Java

面向对象编程的四大特性及其含义?

- 技术点：面向对象编程特点
- 思路：分条简述每个特性的含义
- 参考回答：

    - 抽象：对现实世界的事物进行概括，抽象为在计算机虚拟世界中有意义的实体
    - 封装：将某事物的属性和行为包装到对象中，构成一个不可分割的独立实体，数据被保护在抽象数据类型的内部，并且尽可能地隐藏内部的细节，只保留一些对外接口使之与外部发生联系
    - 继承：子类继承父类，不仅可以有父类原有的方法和属性，也可以增加自己的或者重写父类的方法及属性
    - 多态：允许不同类的对象对同一消息做出各自的响应

String、StringBuffer和StringBuilder的区别?

- 技术点：String
- 参考回答：

    - String是字符串常量，而StringBuffer、StringBuilder都是字符串变量，即String对象一创建后不可更改，而后两者的对象是可更改的：
    - StringBuffer是线程安全的，而StringBuilder是非线程安全的，这是由于StringBuffer对方法加了同步锁或者对调用的方法加了同步锁
    - String更适用于少量的字符串操作的情况，StringBuilder适用于单线程下在字符缓冲区进行大量操作的情况，StringBuffer适用于多线程下在字符缓冲区进行大量操作的情况

String a=""和String a=new String("")的的关系和异同?

- 技术点：String
- 参考回答：

    - 通过String a=""直接赋值的方式得到的是一个字符串常量，存在于常量池；注意，相同内容的字符串在常量池中只有一个，即如果池已包含内容相等的字符串会返回池中的字符串，反之会将该字符串放入池中
    - 通过new String("")创建的字符串不是常量是实例对象，会在堆内存开辟空间并存放数据，且每个实例对象都有自己的地址空间

- 引申：对于用String a=""和String a=new String("")两种方式定义的字符串，判断使用equals()、"=="比较结果是什么

Object的equal()和==的区别?

- 技术点：equal()、==
- 参考回答：
    - equals()：是Object的公有方法，具体含义取决于如何重写，比如String的equals()比较的是两个字符串的内容是否相同
    - "==" ：对于基本数据类型来说，比较的是两个变量值是够是否相等，对于引用类型来说，比较的是两个对象的内存地址是否相同
- 引申：对于用String a=""和String a=new String("")两种方式定义的字符串，判断使用equals()、"=="比较结果是什么

装箱、拆箱什么含义?

- 技术点：装箱、拆箱
- 参考回答：装箱就是自动将基本数据类型转换为包装器类型，拆箱就是自动将包装器类型转换为基本数据类型

int和Integer的区别?

- 技术点：基本数据类型、引用类型
- 参考回答：
    - Integer是int的包装类，int则是java的一种基本数据类型
    - Integer变量必须实例化后才能使用，而int变量不需要
    - Integer实际是对象的引用，当new一个Integer时，实际上是生成一个指针指向此对象；而int则是直接存储数据值
    - Integer的默认值是null，int的默认值是0

遇见过哪些运行时异常？异常处理机制知道哪些?

- 技术点：Java异常机制
- 思路：对Throwable异常进行分类说明每种异常的特点和常见问题，简述几种常见异常处理机制
- 参考回答：
- （1） Throwable继承层次结构，可见分成两大类Error和Exception：
    - Error（错误）:指程序无法恢复的异常情况，表示运行应用程序中较严重的问题；发生于虚拟机自身、或者在虚拟机试图执行应用时，如Virtual MachineError（Java虚拟机运行错误）、NoClassDefFoundError（类定义错误）；属于不可查异常，即不强制程序员必须处理，即使不处理也不会出现语法错误。 Exception（异常）:指程序有可能恢复的异常情况，表示程序本身可以处理的异常。又分两大类：
        - RuntimeException（运行时异常）：由程序自身的问题导致产生的异常；如NullPointerException（空指针异常）、IndexOutOfBoundsException（下标越界异常）；属于不可查异常。
        - 非运行时异常：由程序外部的问题引起的异常；除了RuntimeException以外的异常，如

FileNotFoundException（文件不存在异常）；属于可查异常，即强制程序员必须进行处理，如果不进行处理则会出现语法错误。

- （2）常见的异常处理机制有：
    - 捕捉异常：由系统自动抛出异常，即try捕获异常->catch处理异常->finally 最终处理
    - 抛出异常：在方法中将异常对象显性地抛出，之后异常会沿着调用层次向上抛出，交由调用它的方法来处理。配合throws声明抛出的异常和throw抛出异常
    - 自定义异常：继承Execption类或其子类

什么是反射，有什么作用和应用?

- 技术点：反射
- 思路：简述反射的定义、功能和应用，详见Java基础之泛型&反射
- 参考回答：
    - 含义：在运行状态中，对于任意一个类都能知道它的所有属性和方法，对于任何一个对象都能够调用它的任何一个方法和属性。
    - 功能：动态性，体现在：在运行时判断任意一个类所具有的属性和方法； 在运行时判断任意一个对象所属的类；在运行时构造任意一个类的对象；在运行时调用任意一个对象的方法；生成动态代理
    - 应用：反射&泛型

什么是内部类？有什么作用？静态内部类和非静态内部类的区别?

- 技术点：内部类
- 思路：
- 参考回答：内部类就是定义在另外一个类里面的类。它隐藏在外部类中，封装性更强，不允许除外部类外的其他类访问它；但它可直接访问外部类的成员。静态内部类和非静态内部类的区别有：
    - 静态内部类是指被声明为static的内部类，可不依赖外部类实例化；而非静态内部类需要通过生成外部类来间接生成。
    - 静态内部类只能访问外部类的静态成员变量和静态方法，而非静态内部类由于持有对外部类的引用，可以访问外部类的所用成员

final、finally、finalize()分别表示什么含义

- 技术点：final、finally、finalize()
- 参考回答：
    - final关键字表示不可更改，具体体现在：
        - final修饰的变量必须要初始化，且赋初值后不能再重新赋值
        - final修饰的方法不能被子类重写
        - final修饰的类不能被继承

- finally：和try、catch成套使用进行异常处理，无论是否捕获或处理异常，finally块里的语句都会被执行，在以下4种特殊情况下，finally块才不会被执行：
    - 在finally语句块中发生了异常
    - 在前面的代码中用了System.exit()退出程序
    - 程序所在的线程死亡
    - 关闭CPU
- finalize()：是Object中的方法，当垃圾回收器将回收对象从内存中清除出去之前会调用finalize()，但此时并不代表该回收对象一定会“死亡”，还有机会“逃脱”

重写和重载的区别?

- 技术点：重写、重载
- 参考回答：重写表示子类重写父类的方法；重载表示有多个同名函数同时存在，区别在于有不同的参数个数或类型

抽象类和接口的异同?

- 技术点：抽象类、接口
- 参考回答：
    - 使用上的区别：一个类只能继承一个抽象类却可以实现多个接口
    - 设计上的区别：接口是对行为的抽象，无需有子类的前提，是自上而下的设计理念；抽象类是对类的抽象，建立于相似子类之上，是自下而上的设计理念

为什么匿名内部类中使用局部变量要用final修饰?

- 技术点：匿名内部类
- 参考回答：
    - 一方面，由于方法中的局部变量的生命周期很短，一旦方法结束变量就要被销毁，为了保证在内部类中能找到外部局部变量，通过final关键字可得到一个外部变量的引用；
    - 另一方面，通过final关键字也不会在内部类去做修改该变量的值，保护了数据的一致性。

Object有哪些公有方法?

- 技术点：Object
- 思路：列举常见的几个公有方法
- 参考回答：
    - equals()： 和==作用相似
    - hashCode()：用于哈希查找，重写了equals()一般都要重写该方法
    - getClass()： 获取Class对象
    - wait()：让当前线程进入等待状态，并释放它所持有的锁
    - notify()&notifyAll()： 唤醒一个（所有）正处于等待状态的线程
    - toString()：转换成字符串

- 引申：equals()和==的不同、在synchronized 同步代码块里wait()和notify()&notifyAll()如何配合、hashCode()和equals()的关系、获取Class对象还有什么方法