CIS 605 – Fall 2024
Assignment Set 5
Due Date: Friday, October 27 @ 11:59 PM

Develop the projects described below using good visual design and program coding practices that include

- Professional Appearance (Layout, placement, spelling, formatting)
- Meaningful title on title bar of form(s)
- Identifiers (names) for objects, variables, and constants are meaningful and follow a consistent naming convention
- General remarks at the start of every class in your program including Class Name, Class Description, Developer Name, Date Created, Date Last Modified
- Descriptive remarks for every method
- Proper indentation & blank line after each full comment line
- All variables and constants are local whenever possible (scope)
- Modular programming – i.e., breaking down a "large" programming task into multiple, independent modules, with each module performing one part of the required functionality.

Caution: If the business logic is contained in the form classes for programs 11 or 12, you will receive zero credit for that program.

Program 11

Create a WaterTank class that has

- 4 instance properties (auto-implemented – public get and private set)
  - radius
  - depth
  - current water level in liters
  - maximum water capacity in liters

  Note: Assume radius and depth measurements will be in meters

- 1 constructor that
  - instantiates object and set its radius and depth using parameters
  - calls a private method (see below) to set the maximum water capacity.

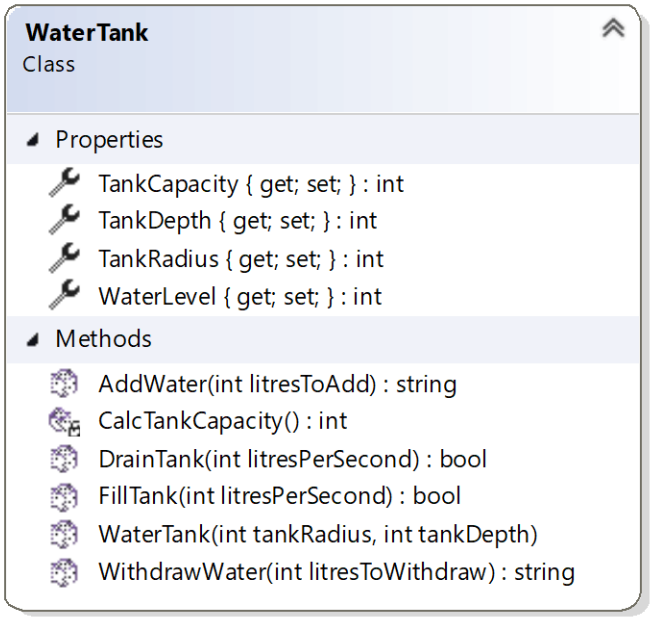
- 5 methods
  - a private method to calculate the maximum water capacity of a tank.
    - Capacity = Pi * radius^2 * depth * 1,000 (1 cubic meter = 1000 liters)
    - **Note:** The capacity calculation will result in a value of type double. Use Math.Floor followed by Convert.ToInt32 to convert this value to type integer.

  - a public method to add water to the tank
    - If the number of liters to be added will not cause the tank to overflow, add the water to the tank. Return an appropriate message a) confirming the liters of water added, and b) the tank's current water level.
    - If the number of liters to be added will cause the tank to overflow, do not add the water to the tank. Instead, return an appropriate message a) indicating that the tank will overflow, b) the tank's current water level, and c) the maximum number of liters that can be added to the tank without overflowing it.

- a public method to withdraw water from the tank
  - If the number of liters to be withdrawn is available in the tank, withdraw the water from the tank. Return an appropriate message a) confirming the liters of water withdrawn, and b) the tank's current water level.
  - If the number of liters to be withdrawn exceeds what the tank currently contains, do not withdraw the water from the tank. Instead, return an appropriate message a) indicating that the tank does not have enough water, and b) the maximum number of liters that can be withdrawn from the tank (i.e., the tank's current water level).

- a public method to fill the tank at a certain rate (i.e., liters per second)
  - If the number of liters that will fill the tank in a second will not cause the tank to overflow, fill the tank. Return a value of true.
  - If the number of liters that will fill the tank in a second will cause the tank to overflow, do not fill the tank. Return a value of false.

- a public method to drain the tank at a certain rate (i.e., liters per second)
  - If the number of liters that will drain from the tank in a second is available in the tank, drain the tank. Return a value of true.
  - If the number of liters that will drain from the tank in a second is not available in the tank, do not drain the tank. Return a value of false.



Create a form class that has appropriate controls and event methods to

- Instantiate a WaterTank object
- Add water to the tank and display the returned message in a label
- Withdraw water from the tank and display the returned message in a label
- Display its current water level
- Display its maximum capacity
- Fill the tank (see note below)
- Empty the tank (see note below)
- Clear and reset the input and output controls
- Exit the application

Note: Use a loop in the event methods for filling and emptying the tank. To keep things simple, assume each loop iteration takes a second. Call the method (i.e., FillTank or DrainTank) within the body of the

loop. The loop should execute as long as the method returns a value of true. After each iteration display the current water level of the tank in a Textbox. Change the following properties of the Textbox: MultiLine to True and ScrollBars to Vertical.

Program 12

Create a Projectile class that has

- 4 instance properties (auto-implemented – public get and private set)
  - initial height (in feet)
  - initial velocity (feet per second)
  - maximum height (in feet)
  - land time (in seconds)

- a constructor that
  - instantiates the object and initializes the properties
  - calls the CalcMaxHeight method to set the maximum height property
  - calls the CalcLandTime method to set the land time property

- a **private** method (FindHeight) that calculates and returns the height (in feet) of the projectile at a certain time (in seconds) after it has been thrown straight up into the air.

  - The formula for calculating the height of a projectile at a certain time (**t** seconds) after it has been thrown up is as follows:

    initial height + (initial velocity * t) – (16 * t^2)

    For example, if the initial height of the projectile is 6 feet and its initial velocity is 100 feet per second, the height of the projectile at 3 seconds after it is thrown up, would be

    6 + (100 *3) – (16 * 3^2) = 162 feet

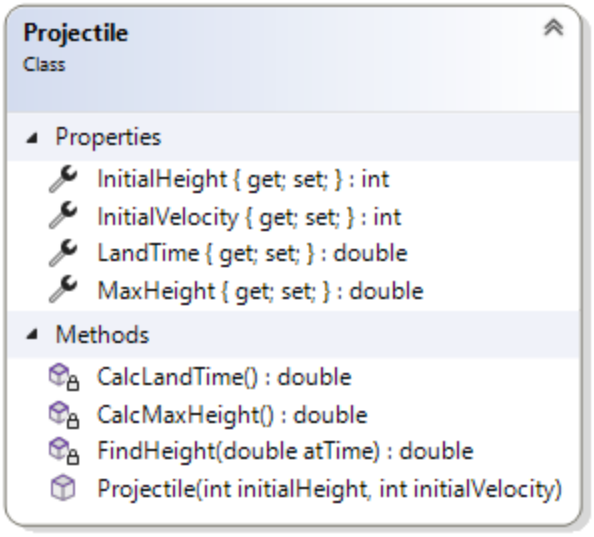
  - The method should receive the time (in seconds) as a parameter

- a **private** method (CalcMaxHeight) that calculates and returns the maximum height of the projectile

  - This method should use the FindHeight method (see above)
  - A projectile will reach its maximum height at **initial velocity/32 seconds** (for example, if the initial velocity of the projectile is 100 feet per second, it will reach its maximum height at 100/32 = 3.125 seconds

- a **private** method (CalcLandTime) that determines and returns the approximate time (in seconds) when the projectile will hit the ground

  - This method should use the FindHeight method (see above)
  - **Hint:** Set up a loop to determine the height of the projectile after every 0.1 second. When the height is no longer a positive number you can assume the projectile has hit the ground.



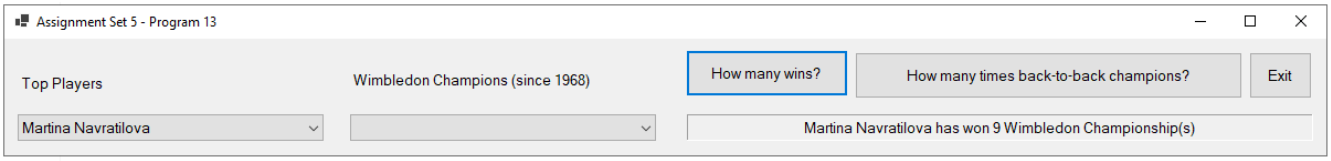
Create a form class that has appropriate controls and event methods to

- Instantiate a Projectile object
- Display the maximum height reached by the projectile
- Display the time it will take for the projectile to reach the ground after it is thrown up into the air
- Clear and reset the input and output controls
- Exit the application

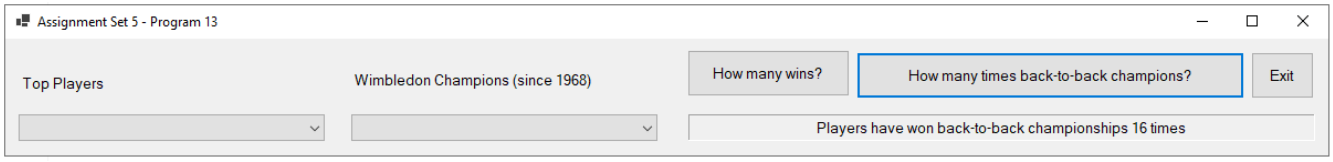
Program 13

Create a form class - you do not need a business logic class for this program. Add two combo or list boxes to the form. Populate one box with a few top tennis players and the other with the names of Wimbledon Singles Champions since 1968. You can populate the items collection of the two boxes by copying and pasting the data from DataForProgram13.xlsx.

Add two buttons to the form. Clicking on one of the buttons (see sample form below) should display in a label the number of times a particular player has won the championship. The user has to select a name from the combo or list box that has the players' names, prior to clicking on the button. If the user clicks on the button without selecting a player's name, display a warning message box.



Clicking on the second button should display the number of times there have been back-to-back champions (see sample form below).



Add an Exit button to close the form.