Lab XII: Encrypting and Decrypting

CPSC 1181

Write a client program to interact graphically with a server that encrypts and decrypts phrases.

Your client program written into the file Clinet.java should use the "Java Interface" file found on the O:drive called Encryptable.java.

Encryptable.java has the port that should be used, as well as the host which is the server (which is the instructor's computer) and the protocol.

Your client program should have

- in the GUI have a JLabel that says 'plain text'
- in the GUI next to the 'plain text' label have a JText field so that the user can enter a phrase and hit 'Enter' and the plain text is then sent to be encrypted. The result from the server is then written next to the label 'encrypted text'
- in the GUI have a JLabel that says 'encrypted text'
- in the GUI next to the 'encrypted text' label have a JText field so that the user can enter a phrase and hit 'Enter' and the encrypted text is then sent to be deciphered. The result from the server is then written next to the label 'plain text'.
- provide three buttons that send to the server that the text is to be encrypted, decrypted or that the client wants to stop.
    - encrypt
    - decipher or decrypt
    - stop

Please refer to the file Encryptable.java for a description of the protocol but here is the idea of the protocol (you may want to read the html file in the directory called docs in the O: drive).

I am using the symbolic names here but they are going to be integers as per the file Encryptable.java

- The server sends to the client the command NAME followed by an integer. The integer corresponds to the client number (name). *The server initiates the session (not the client)*
    - You could display this number as the title of the window (the JFrame).

After that, in any order one of the following

- The server is *expecting* from the client i.e. the client sends
    - ENCRYPT followed by the number of characters that are contained in the 'phrase to encrypt' followed by the characters in the 'phrase to encrypt'. The characters are sent one at a time.
- The server *responds* and sends to the client
    - ENCODED followed by the number of characters that will be sent. And then followed by the actual characters in the 'plain text' that was decrypted. The characters are sent one at a time.

or

- The server is *expecting* from the client i.e. the client sends
    - DECRYPT followed by the number of characters that are contained in the 'phrase to decrypt' followed by the characters in the 'phrase to decrypt'. The characters are sent one at a time.
- The server *responds* and sends to the client
    - PLAIN followed by the number of characters that will be sent. And then followed by the actual characters in the 'encoded text'. The characters are sent one at a time.

or

- The server is *expecting* from the client i.e. the client sends
    - QUIT meaning that the client is done sending to the server.
- The server *responds* and sends to the client
    - DONE meaning that the server is done serving the client.

Use the methods of the class DataInputStream readInt and readChar and the methods from the class DataOutputStream `writeInt` and `writeChar to` talk to the encrypting server. Do not use PrintWriter as it "prints formatted representations of objects to a text-output stream."

For instance, to send to the server the string “HELLO” with `toServer` as show in class

```
toServer.writeInt(ENCRYPT);
toServer.writeInt(5);
toServer.writeChar('H');
toServer.writeChar('E');
toServer.writeChar('L');
toServer.writeChar('L');
toServer.writeChar('O');
toServer.flush();
```

Test your program with short phrases. Only when you know that your program works, decode the following phrase

```
WAMMFSMBJHO G CS HAMZZAMQIDLNGTNDJRBUDNIWDZMGKZYAV
```

Note that the encryption algorithm converts all the lower case letters to upper case letters and “maps” all the non-upper case letter characters to blanks. The encryption algorithm is the Vigenère cipher (in case you want to break it).

Create a new thread of execution for the Client by declaring a class that implements the Runnable interface. That class will implement the ‘run’ method. An instance of that class is then passed as an argument when creating an object of the class Thread which is started subsequentaly.

You must use the Client.java found in the O:drive as a starting program. Fill in the missing code (and there is a lot of missing code). You can change the lines of code already in the file Client.java as much or as little as you want.

To submit today as Lab XII as a single compressed (zip) file

1. The file Client.java with all your code in it.
2. The file Encryptable.java
3. A text file called README.txt with
    a. the decrypted message given the encrypted line
    `WAMMFSMBJHO G CS HAMZZAMQIDLNGTNDJRBUDNIWDZMGKZYAV`
    b. any other comments that you might want to give to the marker

For your information: The following properties of a JTextField can be set as follows

```
JTextField plainTextField = new JTextField("");
plainTextField.setText("MY SECRET MESSAGE");
plainTextField.addActionListener(new EncryptionListener());
plainTextField.setActionCommand("encrypt");
```

now in a listener that implements the ActionListener, you can have code as follows

```
private class EncryptionListener implements ActionListener
   {
      public void actionPerformed(ActionEvent event)
      {
          if ("encrypt".equals(event.getActionCommand()))
          {
```