目录

1. 抽象类和接口区别

抽象类和接口的定义：

抽象类（abstract class）：

使用 abstract 修饰符修饰的类。（如果一个类没有包含足够多的信息来描述一个具体的对象，这样的类就是抽象类。）

实际点来说，一个抽象类不能实例化，因为“没有包含足够多的信息来描述一个具体的对象”。但仍然拥有普通类一样的定义。依然可以在类的实体（**直白点就是能在｛｝里面**)定义成员变量，成员方法，构造方法等。

抽象方法：只声明，不实现。具体的实现由继承它的子类来实现。实际点就是：被 abstract 修饰的方法，只有方法名没有方法实现，具体的实现要由子类实现。方法名后面直接跟一个分号，而不是花括号。例如：public abstract int A();

一个类中含有抽象方法（被 abstract 修饰），那么这个类必须被声明为抽象类（被 abstract 修饰）。

接口（interface）：

定义：接口在 java 中是一个抽象类型，是抽象方法的集合。一个类通过继承接口的方式，从而继承接口的抽象方法。从定义上看，接口是个集合，并不是类。类描述了属性和方法，而接口只包含方法（未实现的方法）。

接口和抽象类一样不能被实例化，因为不是类。但是接口可以被实现（使用 implements 关键字）。实现某个接口的类必须在类中实现该接口的全部方法。虽然接口内的方法都是抽象的（和抽象方法很像，没有实现）但是不需要 abstract 关键字。

1) 接口中没有构造方式（因为接口不是类）

2) 接口中的方法必须是抽象的（不能实现）

3) 接口中除了 static、final 变量，不能有其他变量

4) 接口支持多继承（一个类可以实现多个接口）

抽象类和接口的区别：

(1)默认的实现方法：

① **抽象类可以有默认的方法实现完全是抽象的。**

抽象类中可以有已经实现了的方法，也可以有被 abstract 修饰的方法（抽象方法），因为存在抽象方法，所以该类必须是抽象类。

② **接口根本不存在方法的实现。**

但是接口要求只能包含抽象方法，抽象方法是指没有实现的方法。接口就根本不能存在方法的实现。

(2)子类使用的关键词不一样：

① **实现抽象类使用 extends 关键字来继承抽象类。如果子类不是抽象类的话，它需要提供抽象类中所有声明的方法的实现。**

抽象类虽然不能实例化来使用，但是可以被继承，让子类来具体实现父类的所有抽象方法。但是如果子类将抽象方法没有全部实现，就必须把自己也修饰成抽象类，交于继承它的子类来完成实现。以此类推，直到没有抽象函数。

② **子类使用关键字 implements 来实现接口。它需要提供接口中所有声明的方法的实现。**

接口的实现，通过 implements 关键字。实现该接口的类，必须把接口中的所有方法给实现。不能再推给下一代。（和抽象类相比，抽象类是将梦想传给家族，一代一代去完成。那么接口就是掌门人找大师兄来完成帮派的鸿星伟业，这时候就只有一次希望，要么有能力就实现，没能力就不要接。）

(3)是否有构造器：

① **抽象类可以有构造器**

抽象类是属于类，享有类的所有特性（但是不能实例化），当然包括类的构造方法，也就是构造器。

② **接口不能有构造器**

接口是所有抽象方法的集合，注意，是集合，不是类。当然没有构造方法一说，更别提什么构造器了。

(4)可使用的修饰符：

① **抽象方法可以有 public、protected 和 default 这些修饰**

符

抽象类的目的就是被继承，抽象方法就是为了被重写，所以肯定不能用 private 修饰符，肯定是可以用 public 的。但是 protected 和 default 也是可以的。

② **接口方法默认修饰符是 public。你不可以使用其它修饰符。**

接口就有且只有一个 public 修饰。（感觉抽象类像小儿子各种耍无赖，接口就像私生子，说什么只能是什么）

(5)速度方面：

① **抽象方法比接口速度要快**（抽象方法是小儿子，从小吃的好所以跑的快）

② **接口是稍微有点慢的，因为它需要时间去寻找在类中实现的方法。**（接口是私生子，从小日子苦，营养不良）

(6)增加新方法对子类的影响：

① **如果你往抽象类中添加新的方法，你可以给它提供默认的实现。因此你不需要改变你现在的代码。**

抽象类可以有一些非抽象方法的存在，这些方法被称为默认实现。如果添加一个默认实现方法（不能是抽象方法），就不需要在子类中去实现，所以继承这个抽象类的子类无须改动。

如果你往接口中添加方法，那么你必须改变实现该接口的类。

② **接口中只能添加抽象方法，当你添加了抽象方法，实现该接口的类就必须实现这个新添加的方法。**

因为，定义中说的很清楚，接口的实现必须实现所有的方法。所有，当然包括新添加的方法。

(7)子类能继承的数量：

抽象类在 java 语言中所表示的是一种继承关系，一个子类只能存在一个父类，但是可以存在多个接口。

java 在类的继承上并没有多继承。抽象类属于类，所以可以被继承。但子类只能继承一个父类。

java 为了实现多继承，使用了接口。一个类可以实现多个接口。继承就好比生了孩子，只能有一个爹，但是这个孩子可以学语文，学数学，学英语等等很多东西，而语文、数学、英语就相当于接口。

总的来说，因为 java 中抽象类只有单继承，接口就可以实现多继承。

2. 抽象类的意义

抽象类：

一个类中如果包含抽象方法，这个类应该用 abstract 关键字声明为抽象类。

意义：

1）为子类提供一个公共的类型；

2）封装子类中重复内容（成员变量和方法）；

3）定义有抽象方法，子类虽然有不同的实现，但该方法的定义是一致的。

3. 抽象类与接口的应用场景

接口（interface）的应用场合：

1）类与类之前需要特定的接口进行协调，而不在乎其如何实现。

2）作为能够实现特定功能的标识存在，也可以是什么接口方法都没有的纯粹标识。

3）需要将一组类视为单一的类，而调用者只通过接口来与这组类发生联系。

4）需要实现特定的多项功能，而这些功能之间可能完全没有任何联系。

抽象类（abstract class）的应用场合：

一句话，在既需要统一的接口，又需要实例变量或缺省的方法的

情况下，就可以使用它。最常见的有：

1) 定义了一组接口，但又不想强迫每个实现类都必须实现所有的接口。可以用 abstract class 定义一组方法体，甚至可以是空方法体，然后由子类选择自己所感兴趣的方法来覆盖。
2) 某些场合下，只靠纯粹的接口不能满足类与类之间的协调，还必需类中表示状态的变量来区别不同的关系。abstract 的中介作用可以很好地满足这一点。
3) 规范了一组相互协调的方法，其中一些方法是共同的，与状态无关的，可以共享的，无需子类分别实现；而另一些方法却需要各个子类根据自己特定的状态来实现特定的功能

4. 抽象类是否可以没有方法和属性？

抽象类专用于派生出子类，子类必须实现抽象类所声明的抽象方法，否则，子类仍是抽象类。 包含抽象方法的类一定是抽象类，但抽象类中的方法不一定是抽象方法。

抽象类中可以没有抽象方法，但有抽象方法的一定是抽象类（如 HttpServlet）。但即使抽象类中没有抽象方法，也不能被 new 出来。

没有抽象类方法的抽象类的存在价值在于：类已经定义好了，不能改变其中的方法体（实例化出来的对象满足不了要求），只有继承并重写了他的子类才能满足要求，所以才把它定义为没有抽象方法的抽象类。

5. 接口的意义

1) **重要性:**在 Java 语言中，abstract class(抽象类）和 interface（接口） 是支持抽象类定义的两种机制。正是由于这两种机制的存在，才赋予了 Java 强大的 面向对象能力。

2)

3) **简单、规范性：**如果一个项目比较庞大，那么就需要一个能理清所有业务的架构师来定义一些主要的接口，这些接口不仅告诉开发人员你需要实现那些业务，而且也将命名规范限制住了（防止一些开发人员随便命名导致别的程序员无法看明白）。

4) **维护、拓展性：**比如有一个类，实现了某个功能，突然有一天，发现这个类满足不了需求了，然后又要重新设计这个类，更糟糕是你可能要放弃这个类，那么其他地方可能有引用他，这样修改起来很麻烦。

如果一开始定义一个接口，把功能放在接口里，然后定义类时实现这个接口，然后只要用这个接口去引用实现它的类就行了，以后要换的话只不过是引用另一个类而已，这样就达到维护、拓展的方便性。

5) **安全、严密性：**接口是实现软件松耦合的重要手段，它描叙了系统对外的所有服务，而不涉及任何具体的实现细节。这样就比较安全、严密一些（jdk 中很多方法就是实现了某个接口）。