C# Intermediate: Classes, Interfaces and OOP

POLYMORPHISM

Method Overriding

- Method overriding means changing the implementation of an inherited method.
- If a declare a method as **virtual** in the base class, we can **override** it in a derived class.

```
public class Shape
{
    public virtual Draw()
    {
        // Default implementation for all derived classes
    }
}

public class Circle : Shape
{
    public override Draw()
    {
        // Changed implementation
    }
}
```

- This technique allows us to achieve polymorphism. Polymorphism is a great object-oriented technique that allows use get rid of long procedural switch statements (or conditionals).

Abstract Classes and Members

- Abstract modifier states that a class or a member misses implementation. We use abstract members when it doesn’t make sense to implement them in a base class. For example, the concept of drawing a shape is too abstract. We don’t know how to draw a shape. This needs to be implemented in the derived classes.
- When a class member is declared as abstract, that class needs to be declared as abstract as well. That means that class is not complete.
- In derived classes, we need to override the abstract members in the base class.

```
public abstract class Shape
{
    // This method doesn’t have a body.
    public abstract Draw();
}

public class Circle : Shape
{
    public override Draw()
    {
        // Changed implementation
    }
}
```

- In a derived class, we need to override all abstract members of the base class, otherwise that derived class is going to be abstract too.
- Abstract classes cannot be instantiated.

Sealed Classes and Members

- If applied to a class, prevents derivation from that class.
- If applied to a method, prevents overriding of that method in a derived class.

- The string class is declared as sealed, and that’s why we cannot inherit from it.

```
public sealed class String
{   }
```