高级 C#语言特性

1 继承

1）派生类将继承基类除了构造函数和析构函数的所有成员。

2）与 C++不同，C#只支持类的单一继承。

3）C#提供了关键字 base 来访问基类成员，调用基类构造函数。

4) 可以用 new 关键字来覆盖基类的成员

2 多态

1）C#通过使用关键字 virtual 在基类中定义虚方法，用 override 在派生类中重载虚方法实现多态。

3 抽象(abstract)

1）抽象类用 abstract 关键字声明，只能作为基类， 不能实例化。

2）抽象类可以包含抽象成员或非抽象成员， 如果包含的是抽象成员，不能用{}实现 ,并且不能在派生类中用 base 来访问。抽象方法只能在抽象类中使用。

3）对抽象类不能使用 sealed 关键字。

4)从抽象类派生的非抽象类必须通过重载实现它所继承来的所有抽象成员。

4 密封(sealed)

1）密封类即不能被继承的类，密封类不可能同时又是抽象类。

2）密封方法的目的是使方法所在类的派生类无法重载该方法，所以，密封方法必须是对基类虚方法的重载。

5 操作符重载

1）C#中，所有的操作符都属于类的静态方法（public static）。所有参数都必须在参数列表中列出（其本身不能作为参数）。

2）C++中，操作符不是类的静态方法，并且其自身作为第一个参数。或者，操作符不是类的成员，而是作为 friend 方法，此时需列出所有参数，类似于 C#。

6 类型转换

1）隐式类型转换：低精度到高精度转换(或整数到浮点数转换，精度下降值不变)；十进制 0 到任何枚举类型的转换。

2）显式类型转换：高精度到低精度；数值类型到枚举类型的转换；用 Convert 类进行显式转换（转换失败时抛出异常）。

3）类型的引用转换：派生类类型引用到基类类型引用的转换。null 或基类类型的引用到派生类的转换（转换失败抛出异常）。

4）装箱与拆箱：装箱：值类型的值--拷贝-->object 实例；拆箱：Object 实例-- 拷贝-->值类型的值。

7 结构

1）结构使一种值类型，类是一种引用类型

2）结构不支持继承（但可以实现接口）

3）结构中不能声明其默认构造函数。

4）由于是值类型，结构可以用 new，也可以不用 new 关键字来创建实例。

5）C#文档指出：小于 16 字节的类如果作为结构来处理可能更高效。

8 接口

1）接口组合一系列相关的操作，通过类或结构来实现。

2）接口成员默认的访问方式是 public

3）接口成员可以是方法，性质，索引指示器或事件，但不能是字段。

4）接口支持多继承

5）如果一个类从多个接口继承而来，而其接口中有重复的方法声明， 此时可以用接口的显式成员实现，显式实现只能通过接口调用。

9 集合

1）集合是一组可以通过遍历每个元素来访问的一组对象，推荐利用.net 提供的相关几何类来派生自己的集合。

2）一个集合如果实现了 IEnumerable， 同时也要实现 IEnumerator.

3）Array 与 ArrayList 的比较：

(1)Array 可以定义为多维数组，而 ArrayList 只能是一维的

(2)Array 可以定义自己的下限，而 ArrayList 的下限始终为 0

(3)Array 的元素通常是特定类型，而 ArrayList 的元素都是 Object 类型，因此在操作 ArrayList 的元素时通常要进行装箱拆箱操作，影响性能。

(4)Array 的元素数目是固定的，而 ArrayList 的元素数目是可以自动扩展的。

(5)Array 只提供了单个元素的操作， 而 ArrayList 还提供了操作一个范围内元素的方法。

10 索引器

1）索引器(Indexer)提供了对类的数组式访问功能。

2）一般写成这样： public string this[int ind]{get{} set{}}

11 异常处理

1）三种类型的异常：SystemException; IOException; ApplicationException, 用户自定义的异常需要从 ApplicationException 派生。

2）异常抛出和捕获

```
try
  {
  }
catch([expression])
  {
  }
finally
  {
  }
```

即使 try ,catch 块中存在 break,continue,goto 或 return 语句 ,finally 中的代码始终会被执行。

3）可以包含多个 catch 语句，但出现异常时只会执行第一个匹配的 catch 块， 所以， 如果放在前面的异常类型是后面异常类型的基类或类型相同，就会发生错误。

4)catch 语句处理异常时， 可以重新抛出异常 ,新异常中可以嵌入原异常，这样就可以知道整个异常抛出的过程了。通过 Exception 的 InnerException 属性可以获取原来的异常。

12 委托

1）委托是一种引用类型，它用一组特定的参数和返回类型来封装方法， 类似与 C 中的函数指针，但跟具安全性。

2）组合委托(Multicase Delegate):组合委托只能组合相同类型的委托，且其返回类型必须是 void,可以用+, -, +=, -=来实现。

13 事件

1）订阅事件的类(subscriber)也可以成为发布事件的类(publisher)的用户，当 publisher 产生事件时（委托被执行），所有的用户都将得到通知，并提供对这些事件的响应（和该委托相关联的处理方法被自动执行）。

2）发布事件的类定义用委托声明的事件 ,在用户类中定义响应事件的方法，该方法与事件通过委托关联。

3）声明一个事件一般如下：

```
public delegate void EventHandler(object sender, EventArgs e);
public event EventHandler OnEvent;
```

14 预处理指令

1）与 C++不同， C#没有独立的预处理指令，其预处理指令其实是由编译器来执行的。

2）一行只能有一条预处理指令。

3） #define/#undef; #if/#elif/#else/#endif; #warning/#error; #line;/#region/#endregion

15 属性(Attribute)

1）在类声明，类成员声明等之前，用于指定某些附加信息。

2）把属性应用于程序实体，他们所指定的附加信息可以用在程序的文档说明中做解释型工作，可以在程序运行时用反射技术检索这些信息。

3）属性是通过属性类的声明定义的，派生于 System.Attributes.

4）Conditional,Obsolete 属性的使用。

16 组件与程序集

1）DLL Hell：当多个应用程序试图共享一个公用组件（如某个动态连接库（DLL）或某个组件对象模型（COM）类）时所引发的一系列问题。最典型的情况是，某个应用程序将要安装一个新版本的共享组件，而该组件与机器上的现有版本不向后兼容。虽然刚安装的应用程序运行正常，但原来依赖前一版本共享组件的应用程序也许已无法再工作。

2）程序集包括：程序集清单；类型元数据；MSIL 代码；资源

3）程序集特点：自说明（不用再去查询注册表或其他信息）；并行加载（同一程序的不同版本可以在系统中同时使用）；零影响安装（只需复制）