Project Notes

[Project Folders]

Mocks - mock database
Specs - code coverage (how much of your code your tests cover - program used for this)
Models - concrete classes
Repositories - design patterns / db

REQUIREMENTS ANALYSIS

Non-Functional Requirements

Main

- Basic command line prompt while process is live
    - Info
        - display list of commands, notify user info command is available wherever they see command line prompt
    - Login
        - username and password entry, check if user exists, display appropriate messages if error, set session state to true if successful, update db, enter session
    - Sign up
        - sign up entry form, check if user exists, set session state to true if successful, update db, enter session
    - Exit
        - thread.exit()

Users & Sessions -- *signup, login, logout*

- session saves state of user for every login
- include bool to check if session is live (if live, user is logged in)
- if session state is live, user will not be able to login
    - User will be logged out automatically on program exit
- for each user, instantiate a new object to hold temp data until ready to be stored
- store session state in user object
- while session state live repeat command line prompt
    - Info
        - display list of commands based on user type (username==admin?associate:customer)
    - Logout

        - set session state to false, update db, exit session
    - History
        - list all previous orders based on user type (store/user) and username
    - Menu
        - display all menus (sizes, crusts, presets, toppings)
    - Stores
        - Display all locations
    - Order
        - Location (enter zip):
        - built in command line (select: …)
            - Info
                - display all commands
            - Menu
                - display all menus
            - Preset
                - order #: …
                - select quantity
            - Preset Special
                - select size and crust
                - order #: …
                - select quantity
            - Custom
                - select size and crust
                - select topping
                - select quantity
        - display current balance, prompt (add to order (enter y/n):)
            - If yes, repeat bullet 2
            - If no, increment user balance for datetime(day), update db, exit order function

Menus -- *crust, size, toppings/preset*

- computes pizza cost
- computes total order cost (if multiple pizzas ordered)

~~**Serialized Database**~~

- ~~serialization to store data on file~~
- ~~deserialization to retrieve data (xml)~~

COMPONENT DESIGN

[Relations - each object is a table not a record]

DB PIZZABOX

USER
username, password, name, session state (bool alive)

STORE
store id, *username*, city, state, zip

ORDERS
order id, store id, *username*, date, preset, custom, total pizzas, total order cost

preset<string> = “3Lk2Mn1Sk”
custom<string> = “30Mk2Sk”
L = large, M = medium, S = small
k = thick, n = thin

INVENTORY
store id, preset, custom

Store id = 3803892384
Preset<int> = 100
Custom<int> = 100

If one object contains multiple records (array of users in USER object), it can be used as a table
That would mean each user must be a struct/obj
What would that look like?