1.如何实现类似函数指针的功能

2.组合与继承的区别

3.this域super的区别

4.assert左勇

5.volatile作用-多线程

6.strictfp作用

7.不可变类(immutable class)

8.字符串创建与存储的机制

9.== equals hashcode区别

10.异常处理的原理

11.java IO流的实现机制

12.java socket是啥

13.java NIO

14.序列化

15．GC

16.内存泄漏

17.多线程

使用的原因？如何实现？

多线程同步的实现方法

Sleep wait yield区别

终止线程的方法

守护线程

join作用

18.JDBC处理事务

19.jsp与servlet异同

20.如何用jsp与servlet实现MVC模型

21.ajax是啥

22.javabean EJB？

23.web服务器 web应用服务器

24.web service

25.SOAP 与 REST

26.什么是XML

27.数据库连接池的工作机制

28.spring IOC AOP

29.内连接、外连接

30．数据库事务

存储过程

范式

触发器

游标

视图

30.设计模式

工厂

适配器

观察者

Java中字符只以unicode存在(不选择任何特定编码，直接用它们在字符集中编号)：  
“java中”:JVM中、内存中、代码里声明的每一个char、string类型的变量中

1.i++ ++i

2.0+’0’ 整数+字符串

3.j=j++ java中间缓存变量机制

4.assert

5.main方法可不传参数？

6.三目运算符 右结合性 自动类型提升

运算符优先级

7.32>>32 移位运算 8位一循环，等价于32>>0

8.final finally finalize

package的作用

二．

1.java中传值还是传引用

|  |
| --- |
| 按值传递:当将一个参数传递给一个函数时，函数接收的是原始值的一个副本  按引用传递:当将一个参数传递给一个函数时，函数接收的是原始值的内存地址，而不是值的副本  1.对象就是传引用  2.原始类型就是传值  3.String类型因为没有提供自身修改的函数，每次操作都是新生成一个String对象，所以要特殊对待。可以认为是传值。 |

2.定义在类中的变量会被赋予默认值？static影响？

3.为何静态方法不能访问非静态变量

4. 

5.I/O流

6.序列化 （与持久化关系）

三.

1.递归计算斐波那契数列

2. 



（难）

3．利用1 2 3 4 5这5个·数字，打印出不同的排列，不能有重复

四．java内存管理

1.垃圾收集：优势 收集标准

2.java如何内存管理

3.内存泄漏：解释 引起的原因 哪些情况？

4.clone作用

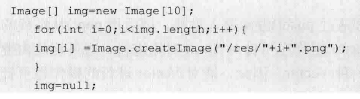
5.泛型与容器

UML

MVC理解:springmvc struts backbone

单例模式写法

5.未释放堆栈 空间原因：



6.栈操作：将1-12月的英文单词压入栈中，再将其取出，Java实现

(1) new stack()

(2)new vector()

7. 链表

8.hash表

9.树

先序 中序 后序遍历

平衡二叉树

10.图

图的深度优先遍历

图的广度优先遍历

哈夫曼编码问题

四叉树

Trie树，单词查找树，字典树

11.排序

稳定性

平均性能，最快的排序算法

有序/无序队列：寻找最小值/估算平均值/找出中间值/找出最大出现的可能性的时间复杂度

(b c d e f g q r s t)查找b，二分查找过程

写冒泡排序：输入10个数，输出排序结果

写选择拍寻：输入10个数，输出排序结果

写插入排序：输入10个数，输出排序结果

写希尔排序：输入10个数，输出排序结果

二分法排序思想？

理解快速排序代码

写归并排序：输入10个数，输出排序结果

辗转相除法，即欧几里得算法，时间复杂度？

数据库索引

1. 常见的5中RuntimeExecption

|  |
| --- |
| ClassCastException(类转换异常)：多态中，可以使用Instanceof 判断，进行规避  ArithmeticException("数学运算异常")：进行if判断，如果除数为0，进行return  NullPointerException：进行if判断，是否为null  ArrayIndexOutOfBoundsException：使用数组length属性，避免越界  IndexOutOfBoundsException(数组越界)  ClassCastException(类型强制转换异常) |

4.1.2

“Java语言中的方法属于类中的成员（member）”：不对

原因：类成员必须静态，而方法可能是静态方法，也可能是非静态方法。

静态方法是类成员，非静态方法是实例成员。

4.1.4 类被加载/类加载时机：静态代码块

|  |
| --- |
| **初始化规定：**   * 创建类的实例 * 访问类的静态变量 (除常量【被final修辞的静态变量】   原因:常量一种特殊的变量，因为编译器把他们当作值(value)而不是域(field)来对待。如果你的代码中用到了常变量(constant variable)，编译器并不会生成字节码来从对象中载入域的值，而是直接把这个值插入到字节码中。这是一种很有用的优化，但是如果你需要改变final域的值那么每一块用到那个域的代码都需要重新编译。   * 访问类的静态方法 * 反射 如( Class.forName(“my.xyz.Test”) ) * 当初始化一个类时，发现其父类还未初始化，则先出发父类的初始化 * 虚拟机启动时，定义了main()方法的那个类先初始化 |

1. 编译即javac过程，将.java文件compile成.class文件，主要是类型、格式检查与编译成字节码文件

加载是指java \*的过程，将.class文件加载到内存中去解释执行，即运行的时候才会有加载一说。

1. 类的加载时机，肯定是在运行时，但并不是一次性全部加载，而是按需动态，依靠反射来实现动态加载，一般来说一个class只会被加载一次，之后就会从jvm的class实例的缓存中获取，谁用谁取就可以了，不会再去文件系统中加载.class文件了。

|  |
| --- |
| 静态代码块在类加载时调用，并且只调用一次。  最常用的作用：给一个对象的属性初始化。  具体的说：  1.当调用一个类的静态变量时，类中的静态代码块会执行。【只有静态代码块会执行】  2.当调用一个类的静态方法时，类中的静态代码块会执行。【只有静态代码块会执行】  3.当创建一个类的实例时，类中的静态代码块、非静态代码块（也叫构造代码块）、创建实例的相应的构造方法都会执行。  调用顺序：静态代码块、非静态代码块、构造方法。 |

4.1.5 java程序初始化顺序

当实例化对象时，对象所在类的所有成员变量首先要初始化，之后才会调用对象所在类的构造函数创建对象。

初始化原则：

|  |
| --- |
| 1.静态对象/变量(只初始化一次)优先于非静态对象/变量(可能多次)初始化，   1. 父类优先于子类 2. 按成员变量定义顺序初始化。   及时变量定义分散了方法定义中，依旧在方法(含构造函数)调用前初始化。 |

初始化方式的执行顺序：

|  |
| --- |
| 父类静态变量、父类静态代码块、  子类静态变量、子类静态代码块  父类非静态变量、父类非静态代码块、父类构造函数  子类非静态变量、子类非静态代码块、子类构造函数 |