Mark as done

In this week's tutorial, you will implement a subscription interface for Pinkman's Pet Finder in accordance with the requirements below:

- There should be two types of user, i.e., premium and basic (we should be able to add other subscription levels to this later, e.g. Gold Member).
- Basic users should have access to the following search filters:
  - type of pet, e.g., dog, cat, guinea pig etc.
  - breed of pet, e.g. poodle, dalmatian
  - desexed status
  - age range
- After a basic user has entered their search info (but before the search results are displayed) an advertisement (pop-out dialog) should be displayed. This advertisement should state that premium users are able to use more search filters (list all those not in the list above, e.g. purebred, hair, etc.) and can see unlimited results. Include the premium fee in a call to action within this ad.
- Basic users should only be able to see a maximum of 3 search results (the message should state how many results there were all up). The search results should show all pet-type relevant data, even if the user has selected *NA*, e.g. it should show the dog's sex even if the user selects *NA*.
- After the user has selected a pet from the dropdown list, an ad should be displayed stating that premium users get 10% off their adoption fee and a $100 vet voucher for their new pet.
- Premium users pay a $14.99/year subscription fee.

- Premium users should be able to search with all current options, and have no ads. Further, they should be notified once they submit their request that they will get 10% off their adoption fee and a $100 vet voucher for their new pet if the adoption proceeds.

Part 1

Pinkman's new Pet Finder is essentially requesting that each type of subscription has the same methods, e.g., to obtain user input for search criteria, display results and place an adoption request, but that these tasks be fulfilled in different ways.

Therefore, it is appropriate to design an interface that contains empty method headers for these tasks, then create two classes called *Premium* and *Basic*, which will implement these methods (Part 2). You will find that a lot of the code currently in *FindAPet* will move to the interface and two classes.

1. First, create the *Subscription* interface, including 3 method headers;
    1. *getUserInput*
        1. return type *DreamPet* - which will represent the user's desired pet
        2. parameter: *Set<String>* - a set of all the available dog breeds
        3. parameter: *PetType* - the type of pet (cat, dog or guinea pig)

    2. *displayResults*
        1. return type *Pet* - the user's chosen pet
        2. parameter: *List* of *Pet*s - the real pets that meet the user's criteria
        3. parameter: *Criteria[]* - an array of Criteria to control what pet features are displayed

    3. *placeAdoptionRequest*
        1. return type *void*
        2. parameter: *Pet* - the real pet chosen by the user

2. Create a constant called *appName* in *Subscription.* This is to be used across all the *JOptionPane*s in the program. Do the same with the icon (delete the icon variable in *FindAPet*).
3. Create a constant called *premiumSubscriptionFee* in *Subscription*. Also create constants for the user discount and voucher amount.
4. Next, create *default* methods for getting individual user input items, e.g., type, breed (HINT: include a *Set<String> allBreeds* parameter), purebred, sex, desexed status, hair, age range and fee range (move the *minMaxValues* method from *FindAPet* to *Subscription*). Most of this code can be obtained from the *FindAPet getUserCriteria* method.
5. Also create 3 *default* methods to get the user's name, email and phone number. Most of this code can be obtained from the *FindAPet getUserDetails* method.
6. Finally, move the *writeAdoptionRequestToFile* method from *FindAPet* to *Subscription*. Add a new parameter (*String preface*) to the list of parameters. This will be used to track whether a user is premium, premium unverified (in the case that the file doesn't load or the premium user database is unavailable, or basic.
   HINT: the filepath should be

```
String filePath = preface+person.getName().replace(" ","_")+"_adoption_request.txt";
```

These methods you've implemented in steps 4 to 6 should be default methods, because they will be used across all implementing classes (since implementing classes combine them in various ways). Note: if you only included their method headers in the interface, you would have a lot of duplicate code, because 'how' the user selects criteria, e.g. breed from a dropdown list, will not vary across classes.

Part 2

Next, create 2 new classes; *Premium* and *Basic*. These will implement *Subscription*, meaning they must contain implementations for *getUserInput, displayResults* and *placeAdoptionRequest*.

Within *Premium*:

1. Create a *Map* with key type *Criteria* and value type *Object*. This will store the user's selections.
2. Create a *String* field called *premiumUserDataFile* (you will assign the *premiumUserInfo.txt* file to it in the constructor).
3. Create a *Map<String, Person>* field to contain the premium users' emails, mapped to Person objects, created using their name, email and ph num (from file).

4. Create a constructor that takes in a String (the *premiumUserInfo.txt* file), and initialise all the fields.
5. To implement the *getUserInput* method:
    1. Call all the default methods in *Subscription* to get user input, adding that input to the *Map* field.
    2. Use the *Map* field, and age range and fee range info to initialise and return a *DreamPet* object.

       Note: if the type is cat or guinea pig, request user input on hair type.
    3. Also, use the *allBreeds* parameter to pass into the *getUserInputBreed* method.

6. Use the code in *FindAPet* to implement the *displayResults* method:
    1. This method should contain a *Map* of pet options the user can choose from.
    2. Iterate through all the pets in the parameter *List*, creating a description by concatenating *Pet* info.
    3. Pass the *Criteria[]* parameter in the *getPetDescription* method call. It'll cause a red line. To fix it, change the parameter type of *DreamPet getDreamPetDescription* to *Criteria[]*, fixing any resulting errors in the method. Code the same change in *Pet getPetDescription*.
    4. Present this info to the user in a drop-down list, saving their decision as a *String.*
    5. Use this *String* to get the associated *Pet* from the *Map*, returning the chosen *Pet*.

7. Implement a *boolean* method called *loadPremiumData*, ensure it:
    1. loads the premium user info file, loop through its elements, creating one *Person* object per line. It should fill the contact info *Map* field with the data in the form email: *Person* (ensure exception handling).
    2. If the file can't load return *false*, other return *true*.

8. To implement *placeAdoptionRequest*:
    1. get the user to enter their email address (this will be used to verify if they are premium users).
    2. call *loadPremiumData* to populate the *Map* with premium user data.
    3. if the method returns *false*, let the user know their account couldn't be found, and use the default methods in *Subscription* to get them to enter their info instead, using their data to initialise the *Person user* field.

       HINT: if the file couldn't be loaded, preface the file path with "*unverified*".

4. if the method returns *true*, check that the *Map* contains the email. If it doesn't, let the user know that no such acct exists, and terminate.
5. Otherwise, use the user's email to initialise the *Person user* field to the corresponding premium user from the *Map* (*"premium_user"* should be concatenated to preface).
6. Call the *writeAdoptionRequestToFile* method to place the adoption request.
7. Remember to let the user know they have a $100 voucher and 10% off their adoption fee if the adoption goes ahead.

Part 3

In *FindAPet main*, you should be initialising the *AllPets* field. You should also have the *JOptionPane* introducing the user (use the constants in the interface in the *JOptionPane* for the app name and icon).

1. Beneath the welcome message, use a *JOptionPane* to ask the user whether they have an account.
2. If the user has an account, create a *Premium* object (remember to pass in the premium user file), assigning it to a *Subscription* variable (coding to an interface). Otherwise, create a *Basic* object.
3. Next, call the *getUserInputType* method on the *user* object, saving the result in a variable called *type*.
4. Next, create a *DreamPet* object, initialising it by calling the *getUserInput* method (remembering to pass in *type*, and *allPets.getAllBreeds(type))* as arguments.
5. As was previously done, pass the *DreamPet* object into *processSearchResults.*
6. In *FindAPet processSearchResults: i*f there are compatible *Pet*s, call *user*'s *displayResults* method, passing in the potential matches and *Criteria.values()*. Pass the user's selected pet (returned by *displayResults*) into *user.placeAdoptionRequest()*.
7. If there are no matches, output a message letting the user know (as previously done) and exit the program.

Part 4

Within *Basic*:

1. Create a *Map* with key type *Criteria* and value type *Object*. This will store the user's selections.
2. Create an *int* field called *numberOfSearchResultsToShow*.

3. Create a constructor to initialise the fields. It should have one int parameter to initialise the *numberOfSearchResultsToShow*.
   HINT: go back to *FindAPet* and ensure you add *3* as an argument in the *Basic* object declaration.
4. Implement the *getUserInput* method using the default methods to get user input for breed, desexed and age range.
    1. You'll have to create a constructor in *DreamPet* that doesn't take fee range
    2. You'll also have to skip checking of fee range in *AllPets findMatch* if the min and max fee values are at their default initialisation value.
    3. Remember to add a message dialog advertising the premium subscription before the method returns.

5. Implement *displayResults* as was done in *Premium*.
    1. Let the user know how many results meet their criteria, but only show them a subset.
    2. Ensure you let them know that only a max of 3 results can be shown on the *Basic* plan.
    3. Pass the *Criteria[]* parameter into the *getPetDescription* method call. Is there a better way to do this? For ideas, see how *Set*s of *Map* keys were included in the interface in *SeekAGeek* Subscription, to offer better control over what features users can see.
    4. If the user chooses to adopt a pet, calculate 10% of the pet's adoption fee, and tell the user they could save this today, as well as the subscription fee, and the $100 vet voucher for premium users.

6. Implement *placeAdoptionRequest* by:
    1. calling the contact info default methods from *Subscription*, using these values to create the *String* file path.
    2. writing the file using *writeAdoptionRequest*.

And you're done! Experiment with various combinations of pet requests to check if your program is fully functional and robust!

Last modified: Tuesday, 30 August 2022, 7:05 AM