Macro - Template - Inheritance - Polymorphism

In this session, you are provided with some sample code that illustrates the following concepts:

- Macro
  - A piece of code in a program that is replaced by the value of the macro
  - See example in **src/macro_template/macro_demo.cpp**
- Template
  - A C++ entity that defines one of the following:
    - A family of classes
    - A family of functions
  - See example in **src/macro_template/template_demo.cpp**
  - See **Figure 1**
- Inheritance
  - The capability of a class to derive properties and characteristics from another class
  - See example in **src/inheritance/inheritance_demo.cpp**
  - See **Figure 2**
- Polymorphism
  - A call to a member function will cause a different function to be executed depending on the type of object that invokes the function
  - See example in **src/inheritance/inheritance_demo.cpp**

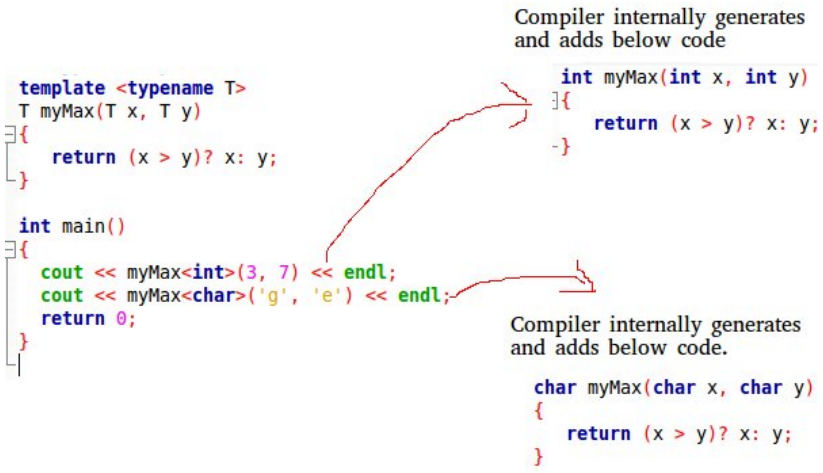

Figure 1.

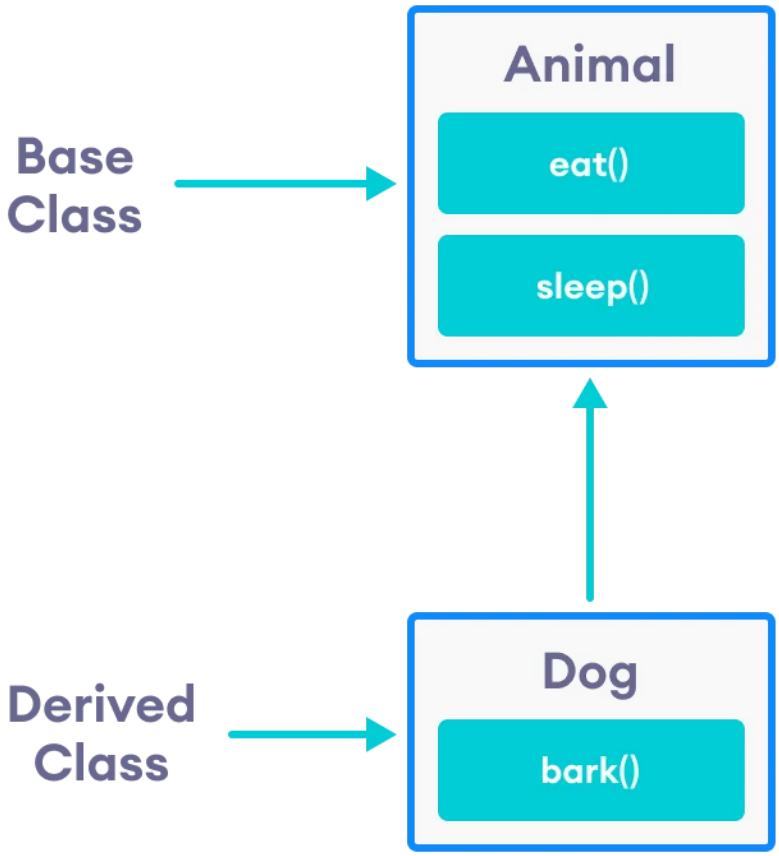

Figure 2.

NOTE:

- In the example code, I put everything (class declaration, class definition, main program,...) in one file.
- This is bad practice - you should NOT do this!!!
    - You need to follow the practice in your previous homework.
- A small practice to you:
    - Refactor/reorganize the sample code to follow the best practice

Deadline: you are required to submit your solution before **12PM on Sunday 27th 2022**.

- Your code needs to be compiled successfully when review
- You have to answer every aspect in your code
- You can discuss the approach with your fellows, but each needs to write and understand the code clearly.
    - You don't have any grades here, you are doing this for your own benefit.
    - In case of getting stuck (even after consulting every material), you can come and ask me in person

SUBMISSION REQUIREMENT:

- your code should be compiled successfully and passed the unit test
- your code needs to conform to Google Coding Style for C++

How to submit:

- at 11:55-12:00 of submission date:
  - push you code (along with the branch with your name, such as PhongND) to the following link: https://git.d-soft.com.vn/dng.pj0018.iot.lab/development/cpptraining
- any early or late submission beyond the specified time will not be accepted

Code review and check:

- will be performed individually, at your workstation, starting from 13:30

Directory Structure

```
├── CMakeLists.txt
├── data
│   ├── data_info.txt
│   └── linux_test.png
├── include
│   └── foo.h
├── main.cpp
└── src
    ├── cmake_images
    │   └── main.cpp
    ├── CMakeLists.txt
    ├── cmake_multiple_sources
    │   └── main.cpp
    ├── inheritance
    │   └── inheritance_demo.cpp
    ├── lib
    │   └── foo.cpp
    ├── macro_template
    │   ├── macro_demo.cpp
    │   └── template_demo.cpp
    └── test
        ├── test_lodepngimg.cpp   # write this in task 2
        └── test_pngppimg.cpp     # write this in task 2
```

Task 1: Image Class

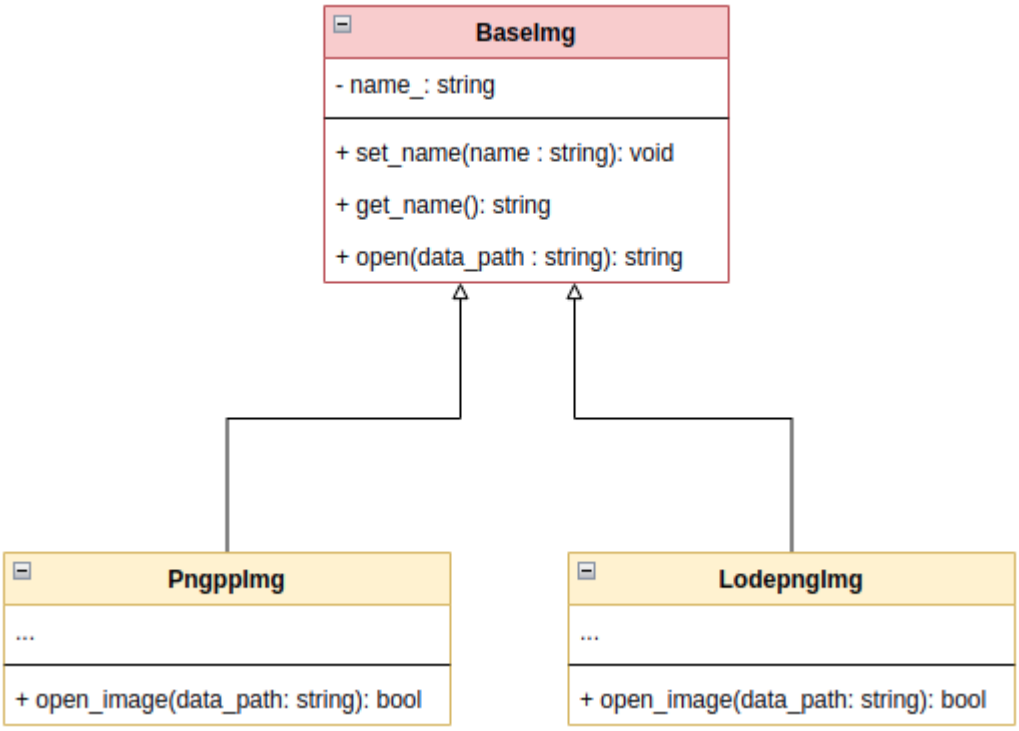

Figure 3.

You are required to implement the class diagram in **Figure 3**.
You need to write a main program to demonstrate the polymorphic call of base class (see below).
The task will utilize:

- Macro:
  - To implement **set**_ and **get**_ functions
- Template
  - Must use, to achieve the following invocation (polymorphic call)
    BaseImg<PngppImg> pngpp_i;
    pngpp_i.open(path_to_data_file); // call open_image() of PngppImg class
- Inheritance
  - With **BaseImg**, **PngppImg**, and **LodepngImg**
- Polymorphism
  - With **open()** and **open_image()**
- As always, you MUST write a unit test for your code
  - This unit test is not only used for your code
    - It will also be used to test code written by your fellow
    - Refer to **Table 1** to know who will test your code
  - For each derived class, write at least 4 tests with the following cases:
    - ValidFilePath
    - InvalidFilePath

        - SetGetCorrectly
        - SetGetIncorrectly
    - NOTE: you must utilize Polymorphism in the test case.
      Do NOT create a plain object such as:
      LodepngImg lodepngimg_obj; // NOT acceptable
- OPTIONAL:
    - Find way to print the object's class name
        - Hint: need to use an external library

No.	**Implementer**	**Tester**
1	Phong	Duc
2	Duc	Tai
3	Tai	Tam
4	Tam	Phong

Table 1.

Task 2: Image Class - Unit Test

As always, you MUST write a unit test for your code:

- This unit test is not only used for your code
    - It will also be used to test code written by your fellow
    - Refer to **Table 1** to know who will test your code
- For each derived class, write at least 4 tests with the following cases:
    - ValidFilePath
    - InvalidFilePath
    - SetGetCorrectly
    - SetGetIncorrectly
- NOTE: you must utilize Polymorphism in the test case.
  Do NOT create a plain object such as:
  LodepngImg lodepngimg_obj; // NOT acceptable
- OPTIONAL:
    - Find way to print the object's class name
        - Hint: need to use an external library

Task 3: Inheritance and Polymorphism

Solve the three tasks present in this link.
I will walk through this with you guys. This is live coding.
NOTE: only start this task after you finish the above two tasks. This is of lower priority than task 1 and task 2.