BME506
Fall 2023 – Lab 5
Object-Oriented Design and Implementation of a simple application

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. **The Appendix has important information.**
4. **ALL code should be compiled and run on the laboratory computers before submission.**

Goals:

The goal of this lab is to develop an object-oriented design of a simple software system that involves a minimum of two separate classes. The classes work together to realize a **queue** of objects and some operations on the **queue**. The software is to be implemented in C++ language. Again, you should remember to use separate header/source files. Also, please make sure to read the Appendix section.

Background:

The software system to be built in this lab has to handle the admission of patients to an obstetrical triage (the waiting/preparation stage at the hospital before entering the birthing unit). The software has to realize a computerized obstetrical triage patient register (i.e. a queue of expectant mothers with signs of impending labour waiting for a bed in the birthing unit). The system should define a minimum of two classes: Patient is represented by a **Patient** class, while the Register (queue of patients) is represented by another class **OTPRegister**. Patients are added to the end of the queue. Patients are then processed in the order of arrival i.e. whoever is first in the queue is processed first—such a queue is known as a FIFO queue (first in first out queue). Figure 1 shows how the queue looks like.

There should be a single register. The register should support the followings:

- contain a queue of patients with no limit on the number of patients in the queue (This unlimited number of elements in the queue is an assumption to simplify the implementation of the software).
- *register* (i.e. enroll) a patient – construct a new patient and add it to the end of the queue.
- *examine* and process a patient from the front of the queue.
- *display* all patients currently in the queue.

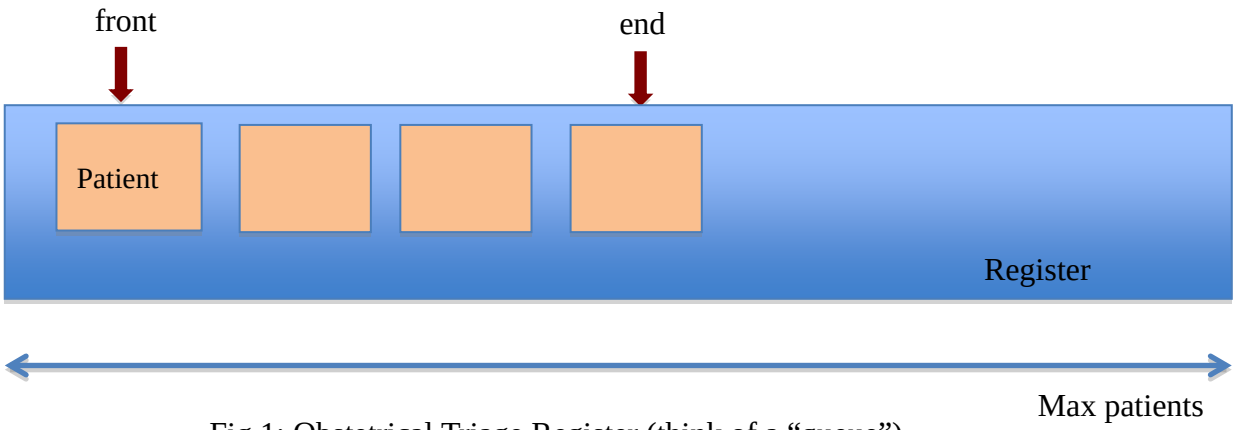

Fig 1: Obstetrical Triage Register (think of a “queue”)

Requirements:

1.1.1. Information defining a Patient includes: name, health card number, condition, cervix dilation (0-10cm).

1.1.2. It is possible to display all the patients present in the queue. You should output the information present in Patient member variables to standard output in such a way that one line of output contains the data about one patient. The first line should display the information of the first patient (i.e. the patient in front of the queue), the second line should display the information of the second patient i.e. the patient behind the first patient and so on.

1.1.3. The condition of a patient can be one of the followings: UNKNOWN, LABOUR, and ADMIT.

1.1.4. When a patient arrives at the registration desk, the attending nurse uses the software to register the patient by entering patient's name and health card number. For a newly arrived patient, her condition should be UNKNOWN by default. A patient in this condition is assumed to have cervix dilation of 0cm. As a result of registration, the patient gets added to the end of the queue.

1.1.5. Examining a patient involves removing the first patient from the queue (i.e. the patient at the front of the queue) and performing an appropriate action depending on her condition as explained below. The examination may or may not result in the patient being added back to the end of the queue.

1.1.5.1. If a patient’s condition is UNKNOWN, the patient is examined for cervix dilation by the nurse. The nurse enters the cervix dilation information to the system.

1.1.5.1.1. If the patient’s cervix dilation is >= 4cm, her condition gets changed to ADMIT, a message is displayed showing the patient’s information, and finally she should be removed from the queue.

1.1.5.1.2. If the patient’s cervix dilation is < 4cm, her condition gets changed to

LABOUR. The patient again gets added back to the end of the queue.

1.1.5.2. If a patient's condition is LABOUR, the patient is examined for cervix dilation by the nurse. The nurse enters the cervix dilation information to the system.

1.1.5.2.1. If the patient's cervix dilation is >= 4cm, her condition gets changed to ADMIT, a message is displayed showing the patient's information, and finally she should be removed from the queue.

1.1.5.2.2. If the patient's cervix dilation is < 4cm, her condition stays the same i.e. LABOUR. The patient again gets added back to the end of the queue.

1.1.6. There should be a textual menu interface that should allow the attending nurse at the registration desk to perform one of the operations *register*, *examine*, or *display*. 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 should design and implement a software that allows a user to register a patient, examine a patient or display all patients present in the queue at any given point of time.

You must name the C++ files as follows:
OTPRegister.h
OTPRegister.cpp
Patient.h
Patient.cpp
Lab5Main.cpp

OTPRegister.h should contain the declaration of the class **OTPRegister. OTPRegister.cpp** should contain the definition of member functions of the class **OTPRegister**. **Patient.h** should contain the declaration of the class **Patient. Patient.cpp** should contain the definition of member functions of the class **Patient**. **Lab5Main.cpp** should contain the `main` function of this C++ program.

The steps to build the C++ project in Netbeans for Lab5 are as follows:

1. Create a Netbeans project named **Lab5Proj** 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 **Lab5Proj** 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 **`Lab5Proj`** has got created automatically under your **Project Location** folder. This **`Lab5Proj`** folder is your Netbeans ***project folder***.

2. Under netbeans project **Lab5Proj**, **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 **`Lab5Proj`** folder.
4. Right-Click on project **Lab5Proj** >
   Select "Properties" >
   Click "Run" >
   On right, choose "Console Type" to be "Standard Output"
   Click [Apply]
   Click [OK]
5. Build the **Lab5Proj** project.
6. Open a Command Prompt window. Go to the folder **`Lab5Proj\dist\Debug\MinGW-Windows\`** (where **`Lab5Proj`** is the Netbeans *project folder*). You should see that the executable **lab5proj.exe** has got created in this folder.
7. Run the executable **lab5proj.exe** from the command prompt.

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 **Lab5Proj** >
   Select "Code Assistance" >
   Select "Clean C/C++ cache and restart IDE"

An example usage of the software is given below. The text in red color is the data entered by the user of the software (e.g. a nurse):

```
prompt> lab5proj.exe
======================
There are currently 0 patient(s) in the list

Please choose an option:
1.  Register Patient
2.  Examine Patient
3.  Display Patients
4.  Quit
>> 1
=============
[Register Patient]
Enter Name: Jane
Enter Health Card Number: 111
.. [Registering]

======================
```

```
There are currently 1 patient(s) in the list

Please choose an option:
1.  Register Patient
2.  Examine Patient
3.  Display Patients
4.  Quit
>> 1
=============
[Register Patient]
Enter Name: Susan
Enter Health Card Number: 222
.. [Registering]

========================
There are currently 2 patient(s) in the list

Please choose an option:
1.  Register Patient
2.  Examine Patient
3.  Display Patients
4.  Quit
>> 3
=============
[Display Patients]
Jane   111  0  UNKNOWN
Susan  222  0  UNKNOWN

========================
There are currently 2 patient(s) in the list

Please choose an option:
1.  Register Patient
2.  Examine Patient
3.  Display Patients
4.  Quit
>> 2
=============
[Examine Patient]
Name: Jane
Health Card Number: 111
Enter Cervix Dilation: 4
.. [Admitting]
Jane  111  4  ADMIT

========================
There are currently 1 patient(s) in the list

Please choose an option:
1.  Register Patient
2.  Examine Patient
3.  Display Patients
4.  Quit
>> 2
=============
[Examine Patient]
Name: Susan
Health Card Number: 222
Enter Cervix Dilation: 1
```

```
.. [Going back to queue]

=======================
There are currently 1 patient(s) in the list

Please choose an option:
1.  Register Patient
2.  Examine Patient
3.  Display Patients
4.  Quit
>> 3
=============
[Display Patients]
Susan  222  1  LABOUR

=======================
There are currently 1 patient(s) in the list

Please choose an option:
1.  Register Patient
2.  Examine Patient
3.  Display Patients
4.  Quit
>> 1
=============
[Register Patient]
Enter Name: Padma
Enter Health Card Number: 333
.. [Registering]

=======================
There are currently 2 patient(s) in the list

Please choose an option:
1.  Register Patient
2.  Examine Patient
3.  Display Patients
4.  Quit
>> 3
=============
[Display Patients]
Susan  222  1  LABOUR
Padma  333  0  UNKNOWN

=======================
There are currently 2 patient(s) in the list

Please choose an option:
1.  Register Patient
2.  Examine Patient
3.  Display Patients
4.  Quit
>> 2
=============
[Examine Patient]
Name: Susan
Health Card Number: 222
Enter Cervix Dilation: 3
.. [Going back to queue]
```

```
======================
There are currently 2 patient(s) in the list

Please choose an option:
1.  Register Patient
2.  Examine Patient
3.  Display Patients
4.  Quit
>> 3
=============
[Display Patients]
Padma  333  0  UNKNOWN
Susan  222  3  LABOUR

======================
There are currently 2 patient(s) in the list

Please choose an option:
1.  Register Patient
2.  Examine Patient
3.  Display Patients
4.  Quit
>> 4
.. [Quiting]
```

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_Lab5.

Copy your Netbeans ***project folder***, **Lab5Proj**, in the above folder. Make sure the folder **Lab5Proj** 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_Lab5.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

Note: please avoid the word **`register`** while naming any class, variable or function since the word **`register`** is a keyword in C++.

In this lab you need to use a data type `deque` provided by C++ library. For this, you need to use the following **#include** statement:

```
#include <deque>
```

You must use `deque` to implement a queue of pointers of the class **Patient**. This queue of pointers of the class **Patient** must be declared as a member variable of the class **OTPRegister** as follows:

```
deque< Patient * >  q;
```

The above declaration enables the variable **`q`** to store pointers to objects of the class Patient in a queued fashion. See the lecture slides for usage of the data type `deque`.