Task 2 – Use of Accessibility Modifiers with Properties

Imagine you want to store a property within the Customer class that could externally (outside the class)be read-only or write-only. Auto-Implemented Properties allow the use of modifiers to specify accessibility.
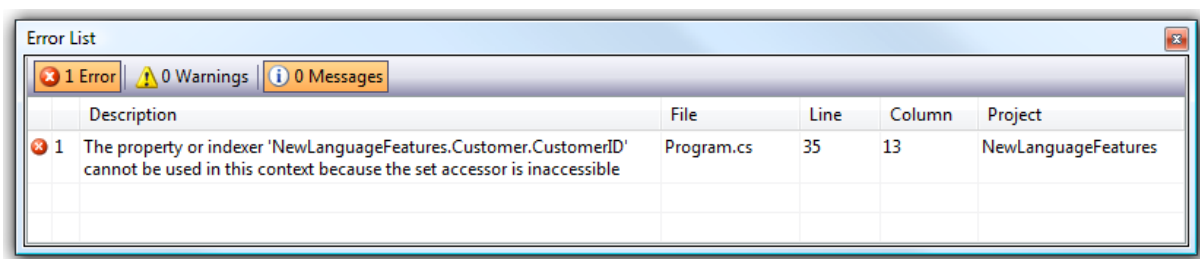
1. Return to the **Customer** class. Suppose you want to add a read-only Customer ID; to do this, add a new property **CustomerID**. Add the `private` modifier to the `set` accessor. Update the ToString() method as well to include this new property.

```
public class Customer
{
    public int CustomerID { get; private set; }
    public string Name { get; set; }
    public string City { get; set; }
    public override string ToString()
    {
        return Name + "\t" + City + "\t" + CustomerID;
    }
}
```

2. In the **Main** method, set the **CustomerID**.

```
static void Main(string[] args)
{
    Customer c = new Customer();
    c.Name = "Maria Anders";
    c.City = "Berlin";
    c.CustomerID = 1;
    Console.WriteLine(c);
}
```

3. Press **Ctrl+Shift+B** to build the solution.

4. Click the **Error List** tab to view the compiler error output.

	Description	File	Line	Column	Project
❌ 1	The property or indexer 'NewLanguageFeatures.Customer.CustomerID' cannot be used in this context because the set accessor is inaccessible	Program.cs	35	13	NewLanguageFeatures

Notice that an error occurs when you attempt to set the CustomerID property directly. This is due to the `private` modifier on `set`. The CustomerID property now behaves as if it were read-only.

5. Return to the **Customer** class and add a constructor to set the **CustomerID**.

```
public class Customer
{
    public string CustomerID { get; private set; }
    public string Name { get; set; }
```

```csharp
        public string City { get; set; }
        public Customer(int ID)
        {
            CustomerID = ID;
        }
        public override string ToString()
        {
            return Name + "\t" + City + "\t" + CustomerID;
        }
    }
```

6. In the **Main** method, update the constructor call to set the **CustomerID**.

```csharp
 static void Main(string[] args)
 {
        Customer c = new Customer(1);
        c.Name = "Maria Anders";
        c.City = "Berlin";
        Console.WriteLine(c);
 }
```

7. Press **Ctrl+F5** to build and run the application. The program now compiles without any errors. Press any key to terminate the application.