Java基础面试题总结

H6 八大基本数据类型:

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

H4 1，面向对象和面向过程的区别

- 面向过程
  - 优点：性能高，节省资源。
  - 缺点：不利于维护，复用和扩展
- 面向对象
  - 优点：易维护、易复用、易扩展。低耦合。
  - 缺点： 性能比面向过程低。

H4 2.，Java 语言有哪些特点

1. 简单易学 面向对象 移植性好
2. 可靠性 安全性
3. 多线程 网络编程 编译与解释并存

H4 3，关于 JVM JDK 和 JRE 最详细通俗的解答

- JRE：是JAVA的运行时环境
- JDK ：是java开发工具包，包含 jre ，编译器和工具， 他能够创建和编译程序。
- jvm：是运行JAVA字节码的虚拟机。不同的系统采用相同的字节码，这样就让移植性大大提高。

使用字节码的好处：

1. 字节码只面向虚拟机，在一定程度上解决了传统解释型语言执行效率低的问题
2. 使Java程序能够在不同的机器上运行。
3. 为什么引进JIT编译器可以让运行速度更快?

> 注： 热点代码：经常需要被调用的方法和代码块
>
> - 因为在： .class->机器码 这一步 jvm 类加载器 会首先 加载 字节码文件 ，然后通过解释器 **逐行** 解释执行，这种方式的执行速度会相对 比较慢。
> - 当引进了 JIT 编译器，(*运行时编译*)。当 JIT 编译器完成第一次编译后，其会将字节码对应的机器码保存下来，下次可以直接使用。而我们知道，机器码的运行效率肯定是高于Java 解释器的。这也解释了我们为什么经常会说 Java 是编译与解释共存的语言

- JIT编译器只需要编译的是热点代码，jvm 会根据每次的运行情况来进行一些优化，因此运行次数越多，执行速度越快
- JDK 9 引入了一种新的编译模式： AOT(Ahead of Time Compilation)，它是直接将字节码编译成机器码，这样就 避免了 JIT 预热等各方面的开销。JDK 支持分层编译和 AOT 协作使用。但是 ， AOT 编译器的编译质量是肯定比不上 JIT 编译器的。

4, Oracle JDK 和 OpenJDK 的对比

1. Oracle JDK 版本将每三年发布一次，而 OpenJDK 版本每三个月发布一 次；
2. OpenJDK 是一个参考模型并且是完全开源的，而 Oracle JDK 是 OpenJDK 的一个实现，并不是完全开源的；
3. Oracle JDK 比 OpenJDK 更稳定。OpenJDK 和 Oracle JDK 的代码几乎 相同，但 Oracle JDK 有更多的类和一些错误修复。因此，如果您想开发 企业/商业软件，我建议您选择 Oracle JDK，因为它经过了彻底的测试和 稳定。某些情况下，有些人提到在使用 OpenJDK 可能会遇到了许多应 用程序崩溃的问题，但是，只需切换到 Oracle JDK 就可以解决问题；
4. 顶级公司正在使用 Oracle JDK，例如 Android Studio，Minecraft 和 IntelliJ IDEA 开发工具，其中 Open JDK 不太受欢迎；
5. 在响应性和 JVM 性能方面，Oracle JDK 与 OpenJDK 相比提供了更好的 性能；
6. Oracle JDK 不会为即将发布的版本提供长期支持，用户每次都必须通过 更新到最新版本获得支持来获取最新版本；
7. Oracle JDK 根据二进制代码许可协议获得许可，而 OpenJDK 根据 GPL v2 许可获得许可

5, Java 和 C++ 的区别

- 都是面向对象的语言，都支持封装、继承和多态
- Java 不提供指针来直接访问内存，程序内存更加安全
- Java 的类是单继承的，C++ 支持多重继承；虽然 Java 的类不可以多 继承，但是接口可以多继承。
- Java 有自动内存管理机制，不需要程序员手动释放无用内存

6, 什么是 Java 程序的主类？应用程序和小程序的主类有何不同？

主类：主类是 Java 程序执行的入口点。一个程序中可以有多个类，但只能有一个类是主类。

- 在 Java 应用程序中，这 个主类是指包含 main（）方法的类。
- 而在 Java 小程序中，这个主类是一个继承自系统类 JApplet 或 Applet 的子类 。
- 应用程序的主类不一定要求是public 类，但小程序的主类要求必须是 public 类。

7，Java 应用程序与小程序之间有那些差别

- 应用程序 是从主线程启动(也就是 main() 方法)。
- applet 小程序 没有 main 方法，主要是嵌在浏览器页面上运行(调用 init()线程 或者 run() 来启动

8， 字符型常量和字符串常量的区别

- 字符型常量：形式上是由单引号引起来的一个字符。相当于一个整形值(ASCII 值），可以参加运算，占两个字节
- 字符串常量：形式上是由双引号引起来的若干个字符，代表的是一个内存地址值，占若干个字节，至少一个 字符为结束标志。

9， 构造器 Constructor 是否可被 override?

- 因为父类的私有属性和构造方法并不能被继承，所以 Constructor 也就不能被 override（重写），
- 但是可以 overload（重载）,所以你可以看到一个类中有多个构造函数的情况（例如：无参构造器，和全参构造器）。

10， 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
  - 方法返回值和访问修饰符可以不同，发生在编译时。
- 重写： 发生在父子类中（**继承中**）
  - 方法名、参数列表必须相同 ，
  - 子类返回值范围 <= 父类 ，
  - 抛出的异常范围小于等于父类 ，
  - 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 则子类就不能重写该方法。

11，Java 面向对象编程三大特性: 封装 继承 多态

- **封装** ：封装就是把一个对象的属性私有化，同时提供一些可以被外界访问的属性的方法
  - 注意： 如果属性不想被外界访问，我们大可不必提供方法给外界访问。但是如果一个类没有提供给外界访问的方法，那么这个类也没有什么意义了。
- **继承** ：继承是 使用已存在的类的定义作为基础建立新类的技术，
  - 新类的定义可以增加新的数据或新的功能，也可以用父类的功能，但不能选择性地继承父类。
  - 通过使用继承我们能够非常方便地 复用 以前的代码。
  - 关于继承的三个注意点：

    1. 子类拥有父类 非私有的属性和方法。
    2. 子类 可以拥有自己属性和方法 ，即子类可以对父类进行 扩展 。

3. 子类可以用自己的方式实现父类的方法.（重写）

- **多态** 多态就是指程序中定义的 引用变量所指向的具体类型 和 通过该引用变量发出的方法调用 在编译时并不确定，而是在程序运行期间才确定
  - 即一个引用变量到底会指向哪个类的实例对象，该引用变量发出的方法调用到底是哪个类中实现的方法，必须在由程序运行期间才能决定。
  - 在Java中有两种形式可以实现多态：
    - 继承（多个子类对同一方法的重写）
    - 接口（实现接口并覆盖接口中同一方法）

H4 12，String StringBuffer 和 StringBuilder 的区别是什么 String 为什么是不可变的

- 可变性：
  - String : **private final char value[]** 因为 final 修饰的变量 --- 不可以变
  - StringBuffer 和 StringBuilder : **char[] value** --- 可以变
- 线程安全：
  - String 和 StringBuffer 都是线程安全的，String是常量,StringBuffer 加了同步锁
  - StringBuilder 不是线程安全的， 因为没有锁
- 性能
  - 每次对 String 类型进行改变的时候，都会生成一个新的 String 对象，然后将指针指向新的 String 对象。
  - StringBuffer 每次都会对 StringBuffer 对象本身进行操作，而不是生成新的对象并改变对象引用。
  - 相同情况下使用 StringBuilder 相比使用 StringBuffer 仅能获得 10%~15% 左右的性能提升，但却要冒多线程不安全的风险。

对于三者使用的总结：

1. 操作少量的数据 = String
2. 单线程操作 -- 字符串缓冲区下 操作大量数据 = StringBuilder
3. 多线程操作 -- 字符串缓冲区下 操作大量数据 = StringBuffer

H4 13，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

H4 14，在一个静态方法内调用一个非静态成员为什么是非法的?

- 静态方法是在 类加载 的时候加载到内存中的，而非静态成员是在 实例化 的时候加载到内存中
- 因此调用时，非静态成员还 不存在 ，因此无法调用。

H4 15，在 Java 中定义一个不做事且没有参数的构造方法的作用?

- Java 程序在执行子类的构造方法之前，如果没有用 super() 来调用父类特定的构造方法，则会调用父类中的 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() 来调用父类中特定的构造方法，则编译时将发生错误。

H4 16， import java 和 javax 有什么区别

- 实际上 java 和 javax 没有区别。这只是一个名字而已。
- 刚开始的时候 JavaAPI 所必需的包是 java 开头的包， javax 当时只是 扩展 API 包 来使用。
- 然而随着时间的推移， javax 逐渐的扩展成为 Java API 的组成部分 。
- 但是， 将扩展从 javax 包移动到 java 包将是太麻烦了 ，最终会破坏一堆现有的代码。
- 因此，最终 决定将 javax 包成为标准API的一部分 。

H4 17， 接口和抽象类的区别是什么

- 接口的方法默认是 public，所有方法在接口中不能有实现(Java 8 开始接口方法可以有默认实现），抽象类可以有非抽象的方法
- 接口中的实例变量默认是 final 类型的，而抽象类中则不一定
- 一个类可以实现多个接口，但最多只能实现一个抽象类
- 一个类实现接口的话要实现接口的所有方法，而抽象类不一定
- 接口不能用 new 实例化，但可以声明，但是必须引用一个实现该接口的对象
- 从设计层面来说，抽象是对类的抽象，是一种模板设计，接口是行为的抽象，是一种行为的规范

H4 18，成员变量与局部变量的区别有那些

1. 从语法形式上看，成员变量是属于类的，而局部变量是在方法中定义的变量或是方法的参数;
2. 成员变量可以被 public,private,static 等修饰符所修饰，而局部变量不能被访问控制修饰符及 static 所修饰；但是，成员变量和局部变量都能被 final 所修饰;
3. 从变量在 内存中的存储方式 来看，成员变量是对象的一部分，而对象存在于 堆内存 ，局部变量存在于 栈内存
4. 从变量 在内存中的生存时间 上看，成员变量是对象的一部分，它随着对象的创建而存在，而局部变量随着方法的调用而自动消失。

5. 成员变量如果没有被赋初值，则会自动以类型的默认值而赋值（一种情况例外被 final 修饰的成员变量也必须显示地赋值）；而局部变量则不会自动赋值。

19，创建一个对象用什么运算符？对象实体与对象引用有何不同？

- new 运算符，new创建对象实例（对象实例在 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，但是一个对象可以有 n 个引用指向它

20，什么是方法的返回值?返回值在类的方法里的作用是什么？

- 方法的返回值 ：是指我们获取到的某个方法体中的代码执行后产生的结果！（前提是该方法可能产生结果）。返回值的作用:接收出结果，使得它可以用于其他的操作!

21，一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,该程序能正确执行吗？为什么？

- 主要作用 是完成对类 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。

22，构造方法有哪些特性

1. 名字与类名相同；
2. 没有返回值，但 不能用void 声明构造函数；
3. 生成类的对象时自动执行，无需调用。

23，静态方法和实例方法有何不同

- 在外部调用静态方法时，可以使用 "类名.方法名" 的方式，也可以使用 "对象名.方法名" 的方式。而实例方法只有后面这种方式。-------- > 调用静态方法可以无需创建对象。
- 静态方法在访问本类的成员时，只允许访问静态成员（即静态成员变量和静态方法），而不允许访问实例成员变量和实例方法；实例方法则无此限制.

24，对象的相等与指向他们的引用相等，两者有什么不同？

- 对象的相等 ，比的是 内存中存放的内容 是否相等。
- 而 引用相等 ，比的是 他们指向的内存地址 是否相等。

25, 在调用子类构造方法之前会先调用父类没有参数的构造方法，其目的是?

- 帮助子类做初始化工作

26, == 与 equals (重要)

- `==` ：基本数据类型比较的是 值 ，引用数据类型比较的是 内存地址
- `equals()` ：它的作用也是判断两个对象是否相等。
- 但它一般有两种使用情况：
  - 情况1：类没有覆盖(重写) equals() 方法。则通过 equals() 比较该类的两个对象时，等价于通过“==”比较这两个对象。
  - 情况2：类覆盖了 equals() 方法。一般，我们都覆盖 equals() 方法来比较两个对象的内容相等；若它们的内容相等，则返回 true (即，认为这两个对象相等)。

27，hashCode 与 equals (重要)

面试官可能会问你：“你重写过 hashcode 和 equals 么，为什么重写 equals 时必须重写 hashCode 方法？”

一,hashcode() 的性质:

对象的散列码．他只能保证离散，不能保证唯一．

也就是说：

- 如果两个对象相等,他们的 hashcode() 一定相等．
- 但是 hashcode() 相等，这两个对象不一定相等；

hashcode() 方法存在的意义是：提高两个对象 比较的效率；

当两个对象进行比较的时候：先比较 hashcode()，如果hashcode() 不相等，说明这两个对象一定不相等，

如果hashcode() 相等 才会 使用 equals() 方法 比较对象的的 值．

- 也就是说：只有当 hashcode() 相等的时候，才能调用 equals()；

如果只重写 equals() 不重写 hashcode()

Object类的 hashcode()方法 无法保证 两个相等对象的 hashcode() 相等，这样也就无法 equals()；

如果只重写 hashcode()，不重写equals()

因为 hashcode() 无法保证唯一性，所以两个不相等的对象,他们的hashcode() 是有可能相等的．

如果不重写equals()，无法比较他们是不是真的相等．

结论:

所以建议两个方法同时重写.

`hashcode()` 是为了提高比较效率的，因为equals() 方法还是比较耗费效率的.

`equals()` 是为了在 hashcode() 无法区分两个对象是否相等的时候，用来比较他们时候真的相等的手段.

28，简述线程，程序、进程的基本概念。以及他们之间关系是什么

- 线程：一个进程可以包括多个线程，他是比进程更小的执行单位，一个进程有独立的内存空间，但是多个线程共享相同的内存空间，因此在多个线程之间切换要比在多个进程之间切换快
- 程序：是存储 指令和数据 的文件，被存储在硬盘或是其他的数据存储设备中，也就是静态的代码
- 进程：是 程序的一次执行过程 ，是系统运行程序的基本单位，因此进程是动态的。一个 进程 就是一个 执行中的程序 。

29, 关于 final 关键字的一些总结

- 修饰变量
  - 基本数据类型，数值一旦初始化后不可被更改
  - 引用类型，初始化后 不可再指向其他的对象
- 修饰类
  - 表明这个类不能被继承。
  - final 类中的所有成员方法都会被隐式地指定为 final 方法。
- 修饰方法 ： 不可被重写。

30, Java序列化中如果有些字段不想进行序列化 怎么办

- 对于不想进行序列化的变量，使用 transient 关键字修饰。
- transient 关键字的作用是：阻止实例中那些用此关键字修饰的的变量序列化；
- 当对象被反序列化时，被 transient 修饰的变量值 不会被持久化和恢复 。
- transient 只能修饰 **变量** ，不能修饰类和方法

31，获取用键盘输入常用的的两种方法

- 方法1：通过 Scanner

```
Scanner input = new Scanner(System.in);
String s = input.nextLine(); input.close();
```

- 方法2：通过 BufferedReader

```
    BufferedReader input = new BufferedReader(new
InputStreamReader(System.in));
    String s = input.readLine();
```

H4 反射的相关方法

- `Class.class` 的形式会使 JVM 将使用类装载器将类装入内存（前提是类还没有装入内存），不做类的初始化工作，返回 `Class` 对象。
- `Class.forName()` 的形式会装入类并做类的静态初始化，返回 `Class` 对象。
- `getClass()` 的形式会对类进行静态初始化、非静态初始化，返回引用运行时真正所指的对象（因为子对象的引用可能会赋给父对象的引用变量中）所属的类的 `Class` 对象。