|  |
| --- |
| MSC SOFTWARE DEVELOPMMENT |
| Advanced Object Technology |
| Assignment 2 |
|  |
| **Jimmy Collins – R00145569 – Jimmy.Collins@mycit.ie** |
| **5/2/2017** |

|  |
| --- |
|  |

Contents

[Section 1 – Introduction 1](#_Toc481151193)

[Section 2 – Demonstrated Usage of Defined Object Technologies 6](#_Toc481151194)

[Template Pattern 6](#_Toc481151195)

[Proxy Pattern - Remote Proxy 7](#_Toc481151196)

[Usage of Database for Data Persistence 8](#_Toc481151197)

[Façade Pattern 10](#_Toc481151198)

[Command Pattern 11](#_Toc481151199)

[MVC Compound Pattern 12](#_Toc481151200)

[Section 3 – System UML Diagram 13](#_Toc481151201)

[Section 4 – Screen Captures of Most Impressive Code Snippets 14](#_Toc481151202)

[Example of Integration with DShield API 14](#_Toc481151203)

[Usage of Apache POI Library to Generate Excel Output 15](#_Toc481151204)

[Getting the Top Elements in an ArrayList 16](#_Toc481151205)

[Section 5 – Evaluation of Work 17](#_Toc481151206)

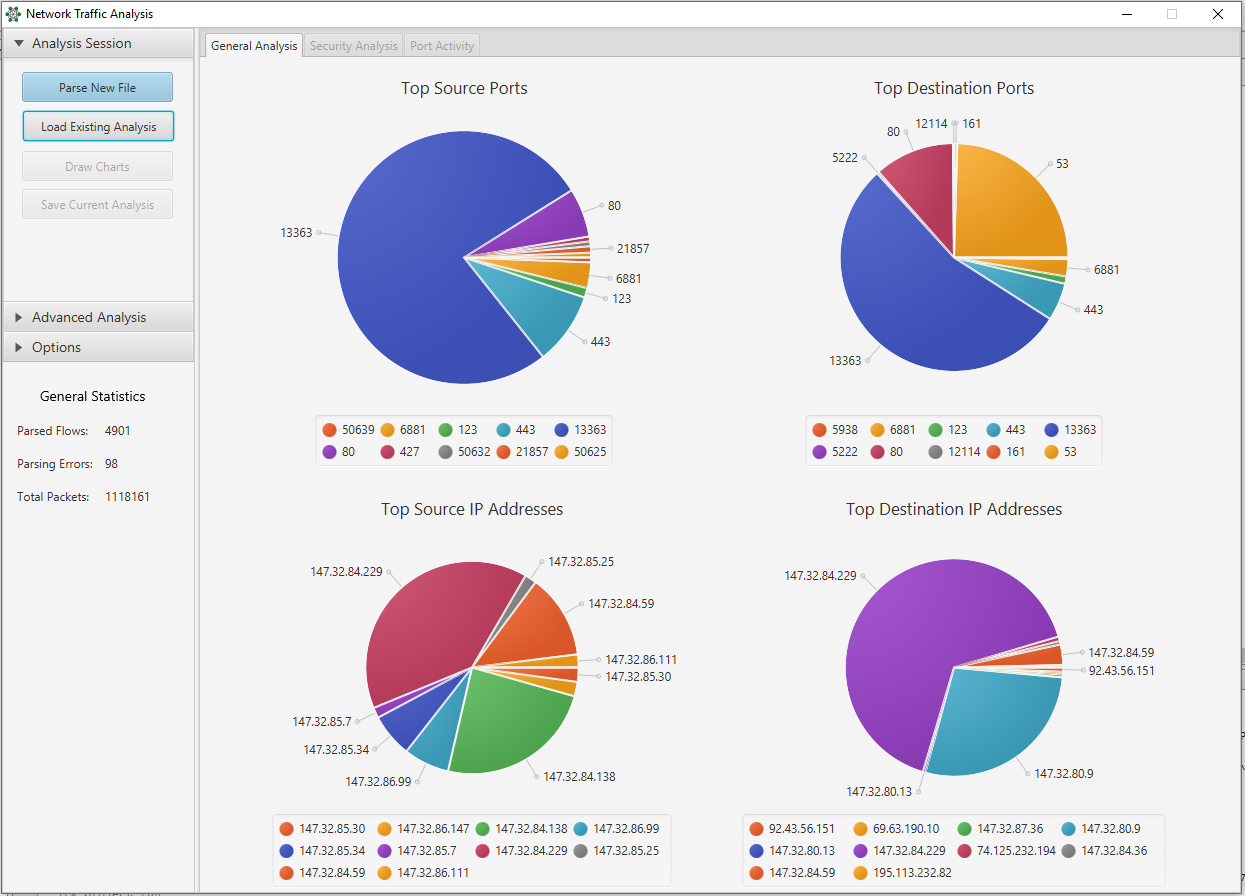
[Appendix 1 – Database Setup Instructions 18](#_Toc481151207)

[Appendix 2 – Possible Development Environment Certificate Issue 19](#_Toc481151208)

## Section 1 – Introduction

Before starting to develop the requirements for Assignment 2, I had some improvements I wanted to make on the Assignment 1 codebase:

* User interface changes to split out general / advanced / settings sections of the user interface, I completed this by using an Accordion control to separate out the options for each section and by consolidating the tabs in the user interface from Assignment 1 into one tab called ‘General Analysis’.



*Figure 1: Main User Interface for Assignment 2*

* Secondly, there were some changes I wanted to make to my strategy pattern implementation from Assignment 1. In the initial code base, each Strategy object was returning a Tab control meaning it was tightly coupled to that UI element. I modified it so it just returns a Chart that can then be added to any UI control (e.g. Tab, HBox etc.)
* Finally, I completed some general code refactoring (removed some duplicate code and other small items I missed in Assignment 1).

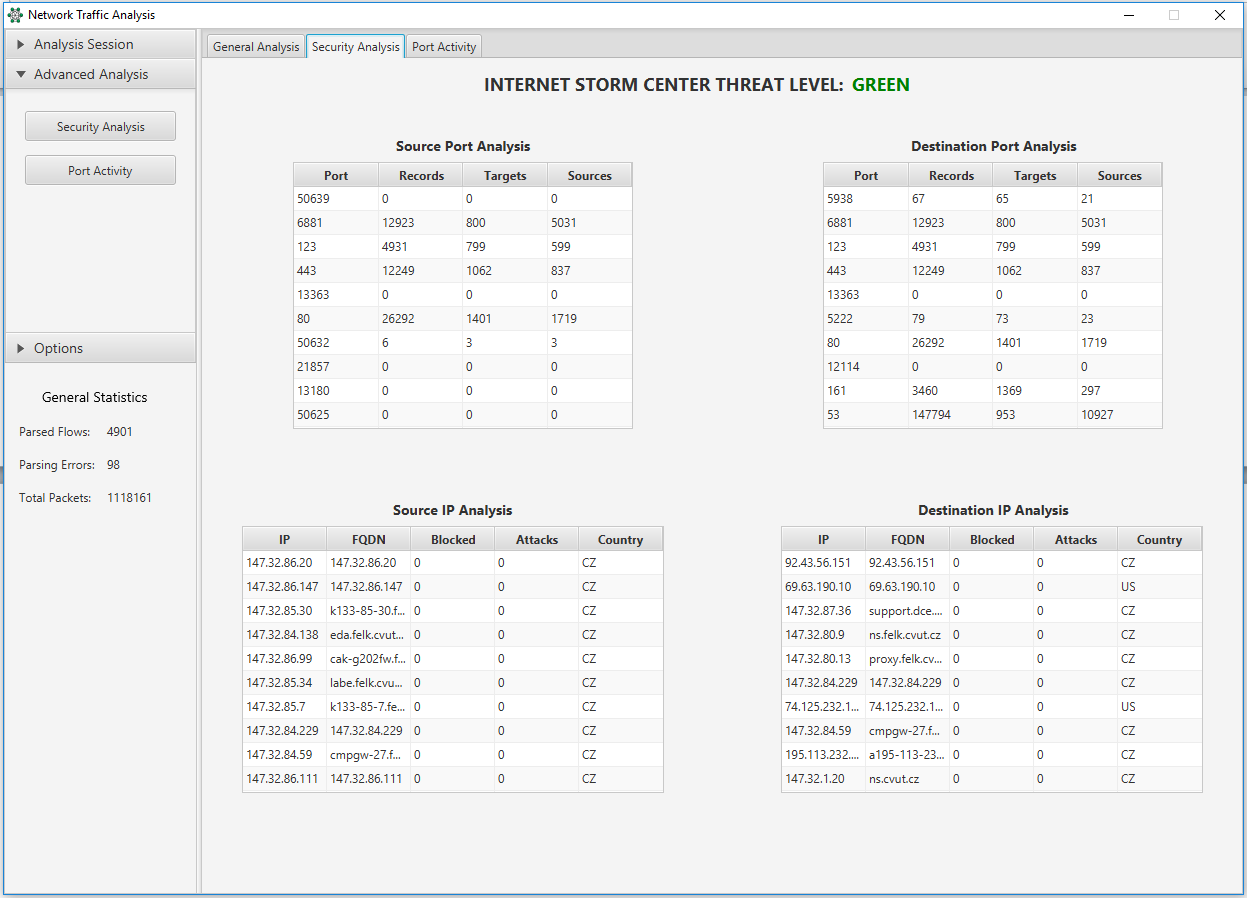
Regarding the Assignment 2 requirements, Section 2 contains details on how each requirement is implemented within the application.

For the data drill down requirements, I decided to offer two main pieces of functionality, both featured around security, and using data from an external API:

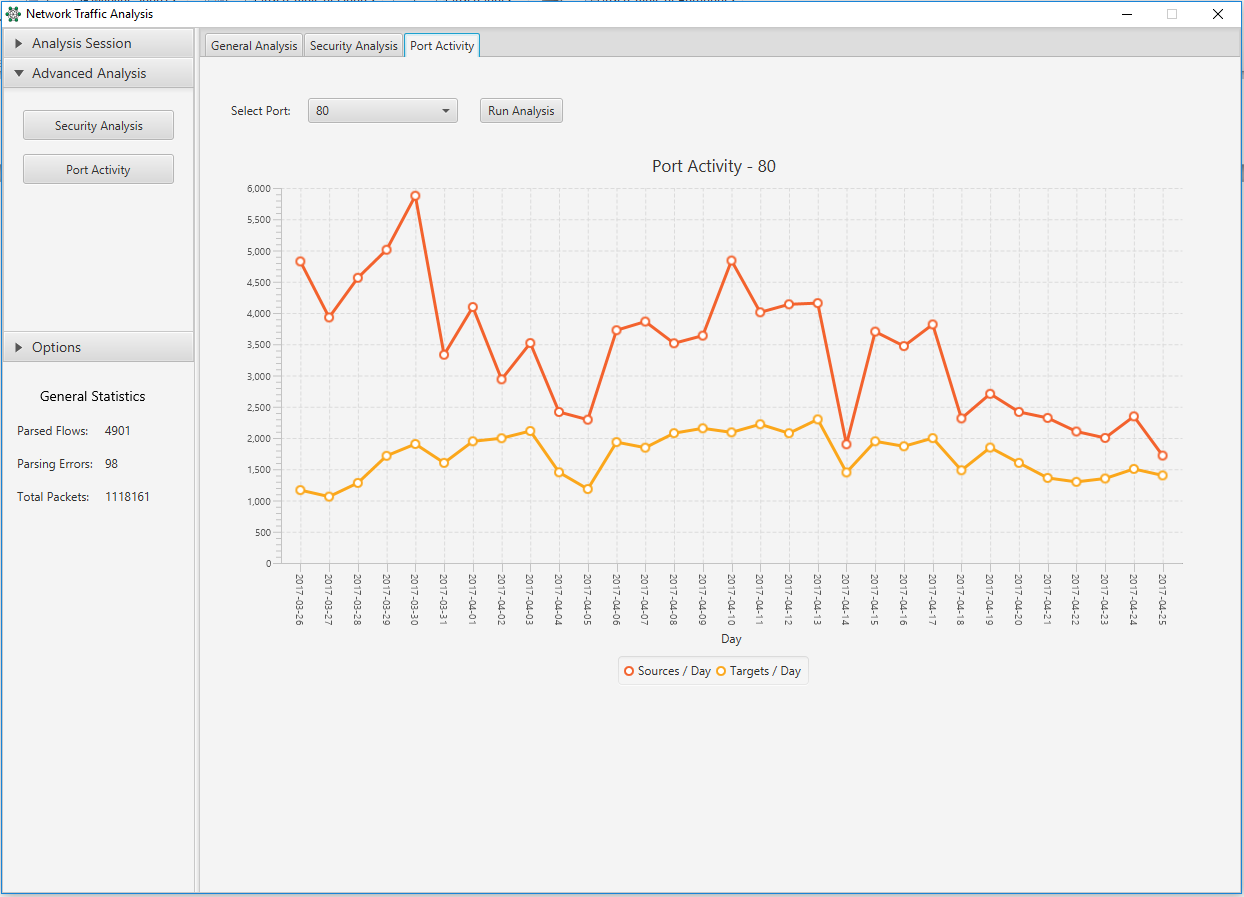
* A security analysis (seen in *Figure 2*) of the top source/destination port numbers and IP addresses. This analysis shows some indicators in the user interface that could allow the user to determine if the IP/port was recently used in an attack or for example if a certain IP has been the source or destination of a lot of malicious traffic.
* For further drill down (seen in *Figure 3*), an analysis can be generated for any source/destination port that outlines how often those ports have been the sources or destinations of attacks over the past 30 days.

The data source for my security related features is the DShield API[[1]](#footnote-1) provided as a free resource by the Internet Storm Center (ISC)[[2]](#footnote-2), part of the SANS Institute. I looked at a number of API’s such as this and the Google Safe Browsing API. I decided to go with the ISC API due to the fact that it is an open, simple API requiring no authentication logic (due to the time constraints of the project).

I also display the current threat level from the ISC in the security analysis user interface.



*Figure 2: Security Analysis UI*

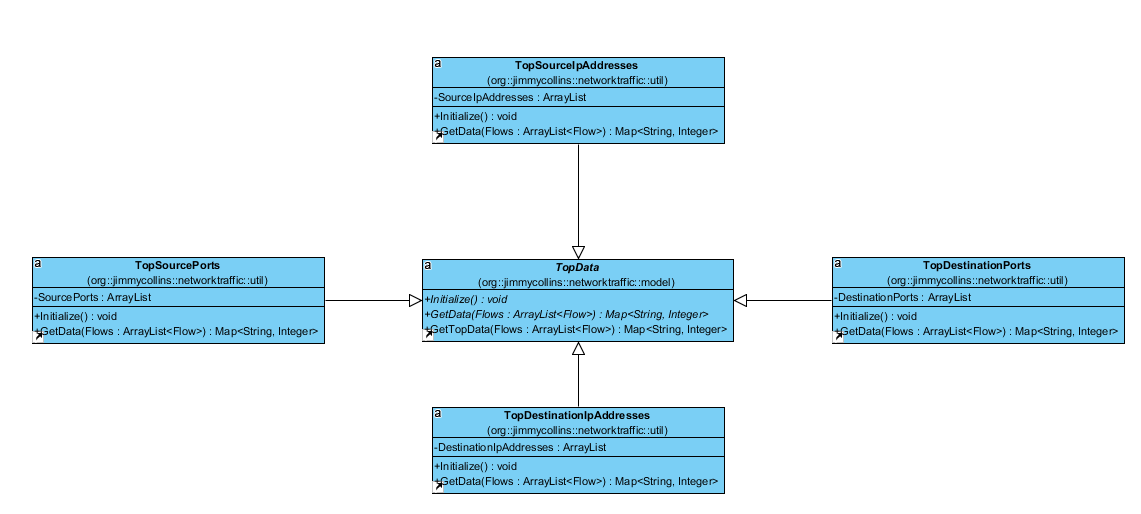


*Figure 3: Port Activity Analysis UI*

## Section 2 – Demonstrated Usage of Defined Object Technologies

### Template Pattern

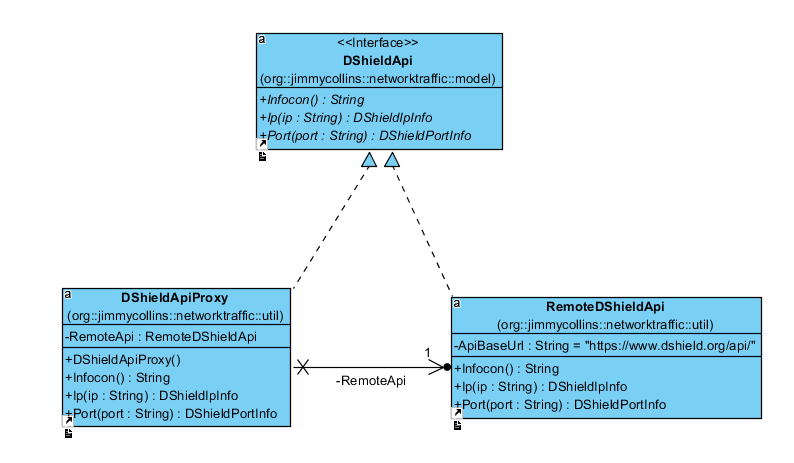
I used the template pattern to refactor some code from Assignment 1, which is the logic that is used to generate the top source/destination ports and top source/destination IP address data from the data parsed. Now the TopData class exists, which is an abstract class with the final method GetTopData, and the helper methods Initialize and GetData which are overridden in each of the concrete classes.



### Proxy Pattern - Remote Proxy

The usage of a remote proxy is demonstrated in UML below. In my application I integrate with the DShield[[3]](#footnote-3) API maintained by the Internet Storm Center (ISC). I decided to approach some of the advanced analysis requirements from a security perspective, so I use this API to return some security information on the top IP addresses and ports that are being parsed from the Netflow file.

**Please see Appendix 2 for the solution to a possible certificate issue that *may occur* the first time you run the application and attempt to connect to the API.**

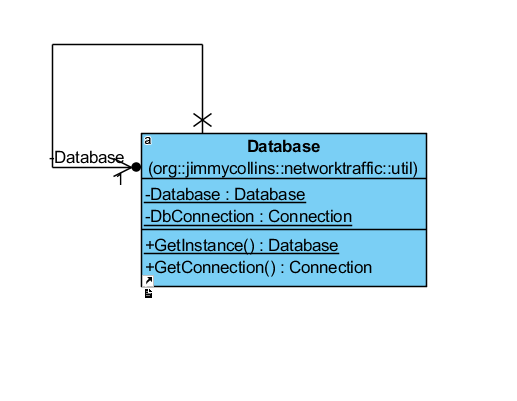


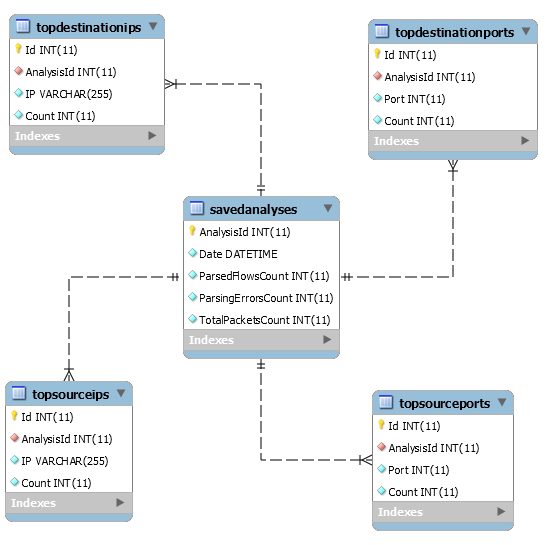
### Usage of Database for Data Persistence

Within the application the user has the ability to save an analysis session to the database. This saves the Top Source/Destination Port and Top Source/Destination IP data to a MySQL database (see **Appendix 1** for the setup details). I consciously chose not to save the data in the ‘Security Analysis’ section, as this data is something that a user would want to check each time an analysis is run, as potential threats are constantly evolving.

The database schema is shown *on the next page*.

Below is shown the class I use to handle the connection to the database, which implements the Singleton Design Pattern.



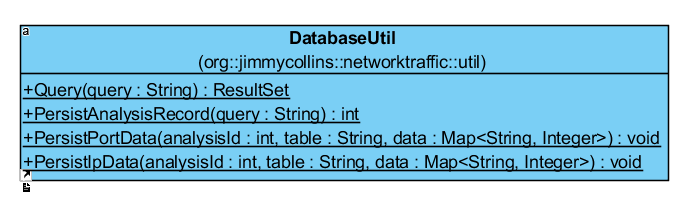


### Façade Pattern

I used the Façade Pattern to refactor my database querying functionality after it had been written, the code was quite complex before this.

I ended up creating a new class, DatabaseUtil, which is a wrapper that contains a set of members that are used for database operations that are easily understood and simple to use.

These members access the database on behalf of the facade user, hiding the implementation details.

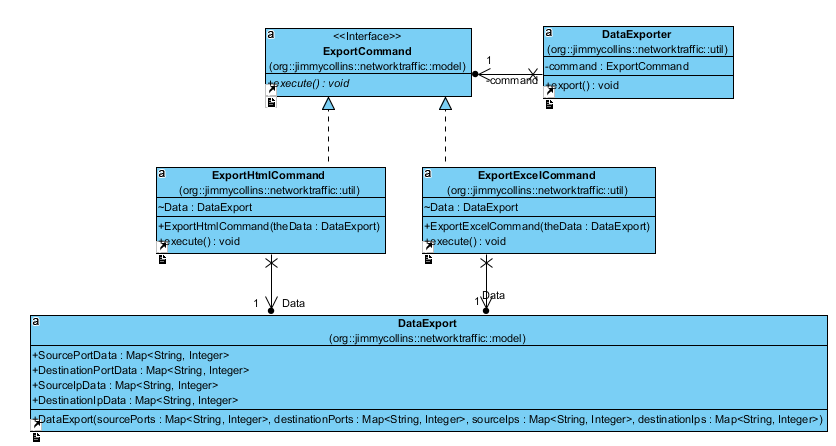


### Command Pattern

I used the command pattern for the functionality that allows a user to export an analysis to different formats (e.g. to email to someone). Currently Microsoft Excel and HTML formats are supported as export formats. I used the Apache POI[[4]](#footnote-4) API to create the Excel document.

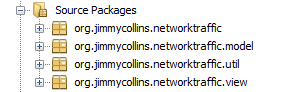
ExportCommand is the interface which both concrete commands ExportExcelCommand and ExportHtmlCommand implement. DataExporter is the receiver and the main UI controller class is the invoker. The DataExport class is just a class used to pass the relevant data to be exported to the constructors of the concrete commands.

**Notes** – adding a new export type could be facilitated easily in this design. I’m also aware that a Strategy Pattern could have been used here to achieve the same functionality.



### MVC Compound Pattern

I am using the Model-View-Controller pattern within my application.



* The application user is interacting with my main user interface (i.e. the View) contained in the org.jimmycollins.networktraffic.view package.
* The action taken by a user is relayed to the Controller for the View.
* The Controller makes requests to Models and associated helper classes to perform the action, and a resulting action is relayed back to the view.

Within the MVC Pattern, I am also using other patterns such as the Strategy Pattern in order to allow the user to perform actions on the user interface (changing the chart types at runtime in the case of the Strategy Pattern).

## Section 3 – System UML Diagram

The UML diagram was too large to reproduce here and maintain readability, so it’s included in the accompanying A1 sized PDF file.

## Section 4 – Screen Captures of Most Impressive Code Snippets

### Example of Integration with DShield API[[5]](#footnote-5)

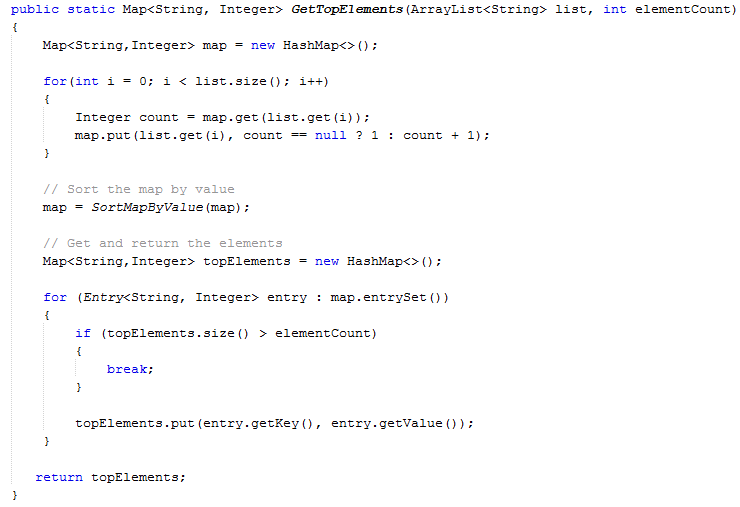


### Usage of Apache POI[[6]](#footnote-6) Library to Generate Excel Output

(*Example truncated due to length – see ExportExcelCommand.java for full source*)



### Getting the Top Elements in an ArrayList



## Section 5 – Evaluation of Work

I believe that overall I have met the requirements set out in terms of using the defined object technologies. Given some more time I would like to have completed some more items such as:

* Styling of the User Interface using CSS – I had planned on creating some CSS to style the UI and gain some exposure to how it’s done in JavaFX, but I didn’t get to it due to the priority of getting the features implemented.
* In the current application, I made the decision to consciously not save the security analysis data to the database as I believe it’s something a user would want to run fresh each time. Given more time, I would have given the user at least the option to save that data also, and also the ability to export it to HTML/MS Excel along with the high level data.
* There are some obvious error conditions that I know are not handled – e.g. what if there is no internet connection and we can’t contact the DShield API? Right now the user would be alerted with the stack trace, but it could be more graceful.
* I would include more in-context help in the form of tooltips and labels for users, especially around the security features so that they know exactly what the data means.
* Finally, I would do a more thorough investigation of the API’s available for port/IP analysis in terms of efficacy and usefulness. The Google Safe Browsing API[[7]](#footnote-7) looked good, but had a steeper learning curve than the DShield API that I ended up using.

## Appendix 1 – Database Setup Instructions

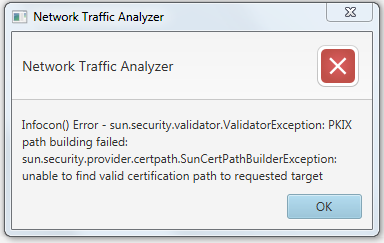
The below steps should be taken to create the necessary schema in MySQL in order for the application to function correctly.

This is used to save the analysis data.

1. Open MySQL Workbench and create a new connection.
2. Open schema-no-data.sql from the db folder in the source tree.
3. Execute this SQL to create the necessary tables and relationships.

## Appendix 2 – Possible Development Environment Certificate Issue

The following issue may occur when you try to run the ‘Security Analysis’ feature under the ‘Advanced Analysis’ menu. I’ve seen this happen on 1/3 machines I’ve been using to develop my project.



This appears to be because the API server has a self-signed certificate.

To work around this issue and get the application running on your local environment, you can add this certificate to the list of trusted certificates of your JVM.

If you see this issue on your environment, please follow the below steps to fix it.

* Go to [www.dshield.org](http://www.dshield.org) and export the certificate. This can be accomplished (using Chrome) as follows:
  + Navigate to [www.dshield.org](http://www.dshield.org) at least once.
  + Next go to chrome://settings/, show advanced settings and click on ‘Manage Certificates…’.
  + In the dialog find the below certificate, and use the export option to export it locally on your machine as a .CER file.



* Next, you need to add this to the list of trusted certificates of your JVM. Go to your JRE\_HOME/bin or JDK/JRE/bin
* Run the below command (substituting the path to your JDK directory and the path to the certification downloaded above):



* If promoted for a password, use the default key-store password ‘changeit’.
* When prompted to trusted the certificate, type ‘yes’.

The application should now work as expected. I’ve successfully tested this procedure.

1. <https://isc.sans.edu/api/> [↑](#footnote-ref-1)
2. <https://isc.sans.edu/> [↑](#footnote-ref-2)
3. <https://www.dshield.org> [↑](#footnote-ref-3)
4. <https://poi.apache.org/> [↑](#footnote-ref-4)
5. <https://www.dshield.org/api/> [↑](#footnote-ref-5)
6. <http://poi.apache.org/> [↑](#footnote-ref-6)
7. <https://developers.google.com/safe-browsing/> [↑](#footnote-ref-7)