Project 2

I have made message passing program using thread.

Program Logic:

I used QHash class for O(1) lookup data structure.

And I made Message class to implement a mechanism for encapsulating.

There are id and data packet. Data packet has 3 numbers.

And some functions are here for encapsulating like setData, getData, setId, getId and so on.

Thread A send Data packet list to Thread B.

Thread B process Data packet list. Thread B waits until Thread A send data. And if Thread A finish sending data, Thread B process Data packet list.

Logic of my program is above.

Implementing Serialization and How can my approach can be extended to enable distributed message Passing

To implement serialization, I can attach size of Data packet in header of packet. In Thread B, parse Data packet and separate each part and then check each part.

After that Thread B can process Data packet.

In order to extend to enable distributed message passing, I can change my project like this;

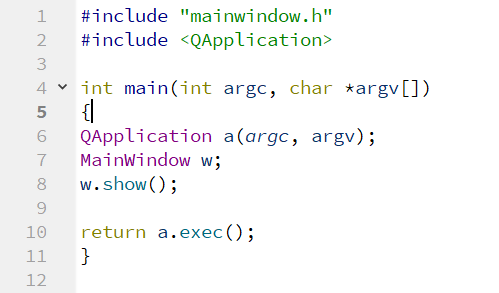
A process the first data packet and then send the second data packet to B. B process the second data packet and then send the third data to C. C also process data and next data will be sent to A again. Each processed data will be stored in storage.

Every process sends data and number of next data.

I can change of data structure. Now in Message class, there are only 2 elements: id, data.

But If need distributed message passing, add some information in message class.

Project :



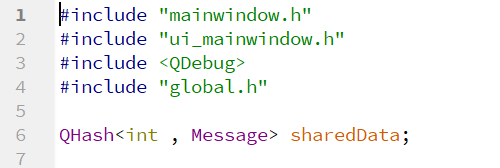
In this IDE first we have to include the argc which is the number of command line arguments and \*argv[] is the array of the character string which is the individual argument provided on the command line as you see on line number 4.

In Line number 7 I make an object of another class and open it on a dialog for User Interface purpose.

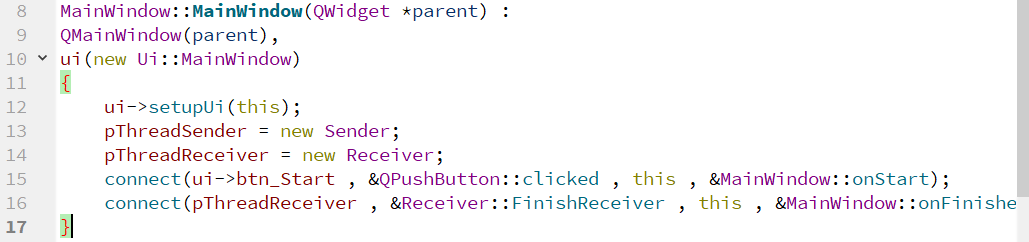
In the below cpp file we include the given header file having the class definition & function defined in this class.



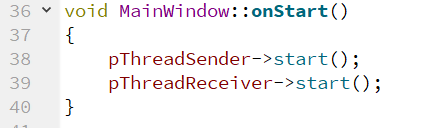
In mainWindow.cpp first we include the essential header files as you see in the below screenshot.



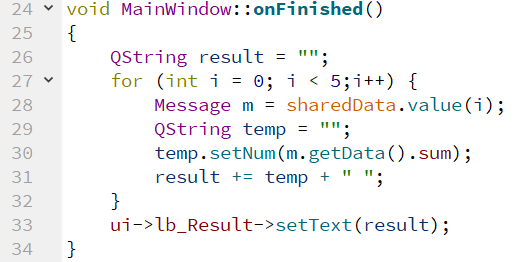
This below code is used to open the User interface which is just called when this class opens. This is the Overloaded Constructor/Parameterized Constructor.



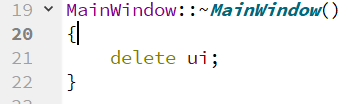
This is the start () function which run on the opening dialog of the class.



The below finished () function is used when it reads the message and then it sets the value on the text field.



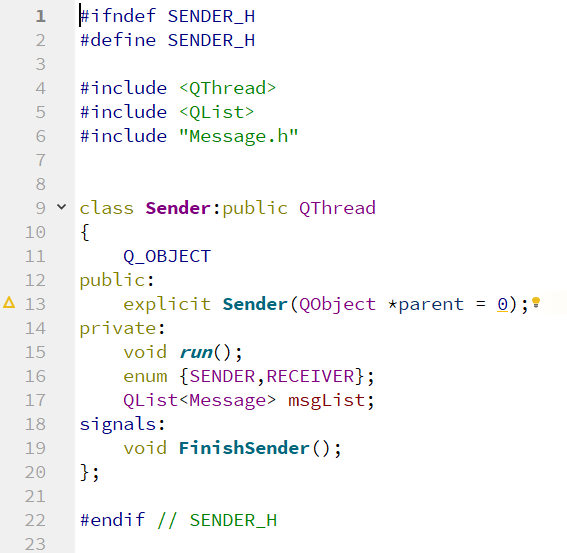
Destructor for the Class in which we close the UI window



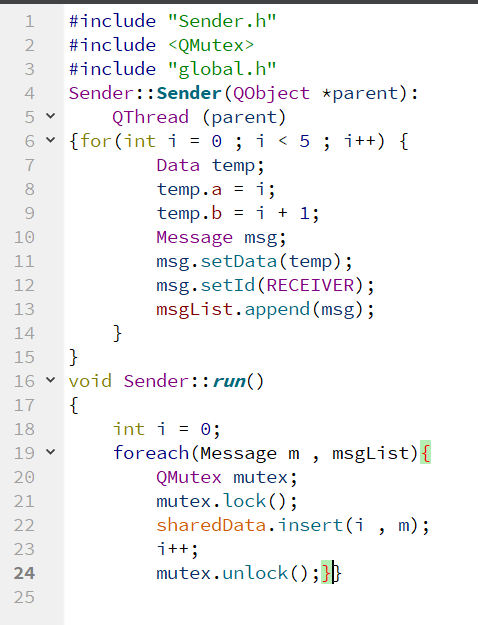
For the Sender classes we have two classes

Sender. H is the Header file.

Sender. CCP is the Source File.



The above class is the header class where we can implement the methods of the senders and defining the linked list in it.



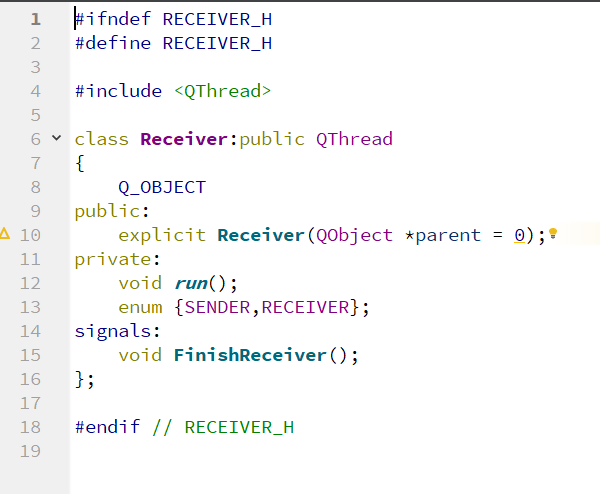
The above code is the CPP file of the sender which sends the code as follow.

13579 code generated by the overloaded constructor

00000 code generated by the run () method.

In the class of the Receiver we build separate classes Receiver.cpp & Receiver. h

In the Receiver .h we apply the concept of Encapsulation and also I inherit the QTHread class in it.



Receiver.cpp File is defined below



This class receives the data from the Sender class and first its Overloaded Constructor is called and then it receive the message as you see below the output screen which are passed from the sender class and then after pressing the start button it runs the run() method and get the data from the sender class and then place in the dialog.

