Object-Oriented Programming in C++

Lab Exercise 4 (2%)

Objective

The objective of this exercise is to deepen your understanding of inheritance, virtual functions, and polymorphism. You should work on the assignment in a **group of two**.

Description

Inheritance and polymorphism are powerful programming mechanisms if properly used. For example, graphical painting programs that work with different types of shapes would benefit from using such techniques. For example, one component of such a program would be to draw all the shapes onto a canvas (screen) at specific locations. A possible inheritance-based implementation would be to have a *shape* base class and then have a derived class for each of the specific shape, e.g. circles and rectangles.

Use the file *le04.cpp* as your starting point (for this exercise it is ok to write all the classes in this one file).

After finishing each part of the exercise call the instructor/TA over to demonstrate the behavior of your program.

a) Write a (non-abstract) base class called *Shape* that has: a *constructor* that takes a *Point* as an argument, a virtual method *void draw()* that draws the shape at a specific location (for our purposes it suffices to output as text the type and location of the object, e.g., "*Drawing Shape at location (1,2)* or *Drawing Circle at location (3,3)*"), and a virtual destructor.

   Then write two derived classes *Circle* and *Rectangle*. Provide the appropriate constructor/destructor, but do not implement the *draw* method yet. What does the above program output?

b) Now implement the *draw* method in the derived classes. Implement it first as non-virtual and then as virtual. What does the program output in those two cases? What is the difference?

c) Write a function *drawAll(...)* that takes a collection of shapes down as an argument and draws them all. Use a container, e.g., vector, to pass the elements down. Does the function print out what you expected? What type of elements does the vector contain? Why did you choose to do it that way?

d) A better design would be to make the *Shape* object abstract, because it is really more of an interface than a type to be instantiated. Change the code accordingly (including the *main* function, where you might have to comment some code out). What changes did you have to make?