A DEEP DIVE INTO C# ABSTRACT CLASS

Abstract class is a special type of class which cannot be instantiated and acts as a base class for other classes. Abstract class members marked as abstract must be implemented by derived classes.

The purpose of an abstract class is to provide basic or default functionality as well as common functionality that multiple derived classes can share and override.

For example, a class library may define an abstract class that is used as a parameter to many of its functions, and require programmers using that library to provide their own implementation of the class by creating a derived class. In C#, System.IO.FileStream is an implementation of the System.IO.Stream abstract class.

```
1. abstract class ShapesClass
2. {
3.  abstract public int Area();
4. }
5. class Square : ShapesClass
6. {
7.  int side = 0;
8.
9.  public Square(int n)
10.   {
11.   side = n;
12.   }
13.   // Override Area method
14.   public override int Area()
15.   {
16.   return side * side;
17.   }
18.  }
19.
20.  class Rectangle : ShapesClass
21.  {
22.   int length = 0, width=0;
23.
24.   public Rectangle (int length, int width)
```

```
25.   {
26.   this.length = length;
27.   this.width = width;
28.   }
29.   // Override Area method
30.   public override int Area()
31.   {
32.   return length * width;
33.   }
34.  }
```

Features of Abstract Class

1. An abstract class cannot be instantiated.
2. An abstract class contain abstract members as well as non-abstract members.
3. An abstract class cannot be a sealed class because the sealed modifier prevents a class from being inherited and the abstract modifier requires a class to be inherited.
4. A non-abstract class which is derived from an abstract class must include actual implementations of all the abstract members of parent abstract class.
5. An abstract class can be inherited from a class and one or more interfaces.
6. An Abstract class can has access modifiers like private, protected, internal with class members. But abstract members cannot have private access modifier.
7. An Abstract class can has instance variables (like constants and fields).
8. An abstract class can has constructors and destructor.
9. An abstract method is implicitly a virtual method.
10. Abstract properties behave like abstract methods.
11. An abstract class cannot be inherited by structures.
12. An abstract class cannot support multiple inheritance.

Common design guidelines for Abstract Class

1. Don't define public constructors within abstract class. Since abstract class cannot be instantiate and constructors with public access modifiers provides visibility to the classes which can be instantiated.
2. Define a protected or an internal constructor within an abstract class. Since a protected constructor allows the base class to do its own initialization when sub-classes are created and an internal constructor

can be used to limit concrete implementations of the abstract class to the assembly which contains that class.

When to use

1. Need to create multiple versions of your component since versioning is not a problem with abstract class. You can add properties or methods to an abstract class without breaking the code and all inheriting classes are automatically updated with the change.
2. Need to to provide default behaviors as well as common behaviors that multiple derived classes can share and override.

Source : http://www.dotnet-tricks.com/Tutorial/csharp/E0K2011113-A-Deep-Dive-into-C#-Abstract-Class.html