Java 是一种基于类和面向对象的计算机编程语言。 面向对象的软件开发的优点如下：

代码模块化开发，从而导致维护和修改方便。

可重用性的代码。

提高了可靠性和代码的灵活性。

增加代码的可读性。

面向对象程序设计包含了许多显著的特点，如封装 ， 继承 ， 多态和抽象 ：

封装

封装提供对象隐藏其内部特征和行为的能力。 每一个对象提供了一些方法，这些方法可以由其他对象来访问和改变其内部的数据。在 Java 中，有三种访问修饰符：公共，私有和保护。 每个修饰符规定不同的访问权限，这种访问无论是来自相同的或外部的包。

一些使用封装的优点列举如下：

每一个对象内部状态是通过隐藏属性实现保护。

它增加了代码可用性和维护性，因为对象的行为可以独立地改变或扩展。

它通过防止对象以不希望的方式彼此相互作用，提高了模块化。

多态性

多态性是编程语言中呈现相同的接口，用于实现不同的基础数据类型

的能力。 一个多态型是一种类型，其操作也可以应用于其他一些类型的值。

继承

继承提供了一个对象获得另一个类的字段和方法称为基类的能力。 继承提供了重用性的代码，可以用来添加额外的功能到现有的类，而不需要修改它。

抽象化

抽象是分离抽象和具体的实现，开发者可以在自己的具体类中实现自己的功能。 Java 支持抽象类，接口暴露的创建和存在，抽象类不包括的所有方法的实际执行。 抽象方法的目的是从类的行为中分离出其实现细节。

抽象和封装之间的差异

抽象和封装是相辅相成的概念。 一方面，抽象专注于一个对象的行为。 另一方面，封装专注于一个对象的行为的执行细节。 封装通常是通过隐藏关于对象的内部状态的信息，因此，可以看作是为了提供抽象化中使用的一种实现策略。

关于 Java 的一般问题

1。 什么是 JVM？ 为什么 Java 是"平台无关的编程语言"？

Java 虚拟机（JVM）是一个过程的虚拟机 ，可以执行 Java 字节码 。每个 Java 源文件编译成字节码文件，它是由 JVM 执行。

Java 不必由程序员为每个单独的平台进行重写或重新编译。 Java 虚拟机，使这一切成为可能，因为它知道特定的指令长度和底层硬件平台的其他特殊情况。

2。 什么是 JDK 和 JRE 的区别？

Java 运行时环境（JRE）是基本的 Java 虚拟机（JVM），包括 Java 执行程序。 它还包括浏览器插件小程序执行。 Java 开发工具包（JDK）是 Java 全功能软件开发工具包，包括 JRE，编译器和工具（如 JavaDoc ，和 Java Debugger ）。

3。"静态"的关键字是什么意思？ 你可以在 Java 重写私有或静态方法？

static 关键字表示该成员变量或方法可以被访问，而无需到它所属的类的实例化。

用户不能覆盖静态方法 ，因为方法重载是基于在运行时动态绑定，而静态方法是在编译时绑定。 静态方法不与任何类的实例相关联，这个概念并不适用。

4。 您可以在静态上下文中访问非静态变量？

在 Java 中静态变量是属于它的类，它的所有实例值保持不变。当类被 JVM 加载时，静态变量被初始化。 如果你的代码试图不在任何实例中访问非静态变量，编译器会报错，因为这些变量尚未创建，他们不与任何实例相关联。

5。 什么是 Java 支持的数据类型？ 什么是自动装箱 Autoboxing 和拆箱 Unboxing？

通过 Java 编程语言支持的八个基本数据类型有：

byte

short

int

long

float

double

boolean

char

Autoboxing 是 Java 编译器提出在基本类型和它们对应的对象包装类之间实现自动转换,。 例如，编译一个 int 转换为 Integer ，double 到 Double 等。 如果反过来转换操作称为 unboxing 。

6。 什么是函数重载？

当同一个类中两个或多个具有完全相同的名称但参数不同的 Java 方

法时，会发生重载。 在另一方面，方法覆盖定义：一个子类重新定义的作为父类相同方法。 覆盖方法必须具有相同的名称，参数列表和返回类型。

7。 什么是构造函数，构造函数重载和拷贝构造函数？

创建一个新的对象时，构造函数被调用。 每个类都有一个构造函数 。 若程序员不提供构造函数的类，Java 编译器（javac）创建一个默认的构造函数。

构造函数重载是类似于 Java 方法重载。 不同的构造函数可以为一个类来创建。 每个构造函数都必须有自己独特的参数列表。

Java不支持如C ＋ ＋的拷贝构造函数，如果你不写你自己的构造函数，Java 不会创建一个默认的拷贝构造函数。

8。 Java 是否支持多重继承？

不，Java 不支持多重继承。 每个类是能够只在一个类扩展，但能够实现多个接口。

9。 接口和抽象类之间的区别是什么？

Java 提供和支持创建抽象类和接口。 这两种实现共享一些共同的特征，但它们在以下功能有所不同：

在一个接口中的所有方法都是抽象。 在另一方面，抽象类可以包含抽象和非抽象方法。

一个类可以实现多个接口，但只能扩展一个抽象类。

为了使一个类实现一个接口，它必须实现其所有声明的方法。 但是，一个类可以不实现抽象类的所有声明的方法。 不过，在这种情况下，子类也必须声明为抽象类。

抽象类可以实现接口，甚至可以不提供接口方法的实现。

在 Java 接口中声明的变量默认情况下是 final。 一个抽象类可以包含非 final 的变量。

Java 接口的成员默认为公用。 一个抽象类的成员可以是私有的，保护或公开。

一个接口是绝对抽象的，不能被实例化。 一个抽象类，也不能被实例化，但如果它包含一个 main 方法可以被调用。

10。 什么是按引用传递和按值传递？

当一个对象是按值传递，这意味着该对象的副本传递。 因此，即使更改了该对象，它不会影响原来的值。

当对象是通过引用传递，这意味着实际的对象不会被传递，而是对象的引用被传递。 因此，由外部方法所做的任何更改，将会影响所有地方。