DogRUs Individual Assignment

Nicolas Petras, nip19@aber.ac.uk

Thursday, 22 March 2018

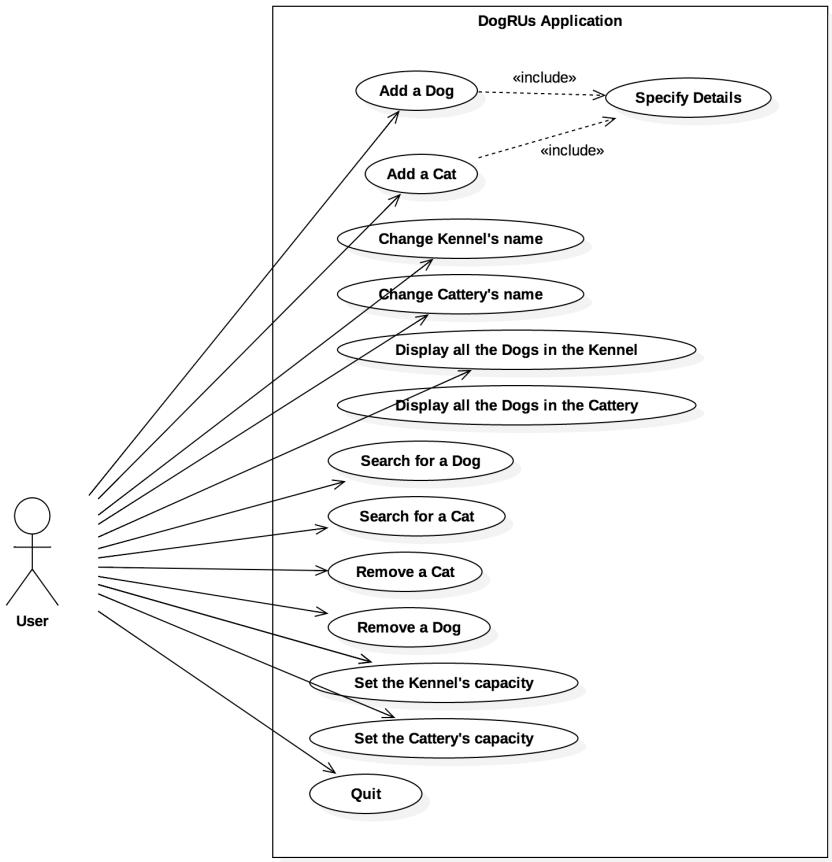

Figure 1: UML Use-Case Diagram

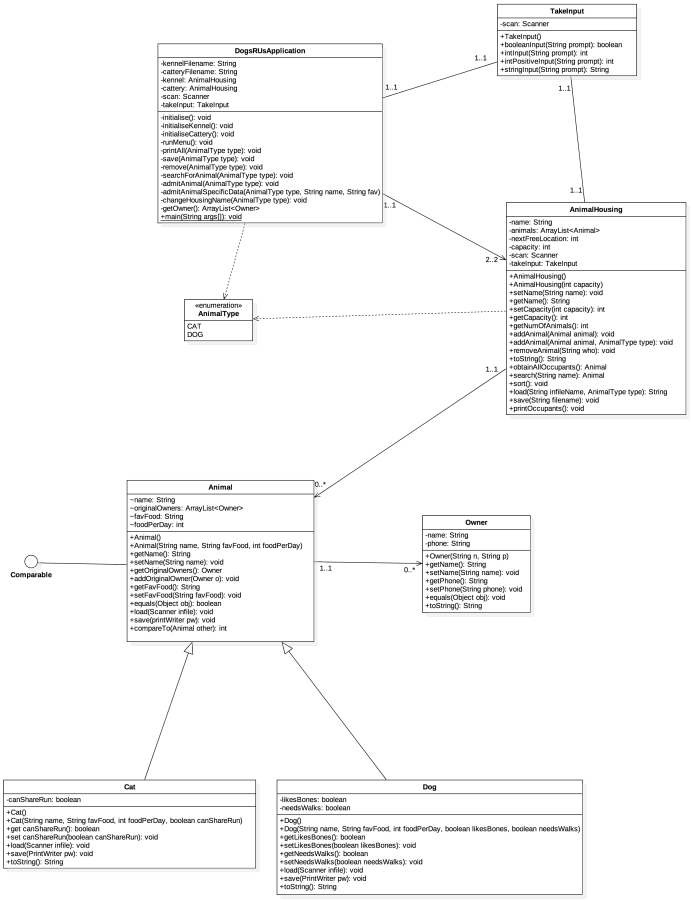

Figure 2: UML Class Diagram

The first task I proceeded with was to **refine** the current code, in various ways, one of which included adding three methods to take three different types of input: String, int and boolean. These three functions prevented *code duplication* and added *validation*. The validation ensured *valid data* was entered and *prevented* the program from *crashing* due to exceptions. These methods have their own class, allowing them to be used throughout the program, and adds to the *cohesion* of the program.

While working on this project I made the effort to **rework** a lot of the code that was provided and comments keep everything consistent. Making it easier for me or another developer to revisit and edit the code in the future. I made sure to catch all exceptions, using multiple `catch` statements within an overall `try-catch` statement, with the *higher level statements* handling *higher level exceptions*.

The next step I undertook was creating an `Animal` and `Cat` class for the purpose of inheritance. Making both the `Cat` and `Dog` classes extensions of the `Animal` class.

I decided to make an `AnimalHousing` class to replace the `Kennel` class, my initial intent was to make this a **super-class** for the existing `Kennel` class and a new `Cattery` class. But it ended up replacing both of these classes because there was no subclass specific variables or methods that needed to be held in `Kennel` or `Cattery`. I wanted messages, and some minor aspects of methods changed for kennels and catteries. I achieved this by using an `AnimalType` (enum) parameter and switch statements.

I made several changes to the `DogsRUsApplication` (previously `KennelDemo`) class to accommodate both **animal types** and their respective housing. One of which, involved removing the `Kennel` object reference, and replacing it with two new `AnimalHousing` references - one for the kennel and another for the cattery. I used the `AnimalType` parameter once again in a lot of the *private utility methods* to provide differing functionality, depending on the `AnimalType` passed to the method. The class does have an *overcrowded* menu, and there are possibly some *bulky* and *inefficient* methods in this class and others, leaving room for improvement. Another area that could see improvement is the functionality of the program - I had several ideas to improve functionality (e.g. using hash map for cats and dogs, giving each a unique ID), but I did not get an opportunity to implement these ideas.

I predict that my grade for this assignment will be in the range of 75-80% since I feel I fulfilled the technical programming requirements to relatively high standard, but my code does have some issues. The documentation contains all the required documents and these documents are executed to a high standard, some aspects of the UML diagrams might be lacking. My code does lack flair marks, there is little to no extra functionality, apart from what is specified in the requirements. Overall I feel my submissions is completed to a high standard, but does still have some mostly minor issues, and lacks extra functionality.

Screenshots - Program Running

Figure 3: Screenshot showing the program being launched, and the kennel and cattery capacities being set

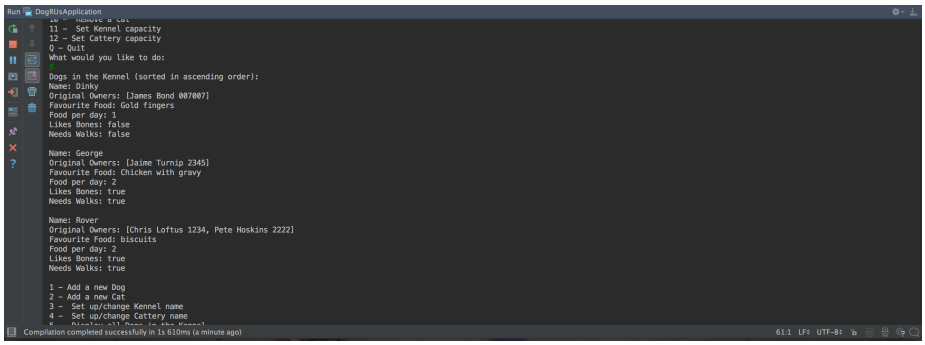

Figure 4: Screenshot showing a list of the dogs currently housed in the kennel, sorted in ascending order

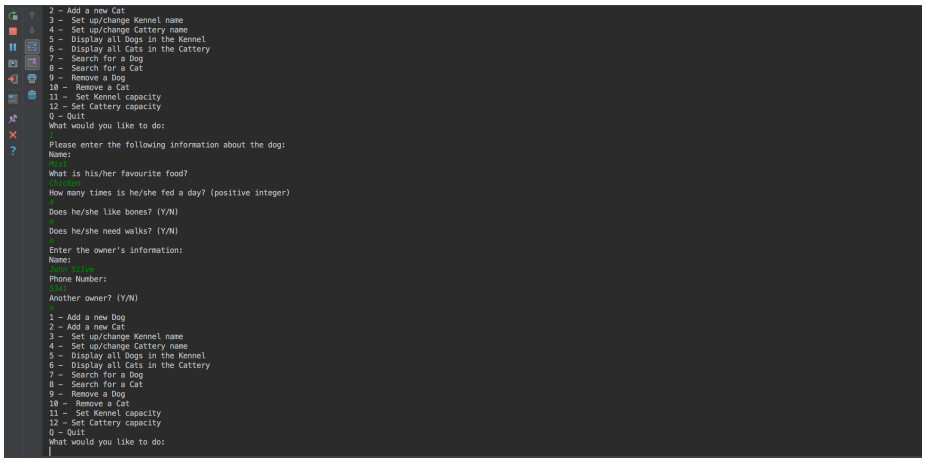

Figure 5: Screenshot showing a new dog being added to the kennel

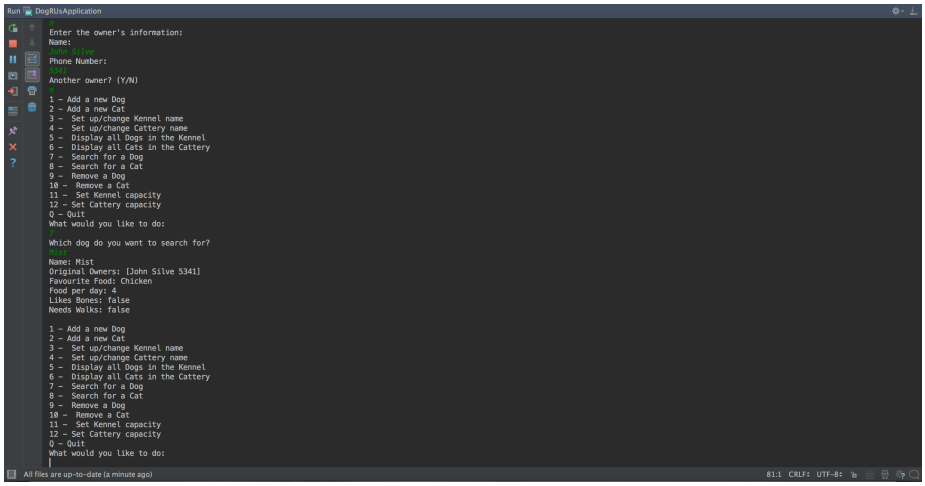

Figure 6: Screenshot showing a dog ‘Mist’ being searched in the current kennel