

CS 12320: Mini Assignment #2

DogsRUS

Due on Monday, April 20th, 2015



Nathan Hughes (nah26@aber.ac.uk)

Contents

UML Diagrams	3
Relationships	3
Owner Class	4
Animal Class	4
Cat Class	5
Dog Class	5
Kennel Class	6
Kennel Demo Class	6
Assignment Write Up	7
Tasks and Solutions	7
Completing the Dog Class	7
Adding the Cat Class	8
Adding the Animal Class	8
Use of "instanceof" operator	9
Alphabetically sort the Animals in Kennel	9
Editing functions to use StringBuilder	10
Evaluation	10
Code Running	11
Launching from Terminal	11
Adding a dog	12
Adding a cat	12
Changing kennel name	13
Displaying all dogs who like bones	13
Displaying all cats that care share a run	14
Searching for dog	14
Searching for cat	15
Removing a dog	15
Removing a cat	16
Changing capacity of kennel	16
Printing all animals alphabetically	17
Exiting program	17

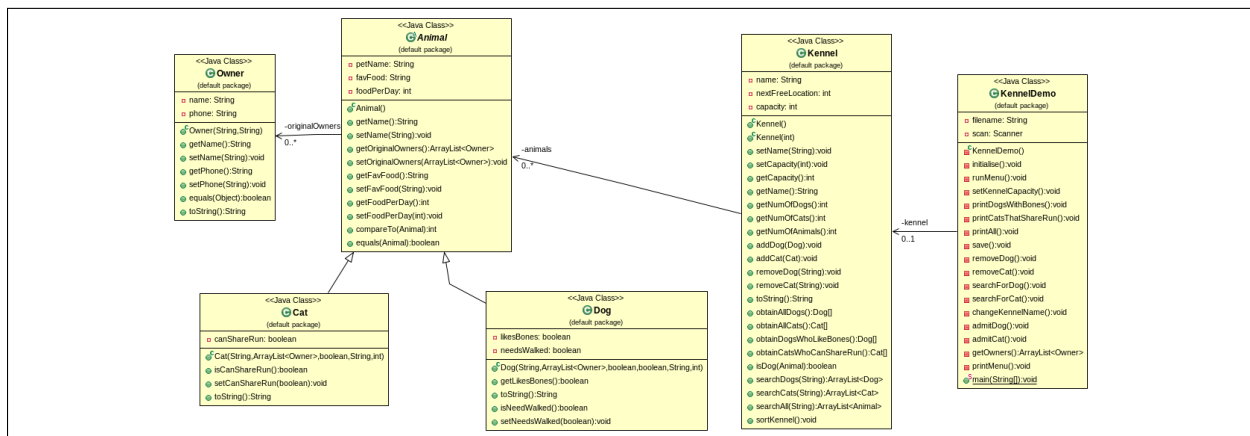
UML Diagrams

Relationships

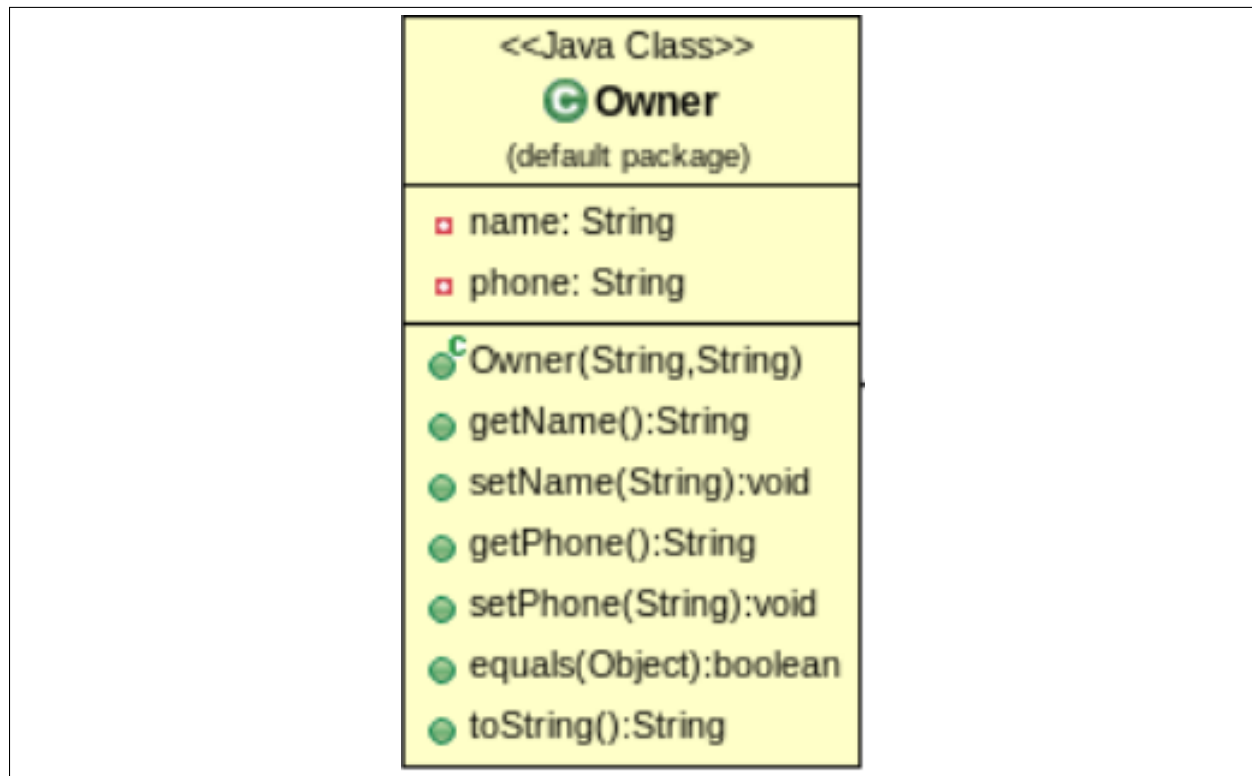
Below are the class diagrams for the DogsRUs application; the 6 classes are shown to have relationships between each other by 2 different types of lines. A solid line depicts an association relationship and a solid line with a boxed arrow shows that they inherit from that parent class.

Beside every class object or function there is a coloured dot, green signifies a publicly accessible element whereas red shows that it is private and cannot be accessed outside of the class itself.

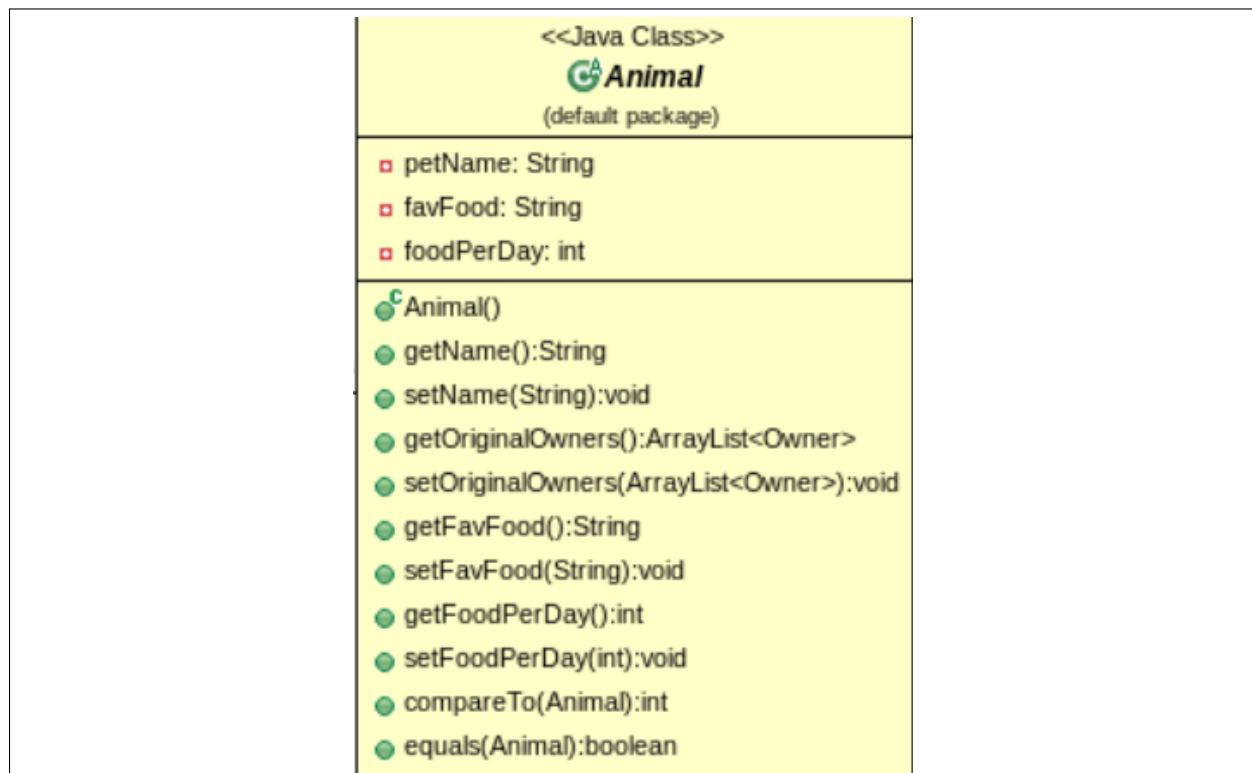
Functions can be identified as they have "()" following their name as well as their return type. Variables/Objects will also have their type noted following their names' also.



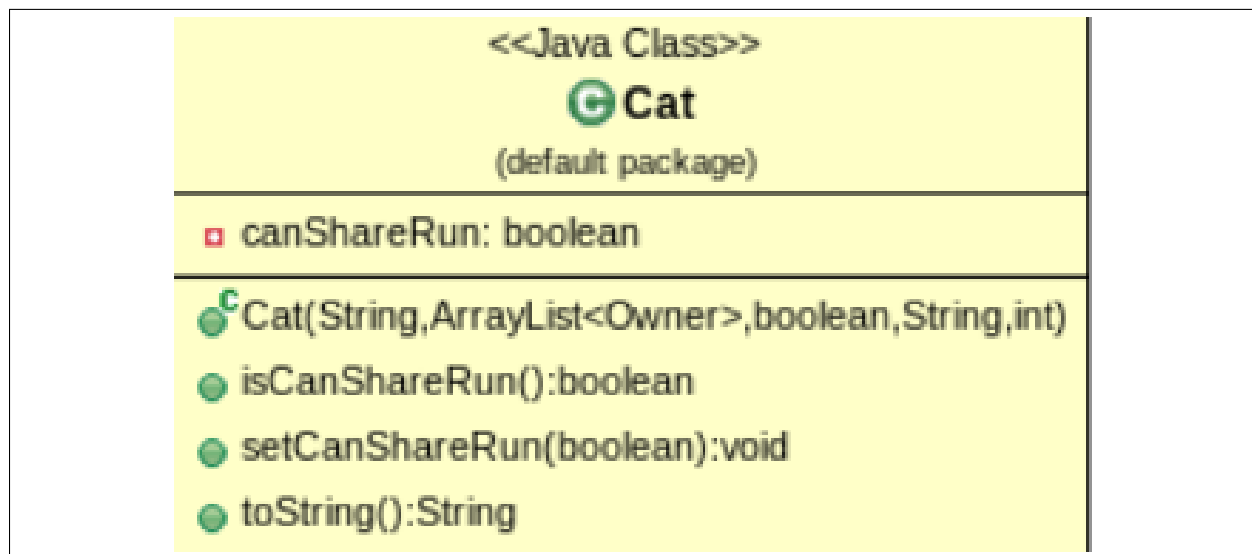
Owner Class



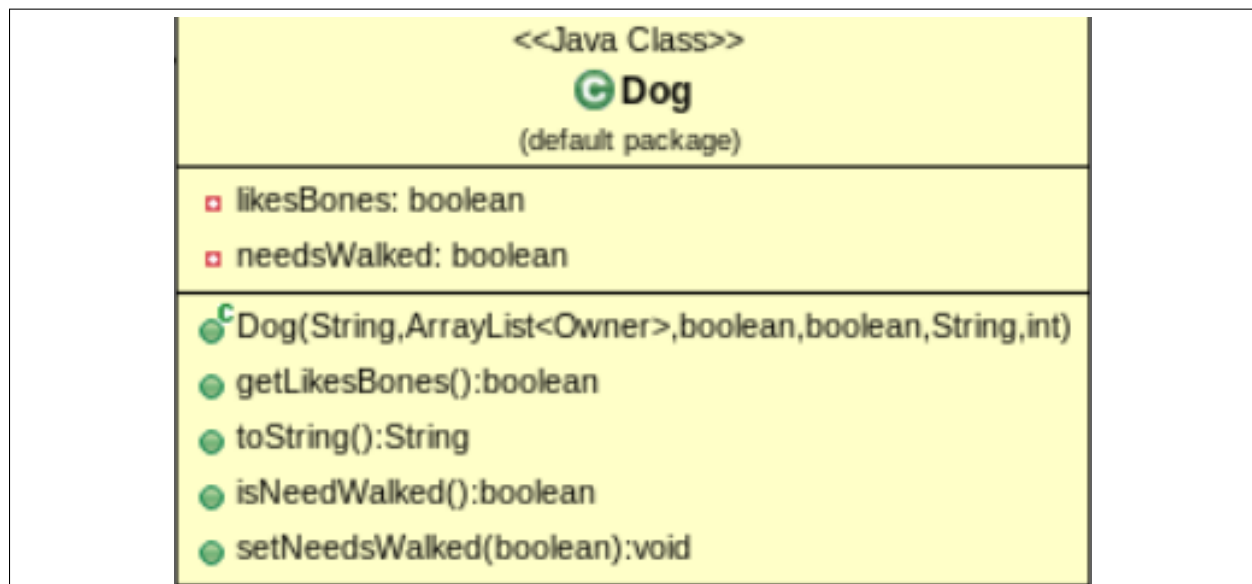
Animal Class



Cat Class



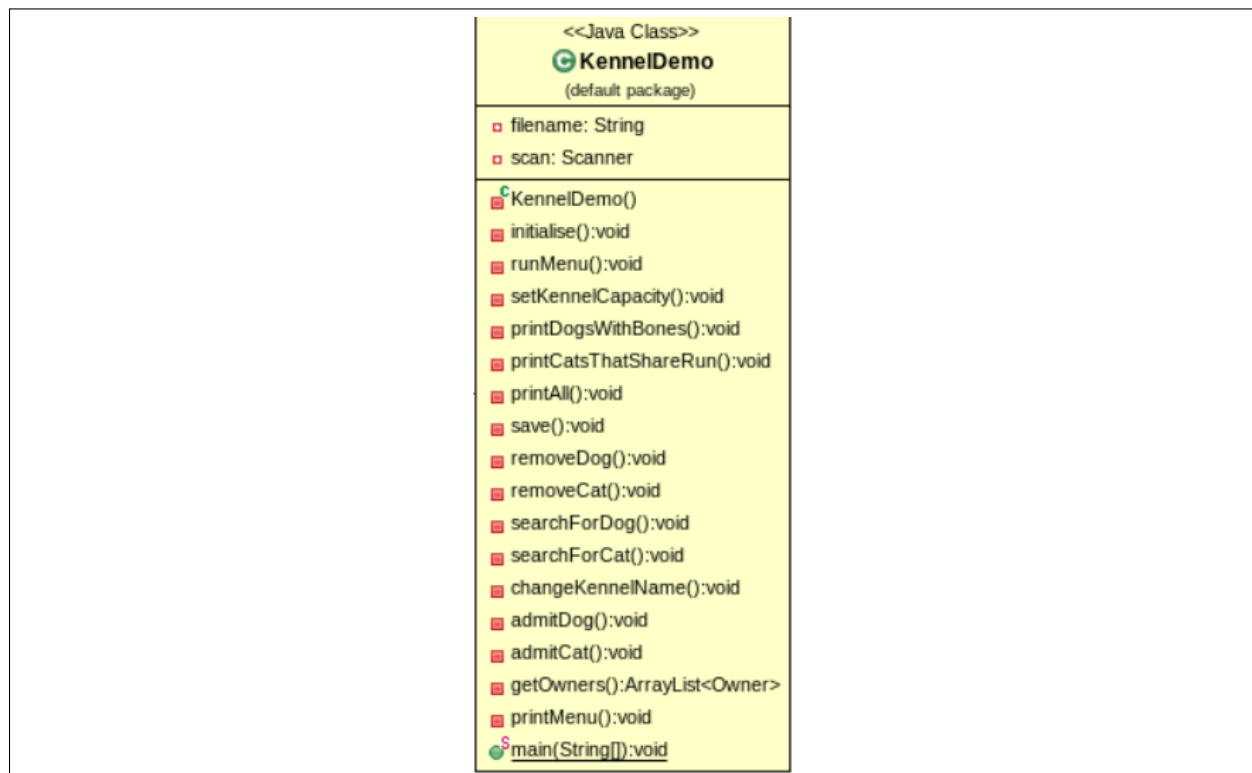
Dog Class



Kennel Class



Kennel Demo Class



Assignment Write Up

Tasks and Solutions

In this section I will outline my solutions to the problems given in this assignment, alongside each problem there will be a code snippet.

Completing the Dog Class

To begin editing this project I first had to change how a lot of things were handled, firstly I knew that the current classes needed augmenting and an extra parent class would be required also. This affected the already present Dog class.

The largest change to this class was that a lot of its methods were no longer required and so they were removed/alterd into its parent class. As well as this the constructor was changed to access the parent class's functions. Also the boolean value of "needWalked" was added to provide the specification requirement.

```
/**
 * Constructor for the dog
 * @param name The dog's name
 * @param owners A list of original owners: a copy is made
 * @param likeBones Does the dog like bones?
 * @param needWalked Does the dog need walked each day?
 * @param food The kind of food it eats
 * @param mealsPerDay Number of feeds per day
 */
public Dog(String name, ArrayList<Owner> owners, boolean likeBones, boolean needWalked, String
    food,
    int mealsPerDay) {
    setName(name);
    setOriginalOwners(new ArrayList<Owner>());

    // We make a true copy of the owners ArrayList to make sure that we
    // don't break encapsulation: i.e. don't share object references with
    // other code
    for(Owner o: owners){
        Owner copy = new Owner(o.getName(), o.getPhone());
        getOriginalOwners().add(copy);
    }
    this.likesBones = likeBones;
    this.needsWalked = needWalked;
    setFavFood(food);
    setFoodPerDay(mealsPerDay);
}
```

Adding the Cat Class

Another requirement for this program was that it would be able to accept Cats as well as Dogs. This meant adding in the cat class. As the code snippet from its constructor shows, it is very similar in how the dog class works, with the exception of having the "canShareRun" option added in.

```
/**
 * Constructor for the cat
 * @param name The Cat's name
 * @param owners A list of original owners: a copy is made
 * @param canShareRun Does the cat share a run with others?
 * @param food The kind of food it eats
 * @param mealsPerDay Number of feeds per day
 */
public Cat(String name, ArrayList<Owner> owners, boolean canShareRun, String food,
           int mealsPerDay) {

    setName(name);
    setOriginalOwners(new ArrayList<Owner>());

    // We make a true copy of the owners ArrayList to make sure that we
    // don't break encapsulation: i.e. don't share object references with
    // other code
    for(Owner o: owners){
        Owner copy = new Owner(o.getName(), o.getPhone());
        getOriginalOwners().add(copy);
    }
    this.setCanShareRun(canShareRun);
    setFavFood(food);
    setFoodPerDay(mealsPerDay);
}
```

Adding the Animal Class

To enable the program to handle both of the different pet classes, a parent class of "Animal" was added. This provided a way of sharing data i.e. reducing code repetition and it also enabled comparisons between any child class of this.

This code extract shows that the Animal class is an abstract class, as it will never need to be created as a variable and it implements the Comparable template class. This was used in the sorting function.

```
public abstract class Animal implements Comparable<Animal> {

    private String petName;
    private ArrayList<Owner> originalOwners;
    private String favFood;
    private int foodPerDay;
}
```

Use of "instanceof" operator

I only used this operator once in my program, and that was to use a Boolean function; the use of "instanceof" determined if the given variable was part of the dog Class or another. It was used throughout the program to determine to distinguish between Dogs and Cats.

```
/**
 * @param animal
 * @returns a value true if the type is of the Dog class
 * and false if it is anything else
 */
public boolean isDog(Animal animal){

    if(animal instanceof Dog){
        return true;
    }
    return false;

}
```

Alphabetically sort the Animals in Kennel

This is another very small but very useful piece of code. By using an interface (shown previously in the Animal Class) I was able to use the "compareTo" function and though polymorphism change its usage to suit the needs of the program. In this instance it evaluates strings regardless of case.

This along with the arraylist class's own functionality it was very easy to sort the entirety of the pets by name regardless of type.

```
/**
 *
 * @param other Animal to compare string to
 * should hopefully ignore the case of the string as well
 * @return
 */
public int compareTo(Animal other){

    return petName.compareToIgnoreCase(other.petName);

}
```

Editing functions to use StringBuilder

The previous incarnation of this method was clunky and inefficient. Therefore it needed to be remodeled and the string builder class helped with this. Through appending each piece of required information to the object it made for returning a tidy easy to read string, simple.

```
/**
 * A basic implementation to just return all the data in string form
 * Using a string builder object
 */
public String toString() {

    StringBuilder results = new StringBuilder();
    results.append("Dog name: " + getName() + "\n");
    results.append("Likes bones?: " + likesBones + "\n");
    results.append("Original Owner(s): " + getOriginalOwners() + "\n");
    results.append("Favourite food: " + getFavFood() + "\n");
    results.append("Needs walked?" + isNeedWalked()+ "\n");

    return results.toString();
}
```

Evaluation

I feel as though this assignment was completed adequately, in that everything that I set out to do, was in the end implemented and completed, and I have been able to erase any bug that I have come across

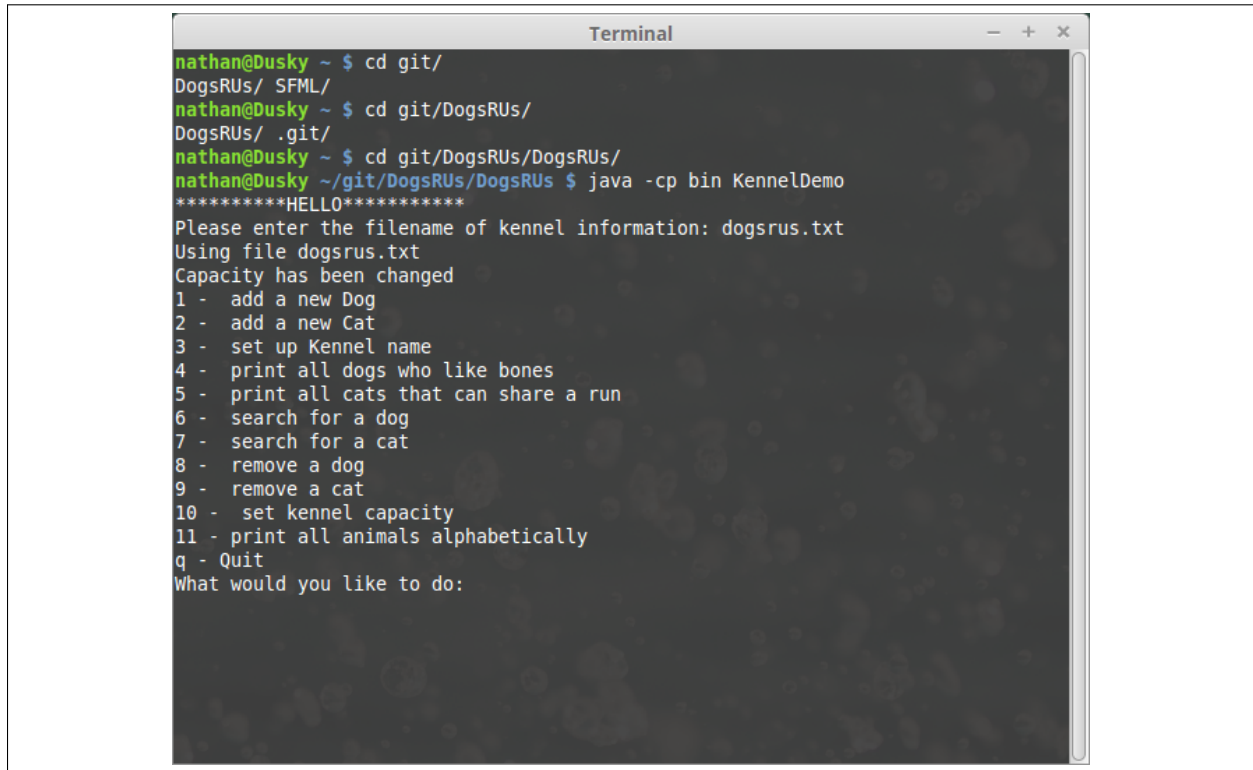
The downfalls that I can see within my own code is that my functions may have ended up being too long and too complicated. I also had the feeling that every addition I made to the Kennel class, I ended up repeating a lot of it in the KennelDemo. Had I had more time and paid more attention to detail I would have liked to fix this up and made the code more presentable

The testing performed for this code was done modularly as each class was tested by itself before being added to the program, this sped the process of bug finding up I think, as it made it easier to pinpoint problems with the code quicker.

If I had to, and perhaps I have missed something but I would award myself a 6/10 in marks for my code. The majority of the marks which I have taken from myself are because of the code being very messy and very much in need of a careful clean up.

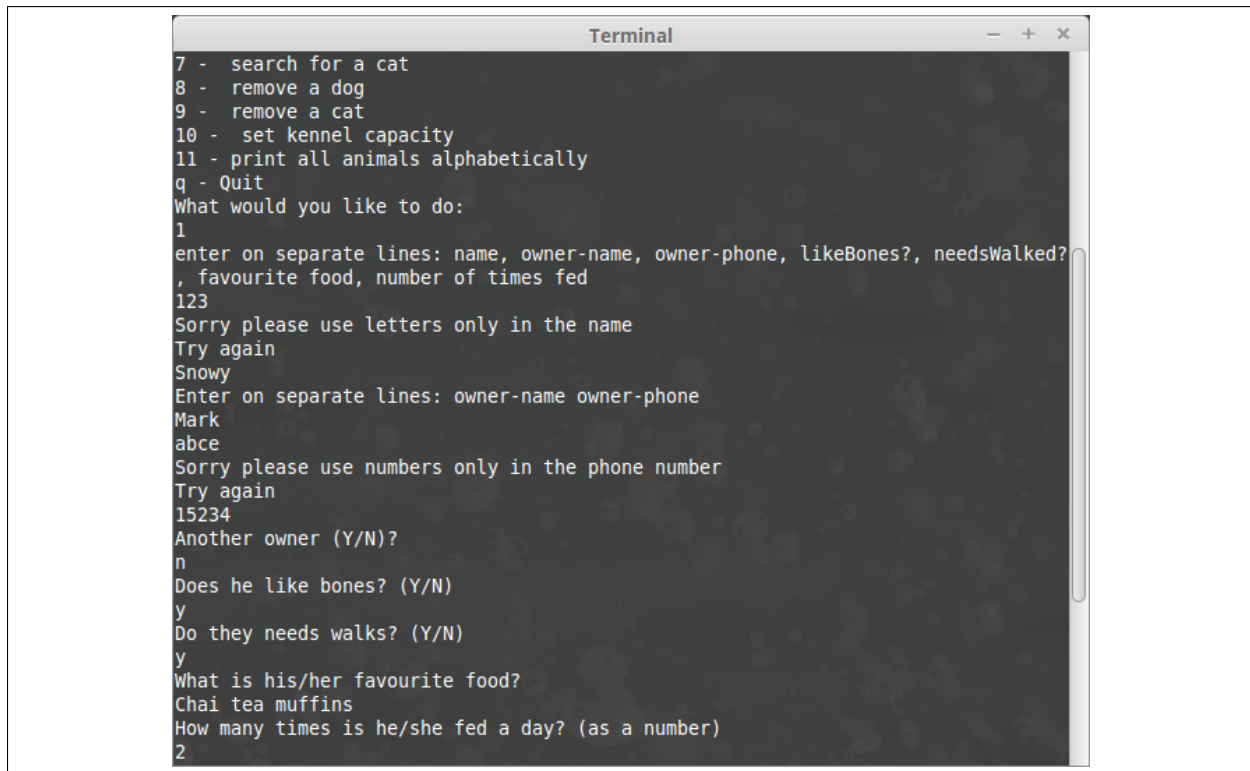
Code Running

Launching from Terminal



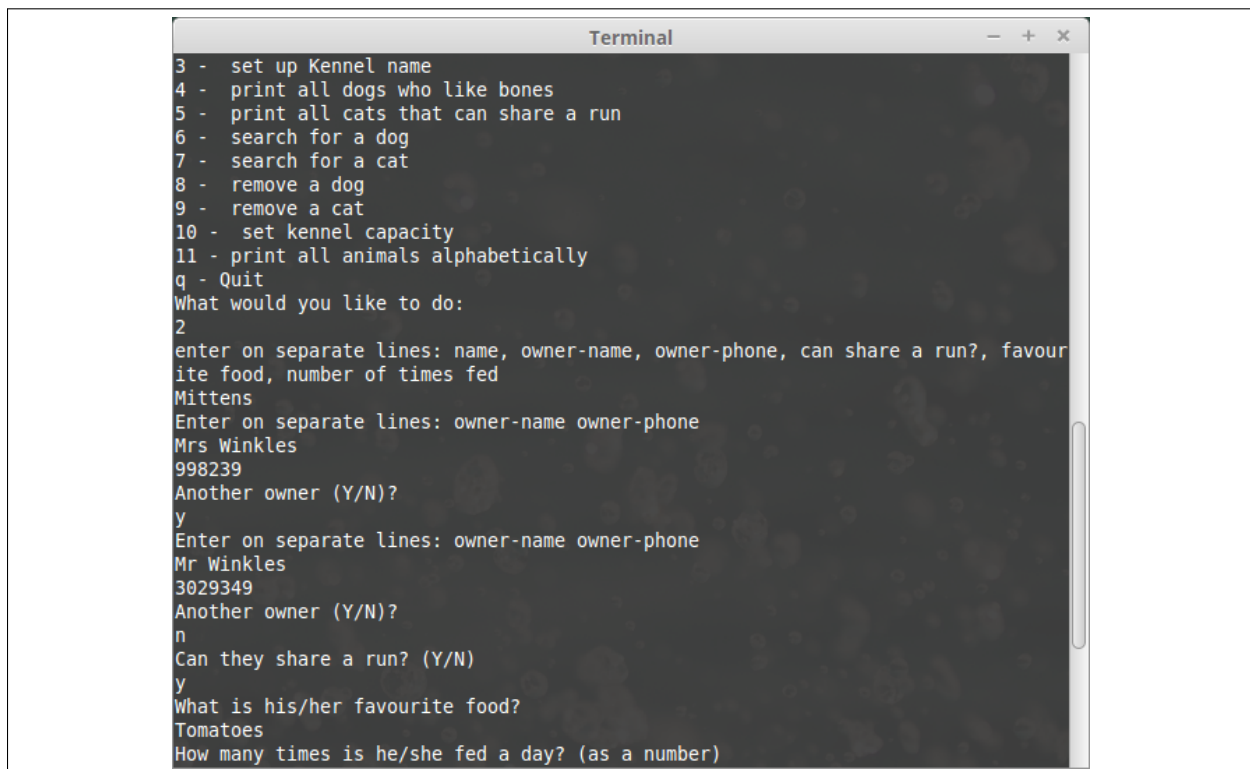
```
Terminal
nathan@Dusky ~ $ cd git/
DogsRUs/ SFML/
nathan@Dusky ~ $ cd git/DogsRUs/
DogsRUs/ .git/
nathan@Dusky ~ $ cd git/DogsRUs/DogsRUs/
nathan@Dusky ~/git/DogsRUs/DogsRUs $ java -cp bin KennelDemo
*****HELLO*****
Please enter the filename of kennel information: dogsrus.txt
Using file dogsrus.txt
Capacity has been changed
1 - add a new Dog
2 - add a new Cat
3 - set up Kennel name
4 - print all dogs who like bones
5 - print all cats that can share a run
6 - search for a dog
7 - search for a cat
8 - remove a dog
9 - remove a cat
10 - set kennel capacity
11 - print all animals alphabetically
q - Quit
What would you like to do:
```

Adding a dog



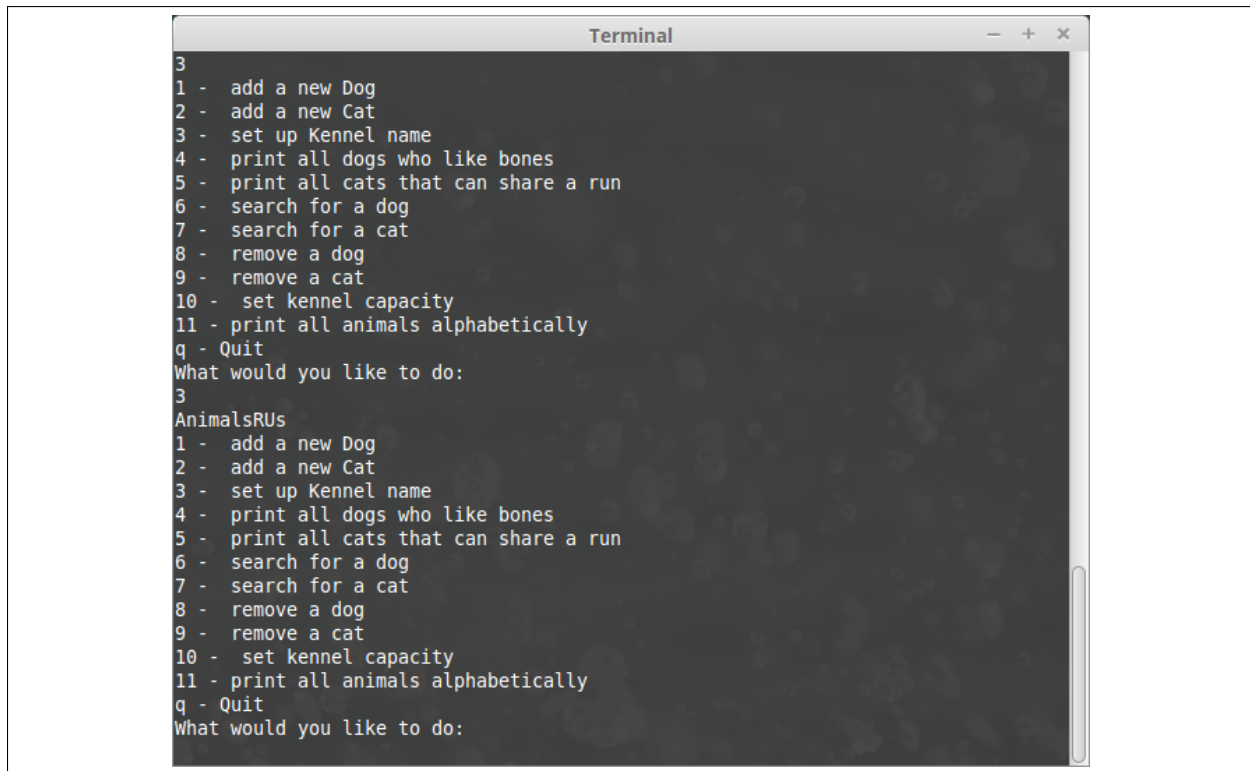
```
Terminal
7 - search for a cat
8 - remove a dog
9 - remove a cat
10 - set kennel capacity
11 - print all animals alphabetically
q - Quit
What would you like to do:
1
enter on separate lines: name, owner-name, owner-phone, likeBones?, needsWalked?,
, favourite food, number of times fed
123
Sorry please use letters only in the name
Try again
Snowy
Enter on separate lines: owner-name owner-phone
Mark
abce
Sorry please use numbers only in the phone number
Try again
15234
Another owner (Y/N)?
n
Does he like bones? (Y/N)
y
Do they needs walks? (Y/N)
y
What is his/her favourite food?
Chai tea muffins
How many times is he/she fed a day? (as a number)
2
```

Adding a cat



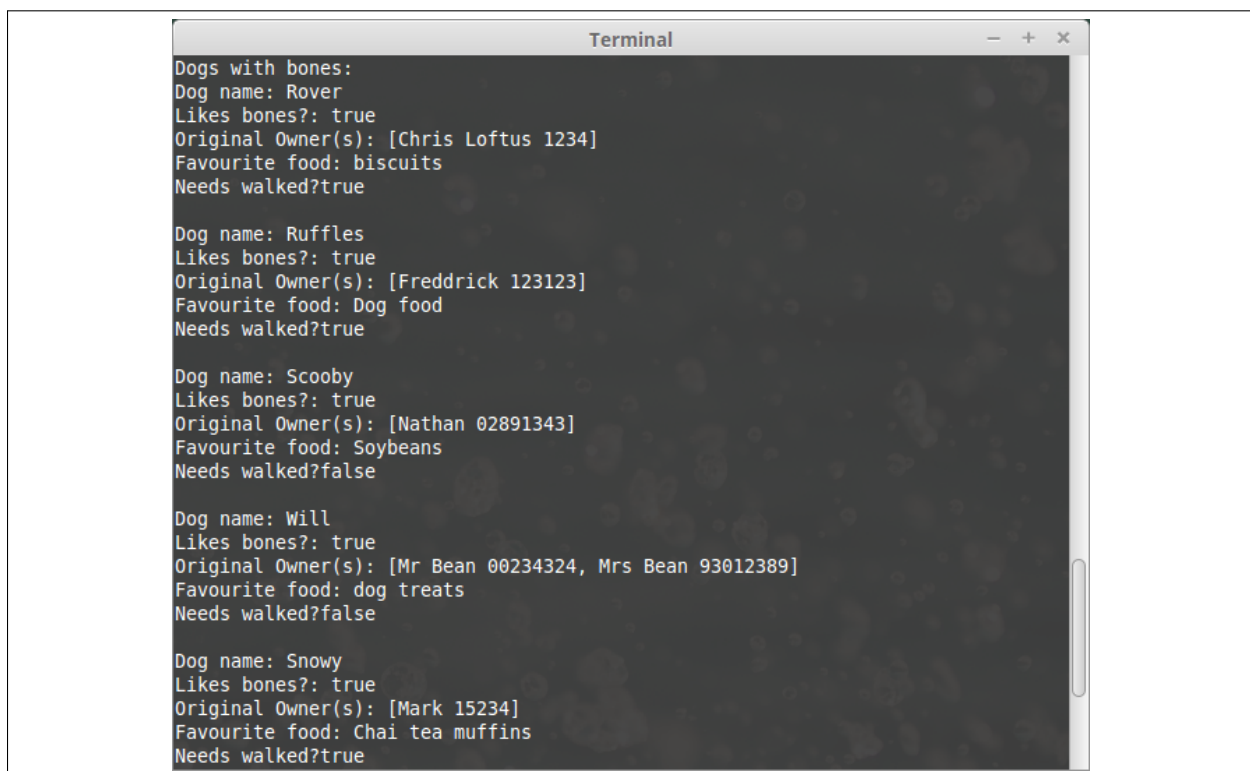
```
Terminal
3 - set up Kennel name
4 - print all dogs who like bones
5 - print all cats that can share a run
6 - search for a dog
7 - search for a cat
8 - remove a dog
9 - remove a cat
10 - set kennel capacity
11 - print all animals alphabetically
q - Quit
What would you like to do:
2
enter on separate lines: name, owner-name, owner-phone, can share a run?, favour
ite food, number of times fed
Mittens
Enter on separate lines: owner-name owner-phone
Mrs Winkles
998239
Another owner (Y/N)?
y
Enter on separate lines: owner-name owner-phone
Mr Winkles
3029349
Another owner (Y/N)?
n
Can they share a run? (Y/N)
y
What is his/her favourite food?
Tomatoes
How many times is he/she fed a day? (as a number)
```

Changing kennel name



```
Terminal
3
1 - add a new Dog
2 - add a new Cat
3 - set up Kennel name
4 - print all dogs who like bones
5 - print all cats that can share a run
6 - search for a dog
7 - search for a cat
8 - remove a dog
9 - remove a cat
10 - set kennel capacity
11 - print all animals alphabetically
q - Quit
What would you like to do:
3
AnimalsRUs
1 - add a new Dog
2 - add a new Cat
3 - set up Kennel name
4 - print all dogs who like bones
5 - print all cats that can share a run
6 - search for a dog
7 - search for a cat
8 - remove a dog
9 - remove a cat
10 - set kennel capacity
11 - print all animals alphabetically
q - Quit
What would you like to do:
```

Displaying all dogs who like bones



```
Terminal
Dogs with bones:
Dog name: Rover
Likes bones?: true
Original Owner(s): [Chris Loftus 1234]
Favourite food: biscuits
Needs walked?true

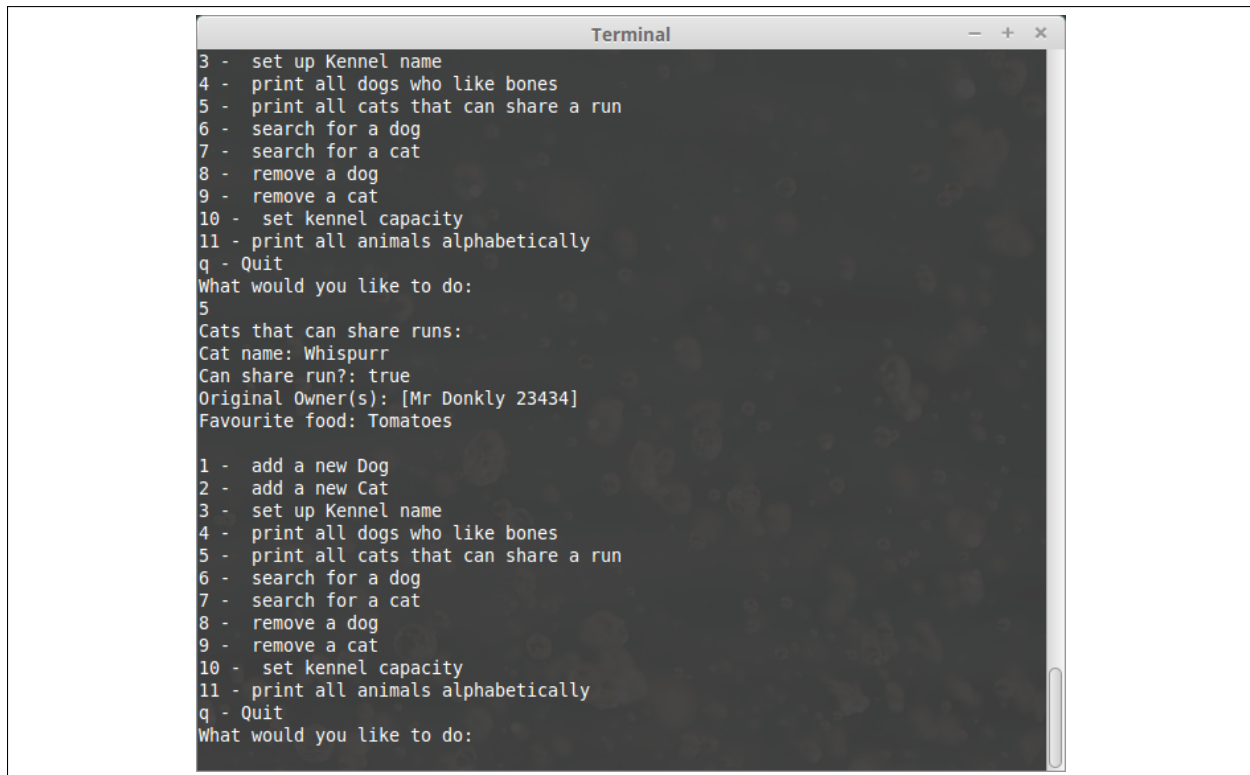
Dog name: Ruffles
Likes bones?: true
Original Owner(s): [Freddrick 123123]
Favourite food: Dog food
Needs walked?true

Dog name: Scooby
Likes bones?: true
Original Owner(s): [Nathan 02891343]
Favourite food: Soybeans
Needs walked?false

Dog name: Will
Likes bones?: true
Original Owner(s): [Mr Bean 00234324, Mrs Bean 93012389]
Favourite food: dog treats
Needs walked?false

Dog name: Snowy
Likes bones?: true
Original Owner(s): [Mark 15234]
Favourite food: Chai tea muffins
Needs walked?true
```

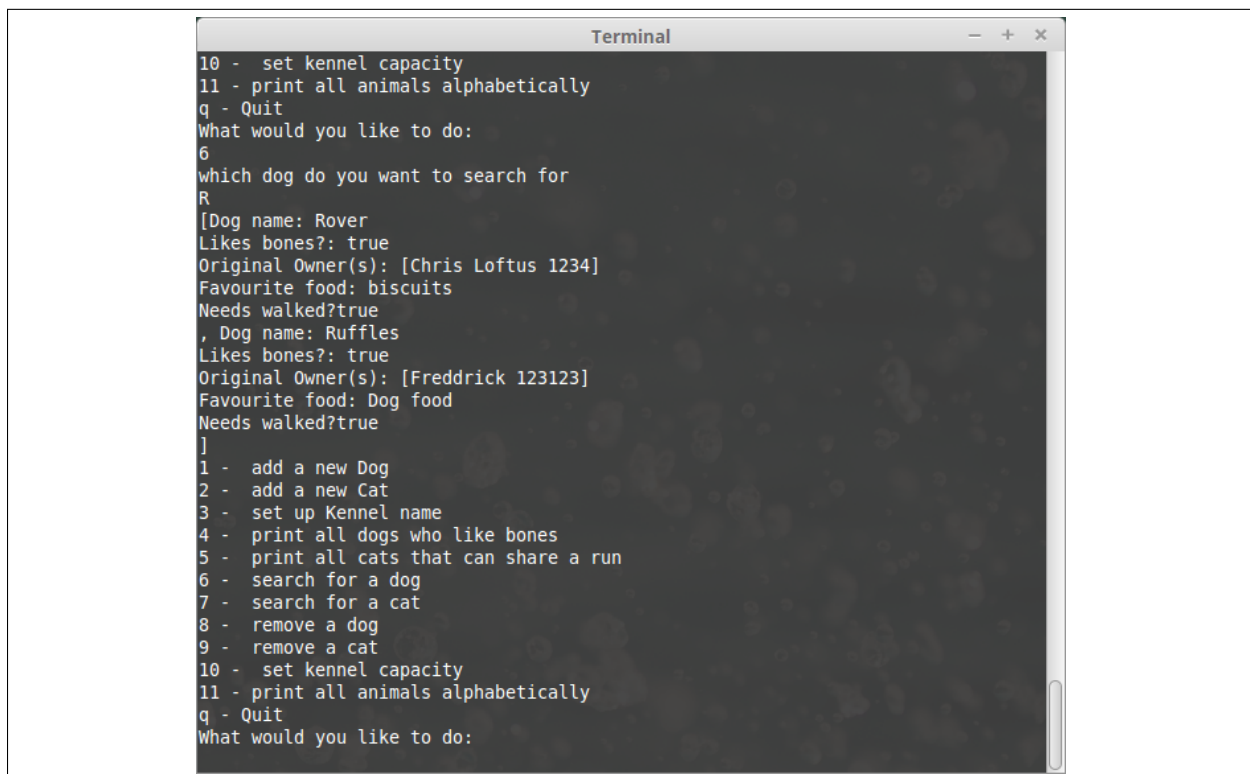
Displaying all cats that care share a run



```
Terminal
3 - set up Kennel name
4 - print all dogs who like bones
5 - print all cats that can share a run
6 - search for a dog
7 - search for a cat
8 - remove a dog
9 - remove a cat
10 - set kennel capacity
11 - print all animals alphabetically
q - Quit
What would you like to do:
5
Cats that can share runs:
Cat name: Whispurr
Can share run?: true
Original Owner(s): [Mr Donkly 23434]
Favourite food: Tomatoes

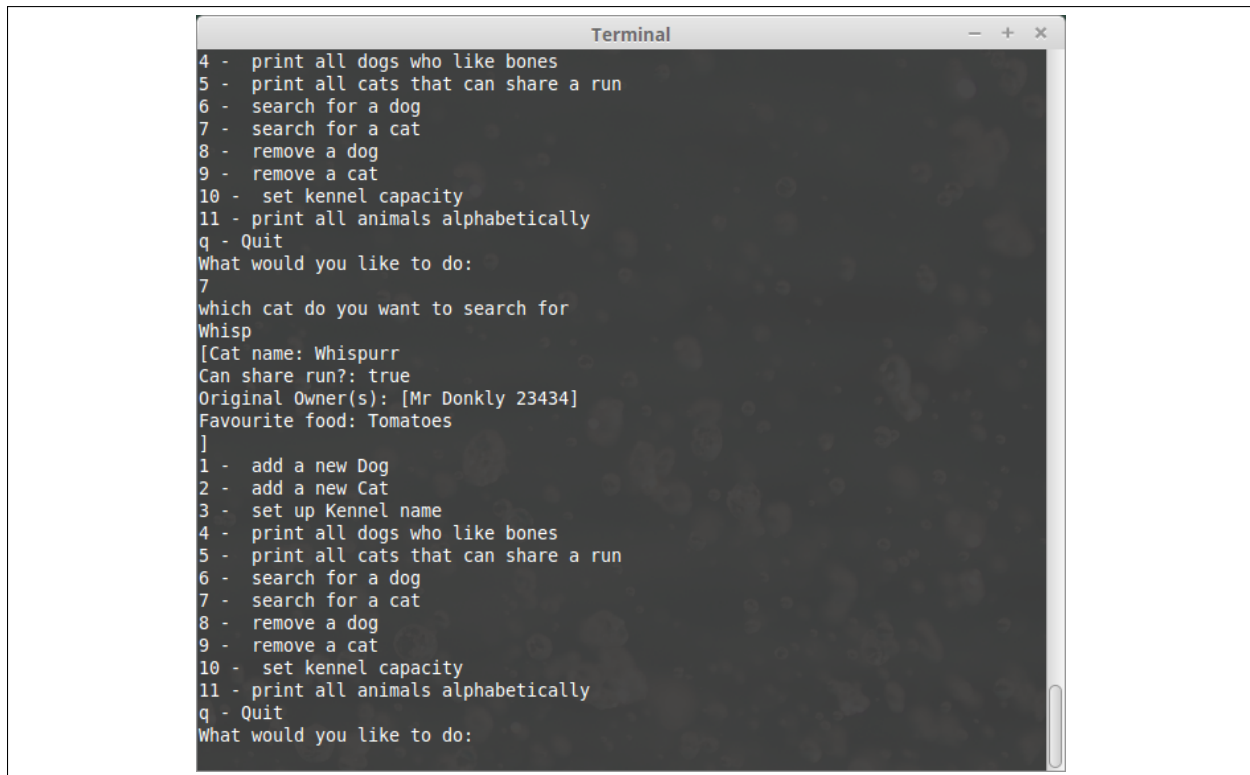
1 - add a new Dog
2 - add a new Cat
3 - set up Kennel name
4 - print all dogs who like bones
5 - print all cats that can share a run
6 - search for a dog
7 - search for a cat
8 - remove a dog
9 - remove a cat
10 - set kennel capacity
11 - print all animals alphabetically
q - Quit
What would you like to do:
```

Searching for dog



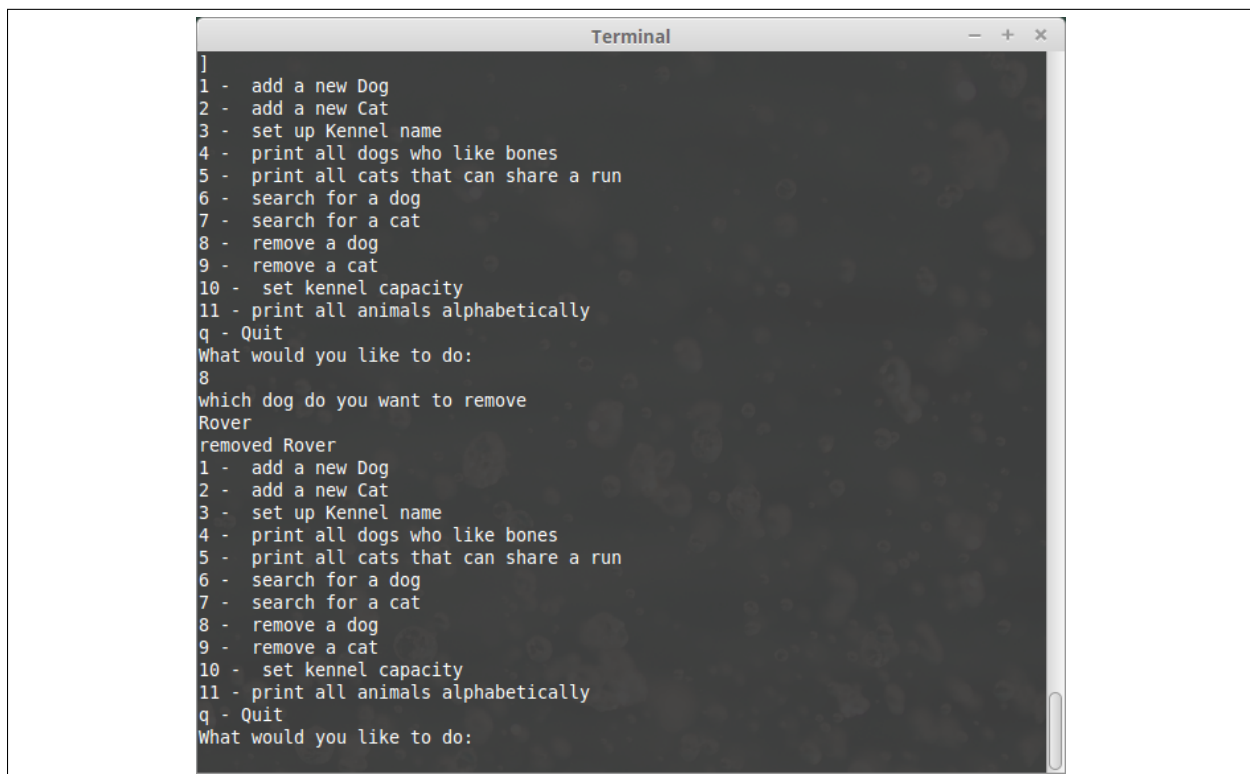
```
Terminal
10 - set kennel capacity
11 - print all animals alphabetically
q - Quit
What would you like to do:
6
which dog do you want to search for
R
[Dog name: Rover
Likes bones?: true
Original Owner(s): [Chris Loftus 1234]
Favourite food: biscuits
Needs walked?true
, Dog name: Ruffles
Likes bones?: true
Original Owner(s): [Freddrick 123123]
Favourite food: Dog food
Needs walked?true
]
1 - add a new Dog
2 - add a new Cat
3 - set up Kennel name
4 - print all dogs who like bones
5 - print all cats that can share a run
6 - search for a dog
7 - search for a cat
8 - remove a dog
9 - remove a cat
10 - set kennel capacity
11 - print all animals alphabetically
q - Quit
What would you like to do:
```

Searching for cat

A terminal window titled "Terminal" with standard window controls. It displays a menu of 11 options. Option 7, "search for a cat", is selected. The program prompts "What would you like to do:" and the user enters "7". It then prompts "which cat do you want to search for" and the user enters "Whisp". The program displays the details for a cat named "Whispurr": "Can share run?: true", "Original Owner(s): [Mr Donkly 23434]", and "Favourite food: Tomatoes". It then returns to the main menu.

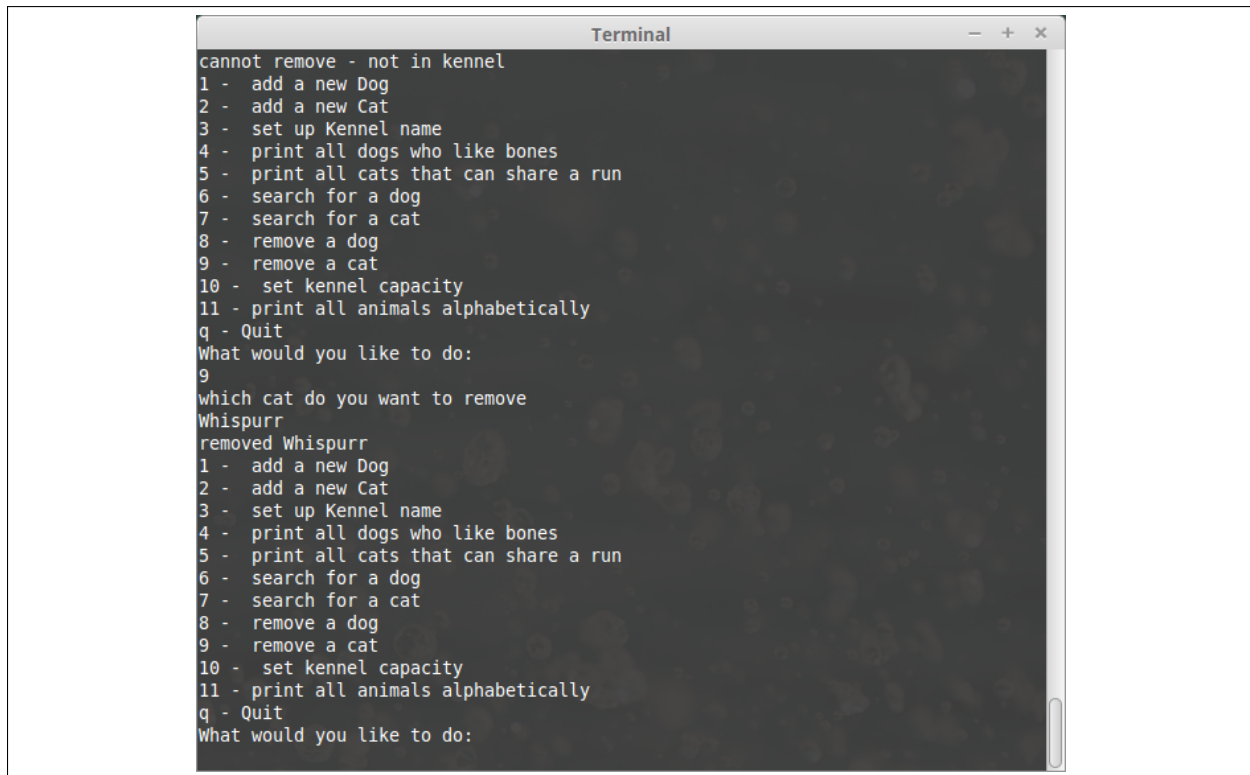
```
Terminal
4 - print all dogs who like bones
5 - print all cats that can share a run
6 - search for a dog
7 - search for a cat
8 - remove a dog
9 - remove a cat
10 - set kennel capacity
11 - print all animals alphabetically
q - Quit
What would you like to do:
7
which cat do you want to search for
Whisp
[Cat name: Whispurr
Can share run?: true
Original Owner(s): [Mr Donkly 23434]
Favourite food: Tomatoes
]
1 - add a new Dog
2 - add a new Cat
3 - set up Kennel name
4 - print all dogs who like bones
5 - print all cats that can share a run
6 - search for a dog
7 - search for a cat
8 - remove a dog
9 - remove a cat
10 - set kennel capacity
11 - print all animals alphabetically
q - Quit
What would you like to do:
```

Removing a dog

A terminal window titled "Terminal" with standard window controls. It displays the same menu of 11 options. Option 8, "remove a dog", is selected. The program prompts "What would you like to do:" and the user enters "8". It then prompts "which dog do you want to remove" and the user enters "Rover". The program displays "removed Rover" and then returns to the main menu.

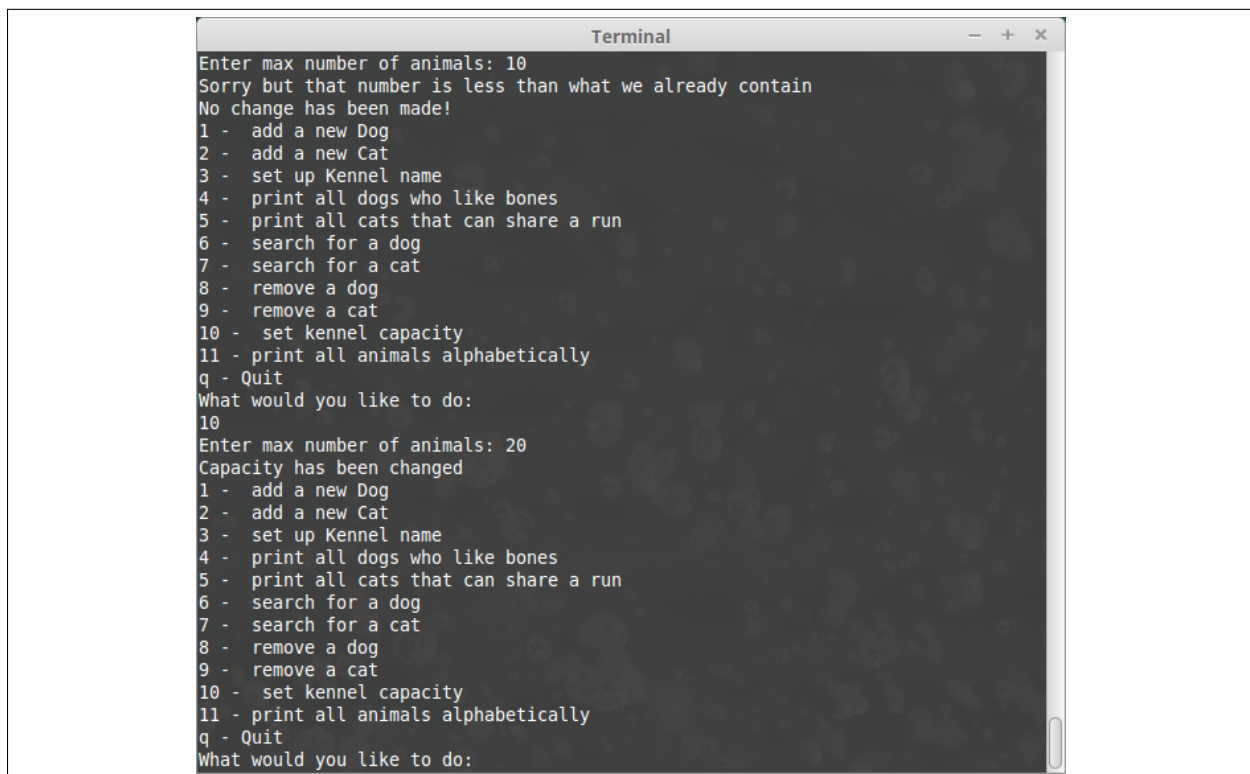
```
Terminal
1 - add a new Dog
2 - add a new Cat
3 - set up Kennel name
4 - print all dogs who like bones
5 - print all cats that can share a run
6 - search for a dog
7 - search for a cat
8 - remove a dog
9 - remove a cat
10 - set kennel capacity
11 - print all animals alphabetically
q - Quit
What would you like to do:
8
which dog do you want to remove
Rover
removed Rover
1 - add a new Dog
2 - add a new Cat
3 - set up Kennel name
4 - print all dogs who like bones
5 - print all cats that can share a run
6 - search for a dog
7 - search for a cat
8 - remove a dog
9 - remove a cat
10 - set kennel capacity
11 - print all animals alphabetically
q - Quit
What would you like to do:
```


Removing a cat



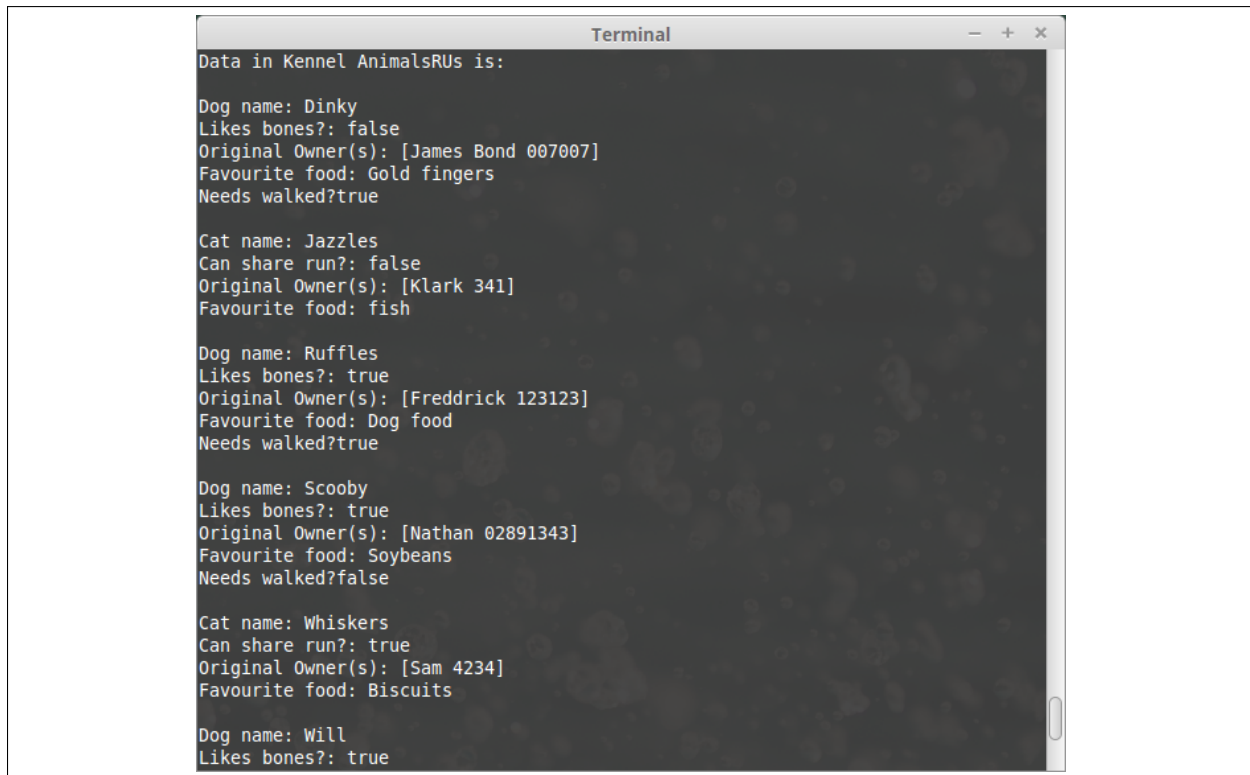
```
Terminal
cannot remove - not in kennel
1 - add a new Dog
2 - add a new Cat
3 - set up Kennel name
4 - print all dogs who like bones
5 - print all cats that can share a run
6 - search for a dog
7 - search for a cat
8 - remove a dog
9 - remove a cat
10 - set kennel capacity
11 - print all animals alphabetically
q - Quit
What would you like to do:
9
which cat do you want to remove
Whispurr
removed Whispurr
1 - add a new Dog
2 - add a new Cat
3 - set up Kennel name
4 - print all dogs who like bones
5 - print all cats that can share a run
6 - search for a dog
7 - search for a cat
8 - remove a dog
9 - remove a cat
10 - set kennel capacity
11 - print all animals alphabetically
q - Quit
What would you like to do:
```

Changing capacity of kennel



```
Terminal
Enter max number of animals: 10
Sorry but that number is less than what we already contain
No change has been made!
1 - add a new Dog
2 - add a new Cat
3 - set up Kennel name
4 - print all dogs who like bones
5 - print all cats that can share a run
6 - search for a dog
7 - search for a cat
8 - remove a dog
9 - remove a cat
10 - set kennel capacity
11 - print all animals alphabetically
q - Quit
What would you like to do:
10
Enter max number of animals: 20
Capacity has been changed
1 - add a new Dog
2 - add a new Cat
3 - set up Kennel name
4 - print all dogs who like bones
5 - print all cats that can share a run
6 - search for a dog
7 - search for a cat
8 - remove a dog
9 - remove a cat
10 - set kennel capacity
11 - print all animals alphabetically
q - Quit
What would you like to do:
```


Printing all animals alphabetically

A terminal window titled "Terminal" with standard window controls. It displays the output of a program that prints data for "Kennel AnimalsRUs". The data is organized into groups for each animal, listing their name, whether they like bones, their original owner(s), their favourite food, and whether they need to be walked. The animals listed are Dinky, Jazzles, Ruffles, Scooby, Whiskers, and Will.

```
Terminal
Data in Kennel AnimalsRUs is:

Dog name: Dinky
Likes bones?: false
Original Owner(s): [James Bond 007007]
Favourite food: Gold fingers
Needs walked?true

Cat name: Jazzles
Can share run?: false
Original Owner(s): [Klark 341]
Favourite food: fish

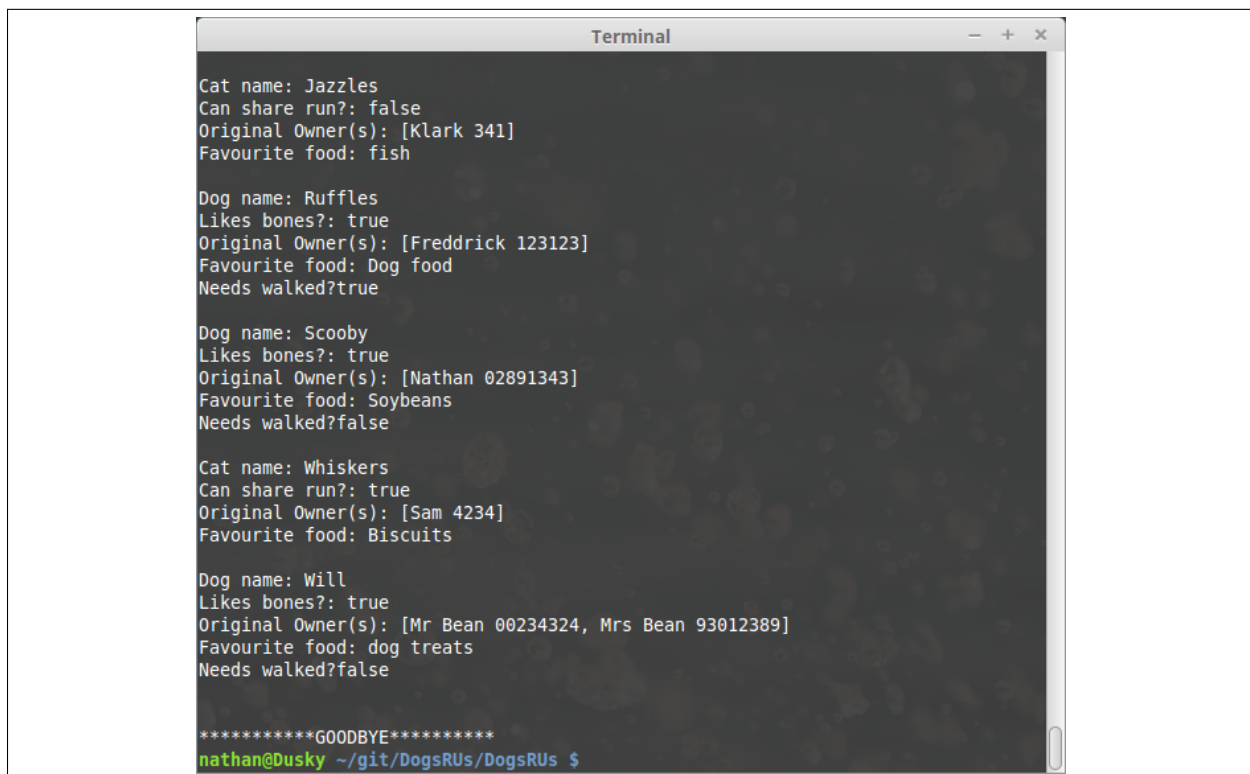
Dog name: Ruffles
Likes bones?: true
Original Owner(s): [Freddrick 123123]
Favourite food: Dog food
Needs walked?true

Dog name: Scooby
Likes bones?: true
Original Owner(s): [Nathan 02891343]
Favourite food: Soybeans
Needs walked?false

Cat name: Whiskers
Can share run?: true
Original Owner(s): [Sam 4234]
Favourite food: Biscuits

Dog name: Will
Likes bones?: true
```

Exiting program

A terminal window titled "Terminal" showing the continuation of the program's output. It lists the details for Jazzles, Ruffles, Scooby, Whiskers, and Will. After the last animal's details, the program prints "*****GOODBYE*****" and the user's prompt "nathan@Dusky ~/git/DogsRUs/DogsRUs \$".

```
Terminal

Cat name: Jazzles
Can share run?: false
Original Owner(s): [Klark 341]
Favourite food: fish

Dog name: Ruffles
Likes bones?: true
Original Owner(s): [Freddrick 123123]
Favourite food: Dog food
Needs walked?true

Dog name: Scooby
Likes bones?: true
Original Owner(s): [Nathan 02891343]
Favourite food: Soybeans
Needs walked?false

Cat name: Whiskers
Can share run?: true
Original Owner(s): [Sam 4234]
Favourite food: Biscuits

Dog name: Will
Likes bones?: true
Original Owner(s): [Mr Bean 00234324, Mrs Bean 93012389]
Favourite food: dog treats
Needs walked?false

*****GOODBYE*****
nathan@Dusky ~/git/DogsRUs/DogsRUs $
```