Properties Tutorial

In this exercise you will learn more about the Properties in C# and how to use them effectively.

Introduction:

A property is a special type of class member that provides an easy way to get & set the value of a private variable. Properties are accessed just like a normal variable but in reality are special methods called accessors. They fulfil the roll of a getters and setters (or accessors and mutators) in other languages, providing a safe way of accessing private members and ensuring that the values that they get set to are valid.

Getters:

Getter methods are generally extremely simple. They simply return a value when called. They look like this in the C# property style:

```
class TimePeriod
{
    private int days;

    public int Days
    {
        get
        {
            return days;
        }
    }
}

class Program
{
    static void Main(string[] args)
    {
        TimePeriod time = new TimePeriod();

        Console.WriteLine(time.Days);
    }
}
```

Compile and run it.

Note how we can access the private member, **days**, using the public property, **Days**.

Questions:

1. What happens if you try to assign a value to Days?

Setters:

Setters are the opposite of getters – they take a value in instead of returning one. They can also optionally modify or validate the data being passed in, to ensure that the value is valid or within a specific range.

```
public int Days
{
    get
    {
        return days;
    }

    set
    {
        days = value;
    }
}
```

This is our full property with setter and getter. Notice the use of the undeclared variable **value**; this represents the value we're assigning to this property. We now have full read and write properties for our private variable.

It is also possible to create more than one property for a variable. For example, the following property allows us to specify the number of days in weeks instead:

```
public int Weeks
{
    set
    {
        days = value * 7;
    }
}
```

Exercise:

We are going to create a money class that converts between different currencies. You should be able to input in one currency using a setter and retrieve another currency using a getter with the appropriate conversion rate. Use Australian Dollars as the base for your conversions.

For example if you write money.AUD = 50, then money.USD should return 40 if the conversion rate was 1 AUD = 0.8 USD.

1. Add this base functionality (ability to get and set different currencies) to your class.
2. Add some functions (or overload some operators) to add and subtract different Money classes and currencies together.