Tutorial – Type Conversion

In this tutorial we are going to create a program that contains two user defined types – RomanNumeral and BinaryNumeral. We will then implement some user-defined conversion methods so that we can convert variables between these two types, both implicitly and explicitly.

Create the RomanNumeral class:

1. Create a new console program and add a new class to your program called *RomanNumeral*.
2. Insert the following code into your new class:

```
class RomanNumeral
{
    private int value;

    public RomanNumeral(int value)
    {
        this.value = value;
    }

    // Declare a conversion from an int to a RomanNumeral. Note the
    // the use of the operator keyword. This is a conversion
    // operator named RomanNumeral:
    static public implicit operator RomanNumeral(int value)
    {
        // Note that because RomanNumeral is declared as a struct,
        // calling new on the struct merely calls the constructor
        // rather than allocating an object on the heap:
        return new RomanNumeral(value);
    }

    // Declare an explicit conversion from a RomanNumeral to an int:
    static public explicit operator int(RomanNumeral roman)
    {
        return roman.value;
    }

    // Declare an implicit conversion from a RomanNumeral to
    // a string:
    static public implicit operator string(RomanNumeral roman)
    {
        return ("Conversion not yet implemented");
    }
}
```

We’ve made three type conversion operators here. These allow the class to be converted to and from other the basic types int and string. Later we’ll add another operator to convert between a class.

The conversions are defined like operators and are named for the type to which they convert. You will notice that they are also static.

To start off with we'll update our main function so that we can test these conversion operators and see how they work.

3. Update the main function as follows:

```
static void Main(string[] args)
{
    RomanNumeral numeral;

    numeral = 10;

    // Call the explicit conversion from numeral to int. Because it is
    // an explicit conversion, a cast must be used:
    Console.WriteLine((int)numeral);

    // Call the implicit conversion to string. Because there is no
    // cast, the implicit conversion to string is the only
    // conversion that is considered:
    Console.WriteLine(numeral);

    // Call the explicit conversion from numeral to int and
    // then the explicit conversion from int to short:
    short s = (short)numeral;

    Console.WriteLine(s);

    Console.ReadKey();
}
```

Run the program and see what output you get.

Exercise:

1. Implement the string conversion operator to convert your roman numeral to a string. (This should output a string showing the number as a roman numeral, not as a decimal value).

2. Add a new class to your program called BinaryNumeral.

   The interface for this class will be similar to the RomanNumeral class.
   It should:
   - store the value as an integer
   - have a constructor that takes an integer
   - an implicit operator to convert from an integer to a BinaryNumeral
   - an explicit operator to convert from a BinaryNumeral to an integer
   - an implicit operator to convert from a BinaryNumeral to a string

   Update the main program as follows to test your BinaryNumeral class:

```
static void Main(string[] args)
{
    BinaryNumeral numeral;

    numeral = 10;

    // Call the explicit conversion from numeral to int. Because it is
    // an explicit conversion, a cast must be used:
    Console.WriteLine((int)numeral);

    // Call the implicit conversion to string. Because there is no
    // cast, the implicit conversion to string is the only
    // conversion that is considered:
    Console.WriteLine(numeral);

    // Call the explicit conversion from numeral to int and
    // then the explicit conversion from int to short:
    short s = (short)numeral;

    Console.WriteLine(s);

    Console.ReadKey();
}
```

3. Add an implicit operator inside the RomanNumeral class to convert from a BinaryNumeral to a RomanNumeral.

   (Hint: a BinaryNumeral can be cast to an int, and the RomanNumeral has a constructor that accepts an int)

   Update your main program to test your new conversion operator:

```
static void Main(string[] args)
{
    RomanNumeral roman;
    roman = 10;
    BinaryNumeral binary;
    // Perform a conversion from a RomanNumeral to a
    // BinaryNumeral:
    binary = (BinaryNumeral)(int)roman;
    // Performs a conversion from a BinaryNumeral to a RomanNumeral.
    // No cast is required:
    roman = binary;
    Console.WriteLine((int)binary);
    Console.WriteLine(binary);

    Console.ReadKey();
}
```