Abstract Classes

As described in the previous example, polymorphism is used when you have different derived classes with the same method, which has different implementations in each class. This behavior is achieved through **virtual** methods that are **overridden** in the derived classes.

In some situations there is no meaningful need for the virtual method to have a separate definition in the base class.

These methods are defined using the **abstract** keyword and specify that the derived classes must define that method on their own.

You cannot create objects of a class containing an abstract method, which is why the class itself should be abstract.

We could use an abstract method in the Shape class:

```cs
abstract class Shape {
   public abstract void Draw();
}
```

As you can see, the **Draw** method is **abstract** and thus has no body. You do not even need the curly brackets; just end the statement with a semicolon.

The Shape class itself must be declared **abstract** because it contains an **abstract** method. Abstract method declarations are only permitted in abstract classes.

Remember, **abstract** method declarations are only permitted in **abstract** classes. Members marked as **abstract**, or included in an abstract class, must be implemented by classes that derive from the abstract class. An abstract class can have multiple abstract members.

Continue