Get-Only Properties

Previously we used properties for field validation. By applying `public` and `private`, we can also use properties to control access to fields.

Recall our imaginary `Area` property. Say we want programs to get the value of the property, but we don't want programs to set the value of the property. Then we either:

- don't include a `set()` method, or
- make the `set()` method private.

This shows approach 1 — don't include a `set()`:

```
public string Area
{
  get { return area; }
}
```

We can still get `Area`, but if we try to set `Area` we get an error:

```
error CS0200: Property or indexer 'Forest.Area' cannot be assigned to (it is
read-only)
```

This shows approach 2 — make `set()` private:

```
public int Area
{
  get { return area; }
  private set { area = value; }
}
```

We can still get `Area`, but if we try to set `Area` in `Main()` we get an error:

```
error CS0272: The property or indexer 'Forest.Area' cannot be used in this
context because the set accessor is inaccessible
```

Notice that in approach 1 we get an error for setting `Area` anywhere. In approach 2 we only get an error for setting `Area` outside of the `Forest` class. Generally we prefer approach 2 because it allows other `Forest` methods to set `Area`.

☑Instructions

1.

In **Forest.cs**, define an `Age` property for the `age` field. It should have a public getter and a private setter.

Hint

To define a `CookTime` property with a public getter and private setter:

```
public int CookTime
{
  get { return cookTime; }
  private set { cookTime = value; }
}
```

2.

In **Program.cs** in `Main()`, try to set the value of `f.Age`. You should see an error.

```
error CS0272: The property or indexer 'Forest.Age' cannot be used in this context
because the set accessor is inaccessible
```

This error means that the private setter prevented us from setting `Age` outside of the class (which is good!).