Name: Aikaterini Karanasiou

Title: Weather CLI application

(A) Installation, build and execution

I implemented initially two applications on Eclipse.

The first application ("WeatherData") is used in order to retrieve weather data from the Open Meteo API (see [1]) and for storing the weather data in the tables of a database. More specifically, the database includes two tables: AVG_TEMPERAURE and DAILY_TEMPERATURE:
"AVG_TEMPERATURE" maintains the weather temperature of each hour from 01-01-2021 until today and "DAILY_TEMPERATURE" includes the weather temperature of the current day. In addition, weather data on the database are updated every day. For achieving this, I used a timer that is expired every 24 hours. When this application starts running, it starts retrieving and storing data (one time) before the timer starts.

Also, the user runs the second application ("CC" or "WeatherCLI") in order to be informed regarding the weather in Athens. Precisely, user can execute the application using the following arguments:

-display for displaying the current weather temperature in Athens
-compare year|month|week for comparing the current weather in Athens with the average weather temperature of last year (or month or week)
-average number_of_month number_of_year for estimating the average weather temperature of specific month of a year
--help for displaying all the possible commands

I tested the above applications on the Ubuntu 18.04 environment.
The following libraries should be installed:

- sudo apt-get install libjsoncpp-dev
  sudo ln -s /usr/include/jsoncpp/json/ /usr/include/json
- sudo apt-get install libboost-all-dev
- sudo apt-get install libcurl4-gnutls-dev
- sudo apt-get install sqlite3

You can build and execute the application as follows:

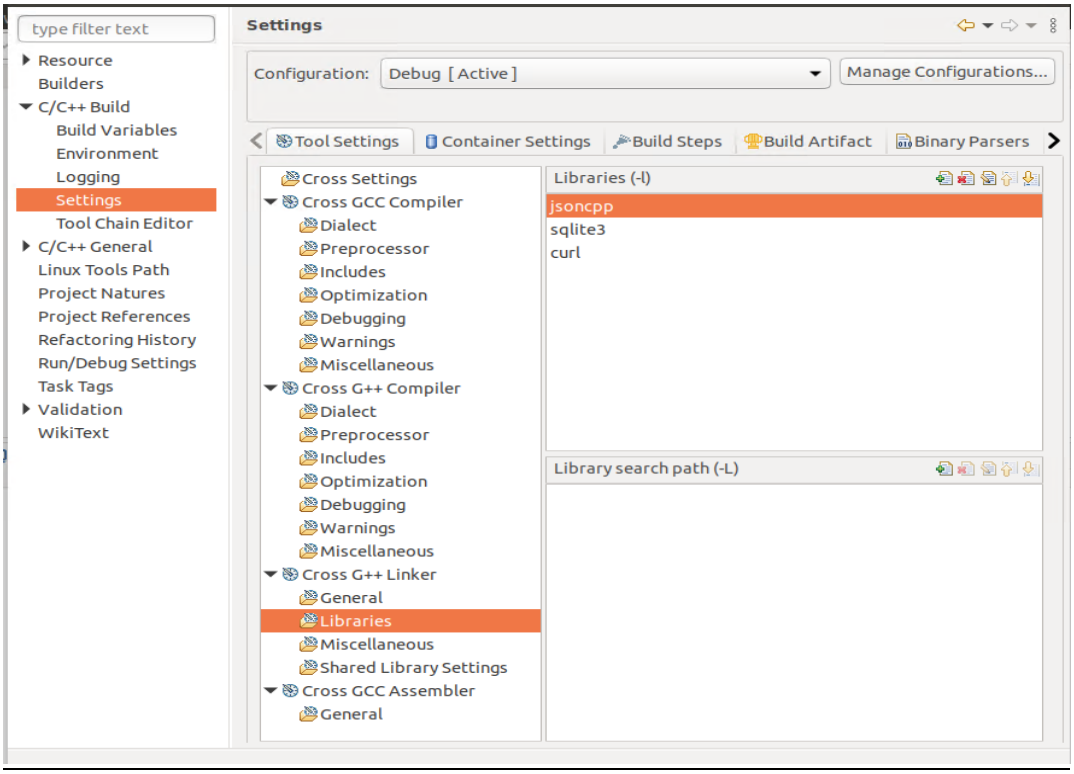
A.1 Using Eclipse

"Eclipse" folder is used for building and executing the applications on the Ubuntu environment with Eclipse.

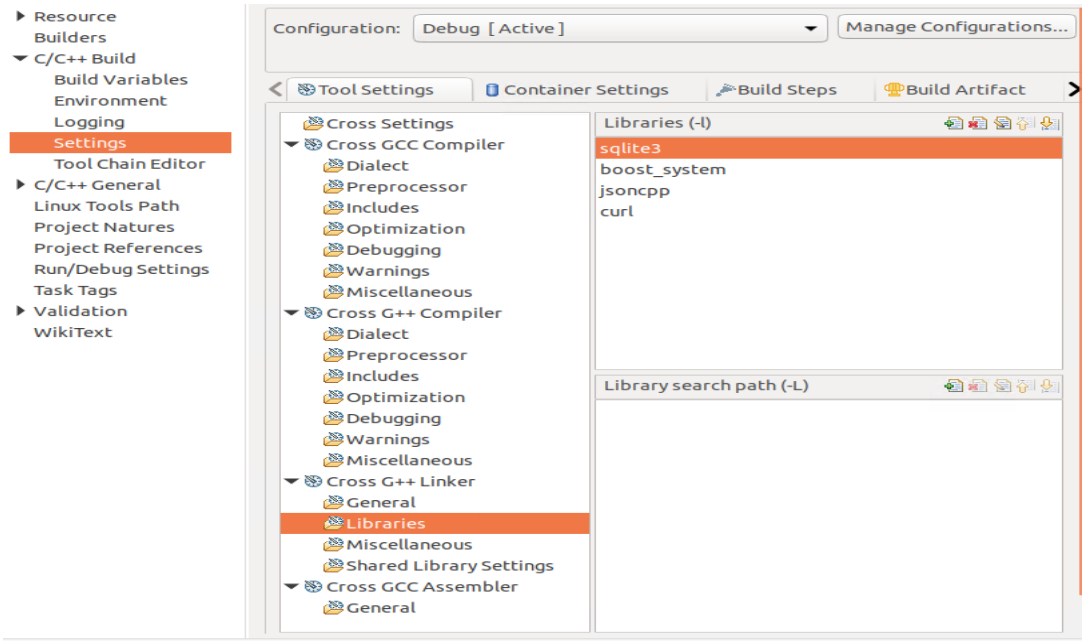
In order to build the applications on Eclipse, you should move "Eclipse/WeatherData", "Eclipse/CC" and "Eclipse/Database" folders on your eclipse workspace.

You have to import also the external libraries for each application under "Properties" option as follows:

“Eclipse/CC” application:



“Eclipse/WeatherData” application



Firstly, you have to execute the “WeatherData” application using the following command:

./eclipse-workspace/WeatherData/Debug/WeatherData.

It takes 5-10 minutes until to retrieve and store the weather data for the first time.

After executing this application, the database will be created under “/eclipse-workspace/Database” folder.

For terminating the application, you should press CLT+C.

Also, you can execute the “CC” application using the following command:

./ eclipse-workspace /CC/Debug/CC

Arguments for “CC” application are mentioned on A paragraph.

A.2 Using Makefiles on Ubuntu environment

“WeatherCLI_app” folder is used for building and executing the applications on the Ubuntu environment without Eclipse.

In order to build the applications on Ubuntu environment, you have to run “make” under

../WeatherCLI/WeatherCLI_app/Build_WeatherData

and

../WeatherCLI/WeatherCLI_app/Build_CC

For executing the applications, you have to use accordingly

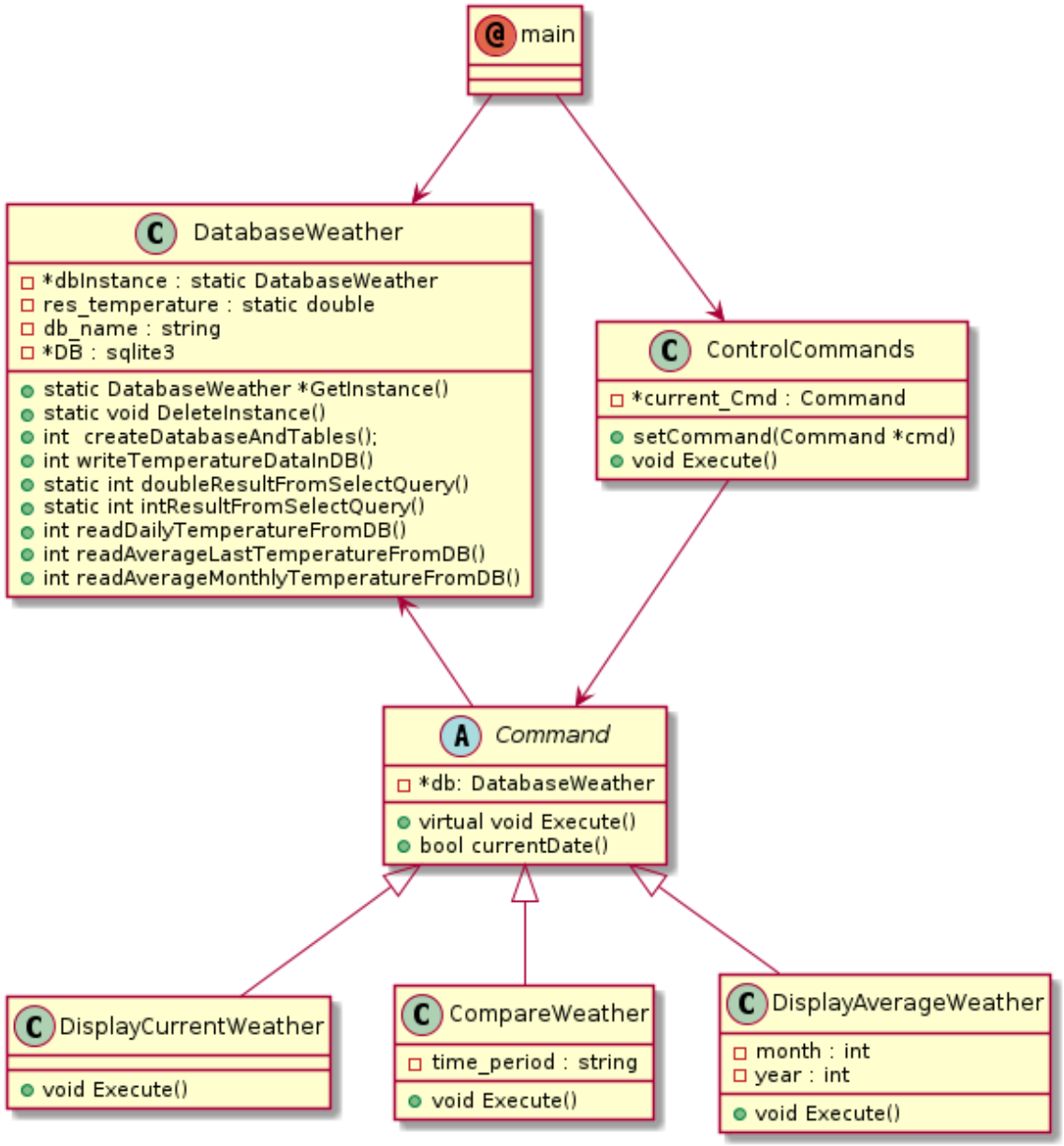
../WeatherCLI/WeatherCLI_app/Build_WeatherData/WeatherData

and

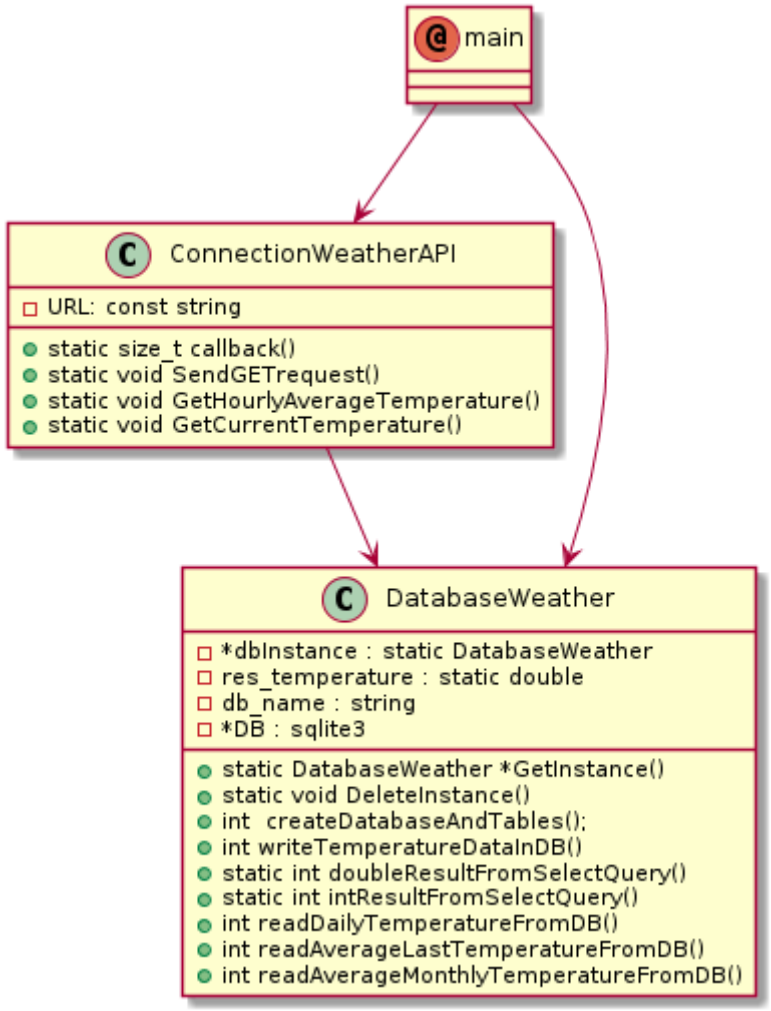
../WeatherCLI/WeatherCLI_app/Build_CC/WeatherCLI

(B) Class Diagrams

“CC” (or “WeatherCLI”) application



"WeatherData" application



(D) Test Examples

D.1 Display Weather

./WeatherCLI –display

```
** readAverageDailyTemperatureFromDB() function called **
Current SELECT SLQ command is: SELECT TEMPERATURE FROM DAILY_TEMPERATURE WHERE YEAR  = 2022 AND MONTH = 9 AND DAY = 18
TEMPERATURE = 30.4
Succeeded to SELECT weather data from the table.
Weather temperature in Athens is: 30.4
```

D.2 Average weather temperature

./WeatherCLI -average 3 2019

```
** readAverageLastTemperatureFromDB() function called **
Current SELECT SLQ command is: SELECT AVG(case when MONTH  = 3 AND YEAR = 2021 then TEMPERATURE end) as AVG_MONTHLY_TEMPERATURE FROM AVG_TEMPERATURE
AVG_MONTHLY_TEMPERATURE = 12.45174731182 8
Succeeded to SELECT weather data from the table.
Average Weather temperature in Athens on Year = 2021 and Month = 3 is 12.4517
```

D.3 Compare weather temperature

./WeatherCLI -compare week

```
** readAverageLastTemperatureFromDB() function called **
Current SELECT SLQ command is: SELECT COUNT(*) FROM AVG_TEMPERATURE
COUNT(*) = 15024
Succeeded to SELECT weather data from the table.
Current SELECT SLQ command is: SELECT AVG(TEMPERATURE) FROM (SELECT TEMPERATURE FROM AVG_TEMPERATURE LIMIT 168 OFFSET 14856)
AVG(TEMPERATURE) = 13.9904761904762
Succeeded to SELECT weather data from the table.
** readAverageDailyTemperatureFromDB() function called **
Current SELECT SLQ command is: SELECT TEMPERATURE FROM DAILY_TEMPERATURE WHERE YEAR  = 2022 AND MONTH = 9 AND DAY = 18
TEMPERATURE = 30.4
Succeeded to SELECT weather data from the table.
Current Weather temperature in Athens (30.4 Celsius) is higher than the average temperature of last week (13.9905 Celsius)
```

./WeatherCLI -compare month

```
** readAverageLastTemperatureFromDB() function called **
Current SELECT SLQ command is: SELECT COUNT(*) FROM AVG_TEMPERATURE
COUNT(*) = 15024
Succeeded to SELECT weather data from the table.
Current SELECT SLQ command is: SELECT AVG(TEMPERATURE) FROM (SELECT TEMPERATURE FROM AVG_TEMPERATURE LIMIT 720 OFFSET 14304)
AVG(TEMPERATURE) = 17.4918055555555
Succeeded to SELECT weather data from the table.
** readAverageDailyTemperatureFromDB() function called **
Current SELECT SLQ command is: SELECT TEMPERATURE FROM DAILY_TEMPERATURE WHERE YEAR  = 2022 AND MONTH = 9 AND DAY = 18
TEMPERATURE = 30.4
Succeeded to SELECT weather data from the table.
Current Weather temperature in Athens (30.4 Celsius) is higher than the average temperature of last month (17.4918 Celsius)
```

./WeatherCLI -compare year

```
** readAverageLastTemperatureFromDB() function called **
Current SELECT SLQ command is: SELECT COUNT(*) FROM AVG_TEMPERATURE
COUNT(*) = 15024
Succeeded to SELECT weather data from the table.
Current SELECT SLQ command is: SELECT AVG(TEMPERATURE) FROM (SELECT TEMPERATURE FROM AVG_TEMPERATURE LIMIT 8760 OFFSET 6264)
AVG(TEMPERATURE) = 17.9189497716894
Succeeded to SELECT weather data from the table.
** readAverageDailyTemperatureFromDB() function called **
Current SELECT SLQ command is: SELECT TEMPERATURE FROM DAILY_TEMPERATURE WHERE YEAR  = 2022 AND MONTH = 9 AND DAY = 18
TEMPERATURE = 30.4
Succeeded to SELECT weather data from the table.
Current Weather temperature in Athens (30.4 Celsius) is higher than the average temperature of last year (17.9189 Celsius)
```

(F) Future Work (Improvements)

According to my implementation, there are two applications where one of them is writing on a database and the second application is reading from the database. Due to this, it is possible a read and write processes to access the database simultaneously and then, one of them is failing.

It seems that SQLite library provides a solution for this issue. Precisely, it can maintain a WAL file where it writes temporarily on it the data that will be committed (see [2] for additional information).

So, I execute the *PRAGMA journal_mode = WAL* sql command (on my code) in order to enable the creation of the WAL file. Also, using this command, more than one applications can have access to the same database on the same time. This command is already added on my solution.

"WAL" value gives access to the database for one database connection and due to this, I should execute one more sql command (before WAL), which is the following: *PRAGMA locking_mode = EXCLUSIVE*.
This command means that only one database connection is used for all the sql commands.

Unfortunately, I observed on my solution that only the journal_mode changed.

In conclusion, using the journal_mode = WAL parameter I observed that several read and write parallel processes are executed correctly. In case of failure, only the read command is failing.

I think that one solution is to use a shared memory that will be used both by the WeatherCLI (or CC) and WeatherData applications. In this case, the first app. could write the data both on the database and shared memory and the other app. can read only from the shared memory.
One example of shared memory is mentioned on [5] link.

(G) Links

[1] Open Meteo API - https://open-meteo.com/en

[2] SQLite - https://www.sqlite.org/index.html

[3] JSONCPP - https://github.com/open-source-parsers/jsoncpp

[4] CURL – https://curl.se/

[5] BOOST Shared Memory -
https://www.boost.org/doc/libs/1_54_0/doc/html/interprocess/quick_guide.html