BME506
Fall 2023 – Lab 7
Design Patterns

Duration: one week.

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

Goals:

The goal of this lab is to demonstrate the application of singleton design pattern. The pattern is to be implemented in C++ language. You should remember to use separate header/source files. The header file (**Lab7.h**) should contain the declaration of members in relevant class(es). The source file (**Lab7.cpp)** should contain the body of relevant member functions, the initialization of the relevant static member variable, and the definition of the **main** function. The partial code has already been provided in the **Appendix** at the end of this file. You **must** use them in your implementation.

Singleton Pattern

A Factory object can be used to create Product objects with specified IDs. We want to design the Factory class as a singleton class. The Product class is already shown below. Apply the singleton design pattern to the Factory class i.e. add the necessary code to the Factory class so that it becomes a singleton class. The **main** function that uses the Factory and Product classes is also shown below. The output obtained on executing the **main** function should be:

```
5
6
```

```
//definition of Product class
class Product {
private:
    int id;
public:
    Product(int i) {id = i;}
    int getID() {return id;}
}
```

```
//definition of the function main
int main (int argc, char* argv[]) {
    Factory* f = Factory::getInstance();
    Product* p1 = f->createProduct(5);
    Product* p2 = f->createProduct(6);
    cout << p1->getID() << endl;
    cout << p2->getID() << endl;
    return 0;
}
```

The steps to build the C++ project in Netbeans for Lab7 are as follows:

1. Create a Netbeans project named **Lab7Proj** 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 **Lab7Proj** 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 **Lab7Proj** has got created automatically under your **Project Location** folder. This **Lab7Proj** folder is your Netbeans ***project folder***.
2. Under netbeans project **Lab7Proj**, right-click on the "Header Files" > "New" > "C++ Header File …". Set the field **"File Name:"** as **`Lab7`**. Set the **"Extension:"** as "**`h`**". Click **[Finish]**.
3. The file **`Lab7.h`** is created.
4. Under netbeans project **Lab7Proj**, right-click on the "Source Files" > "New" > "C++ Source File …". Set the field **"File Name:"** as **`Lab7`**. Set the **"Extension:"** as "**`cpp`**". Click **[Finish]**.
5. The file **`Lab7.cpp`** is created.
6. In **`Lab7.h`** and **`Lab7.cpp`**, add the relevant code.
7. Note: At this point, make sure that your two newly created files **`Lab7.h`** and **`Lab7.cpp`** exists in the **Lab7Proj** folder.
8. Right-Click on project **Lab7Proj** >
   Select "Properties" >
   Click "Run" >
   On right, choose "Console Type" to be "Standard Output"
   Click [Apply]
   Click [OK]
9. Build and run **Lab7Proj** project.

Important Notes:

1. When your 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 **Lab7Proj** >
   Select "Code Assistance" >
   Select "Clean C/C++ cache and restart IDE"

Lab Submission

Deadline: See announcement in D2L for deadline.

ALL code should be compiled 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_Lab7.

Copy your Netbeans ***project folder***, **Lab7Proj**, in the above folder. Make sure the folder **Lab7Proj** contains the relevant **`.h`** and **`.cpp`** files.

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_Lab7.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

Appendix

Partial code of your Lab7.h file is given below:

```
#ifndef LAB7_H
#define LAB7_H

#include <iostream>
using namespace std;

class Product {
private:
    int id;
public:
    Product(int i);
    int getID();
};

//WRITE THE RELEVANT CODE HERE RELATED TO FACTORY CLASS FOLLOWING
//THE SINGLETON PATTERN.
...

#endif /* LAB7_H */
```

Partial code of your Lab7.cpp file is given below:

```
#include "Lab7.h"

Product::Product(int i) {id = i;}

int Product::getID() {return id;}


//WRITE THE RELEVANT CODE HERE RELATED TO FACTORY CLASS FOLLOWING
//THE SINGLETON PATTERN (STATIC MEMBER INITIALIZATION AND BODIES OF
//MEMBER FUNCTIONS).
...


int main (int argc, char* argv[]) {
    Factory* f = Factory::getInstance();
    Product* p1 = f->createProduct(5);
    Product* p2 = f->createProduct(6);
    cout << p1->getID() << endl;
    cout << p2->getID() << endl;
    return 0;
} //end of main
```