DNPExam

Sunday, January 13, 2019   7:12 PM

```
Console Application                               console          [C#], F#, VB      Common/Console
Unit Test Project                                 mstest           [C#], F#, VB      Test/MSTest
NUnit 3 Test Project                              nunit            [C#], F#, VB      Test/NUnit
NUnit 3 Test Item                                 nunit-test       [C#], F#, VB      Test/NUnit
xUnit Test Project                                xunit            [C#], F#, VB      Test/xUnit
Razor Page                                        page             [C#]              Web/ASP.NET
ASP.NET Core Empty                                web              [C#], F#          Web/Empty
ASP.NET Core Web App (Model-View-Controller)      mvc              [C#], F#          Web/MVC
ASP.NET Core Web App                              webapp           [C#]              Web/MVC/Razor Pages
ASP.NET Core Web API                              webapi           [C#], F#          Web/WebAPI
```

```
/* 01 */
var array = new int[3] {1, 2, 3}
var numbers = new List<int>() {1, 2, 3};
string list = String.Join(", ", numbers);

// interpolation
string name = $"{firstName} {lastName}";

// conversions
string s = "100";
int i = int.Parse(s);
// manipulate strings with StringBuilder()

// Structs and Enums
public struct RgbColor {
    public int Red;
    public int Blue;
    public int Green;
}

public enum Directions {
    Up = 0,
    Down = 1
}


/* 02 */

public class Student : Human
{
    public Student(string name)
        :base
    {...}
}

// method overriding: original method must be virtual, other should be "override"
// "hide" base class implementations by using "new" instead of "override" (virtual not necessary)
// BaseClass.x vs x
// "sealed" prevents derivations or overrides

// Indexers: allow instances of class to be indexed just like arrays
var stringCollection = new SampleCollection<string>();
stringCollection[0] = "what";


/* 03 */

// Extension methods: Add methods to an existing class without changing its source code
// are static

public static int WordCount(this String str)
{...}

// Params: do something for every element in the argument
```

```
public static void UseParams2(params object[] list)
    {
        for (int i = 0; i < list.Length; i++)
        {
            Console.Write(list[i] + " ");
        }
        Console.WriteLine();
    }

UseParams(1, 2, 3, 4);

// ref: a reference of the variable is being passed, not the value
// out:

static void Method(out int i)
{ i = 44; }

static void Main()
{
    int value;
    Method(out value);
    Console.WriteLine(value);
}

// Operator overloading

public static Complex operator +(Complex c, Complex d)
{ return new Complex(..) }

// or in shorthand
public static Complex operator +(Complex c, Complex d) => new Complex(..);

// Generics

public class GenericList<T>
{
    public void Add(T value) {..}
    public T this[int index] {..}
}

// Lambda Expressions

static int Square(int number) { return number * number; }
// > can be used inline instead, if more convenient
Func <int, int> square = number => number * number;

// Delegates
public delegate void Del(string message);
public static void DelegateMethod(string message)
{
    Console.WriteLine(message);
}

Del handler = DelegateMethod; // instantiate the delegate
handler("Hello World"); // call the delegate

// multicast Delegates: executes 2 delegates

Del handler = DelegateMethod;
handler += AnotherDelegateMethod;
handler("what");

// predicate delegates:
var cheapBooks 0 books.FindAll(IsCheaperThan10Dollars);

static bool IsCheaperThan10Dollars(Book b) { return b.Price < 10; }


/* 04 */

// Serialize with [Serializable] above the class declaration
```

/* 06 */ - Unit testing

```
[Theory]
[InlineData(-1)]
[InlineData(0)]
[InlineData(1)]
public void ReturnFalseGivenValuesLessThan2(int value)
{
    var result = _primeService.IsPrime(value);

    Assert.False(result, $"{value} should not be prime");
}
```

/* 08 */ - Consuming Web Services = clients

```
dotnet new console
```

/* 09 */ - Exposing Web Services = servers

```
dotnet new webapi
```

/* 14 */ - Security

Common vulnerabilities

- Cross-site scripting: injecting a js on a page (that might send session data or cookie)
- SQL injection
- Cross-site request forgery: links to pages that send malicious requests to 3rd parties
- Open redirect attacks: user gets redirected to malicious site

Authentication based on

- Claims
- Tokens: like Oauth / Facebook