ITP 165: Introduction to C++ Programming

Homework 8: Companion Pet — Due: 11:59pm, October 30th, 2015

Goal

In this homework you will be getting practice with file input and output and, in the process, will be creating a basic and fun companion pet to play with via classes.

Setup

- Create a new project in Visual Studio or Xcode called **homework08** and a new C++ file named **companion.cpp**.
- Each file must begin with the following (replace the name and email with your actual information):

```
// Name
// ITP 165, Fall 2015
// Homework 8
// USC email
```

- Use the input files "Dog.txt", "StarvingDog.txt", and "SadDog.txt" included with this assignment file as sample input to your program.

Part 1: Animal class

- 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. It has 3 `private` member variables:
  - An `int` to hold the animal's hunger meter.
  - An `int` to hold the animal's happiness meter.
  - A `std::string` to hold the name of the pet animal.
- The `class` also has the following `public` member functions:
  - A setter for the happiness member variable
  - A getter for the happiness member variable
  - A setter for the hunger member variable
  - A getter for the hunger member variable
  - A setter for the name member variable
  - A getter for the name member variable
  - A `PrintStats` function that returns nothing, and accepts nothing as input. It should display the name, health, and happiness of the animal.
  - A `Play` function that returns nothing, and accepts nothing as input. It should display a playful message to the console, then also increase the animal's happiness by 10 and hunger by 5.
  - A `Feed` function that returns nothing, and accepts nothing as input. It should display a tasty message to the console, then also decrease the animal's hunger by 10.
- Be sure to comment your `public` member functions (besides the getters and setters) as discussed in lecture.

- It is advised to now test your `class` in main. Be sure to test every function. I would recommend changing the data of a test object and displaying the values to the console to see if your functions are working properly.

Part 2: File Input and Animal Creation

- Inside of main, you will create an object of your `class`. You will also create two `int` variables for hunger and happiness, and one `string` for the name. These variables will be used to store data from the input files.
- Prompt the user from the console for the input file name and open the file by creating an input file stream.
- The input files are in the form:

  Hunger of animal
  Happiness of animal
  Name of animal

  This pattern is the same for all input files used in this homework. You are not guaranteed that animal names are 1 word, so you will need to use some combination of `std::getline` and `inFile.ignore()` (where `inFile` is the `ifstream` variable connected to the input file).
- If the file successfully opened, read in the input file and use the setter functions to set the hunger, happiness, and name of your animal object.
  - If the file does not open properly – tell the user and quit the program.
- It is important to note that when you open any file in your program, it stays open until either the scope of the stream ends OR you manually tell the file to close. In this homework we will practice closing our files when we are done reading/writing to them. To do this, you call the function `inFile.close()` (where `inFile` is the `ifstream` variable connected to the input file), so be sure to close your file once you are done extracting the input data.
- Below is sample output for two possible run-throughs of Part 2. Your output should resemble the following – but doesn't need to match it exactly (user input is in **red**):

```
What is the input file? Dogg.txt
File not found!
Quitting...
Press any key to continue . . .

OR

What is the input file? Dog.txt
Press any key to continue . . .
```

- I would also recommend checking that you properly set the member variables of your animal object via console output before moving on to Part 3.

Part 3: File output

- Our program will always end by attempting to save the animal's data to a file, so let's setup our program for file output.
- Prompt the user for the output file to use, then create an output file stream to open the file.

- If the file has successfully opened, write to the file your animal's stats in the correct format. It should be in the same format as your input file. Remember to close your file when you are done writing.
    - If the file does not open correctly, tell your user that the save failed and quit your program.
- After a successful save, tell your user where their animal info was saved and quit the program.
- Below is sample output for a run-through of Part 3. Your output should resemble the following – but doesn't need to match it exactly (user input is in **red**):

```
What is the input file? Dog.txt
What is the output file? Dog.txt
Saved your doggie to Dog.txt
Quitting...
Press any key to continue . . .
```

- Also, don't forget to open your output file to check if everything worked correctly.

Part 4: User interaction

- Now that we have set up the start and end to our program, let's make some user interaction! The code to Part 4 will be between the code for Parts 2 and 3.
- First, setup a menu of choices for your user with 5 options and save the user's console input into an `int` variable:
    1. Play with the animal
    2. Feed the animal
    3. Rename the animal
    4. Print stats of animal
    5. Save and Quit
- Secondly, setup a switch statement based on the user's choice (don't forget your default case!):
    1. The play case will call the `Play()` function on your animal object
    2. The feed case will call the `Feed()` function on your animal object
    3. The rename case will ask the user for a new name, then will set the animal object's name with this new value
    4. The print stats case will call the `PrintStats()` function on your animal object
    5. The save and quit option will only display a message to the user that a save is going to occur
- Thirdly, and lastly, loop over the menu display and switch statement until the user has entered the Save and Quit choice. Now you should be able to run your whole program from start to finish: create an animal object from a file, interact with your animal until quit, and then at quit save your animal to play with again later!

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**).

```
What is the input file? Dog.txt
1. Play with your doggie?
2. Feed your doggie?
3. Rename doggie?
4. Print stats of your doggie?
0. Save and quit?
What would you like to do with your doggie? 4
***************
Name: Fido the Dog
Hunger: 50
Happiness: 50
***************
1. Play with your doggie?
2. Feed your doggie?
3. Rename doggie?
4. Print stats of your doggie?
0. Save and quit?
What would you like to do with your doggie? 1
Woof! Woof! You have now played with Fido the Dog!
1. Play with your doggie?
2. Feed your doggie?
3. Rename doggie?
4. Print stats of your doggie?
0. Save and quit?
What would you like to do with your doggie? 4
***************
Name: Fido the Dog
Hunger: 55
Happiness: 60
***************
1. Play with your doggie?
2. Feed your doggie?
3. Rename doggie?
4. Print stats of your doggie?
0. Save and quit?
What would you like to do with your doggie? 2
Yummy! You have just fed Fido the Dog.
1. Play with your doggie?
2. Feed your doggie?
3. Rename doggie?
4. Print stats of your doggie?
0. Save and quit?
```

```
What would you like to do with your doggie? 4
***************
Name: Fido the Dog
Hunger: 45
Happiness: 60
***************
1. Play with your doggie?
2. Feed your doggie?
3. Rename doggie?
4. Print stats of your doggie?
0. Save and quit?
What would you like to do with your doggie? 3
New name: Bowzer Yowzer
1. Play with your doggie?
2. Feed your doggie?
3. Rename doggie?
4. Print stats of your doggie?
0. Save and quit?
What would you like to do with your doggie? 4
***************
Name: Bowzer Yowzer
Hunger: 45
Happiness: 60
***************
1. Play with your doggie?
2. Feed your doggie?
3. Rename doggie?
4. Print stats of your doggie?
0. Save and quit?
What would you like to do with your doggie? 0
Time to save your doggie!
What is the output file? MyDog.txt
Saved your doggie to MyDog.txt
Quitting...
Press any key to continue . . .
```

A Note on Style

Be sure to comment your code. Also, 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 **Hw08** folder containing ***only*** the **companion.cpp** file. It must be submitted through Blackboard.

Grading

Item	Points
Part 1: Animal class	10
Part 2: File Input and Animal Creation	10
Part 3: File Output	5
Part 4: User Interaction	10
Total*	**35**

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