Encapsulation means that we are enclosing something, in other words, we are trying to hide something from being accessed. In C#, the act of encapsulation will be setting variables to private, instead of public, when we create classes.

When variables are prevented from accessing from outside of the class, although it may seem that more lines of code are required in the class, it makes the Program code cleaner and more precise. Moreover, whenever there are changes involving those variables, we can simply modify them in the class, instead of modifying multiple lines of code in other files.

When we create a new class, we declare variables. Instead of setting them as public, we set them as private, to apply the concept of encapsulation in the code.

**public** **class** Fraction {

**private** int \_top;

**private** int \_bottom;

**public** Fraction() {

*// sets default value to 1/1 when no parameters are passed in*

        \_top = 1;

        \_bottom = 1;

    }

**public** Fraction(int wholeNumber) {

        \_top = wholeNumber;

        \_bottom = 1;

    }

**public** Fraction(int top, int bottom) {

        \_top = top;

        \_bottom = bottom;

    }

**public** string GetFractionString() {

        string fractionString = $"{\_top}/{\_bottom}";

        return fractionString;

    }

**public** double GetDeciamlValue() {

        double decimalValue = (double)\_top / (double)\_bottom;

        return decimalValue;

    }

}

Above is the code for the reading preparation this week, which demonstrates the concept of encapsulation by setting the variables in private. With such, the main code, or in other words, the public interface would look like the code below:

**class** Program

{

**static** void Main(string[] args)

    {

*// fraction 1: no parameters passed in*

        Fraction fraction1 = new Fraction();

        Console.WriteLine(fraction1.GetFractionString());

        Console.WriteLine(fraction1.GetDeciamlValue());

*// fraction 2: a whole number (numerator = whole number, denominator = 1)*

        Fraction fraction2 = new Fraction(5);

        Console.WriteLine(fraction2.GetFractionString());

        Console.WriteLine(fraction2.GetDeciamlValue());

*// fraction 3: a fraction*

        Fraction fraction3 = new Fraction(1, 4);

        Console.WriteLine(fraction3.GetFractionString());

        Console.WriteLine(fraction3.GetDeciamlValue());

*// fraction 4: a fraction*

        Fraction fraction4 = new Fraction(7, 2);

        Console.WriteLine(fraction4.GetFractionString());

        Console.WriteLine(fraction4.GetDeciamlValue());

    }

}

It looks much cleaner than the codes in previous assignments and would be easier to edit whenever changes are being made.