Lab 02 - Polymorphism

Direction: Submit typed work in the Labs directory of your github repository and/or as an attachment on Google classroom under the accurate Lab02 assessment. All submissions should have the extension h.

Group Lab

Your objective is to write each class below in their own header file. The group members must divide the problems up evenly. Furthermore, no member can write a derived class and the base class it inherits. Likewise, the base class header file must be included in the derived class header file with the include preprocessor directive. Points will be deduced if the files are not linked correctly and if headers of overridden methods do not match. If any member writes both a derived and its inherited base class, the member will be receive any points for the work. Furthermore, if any class is submitted by more than 1 member, no points will be award for the class and there will be a point penalty.

☐ Interface **Shape** whose methods are

- double constant method named `Area()` that takes no parameters.
- double constant method named `Perimeter()` that takes no parameters.

☐ Interface **Counter** whose methods are

- bool method named `Increment()` that takes no parameters.
- bool method named `Decrement()` that takes no parameters.

☐ Interface **NumberSystem** whose methods are

- void method named `Addition()` that takes no parameters.
- void method named `Subtraction()` that takes no parameters.

☐ Derived class **Circle** that publicly inherits **Shape** whose fields and methods are

- private double field named *radius*.
- private double constant static field named *PI* equal to 3.1415926.
- public default constructor that assigns 1 to *radius*.
- public copy constructor.
- public assignment operator.
- public empty destructor.
- public double constant method named `GetRadius()` that takes no parameters. It returns *radius*.
- public double static method named `GetPI()` that takes no parameters. It returns *PI*.
- public double static method named `GetDiameter()` that takes no parameters. It returns the diameter of the circle.
- public void method named `SetRadius()` that takes a double parameter. It assigns the parameter to *radius* only if the parameter is positive; otherwise, it does nothing.
- public overridden `Area()`. It returns the area of the circle.
- public overridden `Perimeter()`. It returns the circumference of the circle.

☐ Derived class **LowerBoundedCounter** that publicly inherits **Counter** whose fields and methods are

- private int field named *counter*.
- private int field named *minimum*.
- public default constructor that assigns 0 to *counter* and 0 to *minimum*.
- public copy constructor.
- public assignment operator.
- public empty destructor.

- public int constant method named `GetCounter()` that takes no parameters. It returns *counter*.
- public int constant method named `GetMinimum()` that takes no parameters. It returns *minimum*.
- public void method named `SetCounter()` that takes an int parameter. It assigns the parameter to *counter* only if the parameter is greater than or equal to *minimum*; otherwise, it does nothing.
- public void method named `SetMinimum()` that takes an int parameter. It assigns the parameter to *minimum* only if the parameter is less than or equal to *counter*; otherwise, it does nothing.
- public overridden `Increment()`. It increments *counter* by 1 and returns true.
- public overridden `Decrement()`. If *counter* is greater than *minimum*, it decrements *counter* by 1 and returns true; otherwise, it just returns false.

☐ Derived class **VectorCalculator** that publicly inherits **NumberSystem** whose fields and methods are

- private double array field named *firstOperand* that has a size of 2.
- private double array field named *secondOperand* that has a size of 2.
- public default constructor that assigns 0 to all the elements of both *firstOperand* and *secondOperand*.
- public copy constructor.
- public assignment operator.
- public empty destructor.
- public double constant method named `GetICoordinate()` that takes a bool parameter. If the parameter equals true, it returns the first element of *firstOperand*; otherwise, it returns the first element of *secondOperand*.
- public double constant method named `GetJCoordinate()` that takes a bool parameter. If the parameter equals true, it returns the second element of *firstOperand*; otherwise, it returns the second element of *secondOperand*.
- public void method named `SetICoordinate()` that takes a bool parameter and a double parameter. If the bool parameter equals true, it assigns the double parameter to the first element of *firstOperand*; otherwise, it assigns the double parameter to the first element of *secondOperand*.
- public void method named `SetJCoordinate()` that takes a bool parameter and a double parameter. If the bool parameter equals true, it assigns the double parameter to the second element of *firstOperand*; otherwise, it assigns the double parameter to the second element of *secondOperand*.
- public overridden `Addition()`. It assigns the sum of first elements of *firstOperand* and *secondOperand* to the first element of *firstOperand*, and assigns the sum of second elements of *firstOperand* and *secondOperand* to the second element of *firstOperand*.
- public overridden `Subtraction()`. It assigns the difference of the first element of *secondOperand* from the first element of *firstOperand* to the first element of *firstOperand*, and assigns the difference of the second element of *secondOperand* from the second element of *firstOperand* to the second element of *firstOperand*.