## Java语言的特点(面向对象、跨平台性、提供了很多内置类库、安全性和健壮性)

面向对象：

跨平台性：Java提供了虚拟机机制，Java编译的程序，通过JVM可以根据当前平台解释成符合当前平台的的机器码，实现同一应用在不同平台上都能运行；

安全性：

Java通常被用在网络环境中，为此，Java 提供了一个安全机制以防恶意代码的攻击。除了Java 语言具有的许多安全特性以外，Java 对通过网络下载的类具有一个安全防范机制（类 ClassLoader），如分配不同的名字空间以防替代本地的同名类、字节代码检查，并提供安全管理机制（类 SecurityManager）让 Java 应用设置安全哨兵。

健壮性：

Java 的强类型机制、异常处理、垃圾的自动收集、安全检查机制等是 Java 程序健壮性的重要保证；

多线程：

## 1.1Java 程序是怎样运行的?(编译-运行-通过JVM初始化)

1. **编译（javac）**

执行 javac xxx.java 把.java文件编译成.class文件

1. **装载字节码(java)**

使用java命令，执行java xxx（.class文件的名称，不带后缀）。java命令将会启动 JVM，并将后面的参数作为初始化类，通过 JVM 内的类加载器将字节码文件装载到内存中。1.2Java类加载器的流程 （加载-验证-准备-解析-初始化）

1. **校验字节码**

类的加载是通过类加载器进行的，加载完后，先由字节码校验器负责检查那些无法执行的明显有破坏性的操作。除了系统类之外，其他类都要被校验。

校验器执行的一些检查：

* 变量要在使用前初始化
* 方法调用与对象引用类型之间要匹配
* 访问私有数据和方法的规则没有被违反
* 对本地变量的访问都在运行的堆栈内
* 运行时堆栈没有溢出

如果这些检查没有通过，类最终不会被加载完成。

1. **解释字节码**

JVM 把每一条要执行的字节码交给解释器，翻译成对应的**机器语言**，最终由操作系统执行。

JVM 解释执行字节码文件就是 JVM 操作 Java 解释器进行解释执行字节码文件的过程。

相关资料：

<https://www.cnblogs.com/ibigboy/p/11184133.html>

<https://www.jianshu.com/p/47ca812ed5d8>

<https://blog.csdn.net/weixin_39638526/article/details/110726141>

## JDK与JRE的区别(是什么?关系?)包含关系

1. JDK（Java Development Kit）即Java程序开发的工具包，是整个Java开发的核心，包含了Java的运行环境（JRE）、Java工具和Java基础的类库。
2. JRE(Java Runtime Enviroment)即Java程序的运行环境包含了虚拟机（JVM）、java类库的Class文件;

## Java的基本数据类型(类型/占多数字节)

|  |  |
| --- | --- |
| 数据类型 | 说明 |
| byte | 1byte |
| short | 2byte |
| int | 4byte |
| long | 8byte |
| float | 4byte |
| double | 8byte |
| char | 2byte |
| boolean | 在Java虚拟机中没有任何供boolean值专用的字节码指令，Java语言表达式所操作的boolean值，在编译之后都使用Java虚拟机中的int数据类型来代替，而boolean数组将会被编码成Java虚拟机的byte数组，每个元素boolean元素占8位。这样我们可以得出boolean类型占了单独使用是4个字节，在数组中是1个字节 |

## 3.1Java 中堆和栈有什么区别？(使用/优势/劣势)

在Java中,[内存](https://so.csdn.net/so/search?q=%E5%86%85%E5%AD%98&spm=1001.2101.3001.7020)分为两种，一种是栈内存，另一种就是堆内存。

堆的作用是用于存储Java中的对象和数组，当我们new一个对象或者创建一个数组的时候，就会在堆内存中开辟一段空间给它，用于存放。

堆其实可以类似的看做是管道，即先进先出，后进后出；堆可以动态地分配内存大小，生存期也不必事先告诉编译器，因为它是在运行时动态分配内存的，但缺点是，由于要在运行时动态分配内存，存取速度较慢。

栈内存是用来执行程序用的，类似水杯，先进入的沉入底部。即先进后出，后进先出；

存储速度比堆块；但缺点是，存在栈中的数据大小与生存期必须是确定的，缺乏灵活性。

## Java中装箱和拆箱

装箱就是将基本数据类型转换为包装器类型；

拆箱就是将包装器类型转换为基本数据类型

## Java访问修饰符(修饰符及作用范围)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 修饰符 | 当前类 | 子类 | 同一包内 | 子孙包 | 不同包 |
| public | 可见 | 可见 | 可见 | 可见 | 可见 |
| protected | 可见 | 可见 | 可见 | 不可见 | 不可见 |
| default（什么都不写） | 可见 | 可见 | 可见 | 不可见 | 不可见 |
| private | 可见 | 不可见 | 不可见 | 不可见 | 不可见 |

private :在同一类内可见;

public :对所有类内可见;

default:在当前类、子类可见、同一包内;

protected:

子类与基类在同一包下：同包下的protected成员变量可见，且对同包子类可见；

子类与基类不在同一包下：子类不能调用基类protected的方法和成员变量;

但是子类实例可以访问其从基类继承而来的protected方法;

## 构造方法、成员变量、静态成员变量三者初始化顺序

1.静态成员变量—》成员变量—》构造方法

## 7.Java代码块执行顺序

(静态代码块（使用static声明）、构造代码块（之间使用｛｝定义）、构造函数、普通代码块(在方法体中))

**静态代码块—构造代码块---构造函数—普通代码块**

<https://www.cnblogs.com/ysocean/p/8194428.html>

## 面向对象的三大特征(封装/继承/多态)

**封装：**

封装就是隐藏对象的属性和实现细节，仅对外公开接口，控制在程序中属性的读和修改的访问级别，将抽象得到的数据和行为（或功能）相结合，**形成一个有机的整体；**

封装的目的是增强安全性和简化编程，使用者不必了解具体的实现细节，而只是要通过外部接口，以特定的访问权限来使用类的成员；

**继承：**

**继承就是子类继承父类的特征和行为**，子类从父类继承方法，使得子类具有父类相同的行为；

**多态：**

多态同一个行为具有多个不同表现形式或形态的能力。是指一个类实例（对象）的相同方法在不同情形有不同表现形式。

多态存在的必要条件：

继承、重写、父类引用指向子类对象

## 为什么Java不支持多继承(总结+原因)

1.简化设计；

## 10.Java的多态(种类及特征)

重载多态和覆盖多态

## 11.重载和覆盖的区别

## 12.接口和抽象类的相同点和不同点(语义和使用)

## 使用:抽象类是为了代码的复用,接口是为了实现多态性;

## \*13.什么是内部内及作用

## 14.Java中static关键字的作用(具体用法)

## 15.为什么要把String设计为不可变的(空间/效率/安全)

## 16.简述下String,Stringbuffer,Stringbuilder(底层实现/线程/是否可变)

## 17.==和equals区别(基本数据类型和引用数据类型)

## 18.简述Object常用方法(equals,split,substr,indexOf...)

## 19.Java中声明一维数组和二维数组的声明方式(语法)

## 20.简述Java异常的分类(结构体系)

## 21.简述下throw 和 throws区别(使用)

## 22.出现在Java程序中的finally代码块是否一定会执行(分情况说明及注意事项)

## 23.final,finally和finalize的区别是什么(使用)

## 24.简述泛型(是什么?解决了什么问题?)

## 25.简述泛型擦除(是什么)

## 26.简述注解(是什么?作用?自定义注解)

## 27.简述元注解(是什么?作用)

## 28.简述Java中的Class对象(是什么)

## 29.Java的反射机制是什么(是什么?有什么作用)

## 30.什么是序列化

## 31.简述序列化和反序列化的实现

## 32.简述Java中的List(子类及特点(底层实现-特点-线程安全性))

## 33.Java中数据安全的基本数据结构

## 34.简述Java中的Set(子类及特点)

## 35.简述Java中的HashMap(底层实现)

## 36.为什么HashMap的线程不安全(底层实现)

## 37.简述Java的TreeMap(底层实现)

## 38.ArrayList,Vector,LinkedList有什么共同点和区别(底层实现-特点-线程安全性)

## 39.HashMap和HashTable有什么区别(底层实现-特点-线程安全性)

## 40.如何决定使用HashMap还是TreeMap(底层原理)

## 41.HashMap中,equals和hashcode之间的关系(基类/作用)

## 42.Collection和Collections类的区别(作用)

## 43.常见的设计模式

## 单例模式:DocumentHelper(DOM4j)|

## 44.xml的解析方式(sax|DOM4j)

## https://developer.aliyun.com/article/599813?spm=a2c6h.17698244.wenzhang.7.73ec7966PLCvDc