COMP20081: Systems Software Report

Group 6

Ryan Dennis (N0804540)

Joshua Maccarthy (N0799202)

Jamie Haywood (N0778608)

2020

# Introduction

The aim of this project is to create a Java application to manage a server and a number of weather station clients. The weather stations will send data concerning their current location to the server at regular intervals, and this information will then be processed and displayed on a GUI for users to view and download.

Users will be required to log in before viewing any data and this process will be handled via a username/password system in a text-based database.

# Features, Design and Implementation

## Features

The application will be made up of four primary components:

* A central server that accepts clients and assigns data handlers that will help to process and display any data that is received.
* A potentially unlimited number of weather clients that will generate data packets and send them to the central server at regular intervals once they have made a connection.
* A graphical user interface through which users will interact with the program. It will update in real time to show the most up-to-date information that the server is receiving from its weather station clients. Users will be able to download logs of all information that has been received by the server.
* A login system using a text-based database. Before users can view any information from the server, they must first login using a username and password combination that will be securely stored in a database.

The program will be created entirely in the Java programming language using the Apache NetBeans IDE.

### Connections

Weather station clients will connect to the central server and subsequently send data via the connection-oriented Transmission Control Protocol (TCP).

TCP was chosen over the connectionless User Datagram Protocol (UDP) primarily for its increased reliability. With UDP, each packet that is sent from the weather station would potentially take a different path and could arrive in different orders. There is no guarantee that the packets would arrive correctly or in the desired order. This makes it largely incompatible with this application, which relies on data arriving regularly and in a specific format so that it can be stored and displayed without issue.

On the other hand, TCP is much more secure. Before the weather stations send any data to the server, they establish a single, secure connection that will remain until either the server or weather station is terminated. Each packet is checked for errors and all of the data will arrive at the server in the correct order, due to TCP only sending data along one path.

TCP also allows the server to accept new clients more simply. A connection is made to a main socket on the server by weather stations when they are first created and the server will then move the connection elsewhere so that new clients can be accepted. Conversely, UDP is much more of a free-for-all, and does not scale up to having a vast number of weather stations as cleanly as TCP. For these reasons, all connections in the application will use TCP only.

### Multithreading

Threads form a core part of the application as it is how the server deals with each new weather station client that connects to it.

The server has a central socket with a known port ID. All weather stations connect to this socket when they are first created, and the server itself will be continuously receiving clients using this socket.

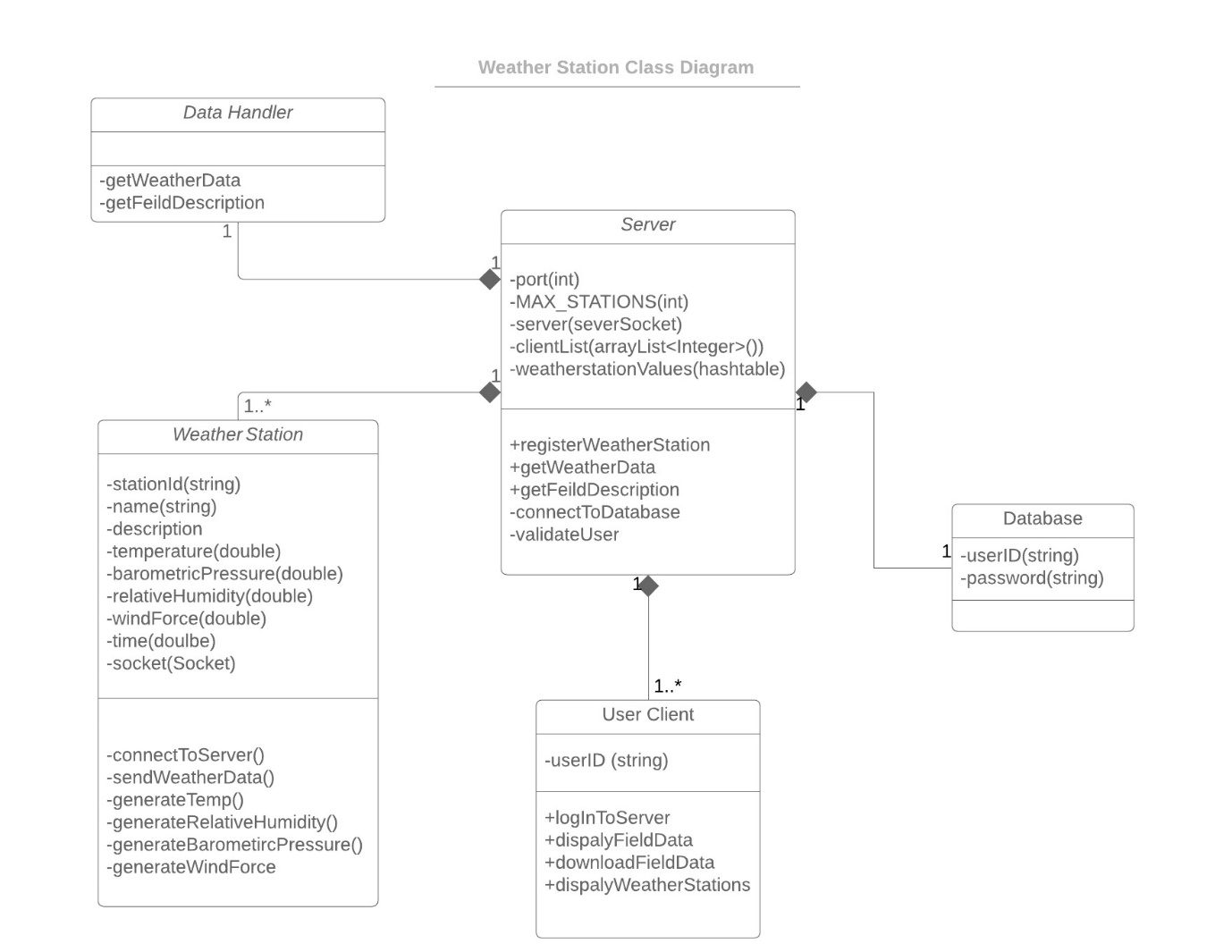
When a connection between the server and a weather station has been made, the multithreading begins. Before the server receives any data, it creates a new data handler, which is assigned to the new client and given a separate thread to run in. This data handler is in charge of recording and processing information that it receives from the weather station it is paired to.

This multithreaded approach allows the server to keep accepting new clients even once it receives a client. If only one thread was used, the server would potentially only be able to accept new clients in larger intervals as it will have to deal with the clients it already has at the same time. This would scale poorly as many clients begin to connect, whereas the multithreading approach can handle a vast number of clients without issue.

## Design

### Class Diagram

The following class diagram was created to illustrate the architecture of the program.

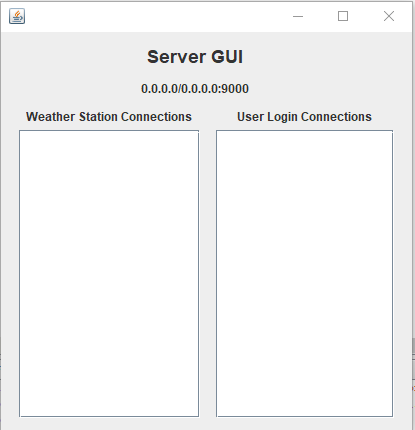


As can be seen in the class diagram, the majority of the program’s functionality comes from the connection between the server and the weather stations. There is only one server maintained throughout the lifespan of the program, but any number of weather stations can connect to it. Additionally, any number of user client interfaces can be created, which query the server to gain access to the weather station’s data in a visual format. Users must provide login credentials to access the GUI, and these usernames and passwords are stored in the text-based login database. Finally, the server has a child class called the Data Handler. An instantiation of this class is allocated to any connecting weather station to handle multithreading.

# Implementation

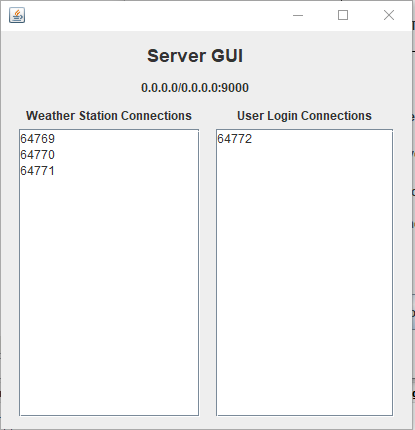
## Software Screenshots

### Server GUI



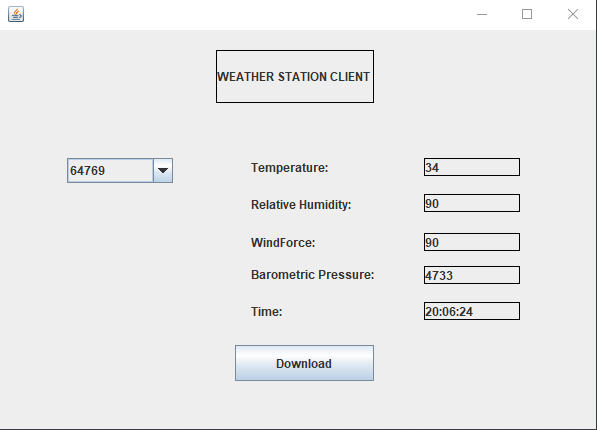
When the server is created, an accompanying interface is opened that displays the port ID and the connections that is has made. These connections are separated into weather stations and user clients.

### Server GUI with Connections



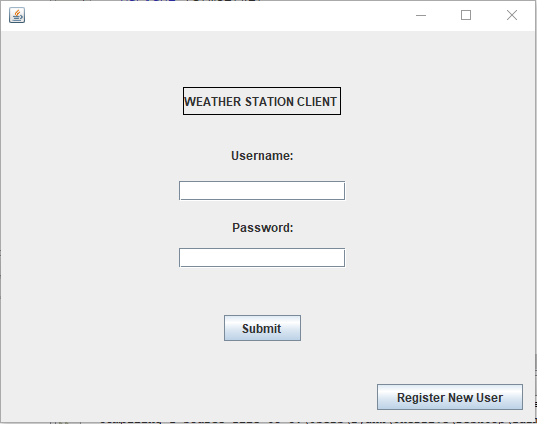
This screenshot shows the same server GUI from before, but this time a number of connections have been made. The listed numbers in both columns are the ID numbers that the clients are allocated as they connect. This GUI is continuously updated as new connections are made.

### User Client View



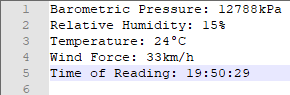
The user client is shown once a user has successfully logged in. Using the drop-down list on the left, the user can select a weather station to view (stations are listed according to their registered ID). The weather values on the right side of the GUI are updated in real time as the server receives new data from the weather station that is being examined.

### User Login Screen



Before users can view any weather station data, they must first login. Consequently, this is the first screen users see when the client is opened. Usernames and their accompanying passwords are stored in a separate text-based database. Once valid credentials have been provided, the user is shown the weather station interface.

### Downloaded Weather Station Data



Through the user client, it is possible to download a set of data from a given weather station. All of the most recent provided data from the weather station in question is saved to a file. Users have the option of specifying where this file is saved through a file browser. Any weather station that is currently connected to the server can have its data saved by a logged-in user.

# Conclusion and Future Work

The solution that has been developed as a result of this project meets all of the aims that it set out to. A central server manages connections with a number of weather clients that generate and send weather data through a secure TCP connection. This data is logged by the server and can be displayed to users via a Graphical User Interface. To access this GUI, users must first input valid login credentials, which are stored in a text-based database. The resulting GUI allows users to view the most up-to-date information from each weather station and download a snapshot of this data if they so choose.

While the current solution satisfies the requirements laid out from the beginning, there is still more that could be added in the future if more time and resources were allocated. One possible improvement that could be made is the option to view the data of multiple weather stations at the same time. Currently, the GUI only allows one weather station to be viewed but allowing multiple stations could allow the user to make comparisons between them.

Another potential area of improvement is data visualisation. In its current state, the weather data provided by the stations is satisfactory, but it is all text-based which could be hard to understand for some users. A possible solution would be to allow for the creation of bar graphs and charts showcasing all of the weather station data in a pleasing, visual format. However, with the current libraries and algorithms employed by the program such a thing would not be possible, hence it is allocated as potential future work.

Extended admin controls are also a possibility. New users can be created, but that is the extent of the management options that the program provides to users. Options to terminate weather stations through the GUI could be one potential option, and the ability to rename station and client IDs to something more human-friendly e.g. a word could be a useful management upgrade.

Overall, the program satisfies the original aims and objectives but also has a good amount of potential to be expanded upon in the future.