Program 1B Requirements for CS221 Spr '23

Triangle and Circle Classes Now Inherit from a Shape Class

Due via Canvas Thursday February 2, at 11:40 A.M.

Overview:

- For the second half of program 1, you will modify your Triangle and Circle classes created for Program 1A to inherit from a new base class called Shape. As before, all names will be exactly as specified here, case included. You will submit a .cpp file and a .h file for each of these three classes and each file will be named the exact name of the class; the submission is 6 files total. You will write your own test driver to exercise and test these classes but you will not submit the test driver to me. I will have my own test driver and will plug your classes into my project to test them. I would like for you to zip the following files and submit the zip file. The files you zip must have the following names:

Shape.cpp	Shape.h
Triangle.cpp	Triangle.h
Circle.cpp	Circle.h

- You will have some methods in the base class (Shape) that the derived classes (Triangle and Circle) will override to provide results specific to their particular shape. They are the previous functions already in place: GetArea() and GetPerimeter. Both of these will return 0 when the Shape class is instantiated because there is no information to calculate them with.
- There will be two methods in the base class, SetShapeType() and SetNumSides() that will be assigned the access level "protected", rather than private or public. This allows the derived classes to set those values in their constructors, but the user/calling code can't change them.
- In essence I want to, in my calling code, be able to create objects of each of these 3 types (Shape, Triangle or Circle) and retrieve the available information on them.
    - **Shape default values**: Not much can be told about a non-specific shape, so on instantiation (in the default constructor) the following default values will be set:

```
shapeType = "undefined shape";
numSides = 0;
borderColor = "clear";
fillColor = "clear";
```

```
And the overridden methods will return:
```

```
GetArea() – returns 0.0
GetPerimeter() = returns 0
```

- **Triangle default values**: The triangle is being simplified over Program1A! It will now be an equilateral triangle (all sides the same) with only one side value as a data member. Perimeter and area calculations will need to be updated accordingly.
    - Set in both constructors:

      shapeType = “triangle” always exactly this
      numSides = 3 always exactly this
      borderColor - pick a default value that is set in both constructors. User can set or retrieve at will. Triangle should have its own unique default color on instantiation.
      fillColor – same as above
      side = 2

- **Circle default values**: The circle class must eliminate the member variables that shape now handles (both the colors and their getters and setters)
    - Set in both constructors:

      shapeType = “circle” always exactly this
      numSides = 1 always exactly this
      borderColor - pick a default value that is set in both constructors. User can set or retrieve at will. Circle should have its own unique default color on instantiation.
      fillColor – same as above
      radius = 1

- Example calling code (the test driver, for example) :

```
#include”Circle.h”
#include”Triangle.h”
#include”Shape.h”
…
 Triangle triangle1;
Circle circle1;
Shape shape1;
// call the appropriate methods and get the appropriate results on each
// object.  For example, shape1’s perimeter should be 0, triangle1’s
// perimeter would be 6 and circle1’s perimeter would be 6.
```

Specific requirements for each class

- **The Shape base class requirements:**
    - The shape class will contain the following private data:

string shapeType	Allowable values: “undefined shape”, “triangle” or “circle”
int numSides	Values: 0, 1 or 3
string fillColor	moved from the previous Circle class definition, can have any string value.
string borderColor	Moved from the previous Circle class definition, can have any string value.

    - The shape class will contain 1 default constructor and one destructor. The destructor in this case will simply display a parting message The default constructor will set the following values:

```
shapeType = "undefined shape";
numSides = 0;
borderColor = "clear";
fillColor = "clear";
```

    - The shape class will contain the following public (unless specified protected) functions:
        - Shape class functions that replace ones previously in the Triangle or Circle classes:

Function name	Return type	Argument(s)	Description
GetFillColor()	string	none	Returns the member variable fillColor
SetFillColor()	void	string	Sets the member variable fillColor to the string argument given.
GetBorderColor()	string	none	Returns the member variable borderColor
SetBorderColor()	void	string	Sets the member variable fillColor to the string argument given.

        - New Shape class getters and setters for new member data:

Function Name	Return Type	Argument(s)	Description
GetShapeType()	string		returns member variable shapeType, which either receives the value “undefined shape” if a direct shape constructor was called or it receives a shape-defined value (“triangle” or “circle”) from a derived class.
SetShapeType()	void	string	Protected access. Sets the value of member variable shapeType – presumably as called by a derived class.
GetNumSides()	Int	None	Returns member variable numSides, which will be 0, 1 or 3 if set by the Shape, Circle or Triangle classes respectively.

SetNumSides()	Void	integer	Protected access. Sets member variable numSides, as described above.

- Shape class functions that will be implemented in the shape class and will also be overridden by the derived classes:

Function Name	Return Type	Argument(s)	Description
GetArea()	float	None	Shape class returns 0.0
GetPerimeter()	integer	None	Shape class returns 1

- **The Triangle derived class requirements:**
    - General: The Triangle class will be modified to inherit from the Shape class. None of its previous data or functions have been moved to the inherited Shape class, so there is no removal. The Triangle has been changed to be equilateral, so there is now only one private side value. It is necessary to add the functions that were designated to be overridden over their implementation in Shape.
    - The Triangle class will contain the following private data:

integer side

    - The triangle classes will contain 2 constructors and one destructor. The destructor in this case will simply display a parting message (Ex: "Good bye cruel world! You won't have the Triangle to kick around anymore!") The constructors will be:
        - A default constructor with no arguments. It will provide the default integer value 2 to the member variable side
        - Another constructor will take 1 integer and two strings as arguments for side, fillColor and borderColor, respectively.
        - Both constructor will set the default values previously specified: shape type = "triangle" and numSides = 3
- The Triangle class will contain the following public functions (note the ones that are overriding a function by the exact same name and interface from the base class).

Function name	Return type	Argument(s)	Description
GetSides()	integer	None	Returns the integer member variable side (all three sides are the same now).
SetSides()	void	integer	Sets all the side member variables to the value passed.
GetArea()	float	None	Overrides the base class function. Calculates the area of the triangle from the member data (as before). *This shouldn't change from before unless you had it wrong or misnamed it.
GetPerimeter()	integer	None	Overrides the base class function. Calculates and returns the perimeter length of the triangle from the member data (as before). *This shouldn't change from before unless you had it wrong or misnamed it.

- If the Triangle class needs any extra functions to perform its duties, they must be private.

- **The Circle derived class requirements:**

    - General: The Circle class will be modified to inherit from the Shape class. The members fillColor and fillBorder and the associated getters and setters will be removed since they are now inherited from Shape. It is necessary to add the functions that were designated to be overridden over their implementation in Shape.

- The Circle class will contain the following private data:

integer radius
Constant float PI, assigned the value of PI to 2 decimal places

- The Circle class will contain 2 constructors and one destructor. The destructor in this case will simply display a parting message as before. The constructors will be:

    - A default constructor with no arguments. It will provide default values to all of the member variables.
    - Another constructor will take 1 integer and 2 strings as arguments and will set the member variables to those values.
    - Both constructors will set the default values previously specified: shape type = "circle" and numSides = 1.

- The Circle class will contain the following public functions:

Function name	Return type	Argument(s)	Description
GetRadius()	integer	None	Returns the integer member variable radius
SetRadius()	void	integer	Sets the member variable radius to the integer argument passed
GetArea()	float	None	Overrides the base class function. Calculates the area of the circle from the member data (as before)
GetPerimeter()	integer	None	Overrides the base class function. Calculates and returns the perimeter length (circumference) of the circle from the member data (as before).

- If the Circle class needs any functions to do its work internally, they will be private.