**IN628 2019 Practical 02.1 – Classes and File Structure**

1. **Exercise 1**

In this practical, you will build two classes representing ***Monsters*** and ***Wizards***. These two classes have ***no data members***. They both have a constructor, and a single method called ***Speak()***. When a ***Monster*** speaks, a message box will pop up containing the following text “I am a Monster…Roar!!!”. When a ***Wizard*** speaks, a message box will pop up containing the following text “I am a Wizard… Expelliarmus!!!”. See the provided demo in the ***Course-Files/Programming 4/*** ***02.1 File Structure*** directory.

* 1. Create a new C++ CLR project or use the empty project created in ***Course-Files/Programming 4/Resources***
  2. Create a ***Form*** with two ***PictureBoxes*** and two ***Buttons***, as shown in the demo. Use the provided images, or find your own if you prefer.
  3. Implement the two classes:
     1. Create a new class via ***Project->Add Class***.
     2. Remember to copy the using statements from ***MyForm.h*** into each new .h file.
     3. Delete anything you want from the .h and .cpp files Visual Studio creates for you except the compiler directives – these are the statements that begin with #.
     4. Place the class definitions in the .h files and place the actual code in the .cpp files – please don’t define ***Speak()*** methods inline.
     5. Remember to preface each method in the .cpp with the name of the class. For example, I have the following line in my ***Monster.cpp*** -**void Monster::Speak()**
     6. Since these classes have no data members and need no initialisation, their constructors will be empty.
  4. Add a ***Monster*** and ***Wizard*** object to the ***MyForm*** class as data members. Remember to use handlers (managed pointers). For example, ***MyForm.h*** file contains this line - **Monster^ monster;**
  5. Instantiate your ***Monster*** and ***Wizard*** object in the ***MyForm\_Load*** event.
  6. Write the handlers for each of the two buttons so that your application behaves like the demo.

**Think about:** An application’s “class architecture” is the collection of classes it contains. Can you think of a better architecture for this program? (Hint: When you have two classes that have nearly identical code, there is almost always a better architecture.)

1. **Optional Extension**
   1. Add a String data member to the ***Monster*** class to hold its name. Pass the value into the constructor, and add it to the output of ***Speak(),*** as shown in the image below.
   2. Add necessary controls to your ***MyForm***, and modify the code of your ***Monster*** class, to allow the user to dynamically change the name of the ***Monster*** at run time.