Layla Oesper

This assignment is due by **10pm on Monday September 25** and is worth 20 points.

1 Goals

The goal of this assignment is to get some more practice using inheritance and interfaces. You will also get more practice building some slightly larger Java programs and using good object oriented programming design.

2 Your Assignment

This is a partner assignment. You will work with the partner that you have been assigned. I highly suggest that you use a timer to trade-off driving and navigating so that you both get an equal opportunity to try out both roles.

For this assignment you'll be creating a vet clinic application where different types of pets (Cats and Dogs) can have records in the system and can be boarded while their owners are out of town. I'll outline below the different classes that you'll need to create for this application. Please make sure to adhere *exactly* to any descriptions about what files, methods or variables should be named. It's important that you build a program that adhere's the the specifications outlined here.

2.1 Pet Class

Create a Pet class (inside `Pet.java`) that stores information about a particular pet. Specifically, your class should have the following instance variables:

Variable	Variable Type
pet name	String
owner name	String
color	String
sex	int (see additional details below)

For the instance variable storing the sex of the pet, the integer values should correspond to one of the following five *public static final int* values: MALE, FEMALE, SPAYED, NEUTERED, OTHER. You should define these five static final values in your class.

You class should implement the **public** methods outlined in Table 1. The `toString()` method should return the pet's name, owner's name, color and sex (use getSex()). Here is an example of the preferred return value by `toString()`.

```
Yawgoo owned by Layla
Color: Brown
Sex: male
```

Pet Methods
`Pet(String name, String ownerName, String color) //Constructor`
`String getPetName()`
`String getOwnerName()`
`String getColor()`
`void setSex(int sexID)`
`String getSex() //returns string equivalent of sex, e.g. SPAYED`
`String toString() //see below for details`

Table 1: Methods to implement in Pet class.

2.2 Boardable Interface

Create a Boardable interface (inside `Boardable.java`) that describes information relating to an animal that can be boarded at the vet clinic. The interface should include the following **public** methods:

Boardable Methods
`void setBoardStart(int month, int day, int year`
`void setBoardEnd(int moth, int day, int year`
`boolean boarding(int month, int day, int year`

See the Cat and Dog classes for what these methods should do when they are implemented. Note: We will assume that the moth will be in the range 1-12, day in the range 1-31, and year will be a four-digit number.

2.3 Cat Class

Create a Cat class (inside `Cat.java`). This class will *extend* the Pet class and *implement* the Boardable interface. In addition to the instance variables and methods inherited from the Pet class, the Cat class should have another instance variable `String hairLength`. The class should also have the following methods.

Cat Methods
`Cat(String name, String ownerName, String color, String hairLength)`
`String getHairLength()`
`String toString() //see below for details`

Here is an example of the preferred return value by `toString()`.

```
CAT
Yawgoo owned by Layla
Color: Brown
Sex: male
Hair: medium
```

In order to implement the `Boardable` interface, you'll need to define new instance variables to store the boarding start and end dates. You should store these as `Date` objects (don't forget to import `java.util.Date` and check out the associated javadocs). You'll also need to implement the methods outline in the `Boardable` interface. To create a `Date` object, you'll want to use the following command `new GregorianCalendar(year, month-1, day).getTime()` where year, month and day are all integers. If you read the javadocs you'll see that `month` is 0 based in this class, which is why we subtract 1 from the month. Lastly, the `boarding` method should return true if the given date is in-between the start and end date (it should also return true if it is equal to either the start or the end date). I suggest looking at the `Date` javadocs to see if there are any useful methods.

2.4 Dog Class

Create a Dog class (inside `Dog.java`). This class will *extend* the Pet class and *implement* the Boardable interface. In addition to the instance variables and methods inherited from the Pet class, the Dog class should have another instance variable `String size`. The class should also have the following methods.

Dog Methods
`Dog(String name, String ownerName, String color, String size)`
`String getSize()`
`String toString() //see below for details`

Here is an example of the preferred return value by `toString()`.

```
DOG
Lana owned by Richard
Color: Red
Sex: spayed
size: large
```

In order to implement the `Boardable` interface, you'll need to define new instance variables to store the boarding start and end dates. You should store these as `Date` objects (don't forget to import `java.util.Date` and check out the associated javadocs). You'll also need to implement the methods outline in the `Boardable` interface. To create a `Date` object, you'll want to use the following command `new GregorianCalendar(year, month-1, day).getTime()` where year, month and day are all integers. If you read the javadocs you'll see that `month` is 0 based in this class, which is why we subtract 1 from the month. Lastly, the `boarding` method should return true if the given date is in-between the start and end date (it should also return true if it is equal to either the start or the end date). I suggest looking at the `Date` javadocs to see if there are any useful methods.

2.5 VetClinic Class

This class does not extend any other classes or implement any interfaces, although it will use the other classes you have developed. You should provide the following **public** methods in this class (you may certainly create other methods if you find them useful).

VetClinic Methods
`VetClinic(String InputFile)`
`public Boardable[] getAllPets()`
`void printAllPets()`
`void printAllBoarding(int month, int day, int year)`
`void printAllOwnedBy(String owner)`

These methods should have the following functionalities:

- `VetClinic(String InputFile)` - This constructor should take the name of a file as an argument and will read in information from the file. You can use the Scanner class to read input from a file. The input file will consist of a series of records for different animals, where each record consists of multiple lines. The first line of the file will indicate the number of animal records in the file. Following that, each animal record will start with either "CAT" or "DOG" on a single line, to signify the type of record. The following line of the file will contain the pet's name, owner's name, its color, sex and either it's hair length (if it is a cat) or its size (if it is a dog). Note, the sex of the animal will an integer value (that should match one of the public static final variables you encoded in your class). You should read each line of the file, create objects for each pet and add them to an array (you'll want to store this as an instance variable for the class). Think carefully about what type of object you think this array should store. An example input file to help you with testing is provided on the Moodle page for this assignment.
- `public Boardable[] getAllPets()` - This method will return the array of objects representing the pets that are currently stored in the instance of a VetClinic.
- `void printAllBoarding(int month, int day, int year)` - Use the appropriate `toString()` method to print to the screen all pets that will be boarding on the specified data. Note, to test this method in a reasonable matter you will need to invoke `setBoardStart` and `endBoardStart` methods a few times on some of the animals since this information is not contained in the file you read in and is not set by the constructors.
- `void printAllOwnedBy(String owner)` - Use the appropriate `toString()` method to print to the screen all pets that have the owner specified by the input parameter `owner`.

2.6 Final Thoughts

- At times you'll have to think carefully about what type of variable you are manipulating when you are calling certain methods. You may have to cast variables between types to get all the functionality to work correctly.
- Keep testing your program as you write it. When I did this assignment I actually wrote a `main()` method for every class that I could use to test out the functionality of the class. This make it a lot easier when I got to writing the `VetClinic` class.

3 Submission and Grading

You'll submit all your files to Moodle as a zipped file. One one partner from each pair needs to submit the assignment. Specifically, put these files into a directory named `[your_last_name_your_partner's_last_name]HW4`, zip this directory, upload it to Moodle. For example, if my partner was Schiller my directory would be named `OesperSchillerHW4` and the resulting file would be `OesperSchillerHW4.zip`.

3.1 Grading

This assignment is worth 20 points. Below is a partial list of the things that we'll look for when evaluating your work.

- Do all of your classes contain the correct variables and methods? Are they named appropriately? Do they have the correct type (public/private/static/final/etc.)?
- When testing your code we will create an object of type `VetClinic` by passing it a file and running your public methods of various classes. I suggest you test your code in a similar manner.
- Do you handle interfaces and inheritance correctly?
- Do your classes exhibit good organization and commenting? Don't forget to follow the Style Guidelines on Moodle.

Start early, ask lots of questions, and have fun! Layla, the lab assistants, and the prefect are all here to help you succeed - don't hesitate to ask for help if you're struggling!