Program Write-Up

https://github.com/wowkyle7/Program1.git

Program Idea

- Pet Store Management System: Program will allow the user to enter data about pets & customers, and store data about what customers are interested in what pet, along with a stock of inventory of each pet in the pet store.

Classes

- Pet (Data Class): Contains attributes for their breed, sex, age, and price
    - Functions: Default constructor, overloaded constructor, deconstructor, getters, setters, print pet
- Customer (Other Class / Data Class): Stores data for a customer like a name, phone number, age, and max budget
    - Functions: Default constructor, overloaded constructor, deconstructor, print customer
- Pet Store (Storage Class): Manages pet inventory and customer information.
    - Functions: Deconstructor, display all pets & customers, dynamically allocate pointers of Pet & Customer
- Driver: The driver simulates a pet store where users can view available pets, purchase pets, and check inventory.

Program flow

- Start
- Create PetStore object
- Prompt user for max number of pets & customers
    - Dynamically allocate pointers in PetStore
- Menu Selection
    - Add Customer
        - Prompt user to enter data
        - Store data in PetStore
        - Increment customer index count by 1
    - Add Pet
        - Prompt user to enter data
        - Store data in PetStore
        - Increment pet index count by 1
    - Remove Customer
        - Call the default constructor to overwrite the most recent customer entry
        - Decrease customer index count by 1
    - Remove Pet
        - Call the default constructor to overwrite the most recent pet entry

        - Decrease pet index count by 1
    - Print All
        - Prints all stored customer data
        - Prints all stored pet data
    - Exit Program
        - Delete pointer in driver
    - Exit Loop
    - End program
        - This calls the deconstructors for the classes, which dynamically deletes all pointers in classes

Individual role breakdown

- Kyle: Pet Store Class, Driver & makefile
- Connor Warnock: Customer Class & makefile
- Matthew Evans: Pet Class & test case file

What did we try to do? Was it successful?

- Our goal was to make a pet store management program capable of managing customers' information & pet inventory. We succeeded in our goal, but some of our desired features such as save & load data to file were not incorporated. This was mainly due to time, but the overall program functions as intended.

What was your process?

- During lab sessions, we used Live Share to work on the files and update them in real-time, and we used Git to work independently. Our lab sessions consisted of working on our assigned files and communicating to ensure everything worked together.

What would you do differently?

- We would better plan the flow of the program rather than simply wing it like we did. This likely would’ve made the code cleaner & more efficient.

What did you learn?

- We learned how to create and use better classes and pointers. One major thing that was new to us was manipulating the pointers from the classes. We also how to operate GitHub which was very handy when it came to sharing work. It was a struggle to decide what we wanted to do and how we wanted to share the work but we eventually sorted everything out.

How will the TA run/test your program?

Manually enter data

- Open CMD in the program's folder
- Compile the program using "*g++ *.cpp -o petstore*" in CMD, then run petstore.exe
    - Alternatively, you can use the makefile and run the petstore.exe
- Follow program flow to add customers/pets, delete customers/pets, and print all data
- Exit program

Use test case

- After compiling the code, run "*Get-Content .\test-file.txt | .\petstore.exe*" in powershell