**NOTES:**

Data validation is “lightly” implemented in this exercise. It is assumed that the user shall provide the correct data type for the input. Below is the first few lines of the main function:

A computer screen shot of a program

Description automatically generated

**Problem 01**

Write a class that extends the LeggedMammal class from the previous laboratory exercise. The class will represent a Dog. Consider the breed, size and is registered. Initialize all properties of the parent class in the new constructor. This time, promote the use of accessors and mutators for the new properties. Instantiate a Dog object in the main function and be able to set the values of the properties of the Dog object using the mutators. Display all the properties of the Dog object using the accessors.

**Solution and Testing 01**

Here we will create a Dog child class that will inherit from the LeggedMammal class from previous exercise. However, when we do this, some fields and methods will become irrelevant. For example, a dog normally has 4 legs and a tail. So, it is extraneous to expose the mutator methods associated with these properties. To tackle this, we will do a private redeclaration of those unnecessary mutators from LeggedMammal and provide default values to them. This will make those inherited methods encapsulated.

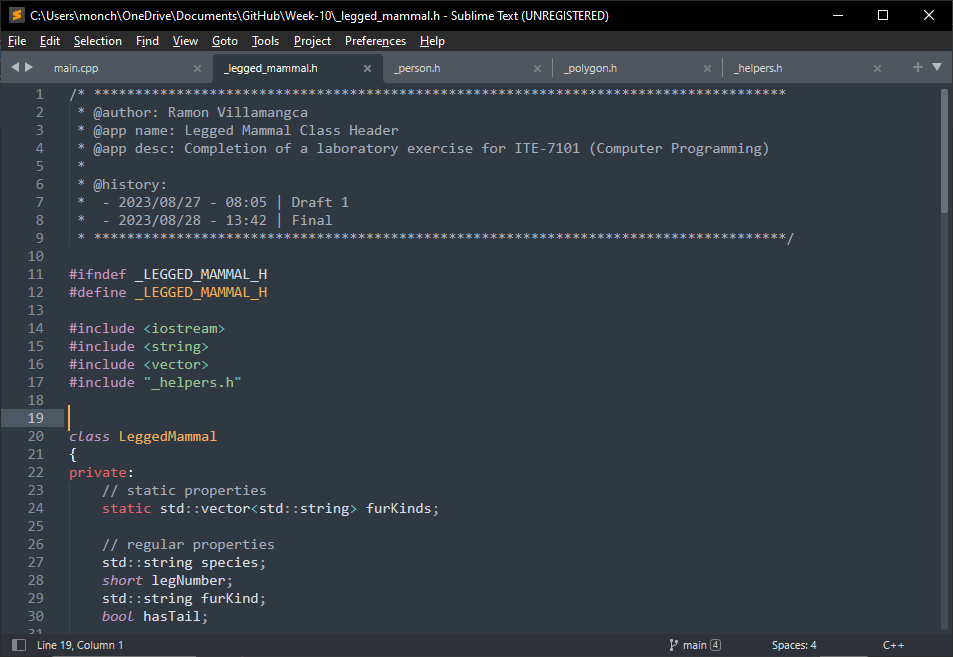
We will then add all the new properties suggested in the problem and also add the “name”, “numBitten” and “isRabid” field. We will then implement all necessary getter/setter methods. We then add another method “isLethal()” that outputs true (“Lethal”) if the dog object is rabid or has bitten at least 10 times and outputs false (“Benign”) otherwise.

We will also overload the output stream operator “<<”, so we can print the class directly by feeding to cout.

In our header, to ensure the header is only included once, we will make use of the simpler directive “#pragma once”, instead of “#ifndef – #define – #endif” combination. The latter being more prone to error.

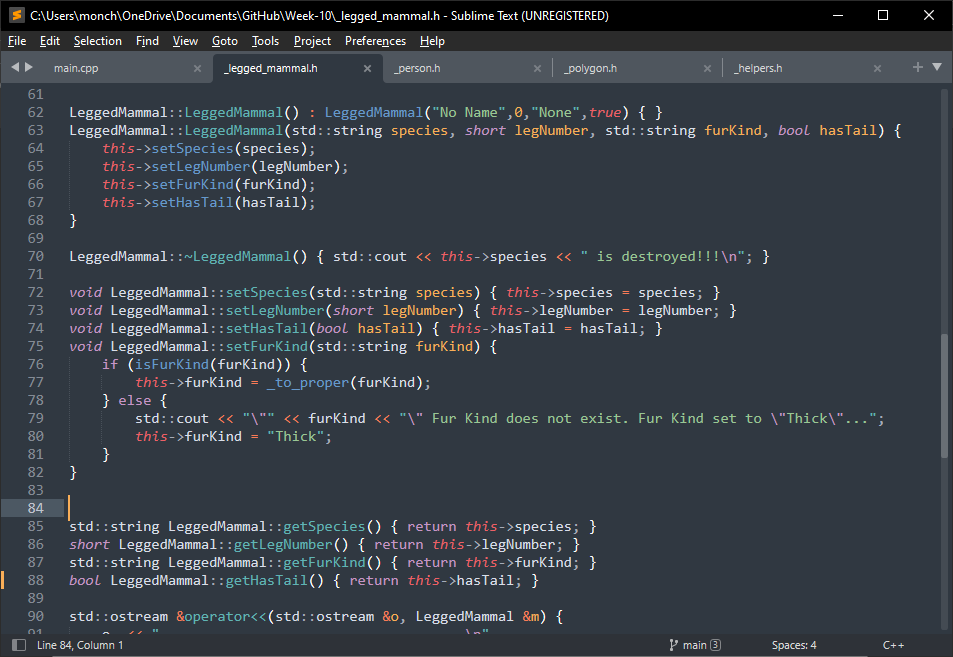
For the testing we created 3 Dog instances to demonstrate the class’s functionalities.

Below is the screenshot of the solution program as well as testing in the terminal.



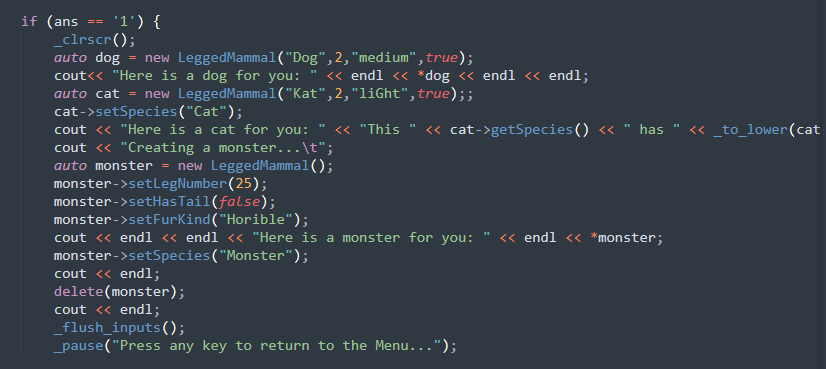
A screen shot of a computer program

Description automatically generated



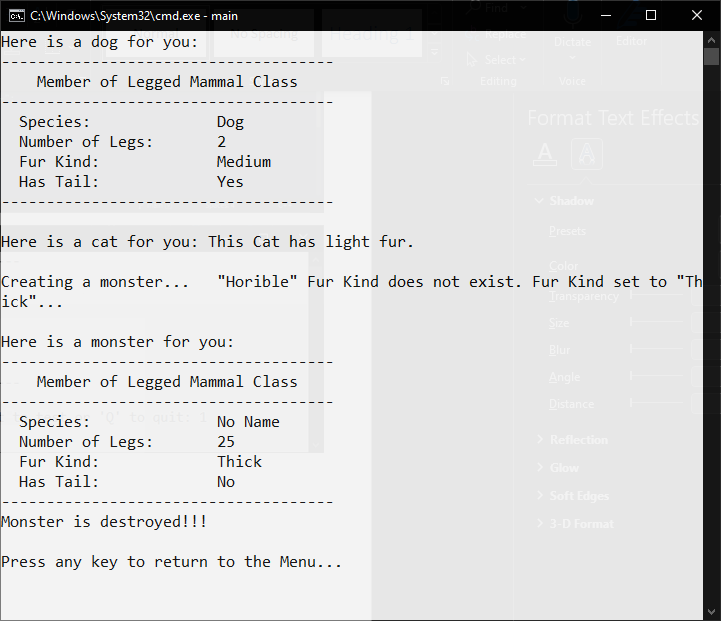
A computer screen shot of a program code

Description automatically generated



A screenshot of a computer

Description automatically generated



**Problem 02**

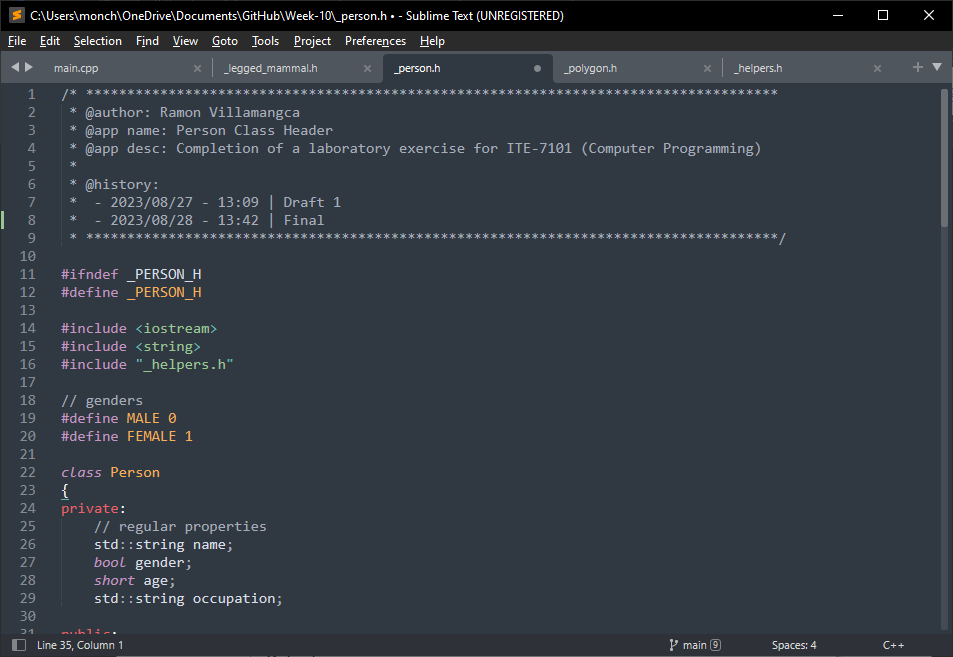
Write a class that extends the Person class from the previous laboratory exercise. The class will represent a Student. Consider the academic program, year in college and enrolled university. Initialize all the properties of the parent class in the new constructor. This time, promote the use of accessors and mutators for the new properties. Instantiate a Student object in the main function and be able to set the values of the properties of the Student object using the mutators. Display all the properties of the Student object using the accessors.

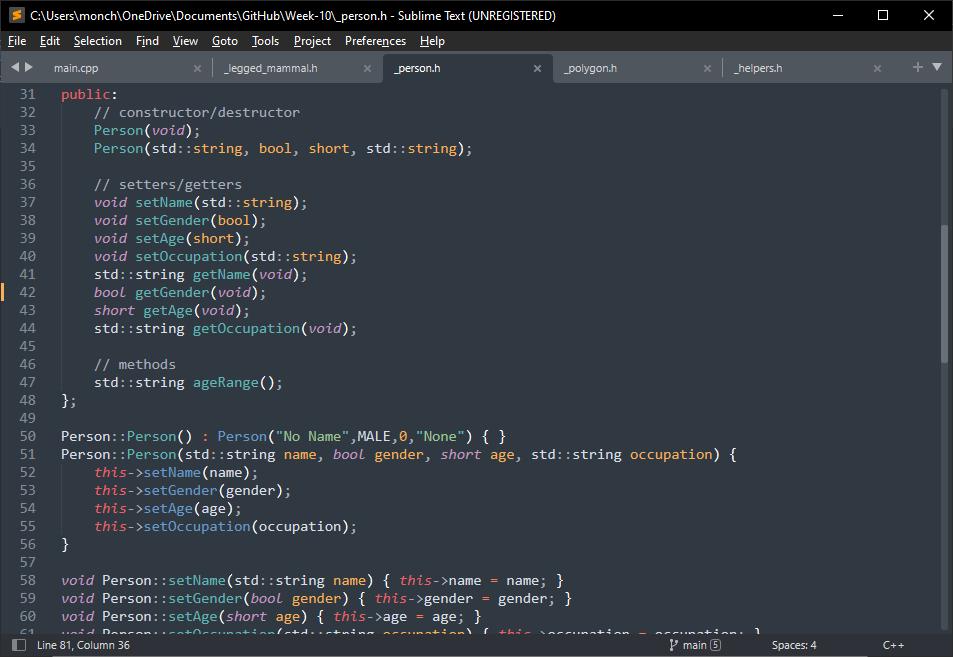
**Solution and Testing 02**

Here we implemented the Person class as per the fields suggested. We implemented both the default and a complete constructor. We provided the usual getters and setters. We added the method “ageRange()” that output a string of either “Baby”, “Child” “Adolescent” or “Adult” depending on the “age” of the person . We then overloaded the output stream operator “<<”, so we can print the class directly by feeding to cout.

For the testing we created 3 Person instances created from user inputs to demonstrate the class’s functionalities.

Below is the screenshot of the solution program as well as testing in the terminal.





A screen shot of a computer program

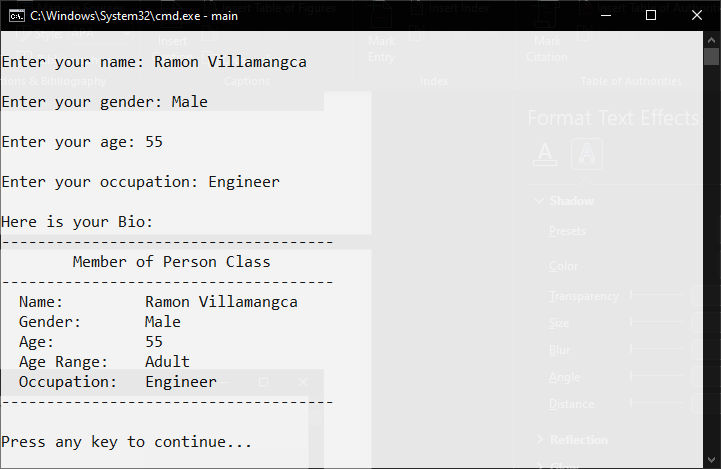
Description automatically generated

A screen shot of a computer

Description automatically generated

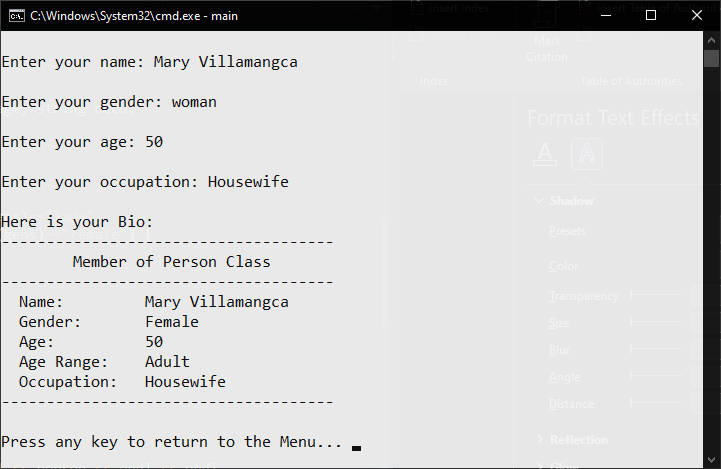
A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated



**Problem 03**

What can you conclude from this activity?

**Answer 03**

The solutions presented may not be the exact solutions the problem creator had in mind. But the solutions work, and do **satisfy the problem specification**. There is always more than one way to solve a programming problem.

There are a few things I’ve learned, though. I have programmed before, but it is only now that I’ve learned that an accessor method ought to return the same data type as the private property it is accessing. It is also the first time that I’ve heard that accessors and mutators have to start with “get” and “set”, respectively. I mean, I have been appending “get”/”set” on my method names quite often, but I didn’t know it is required. Anyway, appending “get”/”set”, is a Java convention. You will be hard-pressed to see a built-in C++ class (e.g. “string”, “vector” classes, etc.) that implements that convention.

------------------------

**NOTE FOR THIS PROBLEM:**

Source code can be found on my GitHub page: <https://github.com/rvillamangca/>.