Clases Abstractas y virtuales

La palabra clave abstract permite crear clases y miembros class que están incompletos y se deben implementar en una clase derivada.

La palabra clave sealed permite impedir la herencia de una clase o de ciertos miembros de clase marcados previamente como virtual.

Clases y miembros de clase abstractos

Las clases se pueden declarar como abstractas si se incluye la palabra clave `abstract` antes de la definición de clase. Por ejemplo:

```
public abstract class A
{
    // Class members here.
}
```

No se pueden crear instancias de una clase abstracta. El propósito de una clase abstracta es proporcionar una definición común de una clase base que múltiples clases derivadas pueden compartir. Por ejemplo, una biblioteca de clases puede definir una clase abstracta que se utiliza como parámetro para muchas de sus funciones y solicitar a los programadores que utilizan esa biblioteca que proporcionen su propia implementación de la clase mediante la creación de una clase derivada.

Las clases abstractas también pueden definir métodos abstractos. Esto se consigue agregando la palabra clave `abstract` antes del tipo de valor que devuelve el método. Por ejemplo:

```
public abstract class A
{
    public abstract void DoWork(int i);
}
```

Los métodos abstractos no tienen ninguna implementación, de modo que la definición de método va seguida por un punto y coma en lugar de un bloque de método normal. Las clases derivadas de la clase abstracta deben implementar todos los métodos abstractos. Cuando una clase abstracta hereda un método virtual de una clase base, la clase abstracta puede reemplazar el método virtual con un método abstracto. Por ejemplo:

```
// compile with: -target:library
public class D
```

```
{
    public virtual void DoWork(int i)
    {
        // Original implementation.
    }
}

public abstract class E : D
{
    public abstract override void DoWork(int i);
}

public class F : E
{
    public override void DoWork(int i)
    {
        // New implementation.
    }
}
```

Si un método `virtual` se declara como `abstract`, sigue siendo virtual para cualquier clase que herede de la clase abstracta. Una clase que hereda un método abstracto no puede tener acceso a la implementación original del método: en el ejemplo anterior, `DoWork` en la clase F no puede llamar a `DoWork` en la clase D. De esta manera, una clase abstracta puede exigir a las clases derivadas que proporcionen nuevas implementaciones de método para los métodos virtuales.