Kenia Benitez

PSID: 1821917

My A-List Pets

For the final project, I will be doing the Tamagotchi Pet game, “A-List Pets”. A-List Pets makes you the owner of a famous cartoon pet. Who wouldn’t want to have Scooby-Doo as their dog? Owning a pet is great and all but owning one that is on everyone’s TV is on the next level. The game is developed in the way where you can interact with the pet while keeping the personality that makes them who they are. This game will allow for the user to pick a specific cartoon pet from the menu and take care of them like you would in real life. The pet’s interactions are based on what the pets love and are known for in their respective shows. The user will oversee feeding, and other regular interactions a pet owner must go through.

The development of the game started off setting up the classes. Originally, the functions for each derived class were going to have different names, but then I realized that it would take too long to create and execute the functions. Because of that, I set virtual functions in the base class in which each derived class could access. I also added the missing statements required, like load and quit.

I started to create the derived classes, I decided to change the last character to Gary from SpongeBob instead of Brian from Family Guy. It felt like this character was a better fit with the other two. The base class will be Pets, and the three derived classes is Garfield, ScoobyDoo, and Gary.

After creating all the classes, I decided to create the load functions to see if they worked properly. I then created the initial loop and menu and continued to construct the entire program. To make the **int main ()** as readable as possible, I created multiple user defined functions. All the user defined functions were created because the function is used multiple time throughout the code.

**int menu ()** will return an integer value of the selection given. This is the main menu of the entire code. The **string newGame ()** initializes the prompt of a new game. In this function, the beginning values are set by letting the user choose the difficulty level. When they do, it will set the status according to a specific threshold set by the level of difficulty. After the beginning value are set, it will upload them to the designated file. If the user enters an incorrect name, the file will not open and set a loop until they do. After all of this, it returns the string of the pet type (Garfield, ScoobyDoo, and Gary) the user chose. The string returned from the **newGame ()** will be used to open the corresponding file in **int main ()**.

After the string is returned, the function **void loadFile** will be called. This will take in the file name, and the statuses. In the main, the addresses of the will be inputted in the functions and taken in as a pointer. This function will open the file and point the data from the file into the address of the designated status from the main. This will also output the beginning status.

After the beginning status is set in the main, the **void actionList ()** is called. It will take in the name of the pet and output the action list for the specific pet they are using. This will then allow the user to input the action they want. So, if the user were to choose option 1, it will be associated with action 1, but the name of that action differs with each derived class.

The last function used was **void execute**, which will take in the specific pet they chose, the beginning statuses, and the beginning choice. Here is where the game is executed. It has specific cases for the type of pet. There, the class is created, and additional cases are set for their actions.

One of the main issues I kept on having was the looping. I did not expect to have so many loops in this code, but to avoid leakage of any way when a user is playing, this was a necessary procedure that had to take place.

Another issue I faced was not being able to take full advantage of polymorphism. I wanted to include this action to have an easier coding process, but I failed to realize that the name of the derived class came directly from the user. I intended to use pointers to make the code shorter, but I kept receiving errors. Because of this, I had to write a specific set of actions for each derived class. It made the code long, and that is the main thing that can be improved upon.

Overall, I made sure to not leave any room for error, and to close out any possible “leakage” the game can face when running.

One thing I forgot to mention was that the definitions of the functions in each class were in the .h file. I decided to do this because the function was not long enough to create a separate template.

In conclusion, creating this game was an enjoyable experience, and I know it can be expanded into an even better game.

**Video link**: <https://youtu.be/nfBv60mwv2g>

Classes:

Graphical user interface, application

Description automatically generated

Graphical user interface, application, Teams

Description automatically generatedUML: