BME506
Fall 2023 – Lab 6
Inheritance and Polymorphism

Duration: two weeks.

Note:

1. Every lab assignment must be done individually.
2. When you name a folder or a file, you **should** avoid spaces in those names. For example, if you need to name a folder as **GreenApple**, you should name it as **GreenApple** instead of **Green Apple.** Naming a folder or a file without spaces in the names is to avoid compilation error that may occur while using MinGW build tools (For details, you may see here: http://www.mingw.org/wiki/Getting_Started).
3. You might need to refer to the APIs of the C++ vector container. The website for the APIs for C++ vector is **www.cplusplus.com/reference/vector/vector/**
4. **ALL code should be compiled and run on the laboratory computers before submission.**

Goals:

The goal of this lab is to explore inheritance and polymorphism.

Requirements:

There are **two parts** in this lab, **Part-1** and **Part-2**. Part-1 involves implementation of a software application. Part-2 involves design of the software application in terms of a UML class diagram.

Part-1

Assume that a software application has four classes **ListOfShapes**, **Shape**, **Rectangle** and **Circle**. Rectangle and Circle are the subclasses of the class Shape. For this, there should be **9** files as follows:
Shape.h and **Shape.cpp**
Rectangle.h and **Rectangle.cpp**
Circle.h and **Circle.cpp**
ListOfShapes.h and **ListOfShapes.cpp**
Lab6Main.cpp that contains the relevant `main` function

The class Shape is an abstract class (i.e. we should not be able to instantiate an object of this class). It has the followings:

- A private member variable *color*.
- A non-default constructor for initializing the member variable *color*.
- A virtual destructor.
- A member function *getColor*(). The class Shape has the implementation of *getColor*() which returns the color of a shape.
- A pure virtual function called *print*().

The class Rectangle has the followings:

- Two private member variables *length* and *breadth*.
- A non-default constructor for initializing the length, breadth and color of a rectangle object.

- A virtual destructor.
- Overrides the virtual function *print*. The implementation of this function should print a rectangle object in a format that is consistent with the sample usage of the application shown later. (While printing, the area of a rectangle is to be computed using the formula (*length* × *breadth*)).

The class Circle has the followings:

- One private member variable *radius*.
- A non-default constructor for initializing the radius and color of a circle object.
- A virtual destructor.
- Overrides the virtual function *print*. The implementation of this function should print a circle object in a format that is consistent with the sample usage of the application shown later. (While printing, the area of a circle is to be computed using the formula (3.14 × radius × radius)).

The class ListOfShapes has the followings:

- One private member variable *shapes* of type `vector<Shape*>`. This member *shapes* stores pointers to multiple rectangle and circle objects.
- A destructor that destroys all the objects, each object being pointed to by a relevant pointer present in the vector. (You might need the member function *size* of vector).
- A member function *addShape*(*Shape**) that adds a pointer to a rectangle object or a circle object to the vector. (You might need the member function *push_back* of vector).
- A member function *removeShape*() that deletes the last element in the vector thereby reducing the number of elements in the vector by one. (You might need the member function *pop_back* of vector. Note that a call to *pop_back* is supposed to destroy the storage for an address in this case. The call to *pop_back* will not destroy the object pointed to by the address. Hence you might first need to get hold of the address (of the object) present in the last element of the vector by using the member function *at* of vector. You then need to explicitly destroy the object).
- A member function *displayShapes*() that iterates over the items present in the vector and outputs the information about each item by calling the virtual function *print* on the item.

There should be a console based command-line menu interface that should allow a user of the application to perform one of the operations **add rectangle**, **add circle**, **remove shape**, **display shapes**. The menu should also support a **quit** operation that should end the application.

Developing the Software

Using the information in the aforementioned section **Requirements**, you are required to implement the software.

The steps to build the C++ project in Netbeans for Lab6 are as follows:

1. Create a Netbeans project named **Lab6Proj** while taking care of the NOTE below.
   NOTE: While creating the project, please **make sure**

- you set the field "**Project Location:**" appropriately. **Be sure to remember this project location** path. (To keep track of your **Project Location** you may do as follows: Once you have created the project, open a File Explorer, and go into your **Project Location** folder).
- once you have chosen your **Project Location**, enter the word **Lab6Proj** in the field "**Project Name:**". Notice that your project name gets appended to your **Project Location** path.
- you **UN-CHECK** the checkbox on the left of "**Create Main File**" (We do so since we do not need any file named main.cpp file in this project).
- you Click **[Finish]** to create the project.
- a new folder named **`Lab6Proj`** has got created automatically under your **Project Location** folder. This **`Lab6Proj`** folder is your Netbeans ***project folder***.

2. Under netbeans project **Lab6Proj**, create the aforementioned C++ header files (that is **.h** files) and source files (that is **.cpp** files).

3. Note: At this point, make sure that your newly created files exist in the **`Lab6Proj`** folder.
4. Right-Click on project **Lab6Proj** >
   Select "Properties" >
   Click "Run" >
   On right, choose "Console Type" to be "Standard Output"
   Click [Apply]
   Click [OK]
5. Build the **Lab6Proj** project.
6. Open a Command Prompt window. Go to the folder **`Lab6Proj\dist\Debug\MinGW-Windows\`** (where **`Lab6Proj`** is the Netbeans *project folder*). You should see that the executable **lab6proj.exe** has got created in this folder.
7. Run the executable **lab6proj.exe** from the command prompt.

Important Notes:

1. When your source code is error-free, it should build successfully (as seen at the compilation process at Netbean's "Output" window). **However,** at times, in the Netbean's source editor window, you may notice red line below a built-in method, such as `size()` method of built-in type string. You may notice this even if your program has built successfully—For example, in the invocation `str1.size()`, you may see a red line below the word `size`. This is misleading. Such a red line should disappear if you perform the following steps:

   Right-Click on project **Lab6Proj** >
   Select "Code Assistance" >
   Select "Clean C/C++ cache and restart IDE"

A sample usage of the application is shown below.

```
prompt> lab6proj.exe

======================
[Shape List]
There are currently 0 shape(s) in the list

Please choose an option:
```

```
1.  Add Rectangle
2.  Add Circle
3.  Remove Shape
4.  Display Shapes
5.  Quit
>> 1
=============
[Add Rectangle]
Enter Color: red
Enter length: 1.5
Enter breadth: 0.3
.. [Adding Rectangle]
======================
[Shape List]
There are currently 1 shape(s) in the list

Please choose an option:
1.  Add Rectangle
2.  Add Circle
3.  Remove Shape
4.  Display Shapes
5.  Quit
>> 2
=============
[Add Circle]
Enter Color: blue
Enter radius: 2.0
.. [Adding Circle]
======================
[Shape List]
There are currently 2 shape(s) in the list

Please choose an option:
1.  Add Rectangle
2.  Add Circle
3.  Remove Shape
4.  Display Shapes
5.  Quit
>> 1
=============
[Add Rectangle]
Enter Color: purple
Enter length: 2.0
Enter breadth: 1.2
.. [Adding Rectangle]
======================
[Shape List]
There are currently 3 shape(s) in the list

Please choose an option:
1.  Add Rectangle
2.  Add Circle
3.  Remove Shape
4.  Display Shapes
5.  Quit
>> 4
=============
[Display Shapes]
```

```
Rectangle L=1.5, B=0.3, area=0.45, red
Circle R=2.0, area=12.56, blue
Rectangle L=2.0, B=1.2, area=2.4, purple

=======================
[Shape List]
There are currently 3 shape(s) in the list

Please choose an option:
1.  Add Rectangle
2.  Add Circle
3.  Remove Shape
4.  Display Shapes
5.  Quit
>> 3
=============
..[Removing Shape]

=======================
[Shape List]
There are currently 2 shape(s) in the list

Please choose an option:
1.  Add Rectangle
2.  Add Circle
3.  Remove Shape
4.  Display Shapes
5.  Quit
>> 4
=============
[Display Shapes]

Rectangle L=1.5, B=0.3, area=0.45, red
Circle R=2.0, area=12.56, blue

=======================
[Shape List]
There are currently 2 shape(s) in the list

Please choose an option:
1.  Add Rectangle
2.  Add Circle
3.  Remove Shape
4.  Display Shapes
5.  Quit
>> 5
=============
.. [Quiting]


prompt>
```

Part-2

In this part, you are required to provide the **UML class diagram** for the design of the above software application. You should use the **Visual Paradigm** to create the class diagram. Once you have created the diagram, scan/crop/capture it and paste it (as image) in a Microsoft Word document named *lab6-Part-2-UML-Class-Diagram.docx*. Next, convert this file into a PDF file and name it ***lab6-Part-2-***

UML-Class-Diagram.pdf. You may use any freely available online converter to do the conversion to PDF.

Lab Submission

Deadline: See announcement in D2L for deadline.

ALL code should be compiled and run on the laboratory computers before submission.

Create a folder. Name it as YourLastname_YourFirstname_bme506_Labnumber.
Example: If student name is John Smith, the name of folder should be
Smith_John_bme506_Lab6.

Copy the followings in the above folder:

1. your Netbeans ***project folder***, **Lab6Proj**. Make sure the folder **Lab6Proj** contains the relevant **.h** and **.cpp** files.
2. your file ***lab6-Part-2-UML-Class-Diagram.pdf***

The above folder must also contain a duly filled and signed standard cover page. The cover page can be found on the departmental web site: Standard Assignment/Lab Cover Page

Compress the above folder as a single zip file that is named
according to the following rule:
YourLastname_YourFirstname_bme506_Labnumber.zip.
Example: Smith_John_bme506_Lab6.zip.

Upload the above zip file on D2L through the "Assessment" > "Assignments" link.

Note: If the code does not compile, the submission will receive a ZERO mark.

Resources

[1] Cplusplus.com Reference [Online]. Available: http://www.cplusplus.com/reference