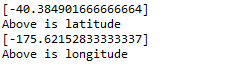
# Implementation of location of individual spectra

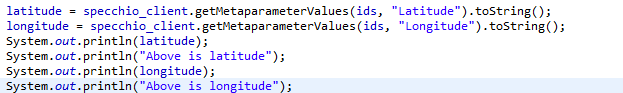
One of the main functional requirements of the dissertation was the ability to grab the location data for a single spectra and export the information into an external mapping software, this came with a set of problems and implementation issues that needed to be worked out. The design choice for this requirement is that the location can be passed through a single class and with the correct method calls open a new browser window that displays location of given spectra.

## Attaining the location

In order to get the location of the individual spectra first the correct spectra had to be attained through the graphical user interface of the SPECCHIO system. Within the current implementation of the spectra data browser there is the ability to check the state of the files that are currently highlighted by the user, this would in turn give a local array list of current ID’s of spectra that are selected by the user.   
The implemented code takes the current id selected by the user and runs a matlab method to return both the latitude and longitude for the spectra with that given id. This matlab method returns a MatLabAdaptedArrayList which contains the locations for any spectra selected with the given id’s.   
This matlab method would have to be run twice once for the latitude and once for the longitude to get the correct location data.

## Getting the correct string for the location

After attaining the location of the given spectra it was then required that it be converted to a string in order to be inserted into a url that would later be passed onto the web browser.   
The chosen way of parsing the correct String was to run javas toString method on the MatLabAdaptedArrayList. This would have to be run twice for both the latitude and longitude.   
This would return the results below.

Taken with a print ln function for ease of showing raw data output

The output contained the useful information required about the given spectra but the string would have to be altered to remove the brackets that contained the information.  
In order to complete this the java method of substring was called to remove both the first and last parts to each string to get the desired results. To ensure that a latitude and longitude of any length would always be consistent with MatLabs output the string was checked for length before removing the end of the string.   
These strings are then saved in their local variable equivalents of their desired output and are then passed through to a new class called maps processing.

Implementing with the current implementation of the SPCCHIO application

A new class had to be created that was named MapsProcessing this class exists to deal with the backend logic issues that are required for dealing with the processing of the latitude and longitude as well as opening the new browser window. The other reason for having this new class is to strive for consistent design layout throughout the program as the QueryBuilder class is strictly front end graphical user interface along with the threads used to control any logic from the backend. So the maps processing class was created and placed in the proc\_modules package of the SPECCHIO application.

To keep follow the design decisions laid out by the original authors of the SPECCHIO application it was decided upon that the implementation of the location would be called through the QueryBuilder and threads would be used to ensure the program can continue if any long standing processes are found.

Thread usage

The current implementation of the SPECCHIO application for the graphical user interface extends the subclass of thread. This allows the program to continue working should a thread not work, obviously checks will be in place if a thread does not complete but this allows the program to not hang entirely. The design decision here is clear in that the system has to deal with a lot of graphical renderings and long running SQL processes.

The location code although not as CPU intensive can cause hangs as it has an SQL check against the identification codes that the spectra data uses within the database so the choice was made to continue using threads to implement the location checks.

The thread is simple and calls for the thread to begin when a button is clicked within the GUI. This thread then gets the location data based on the current ids and passes them to a new instance of the maps processing class so the data can be manipulated and a new browser can be opened.

Refactoring code for good design

Upon initial creation the logic based problems for the code were inserted into the thread, although this did initially serve as a good basis to ensure that the location data was being collected correctly implementation could not continue this way as the application has a clear design for both the front and back end.   
All logic would be passed through to the Maps processing class to be dealt with and would in turn be called by the thread.

Implementation issues  
  
When implementing the location based spectra collection there were a series of challenges that had to be overcome to achieve the correct data that would later be used, bulleted below are a series of issues that had to be dealt with.

* Null spectra locations from Matlabs getMetaparameterValues method
* Multiple spectra per toString
* Compatibility issues with different operating systems

### Null spectra locations

Within the given spectra the location data may not be present but this does not return a null when the toString method is called on the getMetaparametesValues, this problem has been dealt with through a simple method that checks the string for “[]” which is what would be returned if the location data is null.

### Multiple spectra per toString

The implementation that already exists within the SPECCHIO application has already checked for current id’s that are selected by the graphical user interface. It was decided that instead of implementing addition checks on identity that a button would be created that’s state would only be set to enabled if only one spectra was selected.

### Compatibility issues with different operating systems

The ability to open a new web browser has issues within different operating systems but within java there is a desktop class that allows for checking of application that run natively within the current setup. This allows for the users default browser to be opened with the url that is defined with the location data.

## Addition of Javascript for additional points

Spectral information collection techniques include the ability to mount a spectrometer to the back of moving vehicles, this will give a range of different locations and it was explained that additional functionality would be required in order to map multiple points on google maps.

When using a single point it was possible to simply insert the latitude and longitude into a default URL string which would could be inserted singularly into a new instance of an internet browser.  
When dealing with multiple points a simple URL would not be able to handle multiple latitude and longitude values.

After researching different implementation techniques it was decided upon that the importing and running of java script through the java application would be the best solution to multiple point mapping.

//TODO FINISH THIS

## Advantages for users

The clear advantage of this functional requirement is the ability to get an exact photo and map location of where spectral data is collected, with this knowledge   
  
What could change   
The application currently only exports to google maps, this was a personal preference and the project could easily expand to incorporate apple maps and even bing maps. This preference could be chosen by the user and even saved to their individual account.   
The other main drive to use google maps was the general usage statistics in comparison to their direct web based competitor bing.   
<http://trends.builtwith.com/mapping/Bing-Maps>  
http://trends.builtwith.com/mapping/Google-Maps