Lindsey Wingate

Computer Science 161

Professor Kerlin

**Top Level Design and Design Rationale (TLDDR)**

The methods I used to implement the program were the two main methods in each program (PetFinder and PetGenerator). The main method for PetGenerator generates the number of pets the user inputs. It randomly creates pet data (type, name, satisfaction rating, cost of care, and hours of care required). It writes each set of data as a string to a document entitled “Pets.txt”.

The main method for PetFinder requests an input from the user between 1 to 6 and calls the necessary methods to retrieve the data for each number. If 6 is selected you exit the program. They are listed below:

Option 1: fishNames()

This method pulls the data from the file, checks to see if it is for a fish, and prints the lines with fish names only.

Option 2: averageCatCare()

This method pulls the cat care values from each line with the type “Cat”, averages the data, then prints the result.

Option 3: averageBirdCost()

This method pulls the bird cost from the lines with the type “Bird”, averages the data, then returns the data to the main method which prints it to the screen.

Option 4: Satisfaction()

This method searches the lines for the satisfaction rating of each pet and assigns the corresponding satisfaction rate to highest or lowest satisfaction. It prints the highest satisfying pet’s name and rating to the screen as well as the lowest satisfying pets name and rating.

Option 5: numberOfPets()

This method sorts through each line and indicates how many pets of each type there are. At the end of the document, the results are printed to the screen.

The biggest challenges I came across were local variables and opening the file. I had to learn to print the final answer in each method in the correct set of curly brackets otherwise the loop would print an answer out for each line of the file. I solved this by putting the final print or return statement just before the bracket ending the method. Also, I wanted to find a way to open the file once in the main method so I wouldn’t have to keep entering the same code in the smaller methods to open the file, split the lines, and then close it again. I didn’t find a good way to do this, however, and ended up copying the code and using it in each little method. Although the code still worked properly I was hoping to find a more efficient option.

I learned Java really is a hard coded language. It requires you to specify multiple times what type of data you are passing in and out of methods. Also, it requires a lot of thought beforehand so you don’t write unnecessary code (this is general with most programming). Many times I hard coded things which were eventually replaced with loops to save time. For example, I originally had the same set of code for each type of animal in PetGenerator. When I realized all of it was not needed, I deleted the extra and used one set to generate the data for all of the pets.

I designed my program with the five smaller methods within PetFinder because it made the most sense with individual commands. If there wasn’t an option to select a number because the data just needed to be printed to the screen I may not have used individual methods. However, methods create organization and it is a lot easier to edit them. If I wanted to I could easily change the corresponding data to the number options. I didn’t feel the need to use methods within PetGenerator because it was a smaller amount of code and all the data generated was going into the same string. It would have been more work to pass the necessary variables in and out of the methods to ultimately write them all together into the same line to the file.