一.封装

向对象程序设计中，一个非常重要的技术便是封装，也就是把客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。这样做的好处在于可以使类内部的具体实现透明化，只要其他代码不依赖类内部的私房数据，你便可以安心修改这些代码。此外，这样做也是出于安全方面的考虑，如果代表网上支付卡密码的变量随便就可以被访问到，这样的系统谁还敢用呢？

访问修饰符：

Private:只有类本身能存取.

Protected:类和派生类可以存取.

Internal:只有同一个项目中的类可以存取.

Protected Internal:是Protected和Internal的结合.

Public:完全存取.

二.多态

１、多态：同一操作作用于不同的对象，可以有不同的解释，产生不同的执行结果。在运行时，可以通过指向基类的指针，来调用实现派生类中的方法。

编译时的多态性：

编译时的多态性是通过重载来实现的。对于非虚的成员来说，系统在编译时，根据传递的参数、返回的类型等信息决定实现何种操作。

运行时的多态性：

运行时的多态性就是指直到系统运行时，才根据实际情况决定实现何种操作。C#中，运行时的多态性通过虚成员实现。

编译时的多态性为我们提供了运行速度快的特点，而运行时的多态性则带来了高度灵活和抽象的特点。

２、实现多态：

1.接口多态性。

2.继承多态性。

3.通过抽象类实现的多态性。

３、override关键字：

重写父类中的virtual修饰的方法，实现多态。

三.继承

继承主要实现重用代码，节省开发时间。

１、C#中的继承符合下列规则：

1.继承是可传递的。如果C从B中派生，B又从A中派生，那么C不仅继承了B中声明的成员，同样也继承了A中的成员。Object类作为所有类的基类。

2.派生类应当是对基类的扩展。派生类可以添加新的成员，但不能除去已经继承的成员的定义。

3.构造函数和析构函数不能被继承。除此之外的其它成员，不论对它们定义了怎样的访问方式，都能被继承。基类中成员的访问方式只能决定派生类能否访问它们。

4.派生类如果定义了与继承而来的成员同名的新成员，就可以覆盖已继承的成员。但这并不因为这派生类删除了这些成员，只是不能再访问这些成员。

5.类可以定义虚文法、虚属性以及虚索引指示器，它的派生类能够重载这些成员，从而实现类可以展示出多态性。

２、new关键字

如果父类中声明了一个没有friend修饰的protected或public方法，子类中也声明了同名的方法。则用new可以隐藏父类中的方法。（不建议使用）

３、base关键字

base 关键字用于从派生类中访问基类的成员：

1.调用基类上已被其他方法重写的方法。

2.指定创建派生类实例时应调用的基类构造函数



