Abstract Classes and Shape calculator

By

Jorge Jurado-Garcia

EE2510 Sec. 021, Spring 2021

Week 6&7 lab

Milwaukee School of Engineering

Submitted to:

Professor: Joshua D. Carl, Ph.D

EECS Department

Date Report Submitted: 04/30/21

**Objective**

The objective of this lab is to create a class hierarchy of shapes objects. Where we can let the user pick any number of shapes he wants and after the user is done input the program calculates the volume and surface area or area and perimeter of the objects. We created rectangles, circle, box, and spheres as concrete objects (hierarchy level 3).

**Description**

In my program I created 3 abstract class and 4 concrete files for my objects(rectangle, circle, box ,and sphere). Each object is then divided into to base class of TwoDShape or ThreeDShape that is abstract meaning cannot be created in the program but can be defined. Afterwards there is an arching abstract class named “Shape”. In my program I create a vector of shape pointers. And then depending on what player decided will create a a dynamic memory of the object chosen. This will allow me to call the shape later on when the user is done inputted information and also allow me to read the information (History Tab) when the user wants to revert back on the inputted info.

**Conclusions**

The lab was successful and was able to implement all the required classes and proper object relationship using polymorphism. The toughest challenge for me was on how to implement to\_string() function for each class and printing out the area/perimeter or volume/surface area for each object. I later found out I can print out the values by calling the TwoDShape to\_string() or ThreeDShape to\_string() in my concrete classes. Another problem I had during this lab was deciding how to keep an object if within a different scope. This because in my menu process I allow the user four process which are (1) Insert, (2) Help, (3) Start, (4) History and (5) Exit. The insert option lets the user insert any options and its information only. Help gives a small text about the program, (3) calculates and output results, (4) prints the shape name, and (5) exit deletes dynamic memory and exits program. A lot of times whenever I am in insert or calling the history the information of the object’s variables would not exist because the object would have been deleted after the if scope was done. Therefor dynamic memory was needed. Recognizing the need of to use dynamic memory is something I must work on. What I did love about this lab was learning a refresh on pointers and how to use polymorphism. Overall, the lab was a success.

Console Result:

//////////////////////////////////////////

1 - Insert.

2 - Help.

3 - Start.

4 - History.

5 - Exit.

Please enter your choice and press enter:1

///////////////////////////////////////////

Please select Shape choice.

1 - Rectangle.

2 - Circle.

3 - Box.

4 - Sphere.

Choice: 1

length: 1

width: 5

//////////////////////////////////////////

1 - Insert.

2 - Help.

3 - Start.

4 - History.

5 - Exit.

Please enter your choice and press enter:1

///////////////////////////////////////////

Please select Shape choice.

1 - Rectangle.

2 - Circle.

3 - Box.

4 - Sphere.

Choice: 2

radius: 5.34

//////////////////////////////////////////

1 - Insert.

2 - Help.

3 - Start.

4 - History.

5 - Exit.

Please enter your choice and press enter:1

///////////////////////////////////////////

Please select Shape choice.

1 - Rectangle.

2 - Circle.

3 - Box.

4 - Sphere.

Choice: 3

length: 5

width: 6

height: 50

//////////////////////////////////////////

1 - Insert.

2 - Help.

3 - Start.

4 - History.

5 - Exit.

Please enter your choice and press enter:1

///////////////////////////////////////////

Please select Shape choice.

1 - Rectangle.

2 - Circle.

3 - Box.

4 - Sphere.

Choice: 4

radius: 3.14

//////////////////////////////////////////

1 - Insert.

2 - Help.

3 - Start.

4 - History.

5 - Exit.

Please enter your choice and press enter:2

///////////////////////////////////////////

This program wil allow the user to create any number of circles,rectnagle,boxes, and spheres.All of the data will be stored and once the user us ready.

Insert allows the user to insert a shape as data.

Start will not let the user input any new data and will start the process.

History will tell the user all the shapes he has chosen.

Exit will close the problem out.

Thank you playing.

//////////////////////////////////////////

1 - Insert.

2 - Help.

3 - Start.

4 - History.

5 - Exit.

Please enter your choice and press enter:4

///////////////////////////////////////////

Rectangle

Circle

Box

Sphere

//////////////////////////////////////////

1 - Insert.

2 - Help.

3 - Start.

4 - History.

5 - Exit.

Please enter your choice and press enter:1

///////////////////////////////////////////

Please select Shape choice.

1 - Rectangle.

2 - Circle.

3 - Box.

4 - Sphere.

Choice: 1

length: -10

width: -5

cannot have negative length

cannot have negative width

//////////////////////////////////////////

1 - Insert.

2 - Help.

3 - Start.

4 - History.

5 - Exit.

Please enter your choice and press enter:1

///////////////////////////////////////////

Please select Shape choice.

1 - Rectangle.

2 - Circle.

3 - Box.

4 - Sphere.

Choice: 2

radius: 7.2

//////////////////////////////////////////

1 - Insert.

2 - Help.

3 - Start.

4 - History.

5 - Exit.

Please enter your choice and press enter:4

///////////////////////////////////////////

Rectangle

Circle

Box

Sphere

Rectangle

Circle

//////////////////////////////////////////

1 - Insert.

2 - Help.

3 - Start.

4 - History.

5 - Exit.

Please enter your choice and press enter:4

///////////////////////////////////////////

Rectangle

Circle

Box

Sphere

Rectangle

Circle

//////////////////////////////////////////

1 - Insert.

2 - Help.

3 - Start.

4 - History.

5 - Exit.

Please enter your choice and press enter:3

///////////////////////////////////////////

Calculating Shapes...

Done.

Outputting information.

Rectangle[1,5] TwoDShape||Area: 5 Perimeter: 12

Circle[5.34] TwoDShape||Area: 89.5818 Perimeter: 33.5512

Box[5,6,50] ThreeDShape||Volume: 1500 SurfaceArea: 1160

Sphere[3.14] ThreeDShape||Volume: 97.2582 SurfaceArea: 123.896

Rectangle[0,0] TwoDShape||Area: 0 Perimeter: 0

Circle[7.2] TwoDShape||Area: 162.855 Perimeter: 45.2376

//////////////////////////////////////////

1 - Insert.

2 - Help.

3 - Start.

4 - History.

5 - Exit.

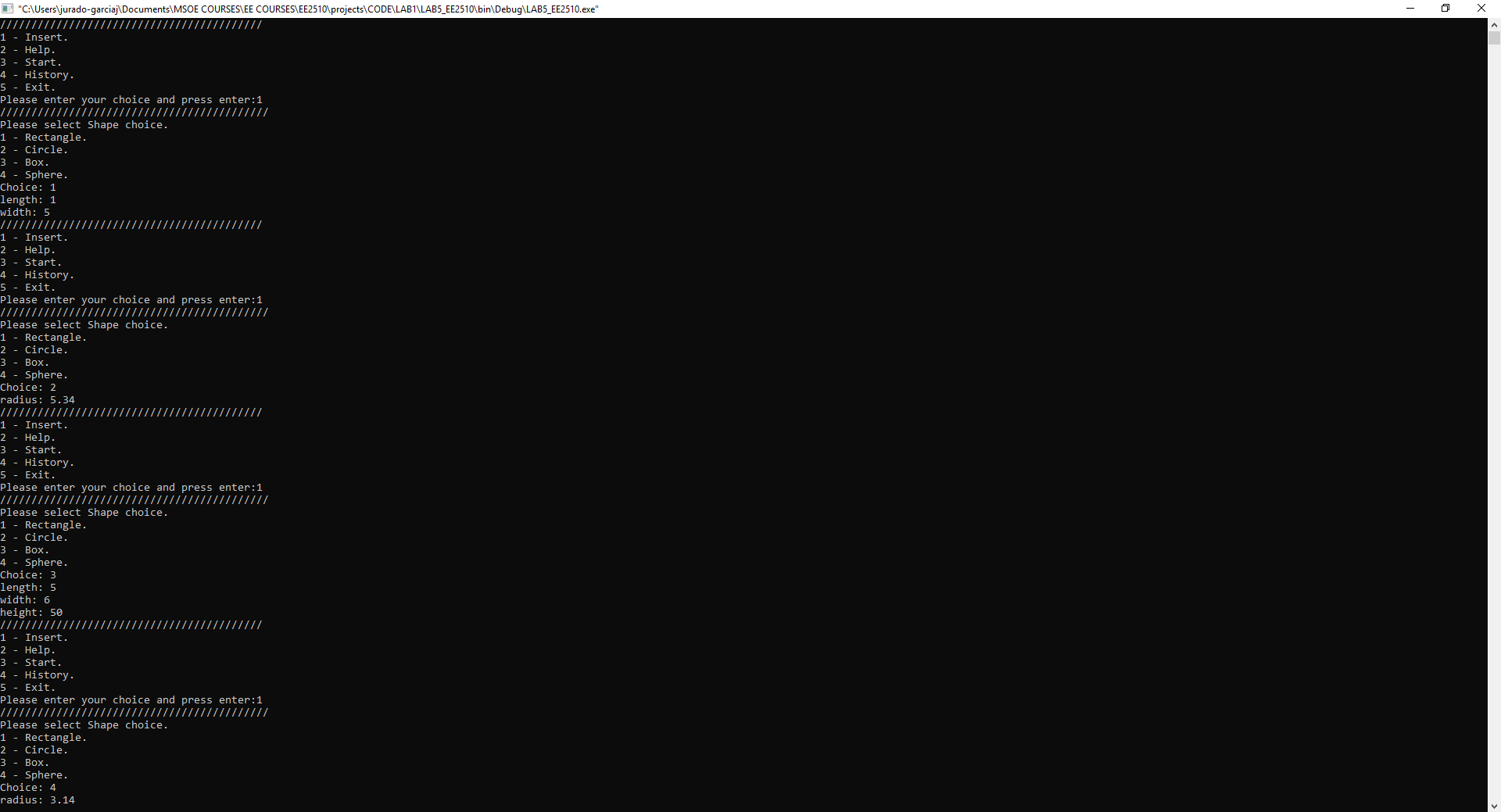
Please enter your choice and press enter:5

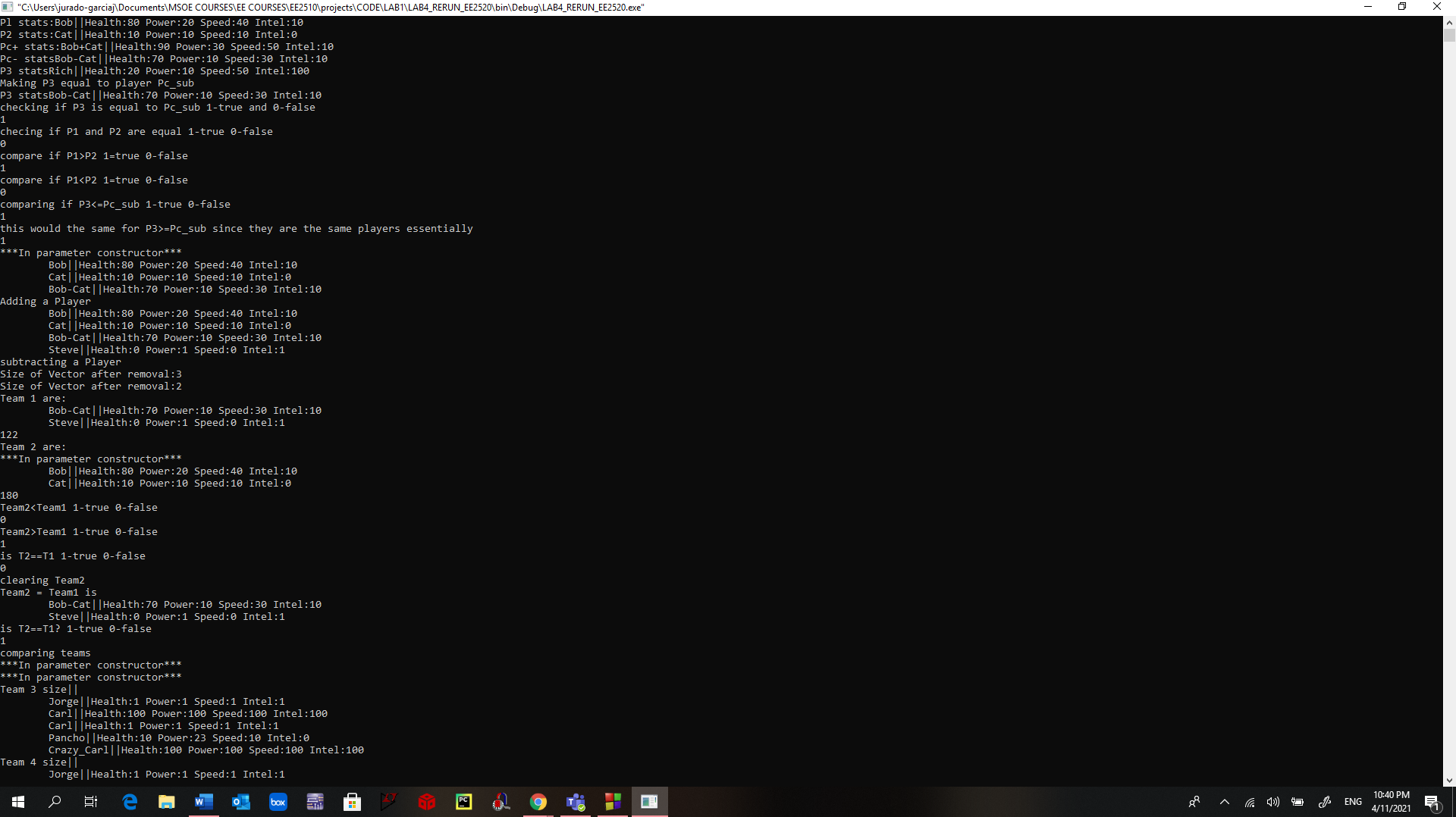
///////////////////////////////////////////

game is done

Process returned 0 (0x0) execution time : 97.007 s

Press any key to continue.





Main File

/\*

\* main.ccp

\*

\* Created on: April 17, 2021

\* Author: Jorge Jurado-Garcia

\* Markups: 4/19 create dynamic memory

\* for shapes objects and for choice 3

\*

\*/

#include <string>

#include "Shape.h"

#include "TwoDShape.h"

#include "ThreeDShape.h"

#include "Rectangle.h"

#include "Circle.h"

#include "Box.h"

#include "Sphere.h"

#include <vector>

**using** **namespace** std**;**

int main**()**

**{**

//creating dynamic memory for rectnagle, circle, box, and square

int choice**;** //variable for first menu

bool gameon **=** **true;**

vector**<**Shape**\*>** shape**;** //vector of base shapes pointers

Shape**\*** shape\_ptr**;**

//user is playing the game

**while(** gameon **!=** **false){**

cout**<<**"//////////////////////////////////////////\n"**;**

cout**<<**"1 - Insert.\n"**;**

cout**<<**"2 - Help.\n"**;**

cout**<<**"3 - Start.\n"**;**

cout**<<**"4 - History.\n"**;**

cout**<<**"5 - Exit.\n"**;**

cout**<<**"Please enter your choice and press enter:"**;**

cin**>>**choice**;**

cout**<<**"///////////////////////////////////////////\n"**;**

**if(**choice **==** 1**){**

//coding for selecting objects

//calles the shape the user is going to input

int shape\_choice**;**

cout**<<**"Please select Shape choice.\n"**;**

cout**<<**"1 - Rectangle.\n"**;**

cout**<<**"2 - Circle.\n"**;**

cout**<<**"3 - Box.\n"**;**

cout**<<**"4 - Sphere.\n"**;**

cout**<<**"Choice: "**;**

cin**>>**shape\_choice**;**

**if(**shape\_choice**==**1**){**

float length**;**

float width**;**

cout**<<**"length: "**;**

cin**>>**length**;**

cout**<<**"width: "**;**

cin**>>**width**;**

Rectangle **\***R **=** **new** Rectangle**(**0**,**0**,**"Rectangle"**);**

R**->**set\_length**(**length**);**

R**->**set\_width**(**width**);**

shape\_ptr **=** R**;** //shape pointer points to the address of R;

shape**.**push\_back**(**shape\_ptr**);**

**}**

**else** **if(**shape\_choice**==**2**){**

float radius**;**

cout**<<**"radius: "**;**

cin**>>**radius**;**

Circle **\***C **=** **new** Circle**(**0**,**"Circle"**);**

C**->**set\_radius**(**radius**);**

shape\_ptr **=** C**;** //shape pointer points to the address of C;

shape**.**push\_back**(**shape\_ptr**);**

**}**

**else** **if(**shape\_choice**==**3**){**

float length**;**

float width**;**

float height**;**

cout**<<**"length: "**;**

cin**>>**length**;**

cout**<<**"width: "**;**

cin**>>**width**;**

cout**<<**"height: "**;**

cin**>>**height**;**

Box **\***B **=** **new** Box**(**0**,**0**,**0**,**"Box"**);**

B**->**set\_length**(**length**);**

B**->**set\_width**(**width**);**

B**->**set\_height**(**height**);**

shape\_ptr **=** B**;** //shape pointer points to the address of B;

shape**.**push\_back**(**shape\_ptr**);**

**}**

**else** **if(**shape\_choice**==**4**){**

float radius**;**

cout**<<**"radius: "**;**

cin**>>**radius**;**

Sphere **\***S **=** **new** Sphere**(**radius**,**"Sphere"**);**

shape\_ptr **=** S**;** //shape pointer points to the address of S;

shape**.**push\_back**(**shape\_ptr**);**

**}**

**}**

**if(**choice **==** 2**){**

cout**<<**"This program wil allow the user to create any number of circles,rectnagle,boxes, and spheres."**;**

cout**<<**"All of the data will be stored and once the user us ready.\n"**;**

cout**<<**"Insert allows the user to insert a shape as data.\n"**;**

cout**<<**"Start will not let the user input any new data and will start the process.\n"**;**

cout**<<**"History will tell the user all the shapes he has chosen.\n"**;**

cout**<<**"Exit will close the problem out.\n"**;**

cout**<<**"Thank you playing."**<<**endl**;**

**}**

**if(**choice **==** 3**){**

cout**<<**"Calculating Shapes..."**<<**endl**;**

**for(**int j**=**0**;** j**<**shape**.**size**();** j**++){**

shape**[**j**]->**calculateAll**();**

**}**

cout**<<**"Done.\n"**;**

cout**<<**"Outputting information.\n"**;**

**for(**int j**=**0**;** j**<**shape**.**size**();**j**++){**

cout**<<**shape**[**j**]->**to\_string**()<<**endl**;**

**}**

**}**

**if(**choice **==** 4**){**

//conduct a for loop of the array

**for(**int j**=**0**;** j**<** shape**.**size**();**j**++){**

cout**<<**shape**[**j**]->**Shape**::**to\_string**()<<**endl**;**

**}**

**}**

**if(**choice **==** 5**){**

gameon **=** **false;**

**}**//end if

**}**//end while

cout**<<**"game is done\n"**;**

**for(**int j**=**0**;** j**<** shape**.**size**();**j**++){**

**delete** shape**[**j**];**

**}**

//vectors are always deleted afterwards

//user stops the game does nothing

/\*

A virtual destructor

- it enables a dynamic dispatch mechanism that makes sure destruction works

fix this by creating shape destructor as a virtual

\*/

**}**

Shape Header File

/\*

Created: Jorge Jurado-Garcia

Date: 4/16/21

modifications:

4/16 creation of virtual class

4/19 adding set/get declaration

adding virtual destructors

\*/

#ifndef SHAPE\_H\_INCLUDED

#define SHAPE\_H\_INCLUDED

#include <stdio.h>

#include <string>

#include <iostream>

**using** **namespace** std**;**

class Shape**{**

private**:**

string name**;**

public**:**

Shape**();** //default constructor

Shape**(**string n**);** //parameter constructor

virtual **~**Shape**();**

set\_name**(**string n**);**

string get\_name**();**

virtual string to\_string**();** //virtual function

//pure virtual function that uses for all subclass

virtual void calculateAll**()=**0**;**

**};**

#endif // SHAPE\_H\_INCLUDED

Shape source file

/\*

\* Shape.ccp

\*

\* Created on: April 16, 2021

\* Author: Jorge Jurado-Garcia

\*

\* Defines the functions for abstract class Shape

\* 4/19/21 Creating set and get functions

\*

\*/

#include "Shape.h"

#include <stdio.h>

#include <string>

#include <iostream>

#include <sstream>

**using** **namespace** std**;**

Shape**::**Shape**(){**

name **=** "no name"**;**

**}**

Shape**::**Shape**(**string n**){**

set\_name**(**n**);**

**}**

Shape**::** **~**Shape**(){**

**}**

Shape**::**set\_name**(**string n**){**

name **=** n**;**

**return** 0**;**

**}**

string Shape**::** get\_name**(){**

**return** name**;**

**}**

string Shape**::** to\_string**(){**

**return** name**;**

**}**

TwoDShape Header File

/\*

Created: TwoDShape

Author: Jorge Jurado-Garcia

Date: 4/16/21

modifications:

4/16 creation of base sub class

4/19 adding virtual constructor

fixing parameter functions for twoDshape

\*/

#ifndef TWODSHAPE\_H\_INCLUDED

#define TWODSHAPE\_H\_INCLUDED

#include <stdio.h>

#include <string>

#include <iostream>

#include "Shape.h"

**using** **namespace** std**;**

class TwoDShape**:** public Shape**{**

protected**:**

float area**;**

float perimeter**;**

public**:**

TwoDShape**();**

TwoDShape**(**string name**);**

virtual **~**TwoDShape**();**

float get\_area**();**

float get\_perimeter**();**

virtual string to\_string**();**

virtual void calculateAll**();**

virtual void calculateArea**()=**0**;**

virtual void calculatePerimeter**()=**0**;**

**};**

#endif // TWODSHAPE\_H\_INCLUDED

TwoDShape Source File

/\*

\* TwoDShape.ccp

\*

\* Created on: April 16, 2021

\* Author: Jorge Jurado-Garcia

\*

\* Defines the functions for abstract class TwoDShape

\*

\*/

#include "TwoDShape.h"

#include <stdio.h>

#include <string>

#include <iostream>

#include <sstream>

**using** **namespace** std**;**

TwoDShape**::**TwoDShape**(){** //will call the Shape class automatically

area **=** 0**;**

perimeter **=** 0**;**

//cannot acces name for base class Shape

**}**

TwoDShape**::**TwoDShape**(**string name**)**

**:**Shape**(**name**){**

//well have to call the shape parameter constructor

//because the default constructor is automatically called

area **=** 0**;**

perimeter **=** 0**;**

**}**

TwoDShape**::** **~**TwoDShape**(){**

**}**

float TwoDShape**::**get\_area**(){**

**return** area**;**

**}**

float TwoDShape**::**get\_perimeter**(){**

**return** perimeter**;**

**}**

string TwoDShape**::**to\_string**(){**

ostringstream out1**;**

ostringstream out2**;**

out1**<<**area**;**

out2**<<**perimeter**;**

string ret**;**

ret **=** "TwoDShape||"**;**

ret **=** ret **+** "Area: " **+** out1**.**str**()** **+** " Perimeter: " **+** out2**.**str**();**

**return** ret**;**

**}**

void TwoDShape**::**calculateAll**(){**

calculateArea**();**

calculatePerimeter**();**

**}**

ThreeDShape Header File

/\*

Created: ThreeDShape

Author Jorge Jurado-Garcia

Date: 4/16/21

modifications:

4/16 creation of base sub class

\*/

#ifndef THREEDSHAPE\_H\_INCLUDED

#define THREEDSHAPE\_H\_INCLUDED

#include <stdio.h>

#include <string>

#include <iostream>

#include "Shape.h"

**using** **namespace** std**;**

class ThreeDShape**:** public Shape**{**

protected**:**

float volume**;**

float surfaceArea**;**

public**:**

ThreeDShape**();**

ThreeDShape**(**string name**);**

virtual **~**ThreeDShape**();**

float get\_volume**();**

float get\_surfaceArea**();**

virtual string to\_string**();**

virtual void calculateAll**();**

virtual void calculate\_Volume**()=**0**;**

virtual void calculate\_SurfaceArea**()=**0**;**

**};**

#endif // THREEDSHAPE\_H\_INCLUDED

ThreeDShape Source File

/\*

\* ThreeDShape.ccp

\*

\* Created on: April 20, 2021

\* Author: Jorge Jurado-Garcia

\*

\* Defines the functions for abstract class TwoDShape

\*

\*/

#include "ThreeDShape.h"

#include <stdio.h>

#include <string>

#include <iostream>

#include <sstream>

**using** **namespace** std**;**

ThreeDShape**::**ThreeDShape**(){** //will call the Shape class automatically

volume **=** 0**;**

surfaceArea **=** 0**;**

//cannot access name for base class Shape

**}**

ThreeDShape**::**ThreeDShape**(**string name**)**

**:**Shape**(**name**){**

//well have to call the shape parameter constructor

//because the default constructor is automatically called

volume **=** 0**;**

surfaceArea **=** 0**;**

**}**

ThreeDShape**::** **~**ThreeDShape**(){**

**}**

float ThreeDShape**::**get\_volume**(){**

**return** volume**;**

**}**

float ThreeDShape**::**get\_surfaceArea**(){**

**return** surfaceArea**;**

**}**

string ThreeDShape**::**to\_string**(){**

ostringstream out1**;**

ostringstream out2**;**

out1**<<**volume**;**

out2**<<**surfaceArea**;**

string ret**;**

ret **=** "ThreeDShape||"**;**

ret **=** ret **+** "Volume: " **+** out1**.**str**()** **+** " SurfaceArea: " **+** out2**.**str**();**

**return** ret**;**

**}**

void ThreeDShape**::**calculateAll**(){**

calculate\_Volume**();**

calculate\_SurfaceArea**();**

**}**

Sphere Header file

/\*

Created: Sphere.h

Author: Jorge Jurado-Garcia

Date: 4/20/21

modifications:

\*/

#ifndef SPHERE\_H\_INCLUDED

#define SPHERE\_H\_INCLUDED

#include "ThreeDShape.h"

#include <iostream>

#include <sstream>

**using** **namespace** std**;**

class Sphere**:** public ThreeDShape **{**

private**:**

float radius**;**

public**:**

Sphere**();**

Sphere**(**float r**,** string name**);**

virtual **~**Sphere**();**

void calculate\_Volume**();**

void calculate\_SurfaceArea**();**

virtual string to\_string**();**

**};**

#endif // SPHERE\_H\_INCLUDED

Sphere Source File

/\*

Created: Sphere.ccp

Author: Jorge Jurado-Garcia

Date: 4/20/21

modifications:

\*/

#include "Sphere.h"

#include <iostream>

#include <sstream>

#include <string>

**using** **namespace** std**;**

Sphere**::**Sphere**()**

**:**ThreeDShape**(**"no name"**)**

//only need to initalize in constructors

**{**

radius **=** 0**;**

**}**

Sphere**::** Sphere**(**float r**,** string name**)**

//only need to initalize in constructors

**:**ThreeDShape**(**name**)**

**{**

**if(**r**>=**0**){**

radius **=** r**;**

**}**

**else{**

radius **=** 0**;**

**}**

**}**

Sphere**::** **~**Sphere**(){**

**}**

void Sphere**::** calculate\_Volume**(){**

//area in this case is inherit from TwoDshape

volume **=** 3.1415**\***radius**\***radius**\***radius**\*(**4**/**3**);**

**}**

void Sphere**::** calculate\_SurfaceArea**(){**

//perimeter in this case is inherit from TwoDShape

surfaceArea **=** 4**\***radius**\***radius**\***3.1415**;**

**}**

string Sphere**::** to\_string**(){**

string ret**;**

ostringstream r**;**

r**<<**radius**;**

ret **=** Shape**::**to\_string**()** **+**"["**+** r**.**str**()** **+** "] " **+** ThreeDShape**::**to\_string**();**

**return** ret**;**

**}**

Box header file

/\*

Created: Box.h

Author: Jorge Jurado-Garcia

Date: 4/20/21

modifications:

\*/

#ifndef BOX\_H\_INCLUDED

#define BOX\_H\_INCLUDED

#include <stdio.h>

#include "ThreeDShape.h"

#include <string>

#include <iostream>

**using** **namespace** std**;**

class Box**:** public ThreeDShape **{**

private**:**

float length**;**

float width**;**

float height**;**

public**:**

Box**();**

Box**(**float len**,** float w**,** float h**,** string name**);**

virtual **~**Box**();**

set\_length**(**float len**);**

set\_width**(**float w**);**

set\_height**(**float h**);**

void calculate\_Volume**();**

void calculate\_SurfaceArea**();**

virtual string to\_string**();**

**};**

#endif // BOX\_H\_INCLUDED

Box Source File

/\*

Created: Box.ccp

Author: Jorge Jurado-Garcia

Date: 4/20/21

modifications:

\*/

#include "Box.h"

#include <iostream>

#include <sstream>

**using** **namespace** std**;**

Box**::**Box**()**

**:**ThreeDShape**(**"no name"**)**

//only need to initalize in constructors

**{**

length **=** 0**;**

width **=** 0**;**

height **=** 0**;**

**}**

Box**::** Box**(**float len**,** float w**,** float h**,** string name**)**

//only need to initalize in constructors

**:**ThreeDShape**(**name**)**

**{**

length **=** set\_length**(**len**);**

width **=** set\_width**(**w**);**

height **=** set\_height**(**h**);**

**}**

Box**::** **~**Box**(){**

**}**

Box**::** set\_length**(**float len**){**

**if(**len **>=** 0**){**

length **=** len**;**

**}**

**else{**

length **=** 0**;**

cout**<<**"cannot have negative length"**<<**endl**;**

**}**

**return** 0**;**

**}**

Box**::** set\_width**(**float w**){**

**if(**w **>=**0**){**

width **=** w**;**

**}**

**else{**

width **=** 0**;**

cout**<<**"cannot have negative width"**<<**endl**;**

**}**

**return** 0**;**

**}**

Box**::** set\_height**(**float h**){**

**if(**h **>=**0**){**

height **=** h**;**

**}**

**else{**

height **=** 0**;**

cout**<<**"cannot have negative width"**<<**endl**;**

**}**

**return** 0**;**

**}**

void Box**::** calculate\_Volume**(){**

//area in this case is inherit from TwoDshape

volume **=** length **\*** width **\*** height**;**

**}**

void Box**::** calculate\_SurfaceArea**(){**

//perimeter in this case is inherit from TwoDShape

surfaceArea **=** 2**\*(** width**\***length **+** height**\***length **+** height**\***width**);**

**}**

string Box**::** to\_string**(){**

string ret**;**

ostringstream l**;**

ostringstream w**;**

ostringstream h**;**

l**<<**length**;**

w**<<**width**;**

h**<<**height**;**

ret **=** Shape**::**to\_string**()** **+**"["**+** l**.**str**()** **+** "," **+** w**.**str**()** **+** "," **+** h**.**str**()** **+** "] " **+** ThreeDShape**::**to\_string**();**

**return** ret**;**

**}**

Rectangle Header file

/\*

Created: Jorge Jurado-Garcia

Date: 4/16/21

modifications:

4/16 creation of virtual class

4/19 add set/get adding ~Rectangle

\*/

#ifndef RECTANGLE\_H\_INCLUDED

#define RECTANGLE\_H\_INCLUDED

#include <stdio.h>

#include "TwoDShape.h"

#include <string>

#include <iostream>

**using** **namespace** std**;**

class Rectangle**:** public TwoDShape **{**

private**:**

float length**;**

float width**;**

public**:**

Rectangle**();**

Rectangle**(**float len**,** float w**,** string name**);**

virtual **~**Rectangle**();**

set\_length**(**float len**);**

set\_width**(**float w**);**

void calculateArea**();**

void calculatePerimeter**();**

virtual string to\_string**();**

**};**

#endif // RECTANGLE\_H\_INCLUDED

Rectangle Source File

/\*

\* Rectangle.ccp

\*

\* Created on: April 16, 2021

\* Author: Jorge Jurado-Garcia

\*

\* Defines the functions for abstract class TwoDShape

\* 4/19 adding set definitions and changing float area to area

\* 4/20 changing code to set\_lengt and set\_width

\*/

#include "Rectangle.h"

#include <iostream>

#include <sstream>

**using** **namespace** std**;**

Rectangle**::**Rectangle**()**

**:**TwoDShape**(**"no name"**)**

//only need to initalize in constructors

**{**

length **=** 0.0**;**

width **=** 0.0**;**

**}**

Rectangle**::** Rectangle**(**float len**,** float w**,** string name**)**

//only need to initalize in constructors

**:**TwoDShape**(**name**)**

**{**

length **=** set\_length**(**len**);**

width **=** set\_width**(**w**);**

**}**

Rectangle**::** **~**Rectangle**(){**

**}**

Rectangle**::** set\_length**(**float len**){**

**if(**len **>=** 0**){**

length **=** len**;**

**}**

**else{**

length **=** 0**;**

cout**<<**"cannot have negative length"**<<**endl**;**

**}**

**return** 0**;**

**}**

Rectangle**::** set\_width**(**float w**){**

**if(**w **>=**0**){**

width **=** w**;**

**}**

**else{**

width **=** 0**;**

cout**<<**"cannot have negative width"**<<**endl**;**

**}**

**return** 0**;**

**}**

void Rectangle**::** calculateArea**(){**

//area in this case is inherit from TwoDshape

area **=** length **\*** width**;**

**}**

void Rectangle**::** calculatePerimeter**(){**

//perimeter in this case is inherit from TwoDShape

perimeter **=** **(**2**\***length**)+(**2**\***width**);**

**}**

string Rectangle**::** to\_string**(){**

string ret**;**

ostringstream l**;**

ostringstream w**;**

l**<<**length**;**

w**<<**width**;**

ret **=** Shape**::**to\_string**()** **+**"["**+** l**.**str**()** **+** "," **+** w**.**str**()** **+** "] " **+** TwoDShape**::**to\_string**();**

**return** ret**;**

**}**

Circle Header File

/\*

\* Circle.h

\*

\* Created on: April 19, 2021

\* Author: Jorge Jurado-Garcia

\*

\* Defines the functions for abstract class TwoDShape

\*

\*/

#ifndef CIRCLE\_H\_INCLUDED

#define CIRCLE\_H\_INCLUDED

#include "TwoDShape.h"

#include <iostream>

#include <sstream>

**using** **namespace** std**;**

class Circle**:** public TwoDShape **{**

private**:**

float radius**;**

public**:**

Circle**();**

Circle**(**float r**,** string name**);**

virtual **~**Circle**();**

set\_radius**(**float r**);**

void calculateArea**();**

void calculatePerimeter**();**

virtual string to\_string**();**

**};**

#endif // CIRCLE\_H\_INCLUDED

Circle Source File

/\*

\* Circle.h

\*

\* Created on: April 19, 2021

\* Author: Jorge Jurado-Garcia

\*

\* Defines the functions for abstract class TwoDShape

\*

\*/

#include "Circle.h"

#include <iostream>

#include <sstream>

**using** **namespace** std**;**

Circle**::**Circle**()**

**:**TwoDShape**(**"no name"**)**

//only need to initalize in constructors

**{**

radius **=** 0**;**

**}**

Circle**::** Circle**(**float r**,** string name**)**

//only need to initalize in constructors

**:**TwoDShape**(**name**)**

**{**

radius **=** set\_radius**(**r**);**

**}**

Circle**::** **~**Circle**(){**

**}**

Circle**::** set\_radius**(**float r**){**

**if(**r **>=** 0**){**

radius **=** r**;**

**}**

**else{**

radius **=** 0**;**

cout**<<**"cannot have negative length"**<<**endl**;**

**}**

**return** 0**;**

**}**

void Circle**::** calculateArea**(){**

//area in this case is inherit from TwoDshape

area **=** 3.1415**\***radius**\***radius**;**

**}**

void Circle**::** calculatePerimeter**(){**

//perimeter in this case is inherit from TwoDShape

perimeter **=** **(**2**\***radius**)\***3.1415**;**

**}**

string Circle**::** to\_string**(){**

string ret**;**

ostringstream r**;**

r**<<**radius**;**

ret **=** Shape**::**to\_string**()** **+**"["**+** r**.**str**()** **+** "] " **+** TwoDShape**::**to\_string**();**

**return** ret**;**

**}**