. 常见的 IO 流类有哪些？

字节流：`FileInputStream`、`BufferedInputStream`、`ObjectInputStream`。

字符流：`FileReader`、`BufferedReader`、`InputStreamReader`。

3. NIO 和传统 IO 的区别？

- 传统 IO 是阻塞式（BIO），基于流（Stream）。

- NIO 是非阻塞式，基于通道（Channel）和缓冲区（Buffer），支持选择器（Selector）实现多路复用。

4. 什么是零拷贝（Zero Copy）？

- 通过 `FileChannel.transferTo()` 或内存映射（MappedByteBuffer）直接在内核态完成文件传输，减少数据拷贝次数，提升性能。

5. 如何实现大文件的高效读写？

- 使用缓冲流（如 `BufferedInputStream`）或 NIO 的 `FileChannel` + `ByteBuffer`。

6. 序列化和反序列化的作用是什么？如何实现？

- 序列化：将对象转为字节流，用于网络传输或持久化。

- 实现：类实现 `Serializable` 接口，通过 `ObjectOutputStream` 写入。

- 注意：`transient` 修饰的字段不参与序列化。

7. Java 中如何关闭 IO 流？

- 在 `finally` 块中手动关闭，或使用 try-with-resources

URL解析至页面展示精简流程

1. URL解析

- 分解协议、域名、端口、路径；检查HSTS强制HTTPS。

2. DNS查询

- 浏览器缓存 → 系统缓存 → 路由器 → ISP递归查询（根域→顶级域→权威域）→ 获取IP

3. 建立连接

- TCP三次握手：SYN → SYN-ACK → ACK。

- HTTPS：TLS握手（密钥交换+证书验证）。

4. 发送请求

- 浏览器发送HTTP请求（方法+头部），服务器处理（静态资源或动态生成）。

5. 接收响应

- 状态码（200/304等），返回HTML及资源（CSS/JS/图片）。

6. 解析渲染

- 构建DOM：解析HTML生成树结构，遇CSS构建CSSOM，遇JS可能阻塞（除非async/defer）。

- 渲染树：合并DOM+CSSOM，计算布局（Layout），绘制像素（Paint），合成层显示（Composite）。

7. 关闭连接

- TCP四次挥手（FIN → ACK）。

主线程如何捕获子线程异常

thread.setUncaughtExceptionHandler(new Thread.UncaughtExceptionHandler() {

public void uncaughtException(Thread t, Throwable e) {

}

});

completableFuture.exceptionally(throwable -> {

*// 异常处理逻辑*

return null;

});

String为什么用final修饰

final防止子类破坏不可变性

安全性，防止关键字符串被篡改

线程安全，不可变对象天然线程安全

安全性示例：

//假设 String 可变，攻击者可能修改关键路径

String filePath = "/safe/path/secret.txt";

filePath.replace("/safe/", "/hacked/"); // 假设这会原地修改内容

增强for和for的区别

1.普通for循环可以获取元素索引，增强for循环无法获取元素索引

2.增强for循环比普通for循环代码更简洁

数组（Array），两者性能几乎相同，foreach循环会被编译器优化为普通for循环

LinkedList，foreach循环性能显著优于普通for

if else 简化

1.Map + 函数式接口

适用场景：简单的条件映射，适合替换 `switch-case`。

```java

Map<String, Runnable> actionMap = new HashMap<>();

actionMap.put("start", () -> System.out.println("Starting..."));

actionMap.put("stop", () -> System.out.println("Stopping..."));

String command = "start";

actionMap.getOrDefault(command, () -> System.out.println("Unknown command")).run();

1. 责任链模式
2. 卫语句提前返回，适用场景，减少嵌套层次
3. Optional 处理空值

\*\*适用场景\*\*：替换 `if (obj != null)` 的嵌套判断。

if (user != null) {

Address address = user.getAddress();

if (address != null) {

return address.getCity();

}

}

return "Unknown";

// 使用 Optional

return Optional.ofNullable(user)

.map(User::getAddress)

.map(Address::getCity)

.orElse("Unknown");

DTO

DTO（Data Transfer Object），例如在前后端交互、微服务间通信时使用

DO（Domain Object）数据库表对象

BO（Business Object）业务对象，是包含业务逻辑的对象

加密算法

对称加密算法: 加密和解密使用同一个密钥-----单密钥加密DES、 AES  
不对称加密算法：加密和解密使用不同的钥-----（公钥，私钥）---安全性高 RSA哈希算法：MD5 SHA256

数据库存储密码：MD5加密（不可逆性）

https和http区别

http默认端口号80；https默认443

http明文传输；https SSL/TLS加密

JWT结构

Header，描述令牌类型和签名算法

Payload，主体部分，用户身份及额外信息

Signature，验证令牌完整性，防止数据篡改

用户登录后返回JWT，前端存储