

Introduction to Programming

Level 4

Programming Portfolio Report

[S256337]

[2129 Words]

# Background

The main goal of the project was to create a database that stores information on recently spotted whales in different areas, this includes being able to navigate through a menu with multiple options, such as ADD, REMOVE, SEARCH, VIEW ALL, SAVE and lastly CHANGE. Each of the menu options having the ability to navigate back to the original state.

The database stores different details on the whales including the name of the species, the quantity of whales that were seen in the area and the location they were spotted in, displayed within the console database and a created text file.

A separate text file that logs the activity of the database will be created upon starting the programme.

# 2.0 Implementation details

**The “main” Method**

The “main” method is where the ArrayLists are created when the application is run, this helped during early testing with console data formatting as variables could be added manually using the “.add” method and would already exist within the database on creation. The reason for using Arraylists in the project is because unlike arrays ArrayLists are resizable, meaning that elements can be added or removed, a key part to the database, see figure 1.

The ArrayLists created include:

* Whales contains the species stored as a String.
* whalesLog contains a line of text on changes made to the database stored as a String.
* QTY contains the number of spotted whales in the given area stored as a String.
* Location contains the geographical location of where the whale was spotted stored as a String value.

Text

Description automatically generated

Figure

The idea behind using the “main” method in development is to use it as a base to load other methods, this means that as soon as the run button is pressed a series of methods then leading to the main menu “Home” method is loaded, when the user wants to go back to the “Home” menu it would save re-running methods that were only required on start-up. The methods that run include “createFile”, “Save” and then finally “Home”, each importing all the different ArrayLists along with the stored integer listLength. Figure 2 shows the methods and the Arraylists that are being carried over when called.

Text

Description automatically generated

Figure

**The “home” Method**

The “Home” method contains the displaying of menu options for the user to choose between, this works by printing options that the user can choose between and then importing a scanner input recognisable by “if” and “else if” statements, this in turn leads to the appropriate method being loaded based off user input shown in figure 4. Figure 3 shows how the menu items are displayed in the console directly when the application is run. User input can be typed underneath the menu options to make a selection in either capital letters or lowercase, this was done using the method “.equalsIgnoreCase()”. User keyboard input is done through importing the Java utility Scanner and then named as keyboardInput, this is then saved to a variable called ID which can be referenced within the “if” or “else if” statements.

Text

Description automatically generated

Figure

Text

Description automatically generated

Figure

In case the user enters anything other than a menu option then the method uses an “else” statement that covers everything outside the other statement parameters to give the user an error message which also then re-runs the “Home”, taking the user back to the original state. This shows in the console as in figure 5.

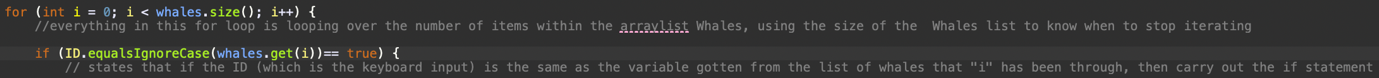
Text

Description automatically generated

Figure

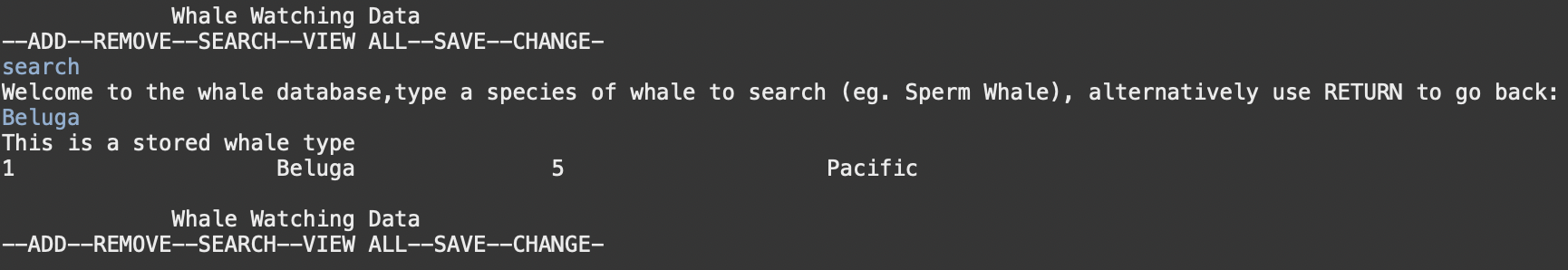
**The “change”, “remove” and “search” Methods**

Searching through the database proved difficult when creating the project, the user needs to see if the typed input matches one of the stored Whale types in the database. When searching through a list for a specific item the choice within the project was to use a “for” loop, this needed to iterate through each individual stored Whale name and then stop on the specific one that had been searched, finally giving the user an answer to whether it is stored in the database through “if” and “else if” statements. The line of code in figure 6 is how the application can take a user input, store it as a string, then compare it to every item stored within the whales ArrayList. Using a method within a method was the biggest challenge of the project and as seen within the “remove” and “change” method there were a few different ways to achieve the same outcome (see figure 9).



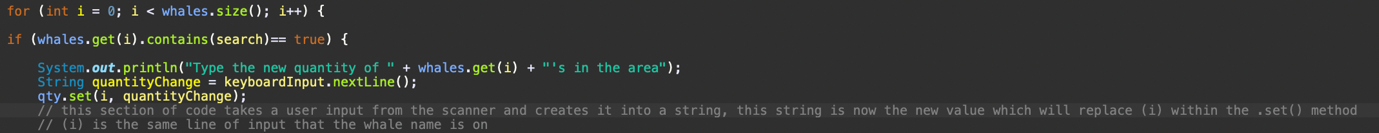
Figure

Once the requirements of the “if” statement are met the user is then told whether the item is stored within the database and shown the entry, the user is then taken back to the ”home” method as shown in figure 7.



Figure

The “change” method uses the same searching loop as the “search” method however presents the user with a question on what they want the quantity to be changed to. This method works off using the set() method and is able to change the first input (i) into a new input that the user enters as seen in figure 8.

****

Figure

The “remove” method works in the exact way as the method to change the quantity, however instead of using the set() method, it uses the remove() to take the item out of the database. Another difference between the two methods is that the “remove” method takes away all 3 items from each of the three ArrayLists they were added to based off the name (see figure 9). Text

Description automatically generated

Figure

When developing the project, it was quickly realised that the same line of code in figure 6 could be reused and although hard to figure out originally, it saved time in the long run whilst keeping the code simplistic. Only minor changes had to be made between the remove, change and search methods for a different user outcome.

**The “add” Method**

Each of the “add” and “remove” methods work in a similar fashion. The key differences between both are such that the “add” method adds user typed items into 3 separate ArrayLists based on questioned asked (see figure 10), this made it easier to add to the log file and helped when referencing within the “remove” method. This method was the easiest to implement overall and could be expanded upon by adding additional questions and creating more ArrayLists for each new subsection.

Text

Description automatically generated

Figure

**The “viewAll” Method**

“viewAll” is a user option within the main menu that can display all the stored whales within the database in a similar list view to what is shown within the text file. To achieve the way that the database shows each item under its desired subheading a method called format() was used. To correctly format the String as shown in figure 11 the syntax must be in the order of :

% [flags] [width] [.precision] [argsize] typechar *as described by Robert Gravelle (2023)*

This same formatting technique is then used for the 3 different ArrayLists (figure 12) and therefore each of the ArrayLists are within the correct columns and all stored in the correct rows, even though the ArrayLists technically aren’t linked with one another, just always presented in the same row which the loop is going through.

**Graphical user interface, application

Description automatically generated**

Figure



Figure

**The “save” and “createFile” Methods**

The “createFile” method is how the application makes both the log file and the main database list as a text file, then loops over the data to print it out. For the data to be written to the file the Java utility java.io.PrintWriter needs to be imported, if the programme needed to read off the file in the future to add to the ArrayLists then PrintReader would need to be imported. The “createFile” method uses a loop to print out the information item by item in the text file. If the file is created successfully then the user is informed in the console, however if an error occurs then this is caught by the error handling methods known as try and catch. Everything to do with the creation of the log file and the database file are within their own try handling methods (figure 13, *line 236 onwards*) that print a message to the console if something fails. To be able to use the error handling methods the Java utility java.io.IOException needs to be imported.

Text

Description automatically generated

Figure

The “save” method was useful in the early development of the programme. The original idea was to have two separate methods one that would create the file and another that would then save, however as the application got further into development more features of the save method were added into the “createFile”. Keeping the “save” method only allows the user to re-run saving the file and creating it over with the new information, however eventually this feature would be removed given more time in development.

**Created Text Files**

The Log file stores and shows the changes made to the database, lines of code throughout the programme can be seen throughout methods adding the logging information to an ArrayList, which is then displayed in the log text file, this is displayed as shown in figure 14.

**Text

Description automatically generated**

Figure

With the changes shown in figure 14 the whale-database text file is shown to the user in the format shown in figure 15.

**Text

Description automatically generated**

Figure

# 

# 3.0 Discussion

The whale watching database application was finished in a running state with a working simple interface in the console, whilst being able to write to new .txt files. Looking back over the project several things could be improved, this includes how the application writes over the existing .txt file every time the programme is run, this would include formatting the “save” and “createFile” methods slightly differently to allow the application to read off an existing file and then display the information back in the console if it already exists, this could’ve been done by importing the Java Print Reader class, implementing an “if” statement within the main method that states if the file already exists, then read the data and turn it into an appropriate string to add to the ArrayList, this would also include splitting the string to allow it to format itself properly within the adequate subheadings of the database, finally using “.add” to sort into the adequate ArrayList. Implementing this code would therefore prevent the database being cleared when ran and started over. Overwriting the file on each occasion as the application currently does can cause performance issues if the database grows larger.

Other improvements that could have been added to the project include the likes of a better GUI as opposed to using the console for the application, this was considered in the development stages of the application however getting a working application within the set time frame was prioritised. If more time was allocated for the project, learning and using a GUI toolkit such as Swing for Java could have been implemented making the application more user friendly and aesthetically pleasing.

Using the log file, a report sheet is made on changes, however a different user system could be implemented into the application given a bigger time scale, meaning that changes could be linked to a specific person adding layers of security and database monitoring if being used by a larger group of people.

Things that work well with the project are the expandability of the database, it would be simple to add more columns with more details relating to whale spotting. The formatting of the code using multiple methods allows for a neater look, meaning that introducing new menu items through more methods would be a simple process, this also made it much easier to troubleshoot during the development process when trying to solve errors due to bits of code being segmented relative to the method. A change that made a difference to the project was the inclusion of java.util.Date, without this inclusion the .txt log file wouldn’t be accurate to when variables changed within the database, separating different times of access and being an integral part to logging.

The use of ArrayLists was an integral part to creating a working database, however the downside is that each of the different ArrayLists don’t have any relation to one another, they’re only displayed in the correct format. One of the alternative ways the project could have been done was using HashMaps, this could add a table data structure that would give faster performance when retrieving values based on a key (Daniel Strmecki 2023).

# User Flow Chart

Diagram, schematic

Description automatically generated

# UML Diagram

Diagram

Description automatically generated

# References

Gravelle, R. (2023) Formatting strings in java: String.format() method, Developer.com. Available at: https://www.developer.com/java/java-string-format-method/ (Accessed: April 7, 2023).

Strmecki, D. (2023) Arraylist vs. LinkedList vs. HashMap in Java, Baeldung. Available at: https://www.baeldung.com/java-arraylist-vs-linkedlist-vs-hashmap#:~:text=The%20answer%20lies%20in%20the,(1)%20constant%2Dtime. (Accessed: April 7, 2023).