## 一、java基础部分

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

面向对象：Java 是面向对象的编程语言，对象就是面向对象程序设计的核心。所谓对象就是真实世界中的实体，对象与实体是一一对应的，也就是说现实世界中每一个实体都是一个对象，它是一种具体的概念；

**跨平台**：Java 语言使用 Java 虚拟机机制屏蔽了具体平台的相关信息，使得 Java 语言编译的程序只需生成虚拟机上的目标代码，就可以在不同平台上不加修改地运行。因此采用 Java 语言编写的程序具有很好的可移植性。

**安全性：**Java技术之所以适用于网络就是因为它具有完备的安全模式；

<https://www.bianchengquan.com/article/235218.html>

**健壮性：**Java 的强类型机制、异常处理、垃圾回收机制等都是 Java 健壮性的重要保证；

**多线程：**Java 支持多个线程同时执行，并提供多线程之间的同步机制。

## 1.1Java 程序是怎样运行的?(编译-运行-通过JVM初始化)

## https://www.jianshu.com/p/47ca812ed5d8

## 1.2Java类加载器的流程 （加载-验证-准备-解析-初始化）

## https://blog.csdn.net/weixin\_39638526/article/details/110726141

## 2.JDK与JRE的区别(是什么?关系?)

## 3.Java的基本数据类型(类型/占多数字节)

## 3.1Java 中堆和栈有什么区别？(使用/优势/劣势)

## 4.Java中装箱和拆箱

## 5.Java访问修饰符(修饰符及作用范围)

## 6.构造方法、成员变量、静态成员变量三者初始化顺序

## 7.Java代码块执行顺序(静态代码块、构造代码块、构造函数、普通代码块(在方法体中))

## https://www.cnblogs.com/ysocean/p/8194428.html

## 8.面向对象的三大特征(封装/继承/多态)

## 9.为什么Java不支持多继承(总结+原因)

## 10.Java的多态(种类及特征)

## 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