Lab Assignment 2 (Fall 2021)

To do this lab, you will need to use **C#** in **Visual Studio Professional 2019**. You can access this program in **Mohawk Apps**, while either on campus or at home. Alternatively, while on campus a local version can be accessed from the **Start Menu**, or, you can download and install it as described by the instructions in the **Student Resources** sub-section located in the **Modules** section of the course page.

To Be Graded – General Details:

- This program will be marked for 6% of your final grade
- Please examine the **Marking Scheme (https://mycanvas.mohawkcollege.ca/courses/92934/pages/lab-assignment-2-fall-2020#jump)** to see the marks breakdown
- This program needs to have appropriate internal comments, as well as **XML comments** for *every class* and *every method*
- This program also needs to have an appropriate comment block at the top of all code files that contains:
  - Your name and student number
  - The file date
  - The program's purpose
  - Your **Statement of Authorship (https://mycanvas.mohawkcollege.ca/courses/92934/pages/statement-of-authorship)**
- Bundle your project into one Zip file, and upload it to the appropriate **Lab Assignment (https://mycanvas.mohawkcollege.ca/courses/92934/assignments/838002)** on MyCanvas
- Please read about **documentation (https://mycanvas.mohawkcollege.ca/courses/92934/pages/program-documentation)** style
- Programs that are late will be penalized 10% per day (includes each day of a weekend)
- Programs that do not compile or do not include a **Statement of Authorship (https://mycanvas.mohawkcollege.ca/courses/92934/pages/statement-of-authorship)** will be penalized 10% for each

Part A: The Shape of Things to Come

Project Name: Lab2A Create Class: *Various (one for each shape)*

Write a Console App (.NET Framework) that:

- Makes use of an abstract class called **Shape** which represents one single shape
- Creates an *object hierarchy* to include additional classes for:
  - **Rectangle**

  - **Square**
  - **Box**
  - **Cube**
  - **Ellipse**
  - **Circle**
  - **Cylinder**
  - **Sphere**
  - **Triangle**
  - **Tetrahedron**
- Prompts the user via a menu to create an instance of one of your shapes
- Uses polymorphism to store shapes into a list
- Continues to prompt until the user selects the exit option
- Displays a neat, orderly table of all instantiated shapes, their dimensions, areas and, in the case of three-dimensional shapes, their volumes
- The main class (**Lab2**) should have the following features:
  - A list to store shapes of any kind
  - Error checking for user input
  - The **Main( )** method should be highly modularized
- You are to use this unmodified **code** **(https://mycanvas.mohawkcollege.ca/courses/92934/files/17176709/download)** **(https://mycanvas.mohawkcollege.ca/courses/92934/files/17176709/download?download_frd=1)** for the abstract **Shape** class (right-click and save as Shape.cs)
- Each of your derived classes must implement in some way:
  - A **SetData( )** method that prompts the user to enter each dimension for the current shape
  - A **CalculateArea( )** method that calculates the area of the shape, or for a three-dimensional shape, the surface area
  - A **CalculateVolume( )** method that calculates the volume of three-dimensional shapes only
  - An overidden **ToString( )** method that you can use to print each line of the table
  - It is recommended that you code each of your classes in a different .cs class file (ie, *Square.cs*, *Circle.cs*, etc.)
  - You may find Wikipedia useful for determing the formulae needed for area and volume calculations
- You may download this **sample program** **(https://mycanvas.mohawkcollege.ca/courses/92934/files/17176712/download)** **(https://mycanvas.mohawkcollege.ca/courses/92934/files/17176712/download?download_frd=1)** for a demonstration of program behaviour
- The challenging part of this program is in creating a proper hierarchy:
  - It is recommended that you work on one class at a time
  - You **must** create an actual hierarchy that does not simply involve having all classes descend from the base class