**1.请列举出在 JDK 中几个常用的设计模式？**

单例模式（Singleton pattern）用于 Runtime，Calendar 和其他的一些类中。工厂模式（Factory pattern）被用于各种不可变的类如 Boolean，像 Boolean.valueOf，观察者模式（Observer pattern）被用于 Swing 和很多的事件监听中。装饰器设计模式（Decorator design pattern）被用于多个 Java IO 类中。

**2.什么是设计模式？你是否在你的代码里面使用过任何设计模式？**

设计模式是世界上各种各样程序员用来解决特定设计问题的尝试和测试的方法。设计模式是代码可用性的延伸

**3.Java 中什么叫单例设计模式？请用 Java 写出线程安全的单例模式**

单例模式重点在于在整个系统上共享一些创建时较耗资源的对象。整个应用中只维护一个特定类实例，它被所有组件共同使用。Java.lang.Runtime 是单例模式的经典例子。从 Java 5 开始你可以使用枚举（enum）来实现线程安全的单例。

**4.在 Java 中，什么叫观察者设计模式（observer design pattern）？**

观察者模式是基于对象的状态变化和观察者的通讯，以便他们作出相应的操作。简单的例子就是一个天气系统，当天气变化时必须在展示给公众的视图中进行反映。这个视图对象是一个主体，而不同的视图是观察者。

**5.使用工厂模式最主要的好处是什么？在哪里使用？**

工厂模式的最大好处是增加了创建对象时的封装层次。如果你使用工厂来创建对象，之后你可以使用更高级和更高性能的实现来替换原始的产品实现或类，这不需要在调用层做任何修改。

**6.举一个用 Java 实现的装饰模式(decorator design pattern)？它是作用于对象层次还是类层次？**

装饰模式增加强了单个对象的能力。Java IO 到处都使用了装饰模式，典型例子就是 Buffered 系列类如 BufferedReader 和 BufferedWriter，它们增强了 Reader 和 Writer 对象，以实现提升性能的 Buffer 层次的读取和写入。

**7.在 Java 中，为什么不允许从静态方法中访问非静态变量？**

Java 中不能从静态上下文访问非静态数据只是因为非静态变量是跟具体的对象实例关联的，而静态的却没有和任何实例关联。

**8.设计一个 ATM 机，请说出你的设计思路？**

比如设计金融系统来说，必须知道它们应该在任何情况下都能够正常工作。不管是断电还是其他情况，ATM 应该保持正确的状态（事务），想想 加锁（locking）、事务（transaction）、错误条件（error condition）、边界条件（boundary condition） 等等。尽管你不能想到具体的设计，但如果你可以指出非功能性需求，提出一些问题，想到关于边界条件，这些都会是很好的。

**9.在 Java 中，什么时候用重载，什么时候用重写？**

如果你看到一个类的不同实现有着不同的方式来做同一件事，那么就应该用重写（overriding），而重载（overloading）是用不同的输入做同一件事。在 Java 中，重载的方法签名不同，而重写并不是。

**10.举例说明什么情况下会更倾向于使用抽象类而不是接口？**

接口和抽象类都遵循”面向接口而不是实现编码”设计原则，它可以增加代码的灵活性，可以适应不断变化的需求。下面有几个点可以帮助你回答这个问题：

在 Java 中，你只能继承一个类，但可以实现多个接口。所以一旦你继承了一个类，你就失去了继承其他类的机会了。
接口通常被用来表示附属描述或行为如：Runnable、Clonable、Serializable 等等，因此当你使用抽象类来表示行为时，你的类就不能同时是 Runnable 和 Clonable(注：这里的意思是指如果把 Runnable 等实现为抽象类的情况)，因为在 Java 中你不能继承两个类，但当你使用接口时，你的类就可以同时拥有多个不同的行为。
在一些对时间要求比较高的应用中，倾向于使用抽象类，它会比接口稍快一点。
如果希望把一系列行为都规范在类继承层次内，并且可以更好地在同一个地方进行编码，那么抽象类是一个更好的选择。有时，接口和抽象类可以一起使用，接口中定义函数，而在抽象类中定义默认的实现。