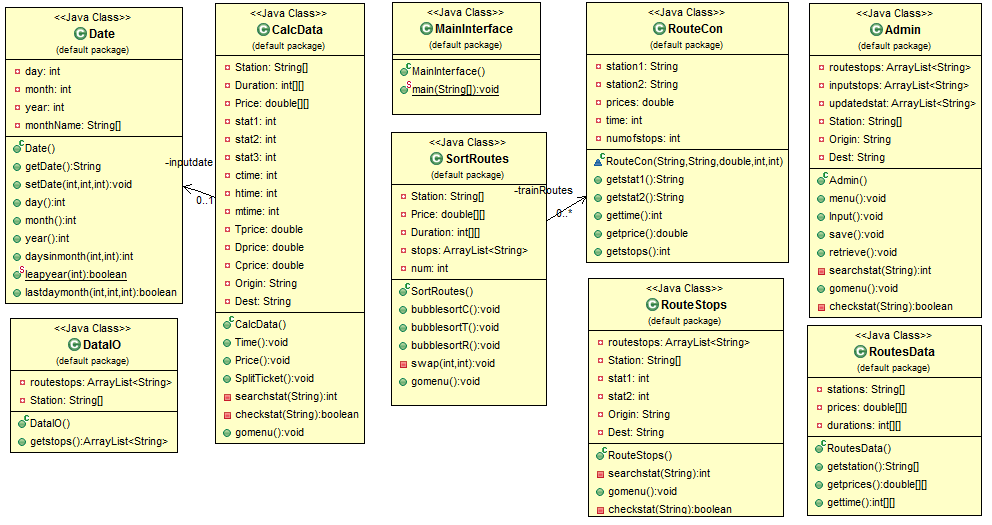
***COA256 Object-oriented Programming & Algorithms***

Assignment 2 – Report

***Design***

In total, there are nine java classes in my solution.  
These classes are MainInterface, CalcData, Date, DataIO, RoutesData, RoutesStops, RouteCon, SortRoutes and Admin.

Below you will see class diagram for this program:

Below I will explain in detail why these classes are necessary for my solution.

DataIO.java

The purpose of DataIO.java class is to import routes (stops) data from default files and store the data within the system, in the ArrayList. A function is also created to use the arraylist in other classes.

MainInterface.java

This class is the main class of the program. It will run when the program start. This class contains the Main Menu output to the screen and allows the user to select the functions they want for this program. There are seven selections, beginning with Time, Price, Route, Spilt Ticket, Sort Routes, Admin and Exit. By selecting the time, the program will run the respective class that it corresponds to. Time, Price and Spilt Ticket will point to CalcData.java, Route will point to RouteStops.java, Sort Routes will point to SortRoutes.java and Admin will point to Admin.java. More detail about these classes can be found below. Exit will terminate the program.

CalcData.java

This class exist to provide the functionality of Price, Time and SplitTicket in the program. The constructor will ask the user to input Origin and Destination of stations they want to find out about and depending on what functionality the user chose in the menu, it will run the corresponding functions in this class such as Time() for time, Price() for Price or SplitTicket() for SplitTicket(). This reason why I have combine these three functionality in one class because they have similar input and output and require similar arrays to calculate them therefore it is more efficient to put them in the same class.

Date Class

For the date class, I have designed it to contain the day, month and year, able to calculate the last day of the month and output the month in words. The date class is necessary to work for the Price functionality, to determine whether the date that the user has inputted is entitled to the discount scheme if the date is the last day of the month.

RoutesData.java

RoutesData class contains fixed data provided in the specification such as the Station’s names, their prices and durations. These data are stored in an array or 2D array and there are respective functions to get the data from the arrays and use them in another class.

RouteCon.java

This class is necessary for the SortRoutes functionality. It is necessary because I need to create constructor define the structure of the data in this order “Origin Station, Destination, Price, Time and NumofStops” in order to create an arraylist which has this structure so I can display the sort in a table format. In this class there are also functions which will be able to return the individual data e.g. NumofStops, in the array.

RouteStops.java

This class exists to provide the functionality of the Routes (Finding stops between stations). This functionality is separated from the Time, Price and SplitTicket functionality because there is a great deal of code and linking (getting the data) from other classes compare to the CalcData.java. Therefore, I decided it is best to separate them, in order for the code to look clear. The user will enter two stations (Origin station and Destination), the class will search the imported Routes data from DataIO array and output the stops if there are any, else it will output an message informing the user there is any stops imported.

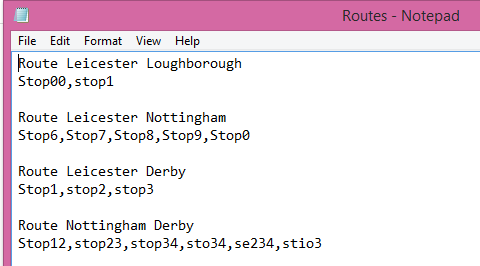
SortRoutes.java

Similar to the previous class, this class exists to provide the functionality of the sorting the routes in the system. This functionality is separated from the Time, Price and SplitTicket functionality due to same reason as RouteStop class. In this class, it will get the data from the system (RouteData.java) and imported data (DataIO) and order them into a single arraylist using the constructor I have previously created. After adding all the data need into the arraylist, it will display the initial list to the screen. The user will have the option to sort the list by time, price or number of stop en route. After selecting the sort, the system will sort the list by bubblesort, (I have create a bubblesort function for each sort) and display them on the screen.

Admin.java

When the user selects the Admin from the Main Menu, the program will direct to this class where it will display an Admin menu containing these four options, Input Route, Save Route, Retrieve Route and Exit. For Input Route functionality it will allow the user to input stop between two stations. For the save route functionality, there will be a selection between saving the routes data into a default file or into a new file. If the user chooses the default file, it will save automatically. If the user chooses the new file, a popup window will appear, allowing the user to enter the name of the new file and save it. For the Retrieve Route functionality, similar to saving a new file, a popup window will appear, allowing the user to select the file to import into the system. Exit will return to the Main menu.

Below are the default Routes data I’m using for my program.



***Functionality***

|  |  |  |
| --- | --- | --- |
| Functionality | Y(Complete)  P (Partial)  N (None) | Comments (e.g. more details on what is not working etc.) |
| Search for Price | Y | Able to read the inputted stations and work out the price based on the date. The month is outputted in words. 10% is calculated if last day of the month is inputted. |
| Search for Travel Time | Y | Able to read the inputted stations and work out the travel time between them. |
| Display Route | Y | Able to read the inputted stations and to list all the inputted stops between the stations. |
| Split Ticket | Y | Able to compare whether a direct journey or indirect journey is cheaper and display that journey on screen. |
| Sort Route Data | Y | Able to list all routes and able to sort these route by price, time and number of stops en routes using bubblesort. |
| Input Route | Y | Able to read the inputted stations and add stops to that route. Able to ask as many stops as you want (no limitation). |
| Load Route (from file) | Y | Able to select a file using the JFileChooser method. The program will read the data from the imported file and use it in the system until it terminate. |
| Save Route (to file) | Y | The users are able to choose whether to save as default file or new file. To save as new file, the user can save the data using the JFileChooser method. The program will read the data to the specified file. |
| Handling dates correctly | Y | Able to input a date and use it in the Price functionality. Able to work out the last day of every month, 30th or 31st, including the February, 28th or 29th. |
| Error handling | Y | Able to catch InputMismatch error, FilenotFound error, thrown exceptions etc. |

***Evidence of Testing***

Error handling are included within the each section.

MainInterface (Main Menu)

|  |  |  |
| --- | --- | --- |
| Input | Expected Output | Actual Output |
| Inputting 1,2,3 or 4 (selecting Time, Price, Route or SplitTicket) in the menu | Ask the user to input a Origin Station | As expected  *See appendix 1.1* |
| Inputting 5 (Selecting Sort Routes) in the menu | Display all the Routes in the menu | As expected  *See appendix 1.2* |
| Inputting 6 (Selecting Admin menu) in the menu | Display the Admin menu | As expected  *See appendix 1.3* |
| Inputting 7 (Selecting Exit) in the menu | A message saying the program will terminate. The program terminated. | As expected  *See appendix 1.4* |
| Inputting anything other than the indicated functions. | An error message telling the user that they didn’t input a function and the program will terminate. The program terminated. | As expected  *See appendix 1.5* |

Search for Travel Time (After selecting 1 in the main menu)

|  |  |  |
| --- | --- | --- |
| Input | Expected Output | Actual Output |
| Inputting the correct Origin and Destination Station | It will display the duration of this journey in hours and minutes and return to menu. | As expected  *See appendix 2.1* |
| The wrong station/  any word | It will return an error message saying it is invalid and prompt the user to enter a correct one. | As expected  *See appendix 2.2* |
| The same station | It will return an error message saying the user has entered the same station and ask the user to enter a correct one. | As expected  *See appendix 2.3* |

Search for Price (After selecting 2 in the main menu)

|  |  |  |
| --- | --- | --- |
| Input | Expected Output | Actual Output |
| Inputting the correct Origin and Destination Station | It will prompt the user to enter the date of this journey | As expected  *See Appendix 3.1* |
| The wrong station/  any word | It will return an error message saying it is invalid and prompt the user to enter a correct one. | As expected  *See Appendix 2.2* |
| The same station | It will return an error message saying the user has entered the same station and ask the user to enter a correct one. | As expected  *See Appendix 2.3* |
| The date (Not last day of the month) | It will display the Date (month in word), the Journey, and the total price of this journey then return to menu. | As expected  *See Appendix 3.2* |
| The date (Last day of the month) | It will display the same thing as the normal inputted date but it will have a message informing the user the price is discounted | As expected  *See Appendix 3.3* |
| The incorrect date | It will throw an error message saying it is an invalid date. | As expected  *See Appendix 3.4* |
| The incorrect format | It will throw an error message saying it is an invalid date. | As expected  *See Appendix 3.5* |

Display Routes (After selecting 3 in the main menu)

|  |  |  |
| --- | --- | --- |
| Input | Expected Output | Actual Output |
| Inputting the correct Origin and Destination Station | It will display all the stops between these stations. If there are none, it will return a message say it has none.  Then it will return to menu | As expected *See Appendix 4.1* |
| The wrong station/  any word | It will return an error message saying it is invalid and prompt the user to enter a correct one. | As expected  *See Appendix 2.2* |
| The same station | It will return an error message saying the user has enter the same station and ask the user to enter a correct one. | As expected  *See Appendix 2.3* |

Split Tickets (After selecting 4 in the main menu)

|  |  |  |
| --- | --- | --- |
| Input | Expected Output | Actual Output |
| Inputting the correct Origin and Destination Station | It will return a message telling the user whether a direct journey or an indirect will be cheaper. Then it will display the journey and the cost of that journey. Then it will return to menu. | As expected  *See Appendix 5.1*  *See Appendix 5.2* |
| The wrong station/  any word | It will return an error message saying it is invalid and prompt the user to enter a correct one. | As expected  *See Appendix 2.2* |
| The same station | It will return an error message saying the user has enter the same station and ask the user to enter a correct one. | As expected  *See Appendix 2.3* |

Sorting routes data (After selecting 5 in the main menu)

|  |  |  |
| --- | --- | --- |
| Input | Expected Output | Actual Output |
| After selecting 5 in the main menu | It will display all 20 routes in the system. Then it will prompt the user to sort the routes by time, price or numofstops | As expected *See Appendix 1.2* |
| Selecting Time (input integer 1) | Routes are sorted by time  Return to main menu | As expected *See Appendix 6.1* |
| Selecting Price (input integer 2) | Routes are sorted by Price  Return to main menu | As expected *See Appendix 6.2* |
| Selecting NumofStops (input integer 3) | Routes are sorted by Stops  Return to main menu | As expected *See Appendix 6.3* |
| Any integer input other than 1-3 | Error message is outputted telling the user that they didn’t enter a sort and return to main menu. | As expected *See Appendix 6.4* |
| Any non-integer input | Error message is outputted, prompting the user to input an integer. | As expected *See Appendix 6.5* |

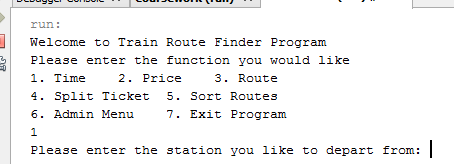
Admin Menu (After selecting 6 in the main menu)   
*Include testing with Input, Save and Retrieve Routes.*

|  |  |  |
| --- | --- | --- |
| Input | Expected Output | Actual Output |
| Selecting Input Route (integer 1) | It ask the user to enter the Origin station, then Destination. | As expected *See Appendix 7.1* |
| After input Route, Entering the correct stations | It will ask the user to input the stops that will go between these stations until the user enter the word Exit. | As expected *See Appendix 7.2* |
| After input Route, entering incorrect stations | It will return an error message saying it is invalid and prompt the user to enter a correct one. | As expected  *See Appendix 2.2* |
| After input Route, entering the same stations | It will return an error message saying the user has enter the same station and ask the user to enter a correct one. | As expected *See Appendix 2.3* |
| After entering the correct stations, enter stops and enter exit. | A message warning the user to save the data and return to Admin menu | As expected *See Appendix 7.3* |
| After entering the correct stations, enter one of the stops as one of the station has been entered. | It will return an error message saying the user has enter stop that is identical to station and ask the user to enter another stop. | As expected *See Appendix 7.4* |
| Selecting Save Route (integer 2) | It will display an option for the user to choose between save the data in a default file or a new file. | As expected *See Appendix 7.5* |
| Selecting Save Route (integer 2) then default file (integer 1) | It will save the data in the default file, output a message say it has succeeded and return to Admin Menu | As expected  *See Appendix 7.6* |
| Selecting Save Route (integer 2) then new file (integer 2) | It will popup a save as window, the user will need to fill the name of the file and click save. Then, it will output a message say it has succeeded and return to Admin Menu | As expected *See Appendix 7.7* |
| Selecting Retrieve Route (integer 3) | It will popup an open as window, the user will need to found the import file and click open. Then, it will output a message say it has succeeded and return to Admin Menu | As expected *See Appendix 7.8* |
| Selecting Return to Main menu (integer 4) | Return to the Main Menu | As expected *See Appendix 7.9* |
| Clicking Cancel on the popup window | It will return a message saying the operation has been cancel and return to the Admin Menu | As expected *See Appendix 7.10* |

DataIO

|  |  |  |
| --- | --- | --- |
| Input | Expected Output | Actual Output |
| Default Routes file is not find | Error message is thrown telling the user that the Routes file is missing. Program continues to run | As expected *See Appendix 8.1* |
| Routes file is corrupted (not in the right format) | Error message is thrown telling the user that it cannot read the Routes file. Program terminates. | As expected *See Appendix 8.2* |

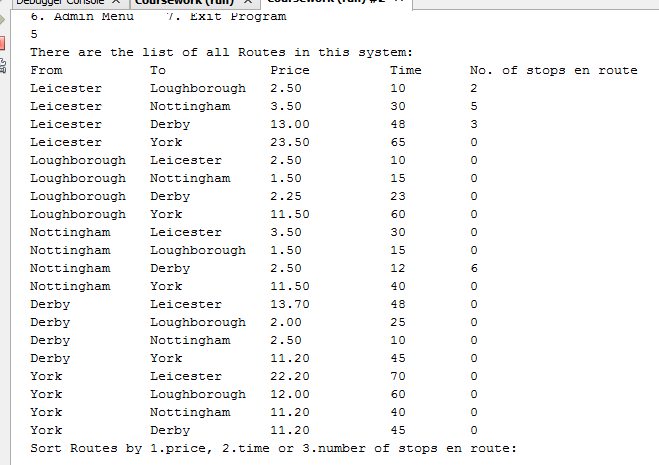
***Appendix***

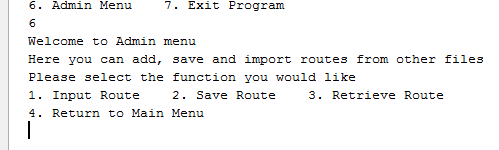
1.1  


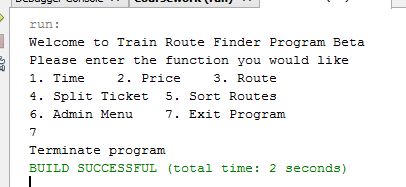


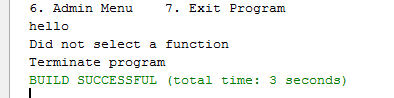


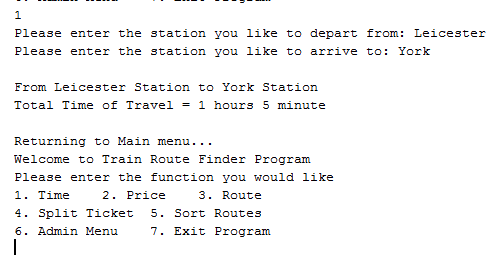


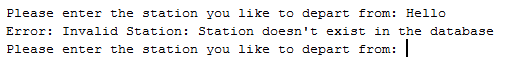
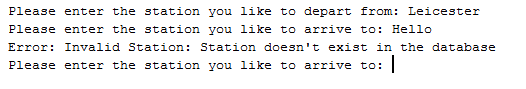
1.2  


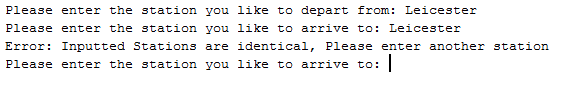
1.3  


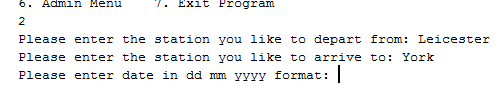
1.4   


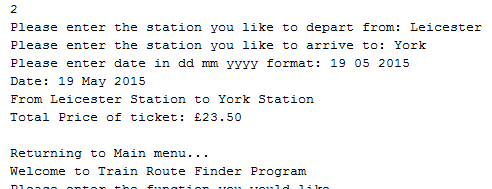
1.5   


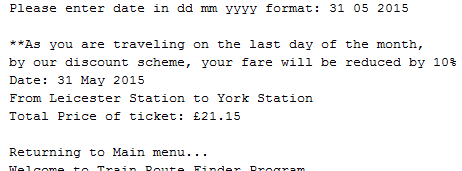
2.1   


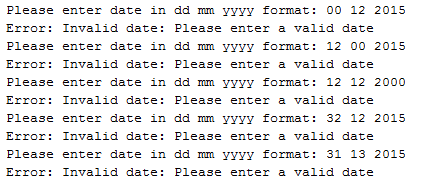
2.2   
 

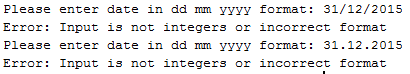
2.3   


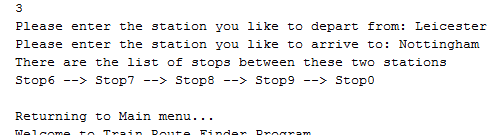
3.1  


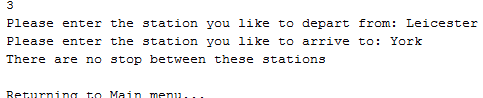
3.2  


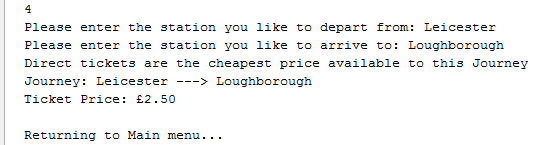
3.3  


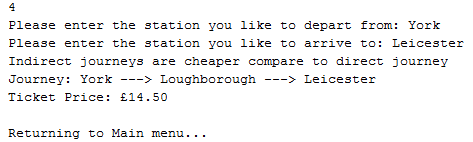
3.4   
 

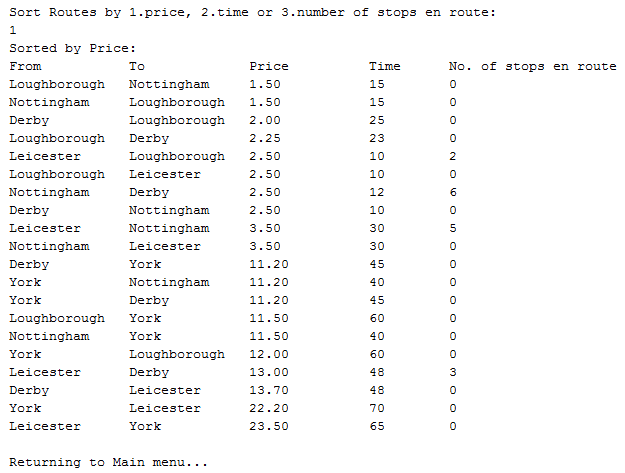
3.5  


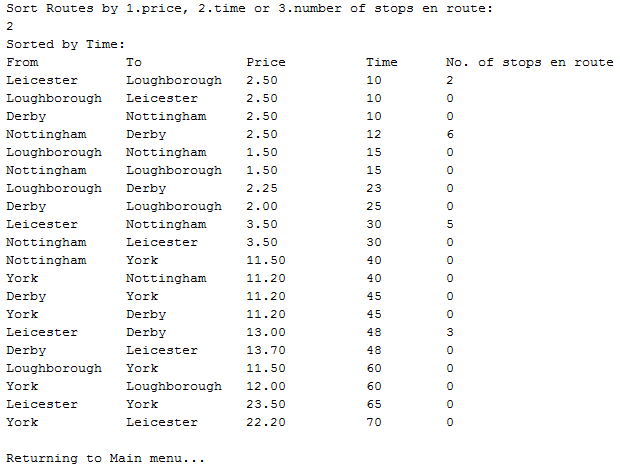
4.1   


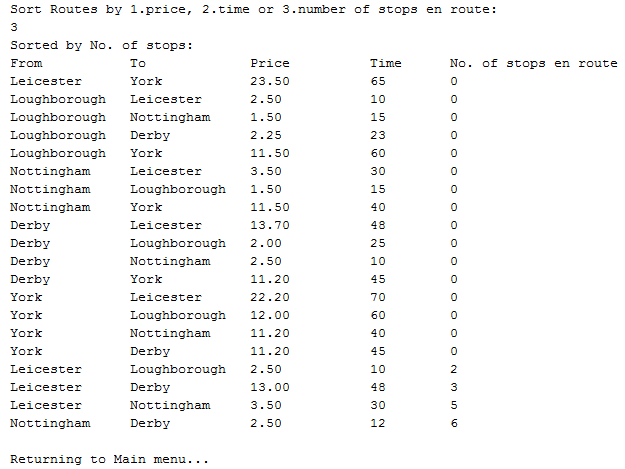


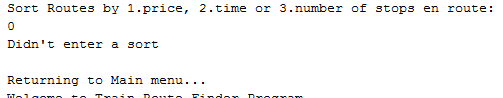
5.1   


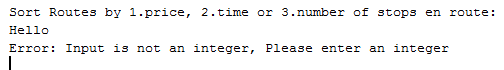
5.2  


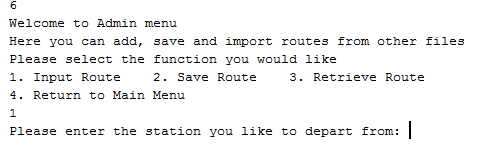
6.1 (See Appendix 1.2 to compare value of table)  


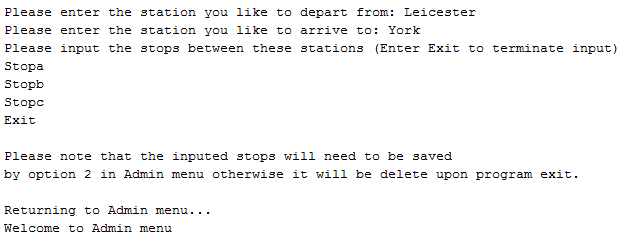
6.2  


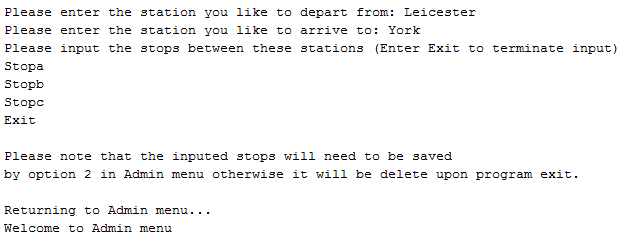
6.3  


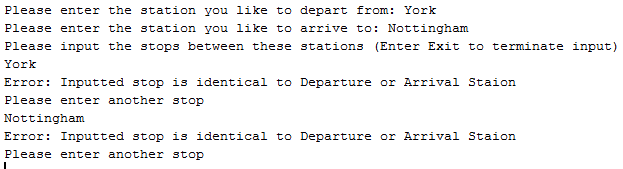
6.4  


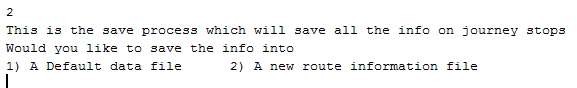
6.5  


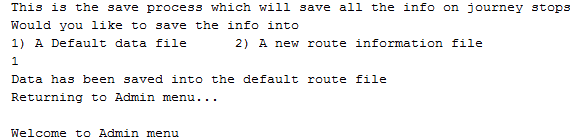
7.1   


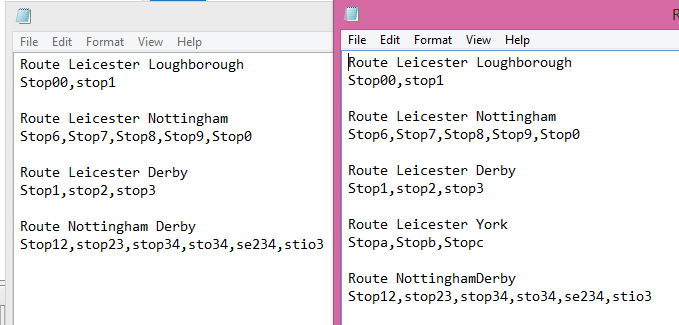
7.2 

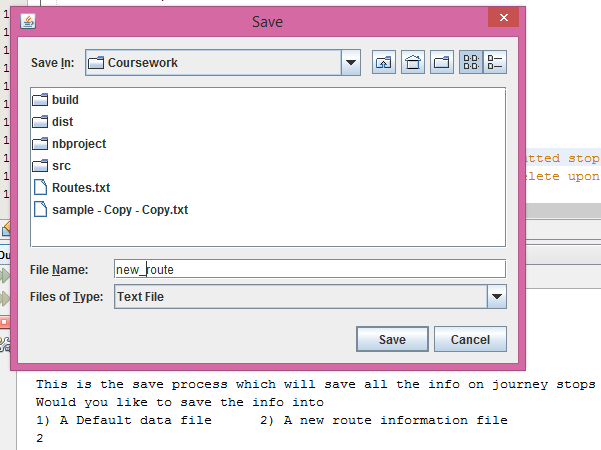
7.3 

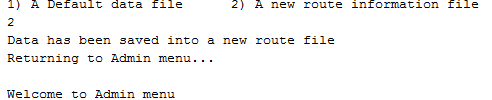
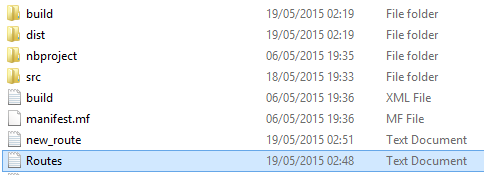
7.4  


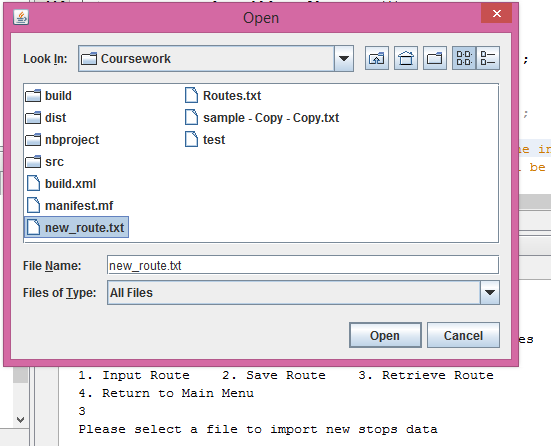
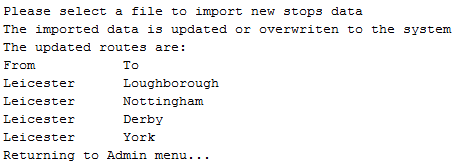
7.5  


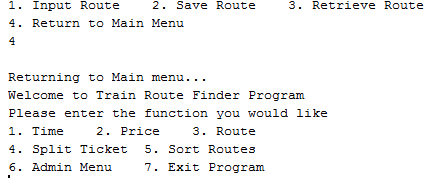
7.6  


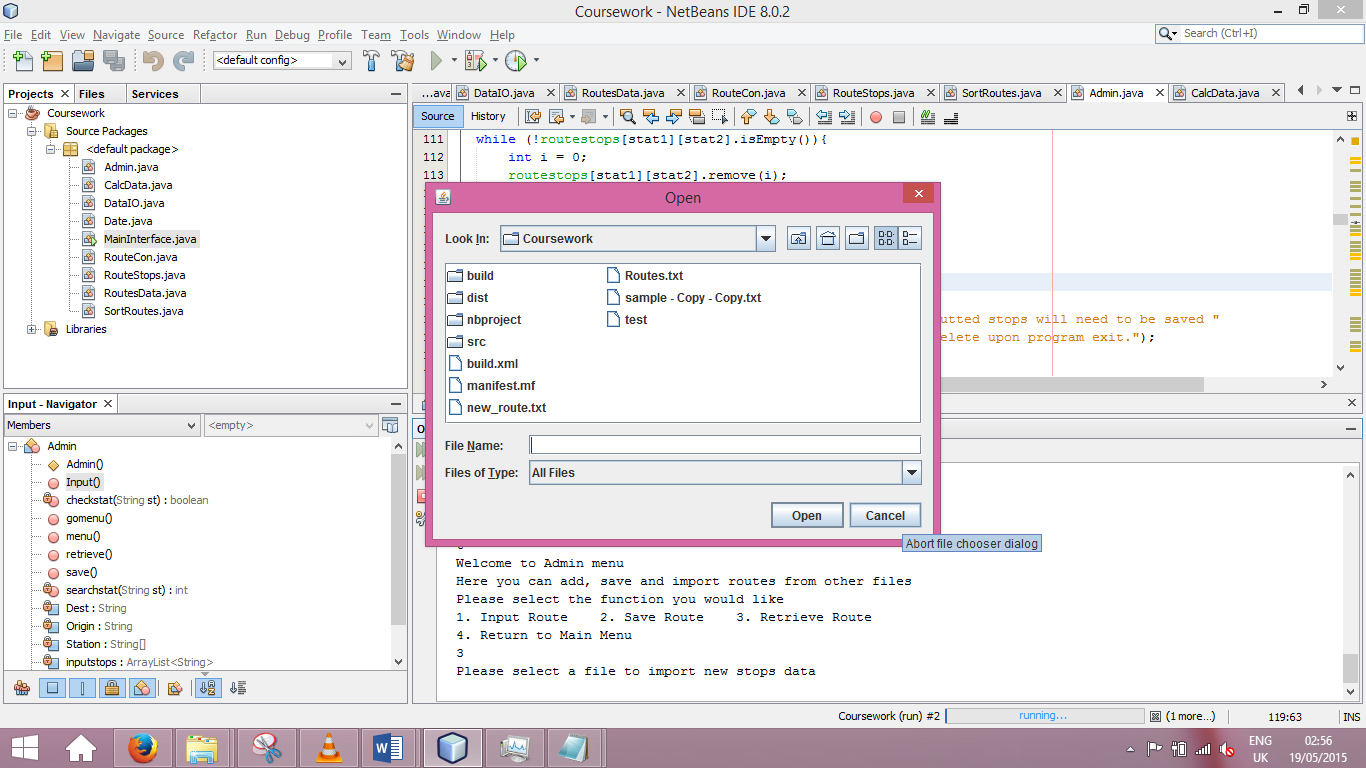
Before After  


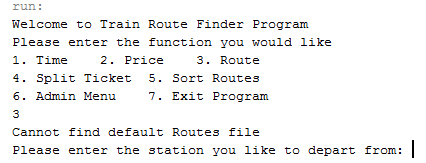
7.7  


7.8  
 

7.9  


7.10  
 

8.1  


8.2

