CODING STYLE

Read the Microsoft C# Coding

- Your homework is to read: https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions
- I expect you to now to adhere to what is outlined in today's class notes in the next assignment, and will be grading you for it

Be consistent with spacing

```
Console.WriteLine($"Type of num1 with value {num1} is {num1.GetType()}");
Console.WriteLine($"Type of num2 with value{num2} is {num2.GetType()}");
```

Don’t use Superfluous Blank Lines

```
public static class Tests
{



    [Test]
    public static void Testliterals()
    {
```

Don’t use explicit variable types if not needed

```
char ch = (char)i;

for (int i = 0; i <= 127; i++)
```

Do use “var” declarations

```
var s = "Hello";
var num1 = 123;
var num2 = 12.34;
var num3 = 1.23f;
var num4 = true;
var c = 't';
var h = 0xFFul;
```

Don’t use Formatted Console WriteLine Instead use String Interpolation

```
Console.WriteLine("{0}", ch);
```

Use nameof operator for identifier names

```
Console.WriteLine($"Type of s with value {s} is {s.GetType()}");
Console.WriteLine($"Type of {nameof(s)} with value {s} is {s.GetType()}");
```

Test: what could be improved?

```
[Test]
public static void TestLetters(char x)
{
    for (int i = 0; i <= 127 ; i++)
    {
        char x = (char)i;
        if (Char.IsLetter(x))
        {
            Console.Write(x);

        }

    }

}
```

Be consistent in formatting

```
Console.WriteLine("Type of Var '"+a+"' is"+a.GetType());
Console.WriteLine("Type of Var '" + b + "' is" + b.GetType());
```

Use string Interpolation instead of Concatenation

```
Console.WriteLine("Type of Var '" + b + "' is" + b.GetType());

Console.WriteLine($"Type of Var '{b}' is {b.GetType()}");
```

Don’t compare Boolean values to true
(and put space after the if)

```
if(Char.IsLetter(currChar) == true)
```

Is this executed 127 or 128 times?
(don’t forget to use “var”)

```
for (int i = 0; i < 127; i++)
```

Loop using “<“ rather than “<=“
(it becomes more explicit, less error prone)

```
for (var i = 0; i < 128; i++)
```

Empty SetUp() not needed (harmless)

```
[SetUp]
public void Setup()
{
}
```

Using a loop is excellent

```
object[] xs = { a, b, c, d, e, f, g };
var i = 0;
foreach (var x in xs )
{
    Console.WriteLine($"Value {i++} is {x} and is of type {x.GetType()}");
}
```

Function name could be replace comment
(also put a space after '//')

```
//Functiont to print type of argument supplied
public void TestLiterals(object salut){
    Console.WriteLine(salut.GetType());
}
```

Use keyword char not Char

```
var aLetter = (Char)i;

if (Char.IsLetter(a))
```

Comment doesn’t add anything here

```
if (Char.IsLetter(aLetter)) { //check if char alphabetical
```

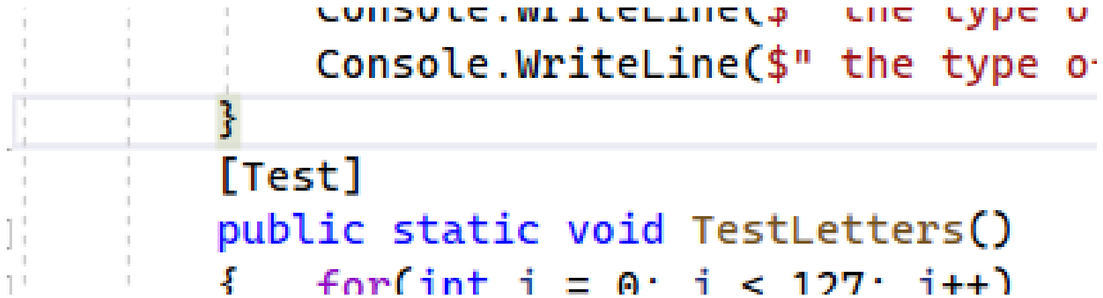
Adding '\0' not necessary
(only because we know it is zero by definition)

```
var ascii = '\0';
for (var i = 0; i < 128; i++)
{
    var x = (char)(ascii + i);
```

Using var is better
(because you don't lose the type info)

```
var tested = "Hello";
object t = tested.GetType();
Console.WriteLine($"\"{tested}\" is of type {t}\n");
```

Put one empty line between functions



Put a space after for
(and a newline after {)

```
public static void TestLetters()
{   for(int i = 0; i < 127; i++)
    {
```

Indent line after “if” statement
(and be careful of superfluous cast or comparison)

```
if(Char.IsLetter(ch) == true )
Console.WriteLine($" the char of {(int)i}
```

Foreach almost always uses a var
(unless the enumerable is not typed)

```
foreach (object value in literals)
```

Consistent Spacing
(and use var)

```
for (char i = (char)0; i<=127;i++)
```

Nice: Clear and Concise
(prefer the space after { and before })

```
object[] literals = {"Hello", 123, 12.34, 1.23f, true, 't', 0xFFul};

object[] xs = { "Hello", 123, 12.34, 1.23f, true, 't', 0xFFul };
```

I like the function
(but the comments take away from it)

```
public static void TestLiterals(object literal)
{
    //get the parsed in literal/object run-time type
    var literalType = literal.GetType();
    //output the literal's data type
    Console.WriteLine($"{literalType}");

}
```

What I am looking for: Part 1

```
[Test]
public static void TestLiterals()
{
    var xs = new object[] { "Hello", 123, 12.34, 1.23f, true, 't', 0xFFul };
    foreach (var x in xs)
    {
        OutputValueAndType(x);
    }
}

public static void OutputValueAndType(object value)
{
    Console.WriteLine($"The value {value} has type {value.GetType()}");
}
```

What I am looking for: Part 2

```
[Test]
public static void TestLetters()
{
    for (var i = 0; i < 128; i++)
    {
        var c = (char)i;
        if (char.IsLetter(c))
        {
            Console.WriteLine($"{i} = {c}");
        }
    }
}
```

Preview of Using LINQ

- There is no logic, so less room for mistakes
- Transforming a collection into another collection using cast
- Filtering the collection using a predicate
- Enumerate the items and output them

```
[Test]
public static void TestLettersUsingLinq()
{
    foreach (var c in Enumerable.Range(0, 128).Cast<char>().Where(char.IsLetter))
        Console.WriteLine($"{(int)c} = {c}");
}
```