Software Craftsmanship Guild

526 S. Main St,  Suite 609

Akron, OH 44311

# Enumerations

Software Craftsmanship Guild

# Lesson Goals

- Learn how Enumerations (enums) are used in C# to simplify code
- Learn some neat tricks for working with them.

# What's an Enum?

- An enumeration, or enum, is a programmer defined type (just like classes are programmer defined types)
- Enums are *value types* that only have two members:
  1. Named constants
  2. Integer values

## Example Enum

Here is an enumeration representing a traffic light. It has three members, Green, Yellow, and Red.

Notice that the members are comma separated.

Be aware that every label has an underlying integer behind it, which is optional. If you don't specify it will default to 0 and count up.

So the two block of code to the right do the same thing

```
public enum TrafficLight
{
    Green,
    Yellow,
    Red
}
```

```
public enum TrafficLight
{
    Green = 0,
    Yellow = 1,
    Red = 2
}
```

# Assigning and Casting

- We can declare variables and properties as the enumeration type, but if we want to actually print or store the underlying integer we must cast it.
- The typical method of casting an enum value is to say (int)Variable
- We can use the Enum.GetName() method if we want to actually display the text value of the enum

Enum in a traffic light

# DEMO

## More About Numbering

By default, it counts up from zero.

If we specify a number then stop specifying, it will simply count up from the last known number

We can also specify a lower number in the middle of the set.

We can also duplicate numbers, and even use a previously defined label as a reference

```
public enum CardSuits
{
    Hearts,     //0
    Clubs,      //1
    Diamonds,   //2
    Spades,     //3
    MaxSuites   //4
}

public enum FaceCards
{
    Jack = 11,      //11
    Queen,          //12, not specified so it counts up
    King,           //13
    Ace,            //14
    NumberOfFaceCards = 4,
    ThisWillBe5,    //5, counts up from last value
    HighestFaceCard = Ace  // note we can reuse
}
```

# Word of Warning

- Even though enums are ints underneath, we can't compare two different enum types, we have to convert them to ints first.

```
public enum Enum1
{
    One
}

public enum Enum2
{
    One
}
```

```
if (Enum1.One == Enum2.One)
{
    // illegal, not the same type
}

if ((int) Enum1.One == (int) Enum2.One)
{

}
```

# Conclusion on Enums

- Enums are really useful for making your code more readable, especially when dealing with relatively static lists, like statuses and labels
- Typically in the case of database work we use enums for “Type” tables… OrderType, CustomerType, EmployeeStatus, etc.
- However if you want to display a real description (with spaces, etc), not just the enum name a small class is more appropriate.