### Java 基础

#### 1. JDK 和 JRE 有什么区别？

* JDK：Java Development Kit 的简称，Java 开发工具包，提供了 Java 的开发环境和运行环境。
* JRE：Java Runtime Environment 的简称，Java 运行环境，为 Java 的运行提供了所需环境。

具体来说 JDK 其实包含了 JRE，同时还包含了编译 Java 源码的编译器 Javac，还包含了很多 Java 程序调试和分析的工具。简单来说：如果你需要运行 Java 程序，只需安装 JRE 就可以了，如果你需要编写 Java 程序，需要安装 JDK。

#### 2. == 和 equals 的区别是什么？

**== 解读**

对于基本类型和引用类型 == 的作用效果是不同的，如下所示：

* 基本类型：比较的是值是否相同；
* 引用类型：比较的是引用是否相同；

代码示例：

String x = "string";

String y = "string";

String z = **new** String("string");

System.out.println(x==y); // true

System.out.println(x==z); // false

System.out.println(x.equals(y)); // true

System.out.println(x.equals(z)); // true

代码解读：因为 x 和 y 指向的是同一个引用，所以 == 也是 true，而 new String()方法则重写开辟了内存空间，所以 == 结果为 false，而 equals 比较的一直是值，所以结果都为 true。

**equals 解读**

equals 本质上就是 ==，只不过 String 和 Integer 等重写了 equals 方法，把它变成了值比较。看下面的代码就明白了。

首先来看默认情况下 equals 比较一个有相同值的对象，代码如下：

**class** **Cat** {

**public** **Cat**(String name) {

**this**.name = name;

}

**private** String name;

**public** String **getName**() {

**return** name;

}

**public** **void** **setName**(String name) {

**this**.name = name;

}

}

Cat c1 = **new** Cat("王磊");

Cat c2 = **new** Cat("王磊");

System.out.println(c1.equals(c2)); // false

输出结果出乎我们的意料，竟然是 false？这是怎么回事，看了 equals 源码就知道了，源码如下：

**public** **boolean** **equals**(Object obj) {

**return** (**this** == obj);

}

原来 equals 本质上就是 ==。

那问题来了，两个相同值的 String 对象，为什么返回的是 true？代码如下：

String s1 = **new** String("老王");

String s2 = **new** String("老王");

System.out.println(s1.equals(s2)); // true

同样的，当我们进入 String 的 equals 方法，找到了答案，代码如下：

**public** **boolean** **equals**(Object anObject) {

**if** (**this** == anObject) {

**return** **true**;

}

**if** (anObject **instanceof** String) {

String anotherString = (String)anObject;

**int** n = value.length;

**if** (n == anotherString.value.length) {

**char** v1[] = value;

**char** v2[] = anotherString.value;

**int** i = 0;

**while** (n-- != 0) {

**if** (v1[i] != v2[i])

**return** **false**;

i++;

}

**return** **true**;

}

}

**return** **false**;

}

原来是 String 重写了 Object 的 equals 方法，把引用比较改成了值比较。

**总结** ：== 对于基本类型来说是值比较，对于引用类型来说是比较的是引用；而 equals 默认情况下是引用比较，只是很多类重新了 equals 方法，比如 String、Integer 等把它变成了值比较，所以一般情况下 equals 比较的是值是否相等。

#### 3. 两个对象的 hashCode() 相同，则 equals() 也一定为 true，对吗？

不对，两个对象的 hashCode() 相同，equals() 不一定 true。

代码示例：

String str1 = "通话";

String str2 = "重地";

System. out. println(String. format("str1：%d | str2：%d", str1. hashCode(),str2. hashCode()));

System. out. println(str1. equals(str2));

执行的结果：

str1：1179395 | str2：1179395

false

代码解读：很显然“通话”和“重地”的 hashCode() 相同，然而 equals() 则为 false，因为在散列表中，hashCode() 相等即两个键值对的哈希值相等，然而哈希值相等，并不一定能得出键值对相等。

#### 4. 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，初始化之后值就不能被修改。

#### 5. Java 中的 Math. round(-1. 5) 等于多少？

等于 -1，因为在数轴上取值时，中间值（0.5）向右取整，所以正 0.5 是往上取整，负 0.5 是直接舍弃。

#### 6. String 属于基础的数据类型吗？

String 不属于基础类型，基础类型有 8 种：byte、boolean、char、short、int、float、long、double，而 String 属于对象。

#### 7. 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 和 StringBuffer、StringBuilder 的区别在于 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，所以在经常改变字符串内容的情况下最好不要使用 String。

StringBuffer 和 StringBuilder 最大的区别在于，StringBuffer 是线程安全的，而 StringBuilder 是非线程安全的，但 StringBuilder 的性能却高于 StringBuffer，所以在单线程环境下推荐使用 StringBuilder，多线程环境下推荐使用 StringBuffer。

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

不一样，因为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") 则会被分到堆内存中。

#### 9. 如何将字符串反转？

使用 StringBuilder 或者 stringBuffer 的 reverse() 方法。

示例代码：

// StringBuffer reverse

StringBuffer stringBuffer = **new** StringBuffer();

stringBuffer. append("abcdefg");

System. out. println(stringBuffer. reverse()); // gfedcba// StringBuilder reverse

StringBuilder stringBuilder = **new** StringBuilder();

stringBuilder. append("abcdefg");

System. out. println(stringBuilder. reverse()); // gfedcba

#### 10. 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。

#### 11. 抽象类必须要有抽象方法吗？

不需要，抽象类不一定非要有抽象方法。

示例代码：

**abstract** **class** **Cat** {

**public** **static** **void** **sayHi**() {

System. out. println("hi~");

}

}

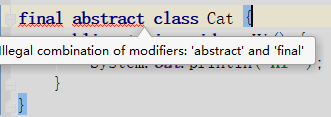
上面代码，抽象类并没有抽象方法但完全可以正常运行。

#### 12. 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。
* 抽象类不能直接实例化，普通类可以直接实例化。

#### 13. 抽象类能使用 final 修饰吗？

不能，定义抽象类就是让其他类继承的，如果定义为 final 该类就不能被继承，这样彼此就会产生矛盾，所以 final 不能修饰抽象类，如下图所示，编辑器也会提示错误信息：



#### 14. 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；接口必须使用 implements 来实现接口。
* 构造函数：抽象类可以有构造函数；接口不能有。
* 实现数量：类可以实现很多个接口；但是只能继承一个抽象类。
* 访问修饰符：接口中的方法默认使用 public 修饰；抽象类中的方法可以是任意访问修饰符。

#### 15. Java 中 IO 流分为几种？

按功能来分：输入流（input）、输出流（output）。

按类型来分：字节流和字符流。

字节流和字符流的区别是：字节流按 8 位传输以字节为单位输入输出数据，字符流按 16 位传输以字符为单位输入输出数据。

#### 16. 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。

### 容器

#### 17. Java 容器都有哪些？

Java 容器分为 Collection 和 Map 两大类，其下又有很多子类，如下所示：

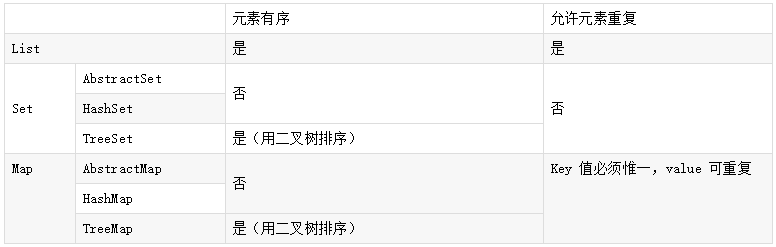
* Collection
* List
  + ArrayList
  + LinkedList
  + Vector
  + Stack
* Set
  + HashSet
  + LinkedHashSet
  + TreeSet
* Map
* HashMap
  + LinkedHashMap
* TreeMap
* ConcurrentHashMap
* Hashtable

#### 18. Collection 和 Collections 有什么区别？

* Collection 是一个集合接口，它提供了对集合对象进行基本操作的通用接口方法，所有集合都是它的子类，比如 List、Set 等。
* Collections 是一个包装类，包含了很多静态方法，不能被实例化，就像一个工具类，比如提供的排序方法： Collections. sort(list)。

#### 19. List、Set、Map 之间的区别是什么？

List、Set、Map 的区别主要体现在两个方面：元素是否有序、是否允许元素重复。



#### 20. HashMap 和 Hashtable 有什么区别？

* 存储：HashMap 运行 key 和 value 为 null，而 Hashtable 不允许。
* 线程安全：Hashtable 是线程安全的，而 HashMap 是非线程安全的。
* 推荐使用：在 Hashtable 的类注释可以看到，Hashtable 是保留类不建议使用，推荐在单线程环境下使用 HashMap 替代，如果需要多线程使用则用 ConcurrentHashMap 替代。

#### 21. 如何决定使用 HashMap 还是 TreeMap？

对于在 Map 中插入、删除、定位一个元素这类操作，HashMap 是最好的选择，因为相对而言 HashMap 的插入会更快，但如果你要对一个 key 集合进行有序的遍历，那 TreeMap 是更好的选择。

#### 22. 说一下 HashMap 的实现原理？

HashMap 基于 Hash 算法实现的，我们通过 put(key,value)存储，get(key)来获取。当传入 key 时，HashMap 会根据 key. hashCode() 计算出 hash 值，根据 hash 值将 value 保存在 bucket 里。当计算出的 hash 值相同时，我们称之为 hash 冲突，HashMap 的做法是用链表和红黑树存储相同 hash 值的 value。当 hash 冲突的个数比较少时，使用链表否则使用红黑树。

#### 23. 说一下 HashSet 的实现原理？

HashSet 是基于 HashMap 实现的，HashSet 底层使用 HashMap 来保存所有元素，因此 HashSet 的实现比较简单，相关 HashSet 的操作，基本上都是直接调用底层 HashMap 的相关方法来完成，HashSet 不允许重复的值。

#### 24. ArrayList 和 LinkedList 的区别是什么？

* 数据结构实现：ArrayList 是动态数组的数据结构实现，而 LinkedList 是双向链表的数据结构实现。
* 随机访问效率：ArrayList 比 LinkedList 在随机访问的时候效率要高，因为 LinkedList 是线性的数据存储方式，所以需要移动指针从前往后依次查找。
* 增加和删除效率：在非首尾的增加和删除操作，LinkedList 要比 ArrayList 效率要高，因为 ArrayList 增删操作要影响数组内的其他数据的下标。

综合来说，在需要频繁读取集合中的元素时，更推荐使用 ArrayList，而在插入和删除操作较多时，更推荐使用 LinkedList。

#### 25. 如何实现数组和 List 之间的转换？

* 数组转 List：使用 Arrays. asList(array) 进行转换。
* List 转数组：使用 List 自带的 toArray() 方法。

代码示例：

// list to array

List<String> list = **new** ArrayList<String>();

list. add("王磊");

list. add("的博客");

list. toArray();// array to list

String[] array = **new** String[]{"王磊","的博客"};

Arrays. asList(array);

#### 26. Iterator 和 ListIterator 有什么区别？

* Iterator 可以遍历 Set 和 List 集合，而 ListIterator 只能遍历 List。
* Iterator 只能单向遍历，而 ListIterator 可以双向遍历（向前/后遍历）。
* ListIterator 从 Iterator 接口继承，然后添加了一些额外的功能，比如添加一个元素、替换一个元素、获取前面或后面元素的索引位置。

### 反射

#### 27. 什么是反射？

反射是在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法；对于任意一个对象，都能够调用它的任意一个方法和属性；这种动态获取的信息以及动态调用对象的方法的功能称为 Java 语言的反射机制。

#### 28. 什么是 Java 序列化？什么情况下需要序列化？

Java 序列化是为了保存各种对象在内存中的状态，并且可以把保存的对象状态再读出来。

以下情况需要使用 Java 序列化：

* 想把的内存中的对象状态保存到一个文件中或者数据库中时候；
* 想用套接字在网络上传送对象的时候；
* 想通过RMI（远程方法调用）传输对象的时候。

#### 29. 动态代理是什么？有哪些应用？

动态代理是运行时动态生成代理类。

动态代理的应用有 spring aop、hibernate 数据查询、测试框架的后端 mock、rpc，Java注解对象获取等。

#### 30. 怎么实现动态代理？

JDK 原生动态代理和 cglib 动态代理。JDK 原生动态代理是基于接口实现的，而 cglib 是基于继承当前类的子类实现的。

### 异常

#### 31. throw 和 throws 的区别？

* throw：是真实抛出一个异常。
* throws：是声明可能会抛出一个异常。

#### 32. final、finally、finalize 有什么区别？

* final：是修饰符，如果修饰类，此类不能被继承；如果修饰方法和变量，则表示此方法和此变量不能在被改变，只能使用。
* finally：是 try{} catch{} finally{} 最后一部分，表示不论发生任何情况都会执行，finally 部分可以省略，但如果 finally 部分存在，则一定会执行 finally 里面的代码。
* finalize： 是 Object 类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法。

#### 33. try-catch-finally 中哪个部分可以省略？

try-catch-finally 其中 catch 和 finally 都可以被省略，但是不能同时省略，也就是说有 try 的时候，必须后面跟一个 catch 或者 finally。

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

finally 一定会执行，即使是 catch 中 return 了，catch 中的 return 会等 finally 中的代码执行完之后，才会执行。

#### 35. 常见的异常类有哪些？

* NullPointerException 空指针异常
* ClassNotFoundException 指定类不存在
* NumberFormatException 字符串转换为数字异常
* IndexOutOfBoundsException 数组下标越界异常
* ClassCastException 数据类型转换异常
* FileNotFoundException 文件未找到异常
* NoSuchMethodException 方法不存在异常
* IOException IO 异常
* SocketException Socket 异常