✔ Done

Well done in completing Part 1: Encapsulating the generic dog features into *DreamDog*, has significantly improved the OO design of Pinkman's Pet finder.

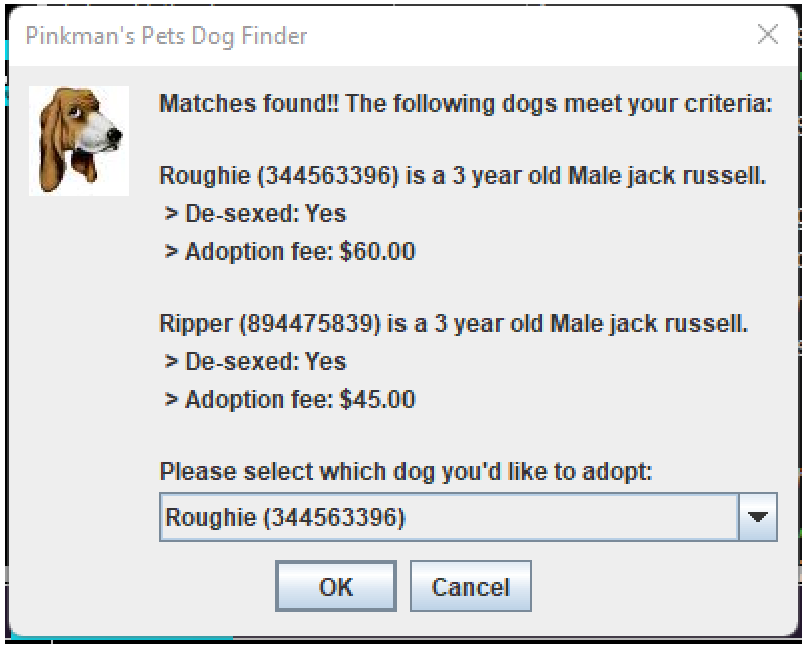
Pinkman has since reached out, with the following message:

"Hi, the program is working very well, and I'm getting good feedback from customers. However, a few people have commented that they would like to know the adoption fee before they put in an application. The fee varies per dog, and I usually work it out based on the dog's age, whether they're purebred or not, or how in-demand they are. Some customers have also been saying it would be better if they could also add purebred in as a criteria (e.g. a drop down list with YES, NO, NA) too, to help them better choose their dog. Could you please update the program?"

First, let's read his message carefully to determine which changes are needed. You should be able to identify 2 changes - the first is to add price to the output before users choose which dog they want (if any), and the second is to add functionality to allow users to choose whether or not they want a purebred dog or not.

Part 1

1. The first task will be to add the adoption fee. Do you think it should go in the *Dog* or *DreamDog* class?
2. If you decided it should go in the *Dog* class, well done! Go ahead and add an adoption fee field, as well as a getter. You'll also have to add adoption fee into the constructor to initialise your field.
3. This should create a red line in *FindADog loadDogs.* Let's go and fix it.
   HINT: look at the file. What index position adoption fee at? Use *parseDouble* to convert it to a *double* before using it in the *Dog* instantiation. Remember to handle exceptions!
4. Now, let's take advantage of our new functionality and go to the *main* method. Add the adoption fee to the *String* describing each dog (*infoToShow*). Your output could look something like this:



```
Matches found!! The following dogs meet your criteria:

Roughie (344563396) is a 3 year old Male jack russell.
 > De-sexed: Yes
 > Adoption fee: $60.00

Ripper (894475839) is a 3 year old Male jack russell.
 > De-sexed: Yes
 > Adoption fee: $45.00

Please select which dog you'd like to adopt:
Roughie (344563396)
```

Part 2

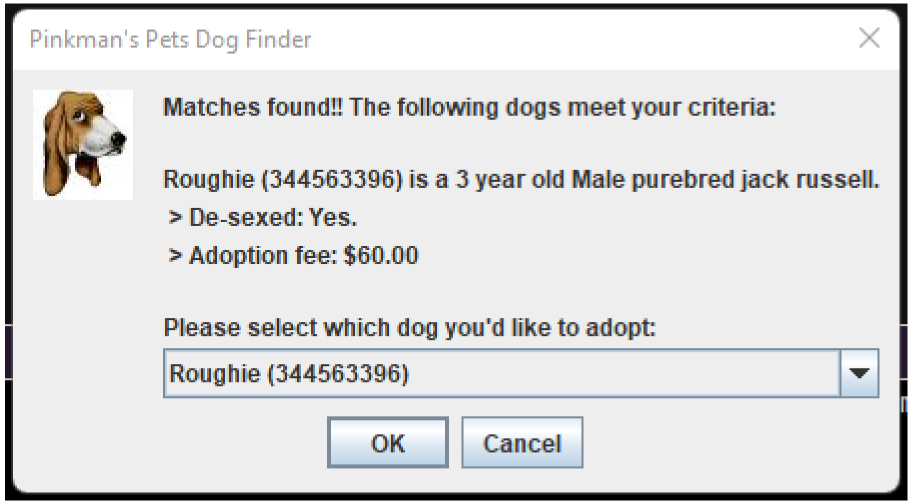
1. Did that feel clunky? We really should delegate the job of describing the *Dog* object to *Dog*, and delegate the job of describing the *DreamDog* features to *DreamDog*...
2. To do this, create a method in *DreamDog* called *getDreamDogDescription*, using the class fields to create and return a *String* that describes the *DreamDog* features in the correct format.
3. Next, create a method in *Dog* called *getDogDescription* to describe the Dog features, remembering to call the *getDreamDogDescription* method to add the real dog's 'dream' features to the String. This is important, so that *Dog* doesn't have to worry about which features *DreamDog* has to print.
4. Remember to go back into *FindADog* main and call the *getDogDescription* method to generate the search results.
5. And it's done! Users can now see the adoption fee for each dog that meets their criteria, and the appropriate tasks have been delegated to the appropriate classes, keeping them nicely decoupled.

Part 3

1. Now, let's build the purebred feature into Pinkman's Pet Finder. First, consider what datatype should be used? What are the range of options Pinkman provided?
2. Because the user selections are well-defined and finite, we should create an *Enum* called *Purebred.*
3. Do you think the new *Purebred* variable should go in the *Dog* or *DreamDog* class?
4. Because it is a generic characteristic that a user can use to filter dogs, it should go in *DreamDog*, so let's add it in! Remember to change the constructor, create a getter, and if purebred="yes", add it to the description.
5. Next, add a purebred parameter in the *Dog* constructor (you should be sensing bad design here!) so that its *DreamDog* can be properly initialised.
6. Now, go into *FindADog*, and use a *JOptionPane* dropdown list to get the user to choose whether or not they want a purebred (or NA)
7. Next, pass the user's choice into the *DreamDog* instantiation
8. Next, in *loadDogs*, read the purebred status from the file, and pass it as a parameter in the *Dog* instantiation.
9. ...and finally, in *AllDogs*, change *findMatch* to consider purebred status too (sensing more bad design?)
10. Test your program to see if it still works:

```
jack russell

Male

Yes

Yes

1

7
```

You should get the following output:



```
Matches found!! The following dogs meet your criteria:

Roughie (344563396) is a 3 year old Male purebred jack russell.
 > De-sexed: Yes.
 > Adoption fee: $60.00

Please select which dog you'd like to adopt:
Roughie (344563396)
```

Fill in the remainder of the info as done previously.

Part 4

1. Now let's address the bad design choices...
2. The first one is the instantiation of *DreamDog* inside *Dog*. Let's decouple those classes by just passing a *DreamDog* object into *Dog* as a parameter instead, instead of breed, sex, etc....
   1. This will require removal of the *DreamDog* fields from the *Dog* constructor, AND
   2. Creation of a *DreamDog* object in the *FindADog loadDogs* method.

   Now, *Dog* and *DreamDog* are decoupled - they are independent of each others' implementations...

3. Let's now address the other bad design choice we noticed.... *AllDogs findMatch* shouldn't be comparing *DreamDog* characteristics. Those searches should be delegated to *DreamDog*:
   1. Create a *boolean* method in *DreamDog* called *compareDreamDogs*.
   2. Move the comparison of breed, sex, desexed status, and purebred status to this method.

3. In *AllDogs findMatch*, call *compareDreamDogs*, passing in the *dogCriteria* object

Now, *AllDogs* and *DreamDog* have been largely decoupled. The code is a lot simpler and cleaner - much better design!

Well done! You have completed tutorial 5! Re-run your program to ensure it still works.

Last modified: Friday, 22 July 2022, 4:24 PM