# Java 复习

作者：yscxy
原文链接：https://ld246.com/article/1615777380010
来源网站：链滴

## ● 请说明Query接口的list方法和iterate方法有什么区别?

考察点：接口

### 参考回答：

①list()方法无法利用一级缓存和二级缓存（对缓存只写不读），它只能在开启查询缓存的前提下使用询缓存；iterate()方法可以充分利用缓存，如果目标数据只读或者读取频繁，使用iterate()方法可以少性能开销。

② list()方法不会引起N+1查询问题，而iterate()方法可能引起N+1查询问题

## ● 请你谈一下面向对象的"六原则一法则"。

考察点：Java对象

### 参考回答：

- **单一职责原则**：一个类只做它该做的事情。（单一职责原则想表达的就是"高内聚"，写代码最终极原则只有六个字"高内聚、低耦合"，所谓的高内聚就是一个代码模块只完成一项功能，在面向对象中如果只让一个类完成它该做的事，而不涉及与它无关的领域就是践行了高内聚的原则，这个类就只有一职责。另一个是模块化，好的自行车是组装车，从减震叉、刹车到变速器，所有的部件都是可以拆和重新组装的，好的乒乓球拍也不是成品拍，一定是底板和胶皮可以拆分和自行组装的，一个好的软系统，它里面的每个功能模块也应该是可以轻易的拿到其他系统中使用的，这样才能实现软件复用的标。）

- **开闭原则**：软件实体应当对扩展开放，对修改关闭。（在理想的状态下，当我们需要为一个软件系增加新功能时，只需要从原来的系统派生出一些新类就可以，不需要修改原来的任何一行代码。要做开闭有两个要点：**①抽象是关键**，一个系统中如果没有抽象类或接口系统就没有扩展点；**②封装可变性**将系统中的各种可变因素封装到一个继承结构中，如果多个可变因素混杂在一起，系统将变得复杂而

乱，如果不清楚如何封装可变性，可以参考《设计模式精解》一书中对桥梁模式的讲解的章节。）

- **依赖倒转原则**：面向接口编程。（该原则说得直白和具体一些就是声明方法的参数类型、方法的返类型、变量的引用类型时，尽可能使用抽象类型而不用具体类型，因为抽象类型可以被它的任何一个类型所替代，请参考下面的里氏替换原则。）**里氏替换原则**：任何时候都可以用子类型替换掉父类型（关于里氏替换原则的描述，Barbara Liskov女士的描述比这个要复杂得多，但简单的说就是能用父型的地方就一定能使用子类型。里氏替换原则可以检查继承关系是否合理，如果一个继承关系违背了氏替换原则，那么这个继承关系一定是错误的，需要对代码进行重构。例如让猫继承狗，或者狗继承，又或者让正方形继承长方形都是错误的继承关系，因为你很容易找到违反里氏替换原则的场景。需注意的是：子类一定是增加父类的能力而不是减少父类的能力，因为子类比父类的能力更多，把能力的对象当成能力少的对象来用当然没有任何问题。）

- **接口隔离原则**：接口要小而专，绝不能大而全。（臃肿的接口是对接口的污染，既然接口表示能力那么一个接口只应该描述一种能力，接口也应该是高度内聚的。例如，琴棋书画就应该分别设计为四接口，而不应设计成一个接口中的四个方法，因为如果设计成一个接口中的四个方法，那么这个接口难用，毕竟琴棋书画四样都精通的人还是少数，而如果设计成四个接口，会几项就实现几个接口，这的话每个接口被复用的可能性是很高的。Java中的接口代表能力、代表约定、代表角色，能否正确的用接口一定是编程水平高低的重要标识。）

- **合成聚合复用原则**：优先使用聚合或合成关系复用代码。（通过继承来复用代码是面向对象程序设中被滥用得最多的东西，因为所有的教科书都无一例外的对继承进行了鼓吹从而误导了初学者，类与之间简单的说有三种关系，Is-A关系、Has-A关系、Use-A关系，分别代表继承、关联和依赖。其中关联关系根据其关联的强度又可以进一步划分为关联、聚合和合成，但说白了都是Has-A关系，合成合复用原则想表达的是优先考虑Has-A关系而不是Is-A关系复用代码，原因嘛可以自己从百度上找到万个理由，需要说明的是，即使在Java的API中也有不少滥用继承的例子，例如Properties类继承了Hahtable类，Stack类继承了Vector类，这些继承明显就是错误的，更好的做法是在Properties类中放一个Hashtable类型的成员并且将其键和值都设置为字符串来存储数据，而Stack类的设计也应该是在Sack类中放一个Vector对象来存储数据。记住：任何时候都不要继承工具类，工具是可以拥有并可以用的，而不是拿来继承的。）

- **迪米特法则**：迪米特法则又叫最少知识原则，一个对象应当对其他对象有尽可能少的了解。再复杂系统都可以为用户提供一个简单的门面，Java Web开发中作为前端控制器的Servlet或Filter不就是一门面吗，浏览器对服务器的运作方式一无所知，但是通过前端控制器就能够根据你的请求得到相应的务。调停者模式也可以举一个简单的例子来说明，例如一台计算机，CPU、内存、硬盘、显卡、声卡种设备需要相互配合才能很好的工作，但是如果这些东西都直接连接到一起，计算机的布线将异常复，在这种情况下，主板作为一个调停者的身份出现，它将各个设备连接在一起而不需要每个设备之间接交换数据，这样就减小了系统的耦合度和复杂度。

# ●请说明内部类可以引用他包含类的成员吗，如果可以，有有什么限制吗？

考察点：类

## 参考回答：

一个内部类对象可以访问创建它的外部类对象的内容，内部类如果不是static的，那么它可以访问创它的外部类对象的所有属性内部类如果是sattic的，即为nested class，那么它只可以访问创建它的外类对象的所有static属性一般普通类只有public或package的访问修饰，而内部类可以实现static，protected，private等访问修饰。当从外部类继承的时候，内部类是不会被覆盖的，它们是完全独立的实，每个都在自己的命名空间内，如果从内部类中明确地继承，就可以覆盖原来内部类的方法。

# ●请说明Comparable和Comparator接口的作用以及它们区别。

考察点：comparable接口

## 参考回答：

1. Java提供了只包含一个compareTo()方法的Comparable接口。这个方法可以个给两个对象排序。体来说，它返回负数，0，正数来表明输入对象小于，等于，大于已经存在的对象。

2. Java提供了包含compare()和equals()两个方法的Comparator接口。compare()方法用来给两个入参数排序，返回负数，0，正数表明第一个参数是小于，等于，大于第二个参数。equals()方法需要个对象作为参数，它用来决定输入参数是否和comparator相等。只有当输入参数也是一个comparato并且输入参数和当前comparator的排序结果是相同的时候，这个方法才返回true。

参考博客链接： https://blog.csdn.net/lsqingfeng/article/details/80342620

# ● 请解释一下extends 和super 泛型限定符

考察点：JAVA泛型

## 参考回答：

（1）泛型中上界和下界的定义

上界<? extend Fruit>

下界<? super Apple>

（2）上界和下界的特点

上界的list只能get，不能add（确切地说不能add出除null之外的对象，包括Object）

下界的list只能add，不能get

（3）上界<? extend Fruit> ，表示所有继承Fruit的子类，但是具体是哪个子类，无法确定，所以调add的时候，要add什么类型，谁也不知道。但是get的时候，不管是什么子类，不管追溯多少辈，肯有个父类是Fruit，所以，我都可以用最大的父类Fruit接着，也就是把所有的子类向上转型为Fruit。

下界<? super Apple>，表示Apple的所有父类，包括Fruit，一直可以追溯到老祖宗Object 。那么我add的时候，我不能add Apple的父类，因为不能确定List里面存放的到底是哪个父类。但是我可以dd Apple及其子类。因为不管我的子类是什么类型，它都可以向上转型为Apple及其所有的父类甚至型为Object 。但是当我get的时候，Apple的父类这么多，我用什么接着呢，除了Object，其他的都不住。

所以，归根结底可以用一句话表示，那就是编译器可以支持向上转型，但不支持向下转型。具体来讲我可以把Apple对象赋值给Fruit的引用，但是如果把Fruit对象赋值给Apple的引用就必须得用cast。

# ● 请说明静态变量存在什么位置?

考察点：类

## 参考回答：

方法区

## ● 请你解释一下类加载机制，双亲委派模型，好处是什么?

考察点：类

## 参考回答：

某个特定的类加载器在接到加载类的请求时，首先将加载任务委托给父类加载器，依次递归，如果父加载器可以完成类加载任务，就成功返回；只有父类加载器无法完成此加载任务时，才自己去加载。

使用双亲委派模型的好处在于Java类随着它的类加载器一起具备了一种带有优先级的层次关系。例如类ava.lang.Object，它存在在rt.jar中，无论哪一个类加载器要加载这个类，最终都是委派给处于模型顶端的Bootstrap ClassLoader进行加载，因此Object类在程序的各种类加载器环境中都是同一个类相反，如果没有双亲委派模型而是由各个类加载器自行加载的话，如果用户编写了一个java.lang.Objet的同名类并放在ClassPath中，那系统中将会出现多个不同的Object类，程序将混乱。因此，如果开者尝试编写一个与rt.jar类库中重名的Java类，可以正常编译，但是永远无法被加载运行。

## ● 请你谈谈StringBuffer和StringBuilder有什么区别，底实现上呢?

考察点：类

## 参考回答：

StringBuffer线程安全，StringBuilder线程不安全，底层实现上的话，StringBuffer其实就是比String uilder多了Synchronized修饰符。

## ● 请说明String是否能能继承?

考察点：String

## 参考回答：

不能，char数组用final修饰的。

## ● 请说明”static”关键字是什么意思?Java中是否可以覆盖override)一个private或者是static的方法?

考察点：static变量

## 参考回答：

“static”关键字表明一个成员变量或者是成员方法可以在没有所属的类的实例变量的情况下被访问。

Java中static方法不能被覆盖，因为方法覆盖是基于运行时动态绑定的，而static方法是编译时静态绑的。static方法跟类的任何实例都不相关，所以概念上不适用。

## ● 请说明重载和重写的区别，相同参数不同返回值能重载吗?

考察点：重载

## 参考回答：

重载(Overloading)

（1） 方法重载是让类以统一的方式处理不同类型数据的一种手段。多个同名函数同时存在，具有不的参数个数/类型。

重载Overloading是一个类中多态性的一种表现。

（2） Java的方法重载，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同的定义。

调用方法时通过传递给它们的不同参数个数和参数类型来决定具体使用哪个方法, 这就是多态性。

（3） 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型别作为重载函数的区分标准。

重写（Overriding）

（1） 父类与子类之间的多态性，对父类的函数进行重新定义。如果在子类中定义某方法与其父类有同的名称和参数，我们说该方法被重写 (Overriding)。在Java中，子类可继承父类中的方法，而不需重新编写相同的方法。

但有时子类并不想原封不动地继承父类的方法，而是想作一定的修改，这就需要采用方法的重写。

方法重写又称方法覆盖。

（2）若子类中的方法与父类中的某一方法具有相同的方法名、返回类型和参数表，则新方法将覆盖有的方法。

如需父类中原有的方法，可使用super关键字，该关键字引用了当前类的父类。

（3）子类函数的访问修饰权限不能少于父类的。

# ● 请列举你所知道的Object类的方法并简要说明。

考察点：面向对象

## 参考回答：

Object()默认构造方法。clone() 创建并返回此对象的一个副本。equals(Object obj) 指示某个其他象是否与此对象“相等”。finalize()当垃圾回收器确定不存在对该对象的更多引用时，由对象的垃圾收器调用此方法。getClass()返回一个对象的运行时类。hashCode()返回该对象的哈希码值。 notify(唤醒在此对象监视器上等待的单个线程。 notifyAll()唤醒在此对象监视器上等待的所有线程。toString)返回该对象的字符串表示。wait()导致当前的线程等待，直到其他线程调用此对象的 notify() 方法或otifyAll() 方法。wait(long timeout)导致当前的线程等待，直到其他线程调用此对象的 notify() 方法 notifyAll() 方法，或者超过指定的时间量。wait(long timeout, int nanos) 导致当前的线程等待，到其他线程调用此对象的 notify() 方法或 notifyAll() 方法，或者其他某个线程中断当前线程，或者已过某个实际时间量。

# ● 请讲讲Java有哪些特性，并举一个和多态有关的例子。

考察点：语言特性

## 参考回答：

封装、继承、多态。多态：指允许不同类的对象对同一消息做出响应。即同一消息可以根据发送对象不同而采用多种不同的行为方式。（发送消息就是函数调用）

## 博客链接：https://blog.csdn.net/qq_38238041/article/details/78897893

# ● 请你讲讲wait方法的底层原理

考察点：基础

## 参考回答：

ObjectSynchronizer::wait方法通过object的对象中找到ObjectMonitor对象调用方法 void ObjectMonitor::wait(jlong millis, bool interruptible, TRAPS)

通过ObjectMonitor::AddWaiter调用把新建立的ObjectWaiter对象放入到 _WaitSet 的队列的末尾然后在ObjectMonitor::exit释放锁，接着 thread_ParkEvent->park 也就是wait。

下面正常操作，放几个链接喽

神奇的工作室

神奇的工作室