**project 0: store application**

Sept 30 2019 Arlington .NET / Nick Escalona

**functionality**

* place orders to store locations for customers
* add a new customer
* search customers by name
* display details of an order
* display all order history of a store location
* display all order history of a customer
* input validation
* exception handling
* persistent data (SQL); no products, prices, customers, etc. hardcoded in C#
* logging
* (optional: order history can be sorted by earliest, latest, cheapest, most expensive)
* (optional: get a suggested order for a customer based on his order history)
* (optional: save some or all data to disk in JSON format)
* (optional: load some or all data from disk)
* (optional: display some statistics based on order history)

**structure**

**business logic**

* class library
* contains all business logic
* contains domain classes (customer, order, store, product, etc.)
* documentation with <summary> XML comments (optional: <params> and <return>)
* has no dependency on UI or any input/output considerations

**user interface**

* interactive console application
* has only display- and input-related code
* low-priority component, will be replaced when we move to project 1

**data access**

* class library
* contains scaffolded EF DbContext
* contains data access logic but no business logic
* use repository pattern for separation of concerns

**test**

* at least 10 test methods
* use TDD for some of the application
* focus on unit testing business logic; testing the console app is very low priority

**object model**

**customer**

* has first name, last name, etc.
* (optional: has a default store location to order from)

**order**

* has a store location
* has a customer
* has an order time (when the order was placed)
* can contain multiple product types in the same order
* rejects orders with unreasonably high product quantities
* must have some additional business rules

**location**

* has an inventory
* inventory decreases when orders are accepted
* rejects orders that cannot be fulfilled with remaining inventory
* (optional: more than one inventory item decrements for a given product order, for at least one product)

**product**

**technologies**

* C#/.NET
* Entity Framework
* Azure SQL Database
* xUnit, NUnit, or MSTest
* Serilog or NLog
* (optional: SonarLint extension in VS or VS Code for static analysis)