**继承和聚合的区别**

继承

指的是一个类继承另外的一个类的功能，并可以增加它自己的新功能的能力，继承是类与类或者接口与接口之间最常见的关系；在Java中此类关系通过关键字extends明确标识。

聚合

聚合体现的是整体与部分、拥有的关系，此时整体与部分之间是可分离的，他们可以具有各自的生命周期；比如计算机与CPU、公司与员工的关系等；

**深拷贝和浅拷贝区别**

1.浅拷贝

对基本数据类型进行值传递，对引用数据类型进行引用传递般的拷贝，此为浅拷贝。

2.深拷贝

对基本数据类型进行值传递，对引用数据类型，创建一个新的对象，并复制其内容，此为深拷贝。

序列化

clone 方法 对其内的引用类型的变量，再进行一次 clone()。

**语法糖**

Java作为一种与平台无关的高级语言，当然也含有语法糖，这些语法糖并不被虚拟机所支持，在编译成字节码阶段就自动转换成简单常用语法。一般来说Java中的语法糖主要有以下几种：

**泛型与类型擦除**

**自动装箱与拆箱，变长参数、**

**增强for循环**

**内部类与枚举类**

**switch 支持 String 与枚举**

**方法变长参数**

**try-with-resources**

Java 7开始，jdk提供了一种更好的方式关闭资源，使用try-with-resources语句，

public static void main(String... args) {

try (BufferedReader br = new BufferedReader(new FileReader("d:\\ hollischuang.xml"))) {

String line;

while ((line = br.readLine()) != null) {

System.out.println(line);

}

} catch (IOException e) {

// handle exception

}

}

**Lambda表达式**

**static、final、static final的区别**

final：

final可以修饰：属性，方法，类，局部变量（方法中的变量）

final修饰的属性的初始化可以在编译期，也可以在运行期，初始化后不能被改变。

final修饰的属性跟具体对象有关，在运行期初始化的final属性，不同对象可以有不同的值。

final修饰的属性表明是一个常数（创建后不能被修改）。

final修饰的方法表示该方法在子类中不能被重写，final修饰的类表示该类不能被继承。

对于基本类型数据，final会将值变为一个常数（创建后不能被修改）；但是对于对象句柄（亦可称作引用或者指针），final会将句柄变为一个常数（进行声明时，必须将句柄初始化到一个具体的对象。而且不能再将句柄指向另一个对象。但是，对象的本身是可以修改的。这一限制也适用于数组，数组也属于对象，数组本身也是可以修改的。方法参数中的final句柄，意味着在该方法内部，我们不能改变参数句柄指向的实际东西，也就是说在方法内部不能给形参句柄再另外赋值）。

static：

static可以修饰：属性，方法，代码段，内部类（静态内部类或嵌套内部类）

static修饰的属性的初始化在编译期（类加载的时候），初始化后能改变。

static修饰的属性所有对象都只有一个值。

static修饰的属性强调它们只有一个。

static修饰的属性、方法、代码段跟该类的具体对象无关，不创建对象也能调用static修饰的属性、方法等

static和“this、super”势不两立，static跟具体对象无关，而this、super正好跟具体对象有关。

static不可以修饰局部变量。

static final和final static：

static final和final static没什么区别，一般static写在前面。

static final：

static修饰的属性强调它们只有一个，final修饰的属性表明是一个常数（创建后不能被修改）。static final修饰的属性表示一旦给值，就不可修改，并且可以通过类名访问。

static final也可以修饰方法，表示该方法不能重写，可以在不new对象的情况下调用

## &和&&的区别？

答：&运算符有两种用法：(1)按位与；(2)逻辑与。&&运算符是短路与运算。逻辑与跟短路与的差别是非常巨大的，虽然二者都要求运算符左右两端的布尔值都是true整个表达式的值才是true。&&之所以称为短路运算是因为，如果&&左边的表达式的值是false，右边的表达式会被直接短路掉，不会进行运算。很多时候我们可能都需要用&&而不是&，例如在验证用户登录时判定用户名不是null而且不是空字符串，应当写为：username != null &&!username.equals(“”)，二者的顺序不能交换，更不能用&运算符，因为第一个条件如果不成立，根本不能进行字符串的equals比较，否则会产生NullPointerException异常。

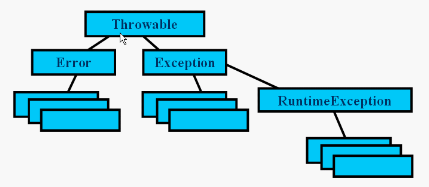
注意：逻辑或运算符（|）和短路或运算符（||）的差别也是如此。

String和StringBuilder、StringBuffer的区别？  
答：Java平台提供了两种类型的字符串：String和StringBuffer/StringBuilder，它们可以储存和操作字符串。其中String是只读字符串，也就意味着String引用的字符串内容是不能被改变的。而StringBuffer/StringBuilder类表示的字符串对象可以直接进行修改。StringBuilder是Java 5中引入的，它和StringBuffer的方法完全相同，区别在于它是在单线程环境下使用的，因为它的所有方面都没有被synchronized修饰，因此它的效率也比StringBuffer要高。

# Java异常分类

一、基本概念

看java的异常结构图



Throwable是所有异常的根，java.lang.Throwable

Error是错误，java.lang.Error

Exception是异常，java.lang.Exception

二、Exception

一般分为Checked异常和Runtime异常，所有RuntimeException类及其子类的实例被称为Runtime异常，不属于该范畴的异常则被称为CheckedException。

①Checked异常

只有java语言提供了Checked异常，Java认为Checked异常都是可以被处理的异常，所以Java程序必须显示处理Checked异常。如果程序没有处理Checked异常，该程序在编译时就会发生错误无法编译。这体现了Java的设计哲学：没有完善错误处理的代码根本没有机会被执行。对Checked异常处理方法有两种

1 当前方法知道如何处理该异常，则用try...catch块来处理该异常。

2 当前方法不知道如何处理，则在定义该方法是声明抛出该异常。

总异常类，既有checkedException又有RuntimeException，所以其中的checkedException必须处理

我们比较熟悉的Checked异常有

Java.lang.ClassNotFoundException

Java.lang.NoSuchMetodException

java.io.IOException

②RuntimeException

Runtime如除数是0和数组下标越界等，其产生频繁，处理麻烦，若显示申明或者捕获将会对程序的可读性和运行效率影响很大。所以由系统自动检测并将它们交给缺省的异常处理程序。当然如果你有处理要求也可以显示捕获它们。

我们比较熟悉的RumtimeException类的子类有

Java.lang.ArithmeticException

Java.lang.ArrayStoreExcetpion

Java.lang.ClassCastException

Java.lang.IndexOutOfBoundsException

Java.lang.NullPointerException

三、Error

当程序发生不可控的错误时，通常做法是通知用户并中止程序的执行。与异常不同的是Error及其子类的对象不应被抛出。

Error是throwable的子类，代表编译时间和系统错误，用于指示合理的应用程序不应该试图捕获的严重问题。

Error由Java虚拟机生成并抛出，包括动态链接失败，虚拟机错误等。程序对其不做处理。

## 类的实例化顺序，比如父类静态数据，构造函数，字段，子类静态数据，构造函数，字段，当 new 的时候， 他们的执行顺序。

此题考察的是类加载器实例化时进行的操作步骤（加载–>连接->初始化）。

父类静态代变量、

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（父类实例成员变量）、

父类构造函数、

子类非静态变量（子类实例成员变量）、

子类构造函数。