Comprehensive Final Lab

This final lab has tasks from throughout the course.

The foundation

Console applications were not a main part of the course, so here are a few specific instructions to get you started.

1. Create a new Console Application Project called "MaintainProducts" and set this as your startup project. (Hint: In Solution Explorer, right-click on the project and choose "Set as startup project").
2. In the Main method, Console.Write("Enter a Product ID: ");
3. Run and test, making sure you can see your message.
4. Have it read in a value from the user.

```
var prodID = Console.ReadLine();
```

5. Debug it to make sure you can read a value from the user.

Printing out a fake product

We've covered the rest of these things in some detail so the instructions get more high-level from this point forward.

6. Create or re-use a business classes Class Library project.
7. Add to it (or re-use) a class called Product. Make sure it has properties that look just like the columns in the Products table.
8. Add a reference to your MaintainProducts project pointing to your business classes project.
9. In your main method, instantiate and print out a made-up product.

Getting a real product

We're printing out a fake product, let's print out a real one.

10. Add Entity Framework to your projects using NuGet.
11. Create a class called NorthwindContext that inherits from DbContext.
12. Add to it a property, a DbSet<Product> called Products.
13. In your main module, instantiate a NorthwindContext.
14. Using a LINQ where method, get the one record pointed to by the prodID read from the user. (HInt: you'll need to Convert to an Int32).
15. Print some details of the one record using Console.WriteLine().

Catching exceptions

If you test with a product ID that you know is in the database, you should see the record. If you test with one that is not in the database, it will throw an exception.

16. Catch that exception and print out a nice message to the user. Tell them to try a different ID.
17. Also note that if you put in something that isn't a number, your convert will throw an exception. Catch that exception also and tell the user they should give you a number.

Your program should now either provide a nice error message or data. Cool.

Updating the record

Let's update a product now.

18. Have your program ask the user if they'd like to update this record.
19. If the user responds with a "yes", your program should ask them for a new product name, quantity per unit and unit price.
20. After the user enters those data points, have EF update the record. Don't forget to SaveChanges().
21. Run and test.

If your program can read and write records, you can be finished.