Delegates

This lab builds on *Exceptions* project

Overview

Our customer has identified an additional requirement (imagine that!).

Some employees are in regions (state, province or city) that imposes an additional income tax.

In order to handle this scenario, the designer decided to add a delegate to the Employee class that can be called to obtain the additional local tax.

The signature of the delegate is `double func(double amt)`. The gross amount being paid is passed in and the local tax is returned.

Here's what you need to do for this lab:

- Use this project as a starting point, or build on *Exceptions*
- Add the tests below to EmployeeTest
- Add a delegate definition to Employee.cs that returns a double and accepts a double as an argument
- Add a property to Employee.cs named LocalTaxMethod that is of the delegate type
- In Pay(amt), modify so that if there is a LocalTaxMethod, it is called and its result is added to the tax

```
[Fact]
public void NullLocalTaxTest()
{
    var e = new Employee("Hank", "Hill", 100, DateTime.Now.AddYears(-5).AddDays(-1
    e.LocalTaxMethod = null;
    e.Pay();
}
[Fact]
public void LocalTaxCalledTest()
{
    var e = new Employee("Hank", "Hill", 100, DateTime.Now.AddYears(-5).AddDays(-1
    bool called = false;
    e.LocalTaxMethod = (amt) => { called = true; return 0; };
    e.Pay();
    Assert.True(called);
}
```

```
}
[Fact]
public void LocalTaxAddedTest()
{
    var e = new Employee("Hank", "Hill", 100, DateTime.Now.AddYears(-5).AddDays(-16
    var nt = e.Pay();
    e.LocalTaxMethod = (amt) => 10;
    Assert.Equal(nt - 10, e.Pay(), 2);
}
```