Introduction to C++ Programming

Laboratory Guide

Lab 2 Inheritance, Ctor, Dtor & Static member

By Yidong Cui (cyd@bupt.edu.cn)

School of Software Engineering
Beijing University of Posts and Telecommunications
2019

1. Introduction

Based on EGE graphics library which were learned in "C Programming Language", practicing "copy constructor / destructor / static member" in C++ with C++11-17 standards.

This lab will take you 2 hours in class and about 2 hours after classes.

2. Aim of this lab

You should master the usage of a certain C++ IDE and you should finish this lab independently. You should comprehend the concepts of "constructor" and "destructor" in C++.

This lab is based on your achievement of Lab 1. You will add constructors and destructor for each class and practicing static variables and functions.

In this lab, **you may use the "refactor" function of your C++ IDE** to reconstruct your programs written in lab 1.

This lab also utilizes EGE graphics library.

3. Contents of this lab

3.1. Create new classes

Define an abstract geometric shape class. This abstract class works as the parent class of all other concrete geometric shape classes (not including classes like "color"). There is a pure virtual function draw() in this abstract class. The "draw()" function is used to draw your geometric shape in the graphics window.

BE AWARE: You also need to **move the common attributes** of the geometry class **to the base class**. For example, the border color of the geometric class.

3.2. Modify existing classes

1) Refactor the geometric shape classes in Lab 1 to make them inherit from the above abstract class.
2) Add a copy constructor and destructor in each class.
3) Add a static member variable to each class to count the corresponding objects being alive. Add a static getter function for the above counter variable.
4) Customize the destructor in each class to show the objects of the class which are alive.

3.3. Creating & destroying objects

The "color" member in any geometric shape class **MUST** be created with "new" operator and destroyed with "delete" operator. Other members in geometric shape class **may be created/destroyed with new/delete**, or just **defined as stack objects/variables** whose lifecycles are managed automatically.

Be aware, when the "color" member in a geometric shape object is created/destroyed with "new/delete", you should make "deep copy" in the copy constructor / destructor of the geometric shape class.

Any **geometric shape object** should ***be created with "new" and destroyed with "delete"***.

3.4. Draw the geometric shapes

In Lab 1, your program will get the shape data being inputted by the user and then draw these geometric shapes in the graphics window.

Now, your program should use "polymorphism" to draw the geometric shape. That is to say, you need to **call the "draw()" function** of any geometric shape object **with its base class pointer or base class reference**.

3.5. General requirements of you program

You may determine any details that are not mentioned above, provided that these details are reasonable for this lab.

4. Considerations for software engineering

1) You need to put all your codes into one "project", no matter which IDE you use.
2) Each class should be declared in one header file and implemented in the corresponding cpp file.
3) You should name your variables, functions, classes according to the coding rules we have learned in this class.
4) You should provide comments for each class. You also should provide comments for some important functions or code block as needed.

5. File version control

You **MUST** use version control system to keep your source codes.

In this lab, you need to make a TAG on the source tree **FIRST** to identify the achievements of Lab 1 before you make any modifications to the source codes.

When you finish all the coding tasks, you need to make a second TAG to identify the achievements of this Lab.

Then you need to export Lab1 source codes and Lab2 source codes. The source codes must be clean without any intermediate or temporary files. You may use TortoiseGit→Export to do so.

6. Important skills you need to master

Once you use pointers and new/delete in your program, you are very likely to meet a certain bug "access violation".

You must learn how to debug your program with debugger provided by your IDE.

The techniques for debugging include "breakpoints", "step in", "step over", "watch", etc.

7. The tools and environment

You should develop your program on Windows, because the EGE graphics library are not usable on other OS platforms. **If you insist working on Linux or MacOS, you need to contact the teacher to negotiate a feasible solution**.

You may use IDEs like Visual Studio, Eclipse CDT or Code::Blocks, and **the most up-to-date versions** are recommended.

If you use Visual Studio, you should install the community release.

Now, for the EGE graphics library, **only Visual Studio is officially supported**. If you are going to use other IDEs, you need to solve the related issues by yourself.

You are NOT allowed use dev-cpp or visual c++ 6.0, because they are old-enough and will not work with C++11/C++17

8. Submission guidelines

You need to submit the following materials

1) The project with your source files. You must do a "clean" before you pack your project files.
2) The executable file.
3) The lab report. (Must written according to the report template)
4) You should add a "readme" file in your project to demonstrate the information of the programmer, the running environment and the description of the result.

All the above files should be packed into a "zip" file. "rar" compress format is **NOT** allowed.

When packing your project files, "7zip", an open source compression software, is recommended.

The name of the zip file should be lab prefix plus your school-number plus your name. For example: Lab2-2017211888-李四.zip