ITP 165: Introduction to C++ Programming

Homework 9: Companion Pet 2.0 Due: 11:59pm, November 6th, 2015

Goal

In this homework you will be upgrading your animal class from homework 8. You will be implementing the same class, but now separated into a header file and an implementation file. Also, you will adding more fun traits and interactions to your animal class. Included with this assignment is the file "FileIO.h" which includes functions to help implement the file input and output needed in this assignment.

Setup

- Create a new project in Visual Studio or Xcode called **homework09** and add three new files named **companion2.cpp, Animal.h,** and **Animal.cpp**.
- Each file must begin with the following (replace the name and email with your actual information):

```
// Name
// ITP 165, Fall 2015
// Homework 9
// USC email
```

- Use the input file "Animal.txt" included with this assignment file as sample input to your program.

Part 1: Animal class – Animal.h and Animal.cpp

- Create a `class` named after your favorite animal. This write-up will use the `Dog` class as an example, but you can use any other animal you'd prefer. This class should NOT include the `<iostream>` library and all flavor text will now be in your main() function.
- The class now has 6 `private` member variables:
  - An `int` to hold the animal's hunger meter.
  - An `int` to hold the animal's happiness meter.
  - An `int` to hold the animal's health meter.
  - An `int` to hold the animal's energy meter.
  - An `int` to hold the animal's age.
  - A `std::string` to hold the name of the pet animal.
- The `class` also has the following `public` member functions:
  - A default constructor that sets the happiness, health, and energy of the animal to 100 and sets the hunger and age to 0, and sets the name to "No Name".
  - A non-default constructor that takes in a parameter for each private member variable, and sets each variable with the parameter values.
  - A setter for each private member variables.
  - A getter for each private member variables.
  - A `Play` function that returns nothing, and accepts nothing as input. It will increase the animal's happiness by 10 and hunger by 5.
  - A `Feed` function that returns nothing, and accepts nothing as input. It will decrease the animal's hunger by 10.

    - A `GiveMedicine` function that returns nothing, and accepts nothing as input. It will decrease the animal's happiness by 20 and increase the animal's health by 20.
    - A `Sleep` function that returns nothing, and accepts nothing as input. It will increase the animal's energy by 20 and increase the animal's age by one.
- Be sure to comment your `public` member functions (but not the getters and setters) with the Name, Purpose, Parameters, and Return as discussed in lecture.

Part 2: Animal interaction – companion2.cpp

- For your third file, you need to be sure to include all libraries needed. For this assignment you will need to include you class header, the `"FileIO.h"` file given to you, and also `<iostream>`.
- FileIO.h has two functions given to you: one for loading animal data from a file into variables, the other for saving animal data from variables into a file. These functions return a `bool` to signify if the load/save was successful. Please read the code and comments in FileIO.h to better understand the code.
- Before main, create a `PrintStats` function that returns nothing, and accepts an object of your animal class passed by reference as input. This function will display to the console the name, health, happiness, hunger, energy, and age of the animal passed in as a parameter.
- Inside of main, you will create five `int` variables for the health, happiness, hunger, energy, and age of your animal and also one `string` for the name. These variables will be used to store data from the input files. Be sure to initialize these variables.
- Next, create an object of your animal class using the NON-default constructor and your initialized variables. This way the program always starts out with a new animal to play with.
- Now you will display a menu of options for your user, this time with 8 options:
    - Feed
        - This case will display flavor text before and after calling the `Feed` function on your animal object.
    - Play
        - This case will display flavor text before and after calling the `Play` function on your animal object.
    - Sleep
        - This case will display flavor text before and after calling the `Sleep` function on your animal object.
    - Give Medicine
        - This case will display flavor text before and after calling the `GiveMedicine` function on your animal object.
    - Rename
        - This case will ask the user for a new name, then will set your animal object's name with this new value.
    - Print Stats
        - This case will call your `PrintStats` function that you created above main with your animal object passed in as a parameter.
    - Save and Quit
        - This case will ask the user for an output file name, then will call the `SaveAnimalData` function from `FileIO.h`, and will tell the user if the save was

successful or not. Be sure to use your animal object's current data as the parameters in the function call.

- Load Animal
  - This case will ask the user for an input file name, then will call the `LoadAnimalData` function from `FileIO.h`. If the load was successful, set your animal object's member variables with this new data, otherwise tell the user there was an error loading the data. Lastly call the `PrintStats` function to display the loaded animal.

- Loop over the menu options until the user has entered the Save and Quit option.

Full Sample Output

Below is sample output for a full run-through of the program. Your output should resemble the following – but doesn't need to match it exactly (user input is in **red**).

```
Welcome to my Doggie Daycare! Here is what we can do:

0. Save and Quit
1. Load Animal
2. Feed
3. Play
4. Sleep
5. Give Medicine
6. Rename
7. Print Stats

What would you like to do? 1
Input file name? Animal.txt
Loading data...
Load successful:
********************
Name: Fido the Dog
Health: 100
Happiness: 100
Hunger: 0
Energy: 100
Age: 0
********************

0. Save and Quit
1. Load Animal
2. Feed
3. Play
4. Sleep
5. Give Medicine
6. Rename
7. Print Stats

What would you like to do? 2
Yum Yum! About to feed Fido the Dog!
Fed your doggie!

0. Save and Quit
1. Load Animal
2. Feed
3. Play
4. Sleep
```

```
5. Give Medicine
6. Rename
7. Print Stats

What would you like to do? 3
Weee so much fun to play with Fido the Dog!
Doggies are so much fun to play with!

0. Save and Quit
1. Load Animal
2. Feed
3. Play
4. Sleep
5. Give Medicine
6. Rename
7. Print Stats

What would you like to do? 4
Time for bed, Fido the Dog.
Your doggie has slept sweetly!

0. Save and Quit
1. Load Animal
2. Feed
3. Play
4. Sleep
5. Give Medicine
6. Rename
7. Print Stats

What would you like to do? 5
Aw, poor Fido the Dog doesn't feel well?
There you go, all better.

0. Save and Quit
1. Load Animal
2. Feed
3. Play
4. Sleep
5. Give Medicine
6. Rename
7. Print Stats

What would you like to do? 6
New name: Scotty Dog
New name set.

0. Save and Quit
1. Load Animal
2. Feed
3. Play
4. Sleep
5. Give Medicine
6. Rename
7. Print Stats

What would you like to do? 7
********************
Name: Scotty Dog
Health: 80
Happiness: 80
```

```
Hunger: 5
Energy: 100
Age: 1
********************

0. Save and Quit
1. Load Animal
2. Feed
3. Play
4. Sleep
5. Give Medicine
6. Rename
7. Print Stats

What would you like to do? 0
Output file name? Output.txt
Preparing to save...
Save successful.
Goodbye!
Press any key to continue . . .
```

A Note on Style

Be sure to comment your code. Also, remember to comment out any extra debug statements.

As we discussed in lecture, it is extremely important that your code is properly indented as it greatly adds to readability. Because of this, if you submit a code file that is not reasonably indented, you will have points deducted.

Likewise, you will lose points if your variable names are not meaningful. Make sure you use variable names that correspond to what you are actually storing in the variables.

Deliverables

1. A compressed **Hw09** folder containing only the **companion2_0.cpp, Animal.h,** and **Animal.cpp** files. It must be submitted through Blackboard.

Grading

Item	Points
Part 1: Animal class – Animal.h and Animal.cpp	20
Part 2: Animal interaction – companion2.cpp	20
Total*	**40**

* Points will be deducted for poor code style, or improper submission.