什么是**接口**？

接口是包含一组虚方法的抽象类型，其中每一种方法都有其名称、参数和返回值。

接口方法不能包含任何实现，CLR允许接口可以包含事件、属性、索引器、静态方法、静态字段、静态构造函数以及常数。但是注意：C#中不能包含任何静态成员。

一个类可以实现多个接口，当一个类继承某个接口时，它不仅要实现该接口定义的所有方法，还要实现该接口从其他接口中继承的所有方法。

什么是**抽象类**？

抽象类提供多个派生类共享基类的公共定义，它既可以提供抽象方法，也可以提供非抽象方法。抽象类不能实例化，必须通过继承由派生类实现其抽象方法，因此对抽象类不能使用new关键字，也不能被密封。如果派生类没有实现所有的抽象方法，则该派生类也必须声明为抽象类。另外，实现抽象方法由overriding方法来实现。

## **三、相同点和不同点**

### **3.1 相同点**

1. 都不能被直接实例化，都可以通过继承实现其抽象方法。

2. 都是面向抽象编程的技术基础，实现了诸多的设计模式。

### **3.2 不同点**

1. 接口支持多继承；抽象类不能实现多继承。

2. 接口只能定义抽象规则；抽象类既可以定义规则，还可能提供已实现的成员。

3. 接口是一组行为规范；抽象类是一个不完全的类，着重族的概念。

4. 接口可以用于支持回调；抽象类不能实现回调，因为继承不支持。

5. 接口只包含方法、属性、索引器、事件的签名，但不能定义字段和包含实现的方法；抽象类可以定义字段、属性、包含有实现的方法。

6. 接口可以作用于值类型和引用类型；抽象类只能作用于引用类型。例如，Struct就可以继承接口，而不能继承类。

通过相同与不同的比较，我们只能说接口和抽象类，各有所长，但无优略。在实际的编程实践中，我们要视具体情况来酌情量才，但是以下的经验和积累，或许能给大家一些启示，除了我的一些积累之外，很多都来源于经典，我相信经得起考验。所以在规则与场合中，我们学习这些经典，最重要的是学以致用，当然我将以一家之言博大家之笑，看官请继续。

### **3.3 规则与场合**

1.         请记住，面向对象思想的一个最重要的原则就是：**面向接口编程**。

2.         借助接口和抽象类，23个设计模式中的很多思想被巧妙的实现了，我认为其精髓简单说来就是：面向抽象编程。

3.         抽象类应主要用于关系密切的对象，而接口最适合为不相关的类提供通用功能。

4.         接口着重于CAN-DO关系类型，而抽象类则偏重于IS-A式的关系；

5.         接口多定义对象的行为；抽象类多定义对象的属性；

6.         接口定义可以使用public、protected、internal 和private修饰符，但是几乎所有的接口都定义为public，原因就不必多说了。

7.         在接口中，所有的方法都默认为public。

8.         “接口不变”，是应该考虑的重要因素。所以，在由接口增加扩展时，应该增加新的接口，而不能更改现有接口。

9.         尽量将接口设计成功能单一的功能块，以.NET Framework为例，IDisposable、IDisposable、IComparable、IEquatable、IEnumerable等都只包含一个公共方法。

10.     接口名称前面的大写字母“I”是一个约定，正如字段名以下划线开头一样，请坚持这些原则。

11.     如果预计会出现版本问题，可以创建“抽象类”。例如，创建了狗（Dog）、鸡（Chicken）和鸭（Duck），那么应该考虑抽象出动物（Animal）来应对以后可能出现风马牛的事情。而向接口中添加新成员则会强制要求修改所有派生类，并重新编译，所以版本式的问题最好以抽象类来实现。

12.     从抽象类派生的非抽象类必须包括继承的所有抽象方法和抽象访问器的实实现。

13.     对抽象类不能使用new关键字，也不能被密封，原因是抽象类不能被实例化。

14.     在抽象方法声明中不能使用 static 或 virtual 修饰符。

**MSDN这样建议**：

1.         **如果预计要创建组件的多个版本，则创建抽象类**。抽象类提供简单易行的方法来控制组件版本。通过更新基类，所有继承类都随更改自动更新。另一方面，接口一旦创建就不能更改。如果需要接口的新版本，必须创建一个全新的接口。

**2.** **如果创建的功能将在大范围的全异对象间使用，则使用接口。抽象类应主要用于关系密切的对象，而接口最适合为不相关的类提供通用功能。**

3.         **如果要设计小而简练的功能块，则使用接口。如果要设计大的功能单元，则使用抽象类**。

4.         **如果要在组件的所有实现间提供通用的已实现功能，则使用抽象类**。抽象类允许部分实现类，而接口不包含任何成员的实现。

## **四、经典示例**

.NET Framework中关于接口和抽象类的使用：

**集合类使用了基于接口的设计**，请关注System.Collections中关于接口的设计实现；

**数据流相关类使用了基于抽象类的设计**，请关注System.IO.Stream类的抽象类设计机制。