Store App Console Version

Let's sell work appropriate stuff.

Overview:

The store manager is a console application used to help ease your shopping!

Models

- You can add more properties if you choose if it makes sense.

Customer

The customer model is supposed to hold the data concerning a customer.

Properties:

- Name
- Address
- Email/Phone number
- List of Orders

StoreFront

The store front contains information pertaining the various store locations.

Properties:

- Name
- Address
- Inventory
- List of Orders

Orders

The orders contain information about customer orders.

Properties:

- Order Line Items
- Location (that the order was placed)
- Total price

LineItems

The line items contain information about a particular product and its quantity.

Properties:

- Product
- Quantity

Products

The Product model is supposed to hold the data concerning a customer.

Properties:

- Name
- Price
- Desc. (optional)
- Category (optional)

Functionalities

Add customer

Models involved:

Customer

Logic to be implemented:

UI

Come up with some user interface that validates customer information based on model properties. So the UI should ask the end user relevant information about the customer they're creating, like the name, address, etc.

Write logic that would be result to:

//Add screen shot of the add restaurant UI

Hint: Use console readline, writeline, and maybe some looping construct to help with validation.

BL

Come up with logic to add a customer. Call the appropriate DL methods to accomplish this. Throw necessary exceptions if needed.

DL

Write data storage specific logic to add a customer to some form of repository. Repo could be a file, a static collection, a DB, etc.

Hint: Use LINQ

Search for a customer

Models involved:

Customer

Logic to be implemented:

UI

Come up with some user interface that searches and gets customer information based on some search parameter.

Note: Present appropriate messages if a customer is not found. Make sure the search parameter makes sense.

Write logic that would be result to:

//Add screen shot of the add restaurant UI

Hint: Use console readline, writeline, and maybe some looping construct to help with validation.

BL

Come up with logic to find a customer. Call the appropriate DL methods to accomplish this. Throw necessary exceptions if needed.

DL

Write data storage specific logic to search a customer from some form of repository. Repo could be a file, a static collection, a DB, etc.

Hint: Use LINQ

View Store Front inventory

Models involved:

StoreFront, Product

Logic to be implemented:

UI

Come up with some user interface that searches and gets store front information based on some search parameter.

Note: Present appropriate messages if a store front is not found. Make sure the search parameter makes sense.

Write logic that would be result to:

//Add screen shot of the add restaurant UI

Hint: Use console readline, writeline, and maybe some looping construct to help with validation.

BL

Come up with logic to find a store front and get its inventory. Call the appropriate DL methods to accomplish this. Throw necessary exceptions if needed.

DL

Write data storage specific logic to search a store front and get its inventory from some form of repository. Repo could be a file, a static collection, a DB, etc.

Hint: Use LINQ

Optional: if no inventory exists or all the storefront is empty. Display a meaningful message and not just an empty space.

Place order

Models involved:

Customer, StoreFront, Orders

Logic to be implemented:

UI

Come up with a user interface that takes in the customer who wants to place an order, which location would the order be placed on, and the finally, the list of items the customer would want to order from a certain store front.

Note: that the UI should be able to find an existing customer and storefront to place the order. Furthermore, you should be able to present the products and quantity of those products available in the store the customer wants to order from. Moreover, you should be able to dictate which product and how much will be added to the order. And finally, you should be able to checkout multiple products and view the order details of the order you just placed.

Write logic that would be result to:

//Add screen shot of the order restaurant UI

Hint: Use console readline, writeline, and maybe some looping construct to help with validation. You may also reuse searching logic that you've already established.

BL

Write logic that would accomplish the following: search customer and store front based on some parameter and assign them to an order, assemble an order object, update the inventory, validate that the current store front inventory can satisfy the order, calculate the order total, return the order details of the new order. Make sure it persists in the data storage.

Hint: Throw necessary exceptions

DL

Write data storage specific logic to search/get a customer/store front, add an order, update the inventory of the store front the order is placed on to some form of repository. Repo could be a file, a static collection, a DB, etc.

Hint: Use LINQ

View order history

Models involved:

Customer/Store Front

Logic to be implemented:

UI

Come up with some user interface that searches for a customer/store front and gets customer/store front order history based on some search parameter.

Note: Present appropriate messages if a customer/store front is not found. Make sure the search parameter makes sense.

Write logic that would be result to:

//Add screen shot of the add restaurant UI

Hint: Use console readline, writeline, and maybe some looping construct to help with validation.

BL

Come up with logic to find a customer/ store front. Call the appropriate DL methods to accomplish this. Throw necessary exceptions if needed.

DL

Write data storage specific logic to search a customer/ store front and get its order history from some form of repository. Repo could be a file, a static collection, a DB, etc.

Hint: Use LINQ

Replenish inventory

Models involved:

StoreFront

Logic to be implemented:

UI

Come up with some user interface that searches for a store front and gets its inventory information based on some search parameter. Also, to ask how much quantity will be added to existing inventory.

Note: Present appropriate messages if a store front is not found. Make sure the search parameter makes sense.

Write logic that would be result to:

//Add screen shot of the add restaurant UI

Hint: Use console readline, writeline, and maybe some looping construct to help with validation.

BL

Come up with logic to find a store front and get its inventory. Call the appropriate DL methods to accomplish this. Throw necessary exceptions if needed.

DL

Write data storage specific logic to search a store front and get its inventory from some form of repository. Repo could be a file, a static collection, a DB, etc.

Hint: Use LINQ

Note: Inventory values should **NEVER** be negative.

App Architecture

Models

The models project would contain all the data structures that you will be utilizing in your BL, DL, and UI. These are the data structures you'll be processing in the BL, DL, & UI.

Project type: classLib

Business Logic

The business project would contain the logic to compute and process the data needed for transactions.

Project type: classLib

Data Access Logic

This data access project would contain the logic to perform CRUD operations on the necessary models to some repository. This also contains repository specific logic on storing and accessing data from different kinds of data storage systems.

Project type: classLib

User Interface

The User Interface (UI) is responsible for displaying information to the end-user. It can also interact with the end-user to ask certain information or what is the next step to take. The end-user should only interact with the UI and nothing else.

Project type: console