University of Calgary
Department of Electrical and Computer Engineering

ENSF 619

Lab 5 – Oct 16, 2020

M. Moussavi, Ph.D. P.Eng

This is a group lab assignment:

You are allowed to work with a partner, which means only groups of two are allowed (groups of three or more **are not allow**).

Objective:

The purpose of this lab is to practice some of the basic object-oriented design concepts such as: aggregation, composition, polymorphism, inheritance, and multiple inheritance in C++. Course material related to exercise B will be discussed on Monday Sept 28th.

Marking Scheme: (30 marks total)

- Exercise A: 20 marks
- Exercise B: 10 marks

Note: Material related to Exercise B will be covered on Monday Sept 30.

Method of Submission and Due Dates:

- You should submit your lab report (in PDF format), and associated files into the **Dropbox** on the D2L.
- Due date Friday October 23rd before 5:00 PM

.

Exercise A - Inheritance in C++ (20 marks)

The concepts of aggregation, composition, and inheritance in C++ and Java in terms of concepts are similar, but they are completely different in terms of implementation and syntax. Also, another major difference between the two languages is that C++ unlike Java supports multiple-inheritance.

What to Do: Single Inheritance in C++:

In this exercise, first you will write the definitions of following classes: Point, Shape, Rectangle, Square, GraphicsWorld, as explained below.

Class Point:

This class is supposed to represent a point in a Cartesian plane. It should have three data members: **x**, and **y** coordinates and an **id** number that its value will be assigned automatically. The first object's **id** number should be 1001, second one should be 1002, and so on. Class Point should also have at least the following member functions:

- `display` - that displays x, and y coordinates of the point in the following format:
  ```
   X-coordinate: ######.##
   Y-coordinate: ######.##
  ```
- A constructor that initializes its data members.
  Note: You are not supposed to define a default constructor in this class. Automatic calls to the default constructor will hide some of the important aspects of this assignment (marks will be deducted if you define a default constructor for this class).

- Access functions, getters and setters, as needed.

- Function `counter()` that returns the number of objects of class Point at any time.

- Two distance functions that return the distance between two points. One of the two must be a static function.

You should create two files for this class: called `point.h and point.cpp`

Class Shape:

This class is the base class or the ancestor of several classes, including class Rectangle, Square, Circle, etc. It should support basic operations and structures that are common among the children of this class. This class should have an object of class `Point` called `origin`, and a `char` pointer called `shapeName` that points to a dynamically allocated memory space, allocated by the class constructor. This class should also have several functions and a constructor as follows:

- A constructor that initializes its data members.
- **No default constructor**
- A destructor that de-allocates the memory space allocated dynamically for `shapeName`
- `getOrigin` – that returns a reference to point origin. The reference should not allow the x and y values of point to be modified through this reference.
- `getName` – that returns the name of the shape.
- `display` – that prints on the screen the shape's name, x and y coordinates of point origin, in the following format:
  ```
      Shape Name:
      X-coordinate:
      Y-coordinate:
  ```
- two `distance` functions
  ```
  double distance (Shape& other);
  static double distance (Shape& the_shape, Shape& other);
  ```
- `move`: that changes the position of the shape, the current x and y coordinates, to `x+dx`, and `y+dx`. The function's interface (prototype) should be;
  ```
    void move (double dx, double dy);
  ```

You should create two files for this class: called `shape.h and shape.cpp`

Class Square:

This class is supposed to be derived from class Shape, and should have one data member, called `side_a`, and in addition to its constructor should have a constructor and several member functions as follows:

- One constructor that initializes its data members with its arguments supplied by the user.
- **No default constructor**. Marks will be deducted if a default constructor is defined for this class
- `area` – that returns the area of a square
- `perimeter`: that returns the perimeter of a square
- `get` and `set` - as needed.
- `display` – that displays the name, x and y coordinates of the origin, side_a, area, and perimeter of a square object in the following format:

```
    Square Name:
    X-coordinate:
    Y-coordinate:
    Side a:
    Area:
    Perimeter;
```

- More Functions, if needed.

You should create two files for this class: called `square.h and square.cpp`

Class Rectangle:

Class Rectangle that is derived from class Square needs to have one more side called `side_b`. This class, in addition to its constructor (no default constructor), should support the following functions:

- `area` – that calculates and returns the area of a rectangle
- `perimeter` – that calculates and returns the perimeter of a rectangle
- `get` and `set` – that retrieves or changes the values of its private data members.
- `display` – that displays the name, and x and y coordinate origin of the shape, side_a, side_b, area, and perimeter of a rectangle object, in the following format:

```
    Rectangle Name:
    X-coordinate:
    Y-coordinate:
    Side a:
    Side b:
    Area:
    Perimeter;
```

- More Functions, if needed.

You should create two files for this class: called `rectangle.h and rectangle.cpp`

Class GraphicsWorld:
This class should have only a function called *run*, which declares instances of the above-mentioned classes as its local variable. This function should first display a short message about the author(s) of the program and then start testing all of the functions of the classes in this system. A sample code segment

of function *run* is given below. You are recommended to use conditional compilation directives to test your code (one class at a time).

```
void GraphicsWorld::run(){
#if 0                   // Change 0 to 1 to test Point
      Point m (6, 8);
      Point n (6,8);
      n.setx(9);
      cout << "\nExpected to dispaly the distance between m and n is: 3";
      cout << "\nThe distance between m and n is: " <<      m.distance(n);
      cout << "\nExpected second version of the distance function also print: 3";
      cout << "\nThe distance between m and n is again: "
           <<  Point::distance(m, n);
#endif                 // end of block to test Point

#if 0                   // Change 0 to 1 to test Square
    cout << "\n\nTesting Functions in class Square:" <<endl;
    Square s(5, 7, 12, "SQUARE - S");
    s.display();
#endif                 // end of block to test Square

#if 0                   // Change 0 to 1 to test Rectangle
      cout << "\nTesting Functions in class Rectangle:";
      Rectangle a(5, 7, 12, 15, "RECTANGLE A");
      a.display();
      Rectangle b(16 , 7, 8, 9, "RECTANGLE B");
      b.display();
      double d = a.distance(b);
      cout <<"\nDistance between square a, and b is: " << d << endl;
      Rectangle rec1 = a;
      rec1.display();
      cout << "\nTesting assignment operator in class Rectangle:" <<endl;
      Rectangle rec2 (3, 4, 11, 7, "RECTANGLE rec2");
      rec2.display();
      rec2 = a;
      a.set_side_b(200);
      a.set_side_a(100);
      cout << "\nExpected to display the following values for objec rec2: " << endl;
      cout << "Rectangle Name: RECTANGLE A\n" << "X-coordinate: 5\n"  << "Y-coordinate: 7\n"
               << "Side a: 12\n" << "Side b: 15\n" << "Area: 180\n" << "Perimeter: 54\n" ;
      cout << "\nIf it doesn't there is a problem with your assignment operator.\n" << endl;
      rec2.display();

      cout << "\nTesting copy constructor in class Rectangle:" <<endl;
      Rectangle rec3 (a);
      rec3.display();
      a.set_side_b(300);
      a.set_side_a(400);
      cout << "\nExpected to display the following values for objec rec2: " << endl;
      cout << "Rectangle Name: RECTANGLE A\n" << "X-coordinate: 5\n"  << "Y-coordinate: 7\n"
      << "Side a: 100\n" << "Side b: 200\n" << "Area: 20000\n" << "Perimeter: 600\n" ;
      cout << "\nIf it doesn't there is a problem with your assignment operator.\n" << endl;
      rec3.display();
#endif                  // end of block to test Rectangle

#if 0                   // Change 0 to 1 to test using array of pointer and polymorphism
      cout << "\nTesting array of pointers and polymorphism:" <<endl;
      Shape* sh[4];
      sh[0] = &s;
      sh[1] = &b;
      sh [2] = &rec1;
      sh [3] = &rec3;
      sh [0]->display();
      sh [1]->display();
      sh [2]->display();
      sh [3]->display();

#endif                 // end of block to test array of pointer and polymorphism
```

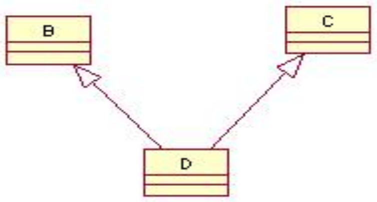
You should create two files for this class: called `graphicsWorld.h and graphicsWorld.cpp`

What to Submit for Exercise A: As part of your lab report (PDF file) submit: your source codes (.cpp and .h files) and the program output showing your code in exercise A. Also submit a zipped file that contains your source files: .cpp and .h for all your classes.

<u>Exercise B - Multiple-Inheritance (10 marks)</u>

This exercise is the continuation of exercise A. You have to complete exercise A and then start this exercise.

C++ allows a class to have more than one parent:



This is called multiple-inheritance. For example if we have two classes called B and C, and class D is derived for both classes (B and C), we say class D has two parents (multiple inheritance). This is a powerful feature of C++ language that other languages such as Java do not support. For the details please refer to your class notes.

What to Do

In this exercise you should add two new classes to your program:

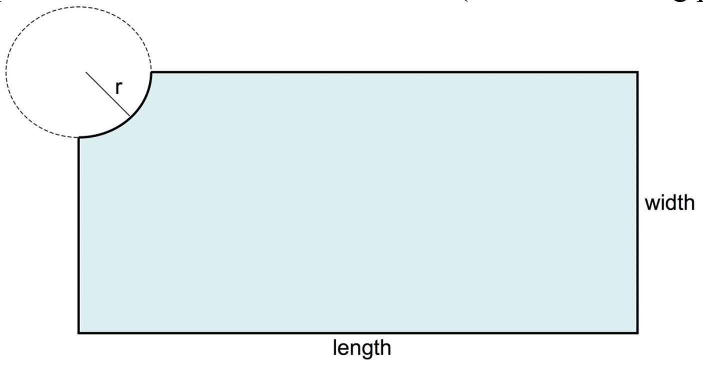
Class Circle:

This class is supposed to be derived from class Shape, and should have a data member radius, and in addition to its constructor it should support the similar functions as class `Rectangle`, such as `area, perimeter, get, set`, and `display`.

You should create two files for this class: called `circle.h and circle.cpp`

Class CurveCut:

CurveCut represents a shape that needs properties of class Rectangle and class Circle. In fact it's a rectangle that its left top corner can have an arch-form cut (see the following picture).



This class must be derived from class Rectangle and class Circle. Where the origins (x and y coordinates) of both shapes are the same. This class in addition to a constructor should support the following functions:

- `area` – that calculates the highlighted area of the above figure.
- `perimeter` – that calculates and returns the perimeter of highlighted areas.
- `display` – that displays the name, x, y coordinates of origin of the shape, width and length (as shown in the picture above), and radius of the cut, in the following format:

```
    CurveCut Name:
    X-coordinate:
    Y-coordinate:
    Width:
    Length:
    Radius of the cut.
```

Note: The radius of the circle must be always less than or equal the smaller of the width and length. Otherwise, program should display an error message and terminate.

You should also add some code to function `run` in class `GraphicsWorld` to:

- test the member functions of class CurveCut.
- use an array of pointers to Shape objects, where each pointer points to a **different object** in the shapes hierarchy. Then test the functions of each class again.

A sample code segment that you can use to test your program is given in the following box (you can add more codes to this function if you want to show additional features of your program).

```
void GraphicsWorld::run(){
    /****************************ASSUME CODE SEGMENT FOR EXERCISE A IS HERE ************************/
#if 0
     cout << "\nTesting Functions in class Circle:" <<endl;
     Circle c (3, 5, 9, "CIRCLE C");
     c.display();
     cout << "the area of " << c.getName() <<" is: "<< c.area() << endl;
     cout << "the perimeter of " << c.getName() << " is: "<< c.perimeter() << endl;
     d = a.distance(c);
     cout << "\nThe distance between rectangle a and circle c is: " <<d;

     CurveCut rc (6, 5, 10, 12, 9, "CurveCut rc");
      rc.display();
      cout << "the area of " << rc.getName() <<" is: "<< rc.area();
      cout << "the perimeter of " << rc.getName() << " is: "<< rc.perimeter();
      d = rc.distance(c);
      cout << "\nThe distance between rc and c is: " <<d;


      // Using array of Shape pointers:
      Shape* sh[4];
      sh[0] = &s;
      sh[1] = &a;
      sh [2] = &c;
      sh [3] = &rc;
      sh [0]->display();
      cout << "\nthe area of "<< sh[0]->getName() << "is: "<< sh[0] ->area();
      cout << "\nthe perimeter of " << sh[0]->getName () << " is: "<< sh[0]->perimeter();
      sh [1]->display();
      cout << "\nthe area of "<< sh[1]->getName() << "is: "<< sh[1] ->area();
      cout << "\nthe perimeter of " << sh[0]->getName () << " is: "<< sh[1]->perimeter();
      sh [2]->display();
      cout << "\nthe area of "<< sh[2]->getName() << "is: "<< sh[2] ->area();
      cout << "\nthe circumference of " << sh[2]->getName ()<< " is: "<< sh[2]->perimeter();
      sh [3]->display();
      cout << "\nthe area of "<< sh[3]->getName() << "is: "<< sh[3] ->area();
      cout << "\nthe perimeter of " << sh[3]->getName () << " is: "<< sh[3]->perimeter();

      cout << "\nTesting copy constructor in class CurveCut:" <<endl;
      CurveCut cc = rc;
      cc.display();

      cout << "\nTesting assignment operator in class CurveCut:" <<endl;
      CurveCut cc2(2, 5, 100,  12, 9,  "CurveCut cc2");
      cc2.display();
      cc2 = cc;
      cc2.display();
#endif
}         // END OF FUNCTION run
```

What to Submit for Exercise B:

1. As part of your lab report (PDF file) submit: Copy of your complete source codes (.cpp and .h files), and the program output showing your code in exercise B works.
2. A zipped file with source file: All `.cpp` and `.h` files