You are REQUIRED to use the VirtualPet files I created and they can NOT be changed.

Step 1: VirtualPet class and derived classes:

Step 1a: VirtualPet class and general instructions

Once you have your project set up carefully read the VirtualPet.h and VirtualPet.cpp files. Note there are two pure virtual functions that any concrete classes you make will be required to override (action() and setType()). Note some of the member variables are protected to make some of the programming easier.

Some tips:

- Note the functions that are const, make sure when you override or re-define any const functions those functions are also const. If you don't, your overrides won't work.

Inheritance Hierarchy:

- Note the VirtualPet class is the abstract base class in an inheritance hierarchy that will have two levels. The second level of the inheritance hierarchy will have two more abstract classes (both of these are derived from VirtualPet): DomesticLandPet and WaterPet. These will be abstract because as you will see they do not override the pure virtual functions in VirtualPet.
- The third level of the inheritance hierarchy will have three concrete derived classes (so these classes will override the pure virtual functions and you will be able to instantiate object from them). The Cat and Dog classes will derive from DomesticLandPet and the Fish class will derive from WaterPet.

Step 1b: DomesticLandPet (derived class, inherits from VirtualPet)
A domestic land pet is a domestic animal with fur that walks on land.

This class is derived from VirtualPet and should use public inheritance.

The class should have the following ADDITIONAL member variables (so these are in addition to the ones inherited from VirtualPet):
- ***licenseFee***: Adopting domestic land pets requires payment of an additional license fee, the license fee can be fractional
- ***vetFees:*** Adopting domestic land pets requires payment of vet fees for a check-up and spaying/neutering.

The class should have the following ADDITIONAL member functions:
- **One Constructor:**
  - The constructor should have five parameters: one of the pet's name, one for the pet's color, one for the base adoption fee, one for the license fee and one for the vet fees. Do not let any of the fees be negative, the fees should have default

    - values of $25 for the adoption fee, $80 for the vet fees and $10 for the license fee.
    - Don't forget to call the base class constructor!!
- **Two ADDITIONAL Set Functions:**
    - Design a set function for the additional member variables: vetFees and licenseFee
        - Do not allow the vetFees or licenseFee to be negative. Use default values of $80 for vetFees and $10 for the license fee.
- **Two ADDITIONAL Get Functions:**
    - Design get functions for vetFees and licenseFee.

<u>The class should OVERRIDE the following member functions:</u>
REMEMBER: any virtual functions that aren't pure that you don't override will inherit the implementation from VirtualPet

- ***setMood*** **Member Function:**
    - First go back and look at how this was implemented in VirtualPet. The only change is domestic pet can have two additional moods: "playful" and "naughty". So it should have the moods VirtualPet has plus these two additional ones. Program it just like setMood in VirtualPet but add these two additional moods. The default mood should still be "content".
- ***calcAdoptionCost*** **Member Function:**
  This member function should calculate the adoptionCost. For a DomesticLandPet:
    - adoptionCost = baseAdoptionFee + vetFees + licenseFee
      Do NOT forget to reuse existing code where you can on this one.
- ***changeMood*** **Member Function:**
  This member function should work like changeMood in VirtualPet with the only difference being there are now 5 moods: content, hungry, sick, playful and naughty. So your random should be from 1 to 5 instead of 1 to 3 to randomly change the mood member variable to one of the five moods. Use changeMood in VirtualPet as a guide to implement this

<u>Step 1c: WaterPet (derived class, inherits from VirtualPet)</u>

A water pet is a virtual pet that lives in the water.

<u>This class is derived from VirtualPet and should use public inheritance.</u>

<u>The class should have the following ADDITIONAL member variables (so these are in addition to the ones inherited from VirtualPet):</u>

- ***waterType:*** The water type can be "fresh" or "salt", no other values are acceptable

<u>The class should have the following ADDITIONAL member functions:</u>

- **One Constructor:**
    - The constructor should have four parameters: one of the pet's name, one for the pet's color, one for the base adoption fee, and one for the waterType. Do not let

- the waterType be anything other than “fresh” or “salt”, use “fresh” as the default value.
  - Don’t forget to call the base class constructor!!
- **One ADDITIONAL Set Function:**
  - Design a set function for the additional member variables: waterType. Do not let the waterType be anything other than “fresh” or “salt”, use “fresh” as the default value.
- **One ADDITIONAL Get Function:**
  - Design a get function for waterType.

<u>The class should OVERRIDE the following member functions:</u>
REMEMBER: any virtual functions that aren’t pure that you don’t override will inherit the implementation from VirtualPet

- ***printInfo*** **Member Function:**
  - This member function should do EXACTLY the same thing printInfo() does in VirtualPet but it should add one line at the bottom:
  - cout << “Water Type “ << waterType << endl;

<u>Step 1d: Cat (derived class, inherits from DomesticLandPet)</u>
Note Cat is a concrete derived class so we must implement all pure virtual functions here, don’t forget const when overriding!!

<u>This class is derived from VirtualPet and should use public inheritance.</u>

<u>The class does not have ANY additional member variables it just uses the inherited ones</u>

<u>The class should have the following ADDITIONAL member functions:</u>

- **One Constructor:**
  - The constructor should have five parameters: one of the pet’s name, one for the pet’s color, one for the base adoption fee, one for the license fee and one for the vet fees. The fees should have default values of $25 for the adoption fee, $80 for the vet fees and $10 for the license fee.
  - This constructor should call the setType() function which we are overriding below.
  - Don’t forget to call the base class constructor!!

<u>The class should OVERRIDE the following member functions:</u>
- ***setType*** Member Function:
  - Don’t forget this one is protected not public (see VirtualPet)
  - Note that this function is void with no arguments (see VirtualPet)
  - All this function should do is set the type member variable equal to “Cat”
- ***action*** Member Function:

- The purpose of this function is to print a statement about what the pet is doing based on the pets mood
- Note this function is void with no arguments and is const (see VirtualPet)
- Here are rules for what to print:

If the mood member variable is “content” print (replace Fluffy with the real name):
“Purr, purr."
"Your cat Fluffy is cuddling in your lap. "

If the mood member variable is “hungry” print (replace Fluffy with the real name):
“MEOW. MEOW.”
"Your cat Fluffy is hungry for some water and fish. "

If the mood member variable is “sick” print (replace Fluffy with the real name):
“Fluffy is quiet and hiding and doesn’t feel well.”
"Time to go to the vet!!”

If the mood member variable is “playful” print(replace Fluffy with the real name):
"Fluffy is playing with a toy mouse. Meow "

If the mood member variable is “naughty” print (replace Fluffy with the real name):
“Scratch. Scratch. Fluffy is scratching the curtains.”
"Time to buy another scratching post or trim nails! "

- ***printInfo*** **Member Function:**
    - This member function should do EXACTLY the same thing printInfo() does in VirtualPet but it should add one line at the bottom (below the mood) that displays the action. So this function should reuse printInfo() from VirtualPet and then call the action function to print the action at the bottom. It might help to look at the sample output in the client program for this one.

Step 1e: Dog (derived class, inherits from DomesticLandPet)

Note Dog is a concrete derived class so we must implement all pure virtual functions here, don’t forget const when overriding!!

This class is derived from VirtualPet and should use public inheritance.

The class does not have ANY additional member variables it just uses the inherited ones

The class should have the following ADDITIONAL member functions:

- **One Constructor:**
    - The constructor should have five parameters: one of the pet’s name, one for the pet’s color, one for the base adoption fee, one for the license fee and one for the

vet fees. The fees should have default values of $25 for the adoption fee, $80 for the vet fees and $10 for the license fee.
- *This constructor should call the setType() function which we are overriding below.*
- Don't forget to call the base class constructor!!

The class should OVERRIDE the following member functions:

- ***setType*** Member Function:
  - Don't forget this one is protected not public (see VirtualPet)
  - Note that this function is void with no arguments (see VirtualPet)
  - All this function should do is set the type member variable equal to "Dog"
- ***action*** Member Function:
  - The purpose of this function is to print a statement about what the pet is doing based on the pets mood
  - Note this function is void with no arguments and is const (see VirtualPet)
  - Here are rules for what to print:

If the mood member variable is "content" print (replace Fluffy with the real name):
"Stretch and snore."
"Your dog Fluffy is sleeping next to you. "

If the mood member variable is "hungry" print (replace Fluffy with the real name):
"RUFF RUFF!!"
"Fluffy is hungry for some food and treats (table scraps preferred)! "

If the mood member variable is "sick" print (replace Fluffy with the real name):
"Whine, cry, sad dog eyes."
"Fluffy doesn't feel well"
"Time to go to the vet!!"
If the mood member variable is "playful" print(replace Fluffy with the real name):
"Fluffy brought you a ball. Throw it and play fetch!"

If the mood member variable is "naughty" print (replace Fluffy with the real name):
"Chew. Chew. Fluffy chewed up a magazine."
"Time to buy more toys or go to obedience class! "

- ***printInfo* Member Function:**
- This member function should do EXACTLY the same thing printInfo() does in VirtualPet but it should add one line at the bottom (below the mood) that displays the action. So this function should reuse printInfo() from VirtualPet and then call the action function to print the action at the bottom. It might help to look at the sample output in the client program for this one. It's the same as the Cat one.

Step 1f: Fish (derived class, inherits from WaterPet)

Note Fish is a concrete derived class so we must implement all pure virtual functions here, don't forget const when overriding!!

This class is derived from VirtualPet and should use public inheritance.

The class does not have ANY additional member variables it just uses the inherited ones

The class should have the following ADDITIONAL member functions:

- **One Constructor:**
    - The constructor should have four parameters: one of the pet's name, one for the pet's color, one for the base adoption fee, and one for the waterType. Use "fresh" as the default value for waterType.
    - This constructor should call the setType() function which we are overriding below.
    - Don't forget to call the base class constructor!!

The class should OVERRIDE the following member functions:

- ***setType*** Member Function:
    - Don't forget this one is protected not public (see VirtualPet)
    - Note that this function is void with no arguments (see VirtualPet)
    - All this function should do is set the type member variable equal to "Fish"
- ***action*** Member Function:
    - The purpose of this function is to print a statement about what the pet is doing based on the pets mood
    - Note this function is void with no arguments and is const (see VirtualPet)
    - Here are rules for what to print:

If the mood member variable is "content" print (replace Fluffy with the real name):

"Swish swish."
"Your fish Fluffy is casually swimming. "

If the mood member variable is "hungry" print (replace Fluffy with the real name):

"SWISH SWISH SWISH!!"
"Fluffy is swimming to the top of the tank looking for food! "

If the mood member variable is "sick" print (replace Fluffy with the real name):

"Fluffy is hiding behind a rock and doesn't feel well."
"Time to call the vet!"

- ***printInfo* Member Function:**
    - This member function should do EXACTLY the same thing printInfo() does in VirtualPet but it should add one line at the bottom (below the mood) that displays

the action. So this function should reuse printInfo() from VirtualPet and then call the action function to print the action at the bottom. It might help to look at the sample output in the client program for this one. It's the same as the Cat one.

Step 1g: Two additional Derived Classes of your choice the represent Virtual Pets of your choice
Add any two derived classes to the inheritance hierarchy that you would like to add (note these should be PETS (like Cat, Dog etc.) not additional Pet Types. They should be COMPLETE and not easy simple additions. Add the member variables, member functions, EVERYTHING that is needed.

Step 2: Create the following client program
The purpose of this program will be to allow the user to create as many virtual pets as they want and save all their virtual pet's names to a file they can later print.

- Create a vector of VirtualPet pointers to represent a list of virtual pets someone has. Make the array size 10.
- Create a file for output and input named PetNames.txt
- Note you may need more variables add them as you need them
- Display the Following menu (you can format it any way you would like):

ADD PET MENU
1. Add a Cat
2. Add a Dog
3. Add a Fish
4. Add a/an ("insert name of the first new derived class you created in step 1g)
5. Add a /an ("insert name of the second new derived class you created in step 1g)
Q. Done Adding Pets

- Loop through the menu above (the ADD PET MENU) and let the user add as many pets as they want to the vector of Virtual Pet pointers you created above using dynamic memory allocation. When the user creates a pet make sure they enter all the relevant information for that type of pet (do NOT use all default values). **In addition, when the user creates a pet add that pet's type and name to your PetNames.txt file. So if the pet was a Cat named Fluffy a line should be saved to the file that says: Cat Fluffy**

- Next display the following menu
  PET PROCESSING MENU
  1. Print All Pet Info
  2. Print PetNames.txt
  Q. Quit

- Loop through the menu above until the user chooses to quit. So let them run it as many times as they would like
- **<u>If the user chooses 1 (on the PET PROCESSING MENU):</u>** Loop through the vector of VirtualPet pointers and polymorphically call changeMood() followed by polymorphically calling PrintInfo()
    - Your output should look like what I have below (don't worry about formatting, spelling or small differences as long as the inheritance and polymorphism are working). Also remember your output will vary since changeMood() is random.

```
These are your pets:

VIRTUAL PET INFO
---------------------
Adoption Cost: 115
Name:          Tasha
Type:          Cat
Color:         Gray
Mood:          naughty
Action:   Scratch scratch. Tasha is scratching the curtains!!
Time to buy another scratching post or trim nails!

VIRTUAL PET INFO
---------------------
Adoption Cost: 115
Name:            Fido
Type:            Dog
Color:           Black
Mood:            naughty
Action:   Chew chew. Fido chewed up a magazine!!
Time to buy more toys or go to obedience class!

VIRTUAL PET INFO
---------------------
Adoption Cost: 25
Name:       Lucky
Type:         Fish
Color:     Yellow
Mood:         sick
Water Type:  fresh
Action:  Lucky is hiding behind a rock and doesn't feel well.
           time to call the vet!
```

- **If the user chooses 2 (from the Pet Processing Menu Above):** Loop through the PetNames.txt file and print all the pet types and names on the screen in a nice format.
- If the user enters an invalid selection, give them a message and have the menu continue until they choose to quit.