JAVA面试宝典

1.final, finally, finalize 的区别

- final 用于声明属性,方法和类, 分别表示属性不可变, 方法不可覆盖, 类不可继承。
- finally 是异常处理语句结构的一部分，表示总是执行。
- finalize 是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法，提供垃圾收集时的其他资源回收，例如关闭文件等，JVM不保证此方法总被调用。

2.抽象类和接口有什么区别

- 抽象类可以有构造方法，接口中不能有构造方法。
- 抽象类中可以有普通成员变量，接口中没有普通成员变量
- 抽象类中可以包含非抽象的普通方法，接口中的所有方法必须都是抽象的，不能有非抽象的普通方法。
- 抽象类中的抽象方法的访问类型可以是public，protected和（默认类型,虽然eclipse下不报错，但应该也不行），但接口中的抽象方法只能是public类型的，并且默认即为public abstract类型。
- 抽象类中可以包含静态方法，接口中不能包含静态方法
- 抽象类和接口中都可以包含静态成员变量，抽象类中的静态成员变量的访问类型可以任意，但接口中定义的变量只能是public static final类型，并且默认即为public static final类型。

3.说说反射的用途及实现

JAVA反射机制是在运行状态中，对于任意一个类都能够知道这个类的所有属性和方法；对于任意一个对象，都能够调用它的任意一个方法和属性；这种动态获取的信息以及动态调用对象的方法的功能称为java语言的反射机制。
主要作用有三：

- 运行时取得类的方法和字段的相关信息。
- 创建某个类的新实例(.newInstance())

- 取得字段引用直接获取和设置对象字段，无论访问修饰符是什么。

用处如下：

- 观察或操作应用程序的运行时行为。
- 调试或测试程序，因为可以直接访问方法、构造函数和成员字段。
- 通过名字调用不知道的方法并使用该信息来创建对象和调用方法。

4.Java程序初始化顺序

当实例化对象时，对象所在的类的所有成员变量首先要进行初始化，只有当所有类成员完成初始化后，才会调用对象所在类的构造函数创建对象
一般遵循三个原则：

- 静态对象（静态变量）优于非静态对象（变量）初始化，其中静态对象只初始化一次，而非静态对象可能会初始化多次
- 父类优于子类进行初始化
- 按照成员变量的定义顺序进行初始化

java初始化工作可以在许多不同的代码块中来完成，执行顺序如下：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

5.clone()方法的作用

当需要clone一个对象实例时，简单的赋值操作只是浅拷贝不能满足需求，使用clone()方法满足需求
Java中的类都默认继承自Object类，Object类中提供了一个clone()方法，这个方法返回一个Object对象的复制，返回的是一个新的对象而不是一个引用

- 实现clone的类首先需要继承Cloneable接口
- 在类中重写Object类的clone()方法，访问修饰符设为public
- 在clone()方法中调用super.clone()
- 把浅复制的引用指向原型对象新的克隆体

6.浅复制与深复制的区别

浅拷贝是指在拷贝对象时，对于基本数据类型的变量会重新复制一份，而对于引用类型的变量只是对引用进行拷贝，没有对引用指向的对象进行拷贝。
而深拷贝是指在拷贝对象时，同时会对引用指向的对象进行拷贝。区别就在于是否对对象中的引用变量所指向的对象进行拷贝。
浅复制：被复制对象的所有变量都含有与原来对象相同的值，而所有对其他对象的引用仍然指向原来的对象，及浅拷贝仅仅复制所考虑的对象，而不复制它所引用的对象。
深复制：把复制对象所引用的对象都复制了一遍

7.重载和重写（覆盖）

重载：就是在类中可以创建多个方法，它们具有相同的名字，但具有不同的参数和不同的定义，重载的时候，方法名要一样，但是参数类型和个数不一样，返回值类型可以相同也可以不相同。无法以返回值类型别作为重载函数的区分标准。
在使用重载要注意以下的几点：

- 在使用重载时只能通过不同的参数样式。例如，不同的参数类型，不同的参数个数，不同的参数顺序（当然，同一方法内的几个参数类型必须不一样，例如可以是 fun(int,float)，但是不能为 fun(int,int)）；
- 不能通过访问权限、返回类型、抛出的异常进行重载；
- 方法的异常类型和数目不会对重载造成影响；
- 对于继承来说，如果某一方法在父类中是访问权限是 priavte，那么就不能在子类对其进行重载，如果定义的话，也只是定义了一个新方法，而不会达到重载的效果。

重写（覆盖）：在子类中定义某方法与其父类有相同的名称和参数，只有函数体内不同
在覆盖要注意以下的几点：

- 覆盖的方法的标志必须要和被覆盖的方法的标志完全匹配，才能达到覆盖的效果；
- 覆盖的方法的返回值必须和被覆盖的方法的返回一致；

- 覆盖的方法所抛出的异常必须和被覆盖方法的所抛出的异常一致，或者是其子类；
- 被覆盖的方法不能为 private，否则在其子类中只是新定义了一个方法，并没有对其进行覆盖。

8.JAVA序列化与反序列化：

序列化：使对象持久化的手段，即指把堆内存中的Java对象数据，通过某种方式把对象存储到磁盘文件中或者传递给其他网络节点（在网络上传输时），这个过程称之为序列化，通俗来说就是将数据结构或者对象转换成二进制串的过程。
反序列化： 把磁盘文件中的数据或者网络节点上的对象数据，恢复成对象模型的过程，也就是将序列化过程中所生成的二进制串转换成数据结构或者对象的过程
序列化的实现：

- 实现Serializable接口，该接口没有需要实现的方法，implements Serializable只是为了标注该对象是可被序列化的
- 使用一个输出流来构造一个对象流，通过使用对象流来完成序列化与反序列化
- ObjectOutStream:通过writeObject()方法实现序列化操作
- ObjectInputStream:通过readObject()方法实现反序列化操作

注：序列化是用来处理对象流的机制，所谓对象流即是将对象的内容进行流化，可以将流化后的对象进行读写工作，也可以将流化后的对象传输于网络之间
序列化的实现：将需要被序列化的类实现Serializable接口，该接口没有需要实现的方法，implements Serializable只是为了标注该对象是可被序列化的，然后使用一个输出流（例如FileOutStream)来构造一个对象流（ObjectOutputStream)对象，接着，使用ObjectOutputStream对象的writeObject（Object obj)方法就可以将参数为obj的对象写出（即保存其状态），要恢复时可以使用其对应的输入流。
序列化的两个特点：

- 如果一个类能够被序列化，那么它的子类也能够被序列化

- 如果static代表类的成员，transient代表对象的临时数据，被声明为这两种类型的数据成员时不能够被序列化的。

在序列化和反序列化的过程中，serialVersionUID起着非常重要的作用，每个类都有一个特定的serialVersionUID，在反序列化的过程中，通过serialVersionUID来判断类的兼容性。如果待序列化的对象与目标对象的serialVersionUID不同，那么反序列化时就会抛出异常。
自定义serialVersionUID有3个优点：
1）提高程序的运行效率
2）提高程序在不同平台上的兼容性
3）增强程序各个版本的可兼容性

9.什么是不可变类

不可变类是指当创建了这个类的实例后就不允许修改它的值了，也就是说一个对象一旦被创建出来，它的成员变量就不能被修改了。
如何创建一个不可变类：

- 使用final和private修饰符修饰类中变量
- 提供带参数的构造器通过参数来初始化属性值
- 仅为该类提供getter方法，不提供setter方法。
- 如果有必要，重写Object类中的equals方法和hashCode方法：在equals方法中根据对象属性值来比较两个对象是否相等，还要保证equals方法判断相等的两个对象的hashCode（）方法的返回值也相等。
- 如果属性中存在引用类型，并且引用变量属性的类型是可变的，必须保护所引用的对象内容不会被改变，不让引用地址相同（name与n不指向同一个地址），既不是同一个对象，但要求对象的内容相同，初始化引用属性时不采用传过来的引用对象，而是新创建一个对象，但二者内容相同，这样就可以避免由于引用属性为可变类所造成的可变性
- 如果一个列成员不是不可变变量，那么在成员初始化或者使用get方法获取该成员变量时，需要通过clone方法来确保类的不可变性

10.hashCode与equals的关系

一般在覆盖equals()方法的同时也要覆盖hashCode()方法，否则就会导致该类无法与所有基于散列值的集合类结合在一起正常运行
如果两个对象根据equals方法比较是相等的，那么hashcode()的返回值一定是相等的，如果两个对象根据equals方法比较是不相等的，那么hashcode()的返回值可能相等也可能不相等

11.常见的运行时异常

空指针异常（NullPointerException）、数组越界异常、类型转换异常、数组存储异常、缓冲区异常、算术异常

12.final关键字

- 修饰类时，表示该类不能被继承（隐式地指定了该类中的方法为final方法）
- 修饰方法时，表示该方法不能被覆盖
- 修饰变量时，表示该变量是常量，对于基本类型，使数值恒定不变，对于对象引用，使引用恒定不变，一旦初始化指向一个对象，就无法再把他改为指向另一个对象，但其对象自身确是可以修改的
- 但当定义域为final但又未给定初值的话（空白final）,编译器必须保证空白final在使用前被初始化
- 当final修饰参数时，无法在方法中更改参数引用所指向的对象、

注:

1）当用final作用于类的成员变量时，成员变量（注意是类的成员变量，局部变量只需要保证在使用之前被初始化赋值即可）必须在定义时或者构造器中（主要针对空白final）进行初始化赋值，而且final变量一旦被初始化赋值之后，就不能再被赋值了。
2)被final修饰的引用变量所指向对象的内容是可变的
3）final和static 的区别：static 是保证变量只有一份，而final保证变量的值不变
4）final变量并不一定保证编译期值就是确定的
5）针对于final参数时，它并不一定能达到“当你在方法中不需要改变作为参数的对象变量时，明确使用final进行声明，会防止你无意的修改而影响到调用方

法外的变量”的效果，无论参数是基本类型还是引用类型。

13.解释一下什么是 servlet

servlet有良好的生存期的定义，包括加载和实例化、初始化、处理请求以及服务结束。这 个生存期由javax.servlet.Servlet接口的init,service和destroy方法表达。

14.说一说 Servlet的生命周期?

servlet有良好的生存期的定义，包括加载和实例化、初始化、处理请求以及服务结束。这 个生存期由javax.servlet.Servlet接口的init,service和destroy方法表达。

Servlet被服务器实例化后，容器运行其 init方法，请求到达时运行其 service方法，service 方法自动派遣运行与请求对应的 doXXX方法（doGet，doPost）等，当服务器决定将实例 销毁的时候调用其destroy方法。 web容器加载servlet，生命周期开始。通过调用servlet的init()方法进行servlet的初始化。通过调用service()方法实现，根据请求的不同调用不同的do***()方法。结束服务，web容器调 用servlet的destroy()方法。

14.SERVLET API中 forward() 与 redirect()的区别?

前者仅是容器中控制权的转向，在客户端浏览器地址栏中不会显示出转向后的地址；后 者则是完全的跳转，浏览器将会得到跳转的地址，并重新发送请求链接。这样，从浏览器 的地址栏中可以看到跳转后的链接地址。所以，前者更加高效，在前者可以满足需要时， 尽量使用forward()方法，并且，这样也有助于隐藏实际的链接。在有些情况下，比如，需要跳转到一个其它服务器上的资源，则必须使用 sendRedirect()方法。