Project Part 6
Name: Vinitha Gadiraju
Title: PetMatcher
Project Summary:
PetMatcher is an app that allows clients to find pets (either a snake or a dog) that match the requirements they desire. There are two kinds of users: 1) pet seekers and 2) shelter administrators. Pet seeking clients can login, create a profile, enter what pet they want, enter what qualities they are looking for in a pet, and enter their housing information. Shelter administrator clients can create accounts (for either a snake or a dog) with their contact information and upload pets to the database along with their qualities. Shelter admins and pets will belong to a specific shelter. Pet seeking clients can look through on pets from the database that match them, and select which ones they are interested in. The pets they are interested in will be removed from the database.

Project Requirements Implemented:

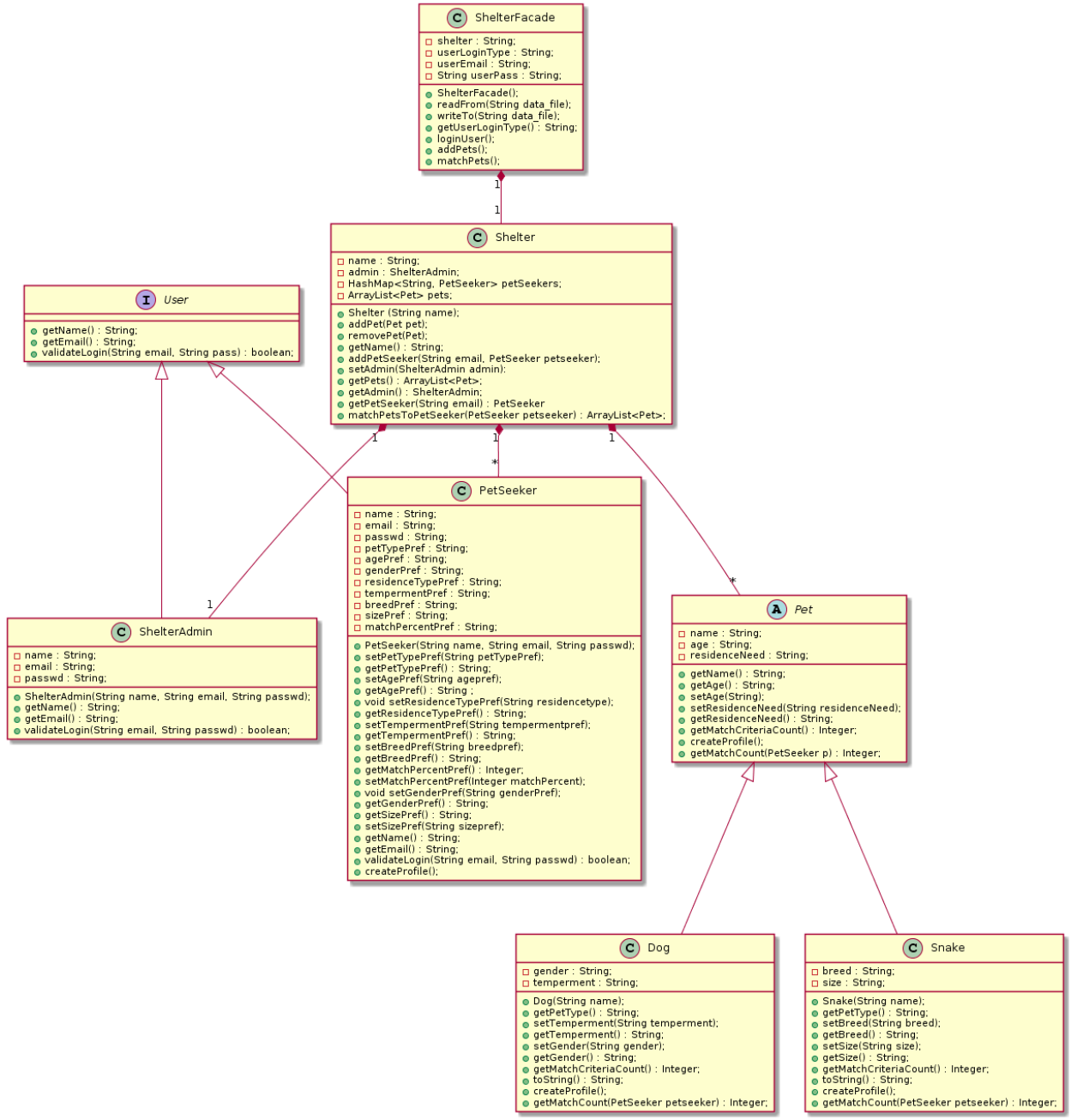
User ID	User Requirements
UR-1	Users should be able to create an account using their email and a password.
UR-2	Users should be able to login using their email and a password.
UR-3	Users should be able to create a profile.
UR-4	Pet seeking users should be able to select preferences.
UR-5	Pet seeking users should be able to select if they want a snake or dog.
UR-6	Pet seeking users should be able to select the age range of the pet they want.
UR-7	Pet seeking users should select the kind of home they live in.
UR-9	Pet seeking users should be able to select the temperament/role of the pet they want.
UR-10	Shelter administrator should be able to create a profile for their shelter.
UR-11	Shelter administrator users should be able to select if they want to create a profile for a dog or a snake.
UR-12	Shelter administrator users should be able to create pet profiles with name and breed.
UR-14	Shelter administrator users should be able to select the age range of the pet.
UR-15	Shelter administrator users should be able to select the kind of home the pet can be in.
UR-17	Shelter administrator users should be able to select the temperament of the pet.
UR-18	Pet seeking users should be able to select or reject a pet when they see it.

Project Requirements not implemented:

UR-8	Pet seeking users should be able to select if they have children in their home.
UR-13	Shelter administrator users should be able to write a bio for the pet profile.
UR-16	Shelter administrator users should be able to select if the pet can be around children or not.

UR-19	Pet seeking users should be able to message their pet matches to inquire more about the pet.

Class Diagram



The original class hierarchy has maintained the same in the class diagram. The pre-planning of subclasses and inheritance with Users and Pets allowed for a sound structure that carried through the entire design process. The major change in the class diagram, from the beginning, is the addition of the ShelterFacade. ShelterFacade is the implementation of the

Façade design pattern into the program. I chose the Façade design pattern because the PetMatcher app is complex in that it involves creation of multiple objects and their sub-classes. The order of object creation is also critical in the sense that Shelter Admins must add pets to the system before the PetSeeker begins to "swipe". I included Login Authentication and Profule Creation in the Façade design pattern since the process tends to be laborious. ShelterFacade hides complex system operations like object creation, the management of pets (addition and removal), user interaction/profiles and writing to an external file. The external file is critical since it serves as the database for all pets in the system.

What have I learned?

Towards the end of my design process, it became clear that not all my project requirements were going to be implemented. It came down to a decision between implementing a messaging feature or re-designing my code to implement a façade design pattern. I eventually chose implementing a design pattern so that the code would have a simplified and cleaner system for the user to interact with. I learned that it was important to clean up the code you have and create a product worthy of using, rather than continue to implement new features on a cluttered back-end.

Time management was a crucial part of this process. With each of the progress reports, I set milestones that I wanted to accomplish so that I would be able to create a viable product by the end of the semester. I front-loaded my schedule, with the main aspects of the application coded by progress report 2. However, once I decided to implement a design pattern as mentioned above, the schedule changed and added a larger workload to the end of my timeline. This simply required a more detailed breakdown of what I would have to accomplish on a weekly basis.

This process was exciting in the sense that I could build a fun application from start to finish that I care about and would use! I feel inspired to continue refining this app and move into a UI phase of implement the mock-ups from my initial report at the beginning of the semester. When even designing the command line interface, I thought about other stakeholders and what would make the application easier and more fun to use. This led to the addition of small emoticons and messages providing feedback through the different stages of interaction in the application. I hope to continue using this knowledge in a more advanced rendering of the application.