# Exercises – OOP 1 – Class Properties and Methods

In this exercise, you will become more familiar with Classes, Properties, and Methods. You will create a pseudo-GIS class library project that will contain a Map class and a Layer class. The Map class will have two properties (Name and LayerCount) and 4 methods (AddLayer, RemoveLayer, GetLayer, and GetLayerByName). The Layer class will have a Name property. See the last page for a class diagram.

1. Download and follow the instructions in <https://github.com/viljoed/gis4x07/raw/master/GIS4x07_ExerciseSetup.docx> to create a Git repository for this exercise. Make sure the name is gis4207-day10.
2. Download and follow the instructions in <https://github.com/viljoed/gis4x07/raw/master/VS_SolutionSetup.docx> to create the Visual Studio solution for this exercise.
3. Replace / Rename from ExerciseSolutionName.sln to MyGIS.sln
4. Open the MyGIS.sln solution
5. Add a Class Library Project called MyGisBLL
6. Add a Console Application Project called MyGisConsole
7. Add a Test Project called MyGisBLLTests
8. In Visual Studio, add 2 classes (Map and Layer) to the MyGisBLL Class Library
9. Commit/push your changes so you can return to this commit if you need to.
10. Modify the scope of the Map and Layer classes to be public
11. Add a read/write Name property to the Map class.
12. Add a read-only Layers property to the Map class.
13. Add a read/write Name property to the Layer class.
14. Commit all changes to the repository.
15. The Map will contain an array of Layer objects and an int to keep track of the number of layers. The following lines need to be added to the top of the Map class:  
      
     private Layer[] \_layers = new Layer[0];  
     private int \_layerCount = 0;
16. Add a read-only property called Layers that returns the array of Layer objects.
17. Create a public AddLayer method that does not return anything to the calling statement and has one parameter called layer that is of the type Layer. The layers passed to the method will be added to the \_layers array.
18. The getter (get {}) of the Layers property will return the \_layers array
19. To add items to the \_layers array, it must first be resized to accommodate the additional layer.

Array.Resize(ref \_layers, \_layers.Length + 1)

1. In the Test Project, add a test for the AddLayer method. Run the test and debug your code until you have this method working
2. Once the test passes, commit the changes to the repository.
3. In the Map class, add a LayerCount property with an associated private **int** called \_**layerCount**.
4. Add the statement to AddLayer that will increment \_layerCount when AddLayer is called.
5. In the Test Project, add another test to make sure the layerCount property is working properly. Once it is working properly, commit the changes to the repository.
6. In the Map class, add a public method called GetLayer that will return a Layer object given an int parameter called layerIndex.  
   HINT: \_layers array contains the map layers
7. In the Test Project, add a test to make sure GetLayer is working properly. Once it is, guess what you should do. (HINT: Commit changes to the repository).
8. In the Map class, add a public method called GetLayerByName that will return a Layer object given a string parameter called layerName. If layerName does not exist in the map, return a null.  
   HINT: You will have to loop through \_layers to find the layer that matches the name passed as an argument.
9. In the Test Project, add a test to make sure GetLayerByName is working properly. Once it is working, commit to the repository.
10. In the Map class, add a public method called RemoveLayer that has one int parameter named layerIndex. If the index is within the range (i.e. not greater than the layerCount – 1), then this method should decrement \_layerCount by 1, remove the specified layer from \_layers, and resize the array to be one element smaller.
11. In MyGisConsole.Main, add statements to:
    1. Create a Map and assign it to a variable called mapCanada and set its name to “Canada”
    2. Create Layer object variables named layerOntario, layerManitoba, and layerAlberta with the Name property set to “Ontario”, “Manitoba”, and “Alberta” respectively.
    3. Add statements to call AddLayer with these three layers.
    4. Add the code to get layerManitoba from the map by index value and Console.WriteLine its name
    5. Add the code to get layerManitoba from the map by name and Console.WriteLine its name
    6. Print out the name of the map and all the layers using Console.WriteLine so that the output looks like:

**Map: Canada  
 Layer 0: Ontario  
 Layer 1: Manitoba  
 Layer 2: Alberta**

1. Add the code to get a layer by index number and pass an index value that does not exist (e.g. 10). What happens?
2. In Map.GetLayer, add a try … catch block to return a null if the index is not valid.
3. In the Test Project, add a test for GetLayer when an invalid index is passed. Once that is working, commit your code.
4. In MyGisConsole.Main, add the if block to check to make sure the layer is not null before attempting to get the Name property from it.
5. Exercise is complete. Commit changes to the repository. Checkout to another folder to make sure all of your code works as expected.

