Programming Guide : Homework 2

This is a quick guide on how to start your homework 2 with the given skeleton code of a client program. The objective of the skeleton is to let students focus on gRpc programming, and also to evaluate your homework fairly by having the same skelton. The skeleton is almost similar to that of hw1.

1. Skeleton consists of :

- **client.h** : contains an abstract class that has three pure virtual functions to be implemented by you and data structures.

2. Derive IClient class and make your own Client.

Every business logic has been implemented except for major functions that use gRpc and communicate with the server. What you are supposed to do is just making a concrete class of IClient and implementing three virtual functions. Following is one possible example of the concrete class.

```
class Client : public IClient {
  public:
    Client(const std::string& hname,  const std::string& uname,  const std::string& p)
        :hostname(hname), username(uname), port(p)
        //, stub_(/* some parameters */)
        {}
  protected:
    virtual int connectTo();
    virtual IReply processCommand(std::string cmd);
    virtual void processTimeline();
  private:
    std::string hostname;
    std::string username;
    std::string port;
    // You can have an instance of the client stub
    // as a member variable.
    std::unique_ptr<ClassNameOfYourStub> stub_;
};
```

```
int Client::connectTo()
{
   // your implementation
   return 1; // return negative if failed
}

IReply Client::processCommand(std::string input)
{
   // your implementation
   IReply ire;
   return ire;
}

void Client::processTimeline()
{
   // your implementation
}
```

2.1 function "connectTo"

In this function, you are supposed to create a stub so that you call service methods in the ***processCommand/porcessTimeline*** function. That is, the stub should be accessible when you want to call any service methods. I recommend you have the stub as a member variable in your own Client class. Please refer to gRpc tutorial how to create a stub.

2.1 function "processCommand"

In this function, you are supposed to parse the input command and call an appropriate service method. Then, you need to return ***IReply*** that contains results taken from the response of the service method. The skeleton will display the result based on the IReply. ***IStatus*** and ***IReply*** is almost same as *Status* and *Reply* of hw1.

```
enum IStatus
{
   SUCCESS,
   FAILURE_ALREADY_EXISTS,
   FAILURE_NOT_EXISTS,
   FAILURE_INVALID_USERNAME,
   FAILURE_INVALID,
   FAILURE_UNKNOWN
};
```

```
struct IReply
{
   grpc::Status grpc_status;
   enum IStatus comm_status;
   std::vector<std::string> users;
   std::vector<std::string> following_users;
};
```

2.3 function "processTimeline"

In this function, you are supposed to get into timeline mode. You may need to call a service method to communicate with the server. Use ***getPostMessage/displayPostMessage*** functions for both getting and displaying messages in timeline mode. You should use them as you did in hw1.

```
// in client.h

std::string getPostMessage()
{
   // this function has been implemented for you.
}

void displayPostMessage(std::string sender, std::string message, std::time_t time)
{
   // this function has been implemented for you
}
```

3. You must invoke "*start_client*" function.

In order to start the business logic provided by the skeleton, you should invoke ***"start_client"*** function that is implemented in IClient when you are ready to go. Following code snippet can be an example.

```
int main(int argc, char** argv) {
  // process command line argument
  // do something else for your program

   Client client(hostname, username, port);
   client.run_client();

   return 0;
}
```

4. Do not print anything on the screen.

The skeleton does print everything on screen for you to interact with a user. You can print some debugging information during the development phase, but you must erase them when you submit your homework.

5. Test your program with given test cases

There are five test cases that cover the most scenarios in homework 2. You can see test cases and its results in the provided excel file. Before testing each test case, remove your data file and restart your server/client program to have the same output as the given output in the excel file. Evaluation will be done with different test cases.

6. Getting started with gRpc

Although gRpc has been installed in the provided VM, you need to get familiar with gRpc so that you can write code. Followings are a good starting point for your homework.

- gRpc C++ tutorial : https://grpc.io/docs/tutorials/basic/c.html
- gRpc C++ API : https://grpc.io/grpc/cpp/index.html