接口是公开的，里面不能有私有的方法或变量，是用于让别人使用的，而抽象类是可以有私有方法或私有变量的，

另外，实现接口的一定要实现接口里定义的所有方法，而实现抽象类可以有选择地重写需要用到的方法，

一般的应用里，最顶级的是接口，然后是抽象类实现接口，最后才到具体类实现。

还有，接口可以实现多重继承，而一个类只能继承一个超类，但可以通过继承多个接口实现多重继承，接口还有标识（里面没有任何方法，如Remote接口）和数据共享（里面的变量全是常量）的作用.

**抽象类和接口的对比**

|  |  |  |
| --- | --- | --- |
| **参数** | **抽象类** | **接口** |
| 默认的方法实现 | 它可以有默认的方法实现 | 接口完全是抽象的。它根本不存在方法的实现 |
| 实现 | 子类使用**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。 | 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 |
| 构造器 | 抽象类可以有构造器 | 接口不能有构造器 |
| 与正常Java类的区别 | 除了你不能实例化抽象类之外，它和普通Java类没有任何区别 | 接口是完全不同的类型 |
| 访问修饰符 | 抽象方法可以有**public**、**protected**和**default**这些修饰符 | 接口方法默认修饰符是**public**。你不可以使用其它修饰符。 |
| main方法 | 抽象方法可以有main方法并且我们可以运行它 | 接口没有main方法，因此我们不能运行它。 |
| 多继承 | 抽象方法可以继承一个类和实现多个接口 | 接口只可以继承一个或多个其它接口 |
| 速度 | 它比接口速度要快 | 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。 |
| 添加新方法 | 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。 | 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。 |

接口和抽象类有什么区别

你选择使用接口和抽象类的依据是什么？

接口和抽象类的概念不一样。接口是对动作的抽象，抽象类是对根源的抽象。

抽象类表示的是，这个对象是什么。接口表示的是，这个对象能做什么。比如，男人，女人，这两个类（如果是类的话……），他们的抽象类是人。说明，他们都是人。

人可以吃东西，狗也可以吃东西，你可以把“吃东西”定义成一个接口，然后让这些类去实现它.

所以，在高级语言上，一个类只能继承一个类（抽象类）(正如人不可能同时是生物和非生物)，但是可以实现多个接口(吃饭接口、走路接口)。

第一点． 接口是抽象类的变体，接口中所有的方法都是抽象的。而抽象类是声明方法的存在而不去实现它的类。  
第二点． 接口可以多继承，抽象类不行  
第三点． 接口定义方法，不能实现，而抽象类可以实现部分方法。  
第四点． 接口中基本数据类型为static 而抽类象不是的。

当你关注一个事物的本质的时候，用抽象类；当你关注一个操作的时候，用接口。

抽象类的功能要远超过接口，但是，定义抽象类的代价高。因为高级语言来说（从实际设计上来说也是）每个类只能继承一个类。在这个类中，你必须继承或编写出其所有子类的

所有共性。虽然接口在功能上会弱化许多，但是它只是针对一个动作的描述。而且你可以在一个类中同时实现多个接口。在设计阶段会降低难度的。

抽象类：对一类事物的抽象。  
接口：对某一行为抽象。  
所以，个人觉得，两者最核心的区别是：**使用动机**

1. difference between abstract and interface class

 If you need to change your design, make it an interface. However, you may have abstract classes that provide some default behavior. Abstract classes are excellent candidates inside of application frameworks.

抽象类：对一类事物的抽象。  
接口：对某一行为抽象。  
所以，个人觉得，两者最核心的区别是：**使用动机**

一般的应用里，最顶级的是接口，然后是抽象类实现接口，最后才到具体类实现。

An abstract class can have shared state or functionality. An interface is only a promise to provide the state or functionality. A good abstract class will reduce the amount of code that has to be rewritten because it's functionality or state can be shared. The interface has no defined information to be shared