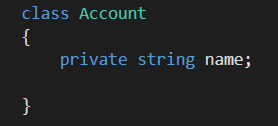
**Lab 3 (20 points)**  
Intro to OOP

Name: Megan Ostrander

Please answer the following questions in Word (or another text editing program). Upload your completed answers directly to Blackboard. Please upload either a Word document or PDF. No other formats will be readable in Blackboard when grading.

1. **Give an example of a get and set method for the following class: (5 points)**



Class Account

{

Private string name;

Console.WriteLine(“Please enter the name:”);

String theName = Console.ReadLine();

name.SetName(theName);

Console.WriteLine($“The name you set is {name.GetName()}”);

}

1. **How would you instantiate an object from Account class? (3 points)**

It would be easiest to instantiate an object at the beginning of a class prior to its use by specifying the type of object it is, the name you want to use for the variable, and telling the class that it is creating a new object of that object. Below is an example.

Account myAccount = new Account();

1. **What is the difference between Public and Private access modifier? (3 points)**

Instance variables declared with the public access modifier can be utilized by all the classes in the program, while private instance variables can only be accessed by specific classes in the program. In other words, anything that calls about a class’s methods can access public variables, but not private variables. Additionally, both are declared within a class prior to the class’s methods, so only the methods and properties of the class can access the private variable for that class.

1. **What is the difference between a class and an object? How many instances of a class can we create? (4 points)**

I am still trying to fully understand how class and objects differentiate, but I like to view it as a real-world object. The book uses a car as an example. The framework or the shell of the car could be seen as the Class of the car, while the individual pieces (i.e. wheels, motor, etc.) that allow the car to function would be the objects. Therefore, Classes are the units that contain objects in programs, and objects are the individual components that run the program. The car as a whole would be the program in this example.

1. UML diagrams help us to think through what a class will have (properties/attributes and methods/operations) in it before we code it out. Consider the following classes. **Add one attribute and one operation to each class.** Think of a real-world application that will use these classes. You do not need to create an image...just fill them in here. **(5 points)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Person** | | - name : string - zipCode : int  - bloodType : string | | + Talk (string sentence, int volume)  + Move (double direction) : double  + Read : string , int  + Speed : int | | |  | | --- | | **Cup** | | - volume : int  - size : string, int | | + Fill (int volume)  + Pour (int volume) | |