ITCS 3166

Midterm Project

For this project, you'll now get to write your own Java Server. Using what you've learned from the Activities, you're going to create two Java classes: A Server class and a Client class. The server will run first, set up a local connection, and wait for the client. The client will then connect to the server and send some data to be processed.

The server will work as a conversion app. The client will ask for the user to enter a temperature in the form "80F" or "16C" to be converted. Note that the client should be able to enter either Fahrenheit or Celsius temperatures. Invalid entries (being anything not in the format "xF" or "xC" where x is any real float value) should be turned down by the client and the program should ask for new entry until the user enters a valid input. The server should then receive the user's entry by any means and convert the given temperature into its counterpart, sending the converted temperature back to the client. The formula for converting temperatures can be found below.

If you're still not sure about Java programming or how to set up the server/client stuff, check the following two links:
https://docs.oracle.com/javase/tutorial/networking/sockets/clientServer.html
http://www.javaworld.com/article/2077322/core-java/core-java-sockets-programming-in-java-a-tutorial.html

When writing this program, here's some things to keep in mind:

- Code written and submitted must be entirely your own. While you may receive help from other students, tutors, TAs, textbooks, or online sources, the code in your project must be written by you and must be your own original code. Copying from any other source will result in a 0 grade and can potentially have further consequences.
- The nature of the app is to collect an entry from the user, send the data to the server, have the server convert the data, send the data back to the client, and display the converted data to the user. The program should run through only once, and both the client and server should terminate once finished. Keep in mind that it's possible for either the server or the client to terminate while the other is sending data – if this happens it'll result in a Connection Reset error. If you encounter this error, make sure that both your classes stay connected beyond sending data.
- For this assignment, the program must be entirely console-based. No GUI applications or stand-alone executables will be accepted for this submission.
- Make sure input is validated before it is sent. If invalid data is entered, the program should repeatedly ask the user for new input until the input entered is valid.

- Make sure you close your IO objects and all your sockets. Leaving them open is a security risk. While the risk is negligible on a local host, it's still good practice to close them when they're done receiving data.
- The server should send information to the console for good debugging to tell the programmer what it's doing during RunTime. Keep in check with when connections are made and when data is sent. Good documentation is also important in this regard.
- This program will require you to compile and run two classes at the same time. Some compilers will not allow this; Eclipse and NetBeans will. If you have a different compiler, check to make sure it will run two main programs simultaneously. If it doesn't, I'd recommend downloading one of the two compilers mentioned; a quick Google search will give you their websites.
- The formulas for converting temperatures are as follows:
    - C = (F – 32) / 1.8
    - F = (C * 1.8) + 32
    - Keep in mind the order of operations!
- Finally, when you've finished your program, zip both the .java files and upload them to Moodle.

Grading Rubric

This assignment is worth 30 points. The point values are graded based on this rubric.

Connectivity	10 points	The server opens a port on a localhost, and the client successfully connects.
User-Input	4 points	The client program successfully obtains user-input, and data validation is done correctly to validate the input obtained. The user is asked for new input if invalid input is given.
User-Friendliness	6 points	The look-and-feel of the program runs well; the client sends good user-friendly output; the server relays information to the console.
App Reliability	5 points	The program correctly sends the user input to the server, the server correctly converts the input, and the server successfully sends the new data back to the client.
Good Practice	5 points	No dead code; no unused imports or variables, IO and Sockets are correctly closed. Good documentation.
Total	30 points	