Object Orientated Report

ReadMe

PROJECT TITLE:Object Orientated Coursework

PURPOSE OF PROJECT:To get good marks on the coursework

VERSION or DATE:

HOW TO START THIS PROJECT:

AUTHORS: Kareem Ahmed

USER INSTRUCTIONS:

Unfortuatly i have had some issues with getting the program to run in command prompt. But it works completely in BlueJ with no issues and it works and an jar exucatable file

except for the file manipulation parts.

To Run in BlueJ right click the Main class and press void main(String args[])

File manipulation

the programs saves all the results of extra routes into a file called "Saving.txt" so that is where the info will go when you press save route

to retrieve route the program reads in a certain way it goes like this

31Cherry,Philly,Holm,;21Chelt,;

so the numbers refer to the stations like this

0 = Leicester

1 = Loughborough

2 = Nottingham

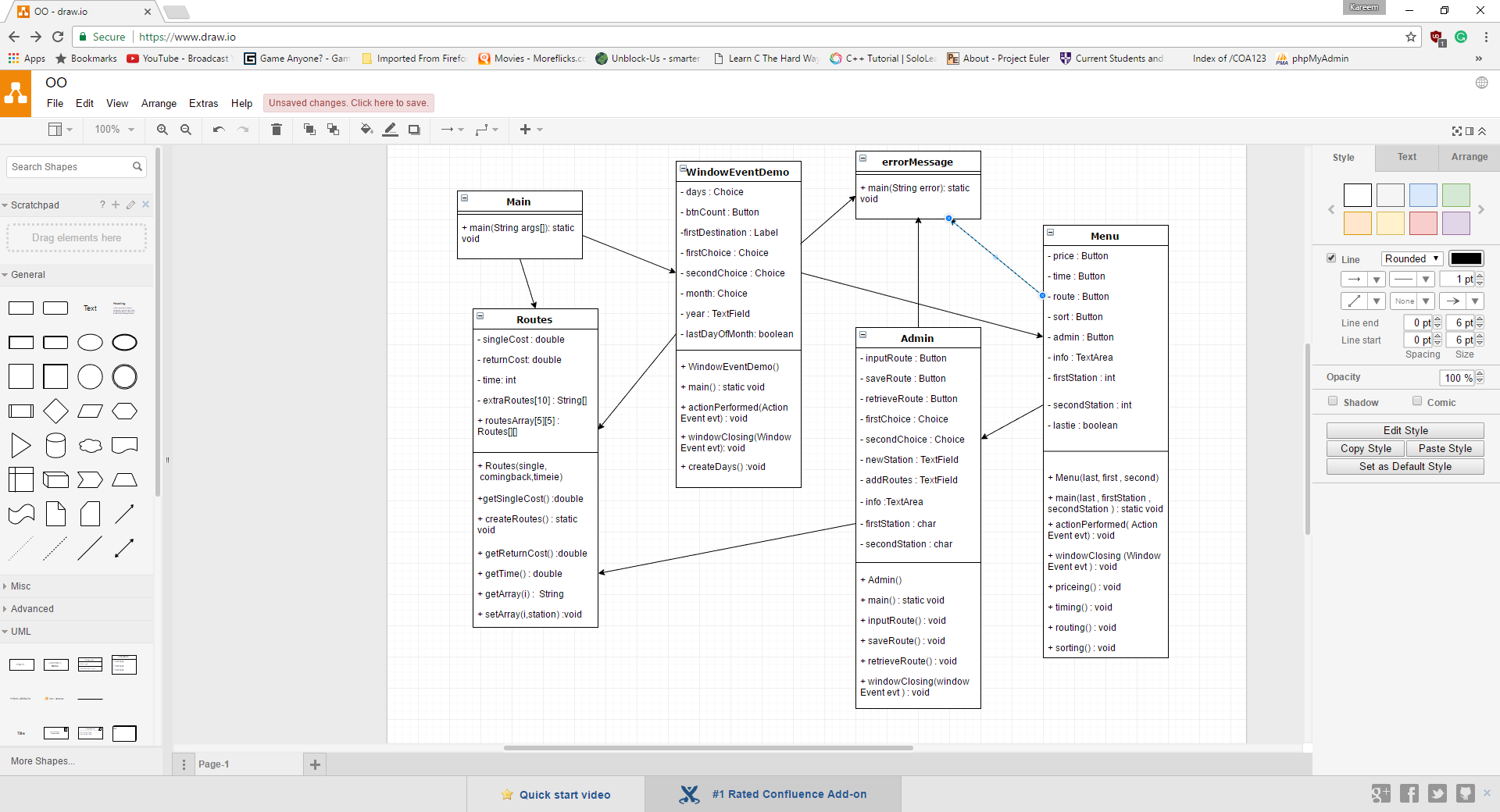
3 = Derby

4 = York

the first number refers to the first station that is picked and the second to the second

after the number you have the names of the extra stations which are split up with commas.

the end of a certain route is shown with the use of a semi colon.



Functionality Working

* You can input two stations
* It will tell you the single and return price
* It will tell how long the route will take
* It will show the stations between the two stations that you picked
* It will also show you the stations in between in alphabetical order
* You can close the program by press the red x
* You can access an admin panel
* You can add routes in between stations
* You can save the current route to a text file
* You can upload any amount of stations in between the stations from a file
* I have also made a user interface to do all of this

Functionality not working

Everything seems to work

Functions and classes

Im going to go through each class and the functions within explaining what they do

Main Class

This class is very small and is only used to just start the program

Main()

The first thing that happen is the Function createRoutes() is called ill explain what this does later but it pretty much just creates the table that we are given with all the routes on. Next the main() function of the WindowEventDemo class is called this starts the gui

WindowEventDemo Class

This classes job is to create the first gui interface the user interacts with

Main()

Creates an instance of the class that’s it

WindowEventDemo()

This function creates a lot of items and labels which are put onto the gui. First there is a drop down menu where you can pick the first train station and another drop down where you can pick the second station. It also has two more drop downs where you can pick the month and day that you are looking for. I decided to do most things in drop downs to reduce the amount of validation that I needed to do. The year is not a drop down the reason I decided this is because I thought it would be too large with every year and it would also become out of date soon so I thought I would leave it as a text area and allow users to enter and do some validation on it to make sure it is correct. That’s pretty much the extent of this function.

ActionPerformed(ActionEvent evt)

This function is called when the submit button is pressed. Firstly, I make a variable called day this is equal to the index of the days drop down and add 1 because drop down indexes start from 0 and the drop down starts at 1, this wasn’t required it just made my life easier. And next check if the index in both of the stations drop down is the same if it is then the same station has been picked twice so then an error message is called. if that’s fine I make a switch case based on the index of the month drop down. Pretty much I use this to work out if there has been the last day of the month and to check if the correct day has been picked for example picking the 31 in February should report an error and it does. After we have confirmed if it’s the final day or not we call the Menu.main() function and pass weather it’s the last day and the indexes of the drop down menus.

CreateDays()

This simply adds all the values from 1 to 31 into the days drop down and really should change this to a loop.

WindowClosing()

This function is called when the red x is pressed and it terminates the program.

ErrorMessage Class

This is called from many places and it creates an onscreen error message which can have customized dialogue

Routes Class

This class creates multiple instances of a class which holds information on every route. Stuff like time and cost and the routes inbetween.

Routes()

This function takes in the single cost , the return cost and the time it takes. And puts it into that instance of that class. It also creates an array which will be used to store the extra routes between stations.

GetSingleCost()

Returns the singlecost value from the instance

CreateRoutes()

This functions pretty much goes through the table we were given and inputs it all into a multidimensional array so that I can have easier access to it later.

GetReturnCost()

Returns the return cost of a route

GetTime()

Returns the time it takes to complete a route

GetArray()

This returns one of the extra routes which is in between the stations

SetArray()

This adds a station to the array extraRoutes[]. Which is where the stations between main stations are stored.

Menu Class

Menu()

This function takes in the weather the day picked is the last of the month, it also asks for the indexes of the first and second station picked. This function pretty much creates the gui which will have a lot of options of choices to pick from. Like sort, price, time, route and admin pressing any of the button will call their functions. Also there is a text area in the middle of the gui which will be updated when a button is pressed.

Main()

Creates a new instance of the class

Actionperformed()

This function goes off if a button is pressed and it is a switch case to find the write button and call the correct function associated with the function.

Pricing()

This happens if the price button is pressed. First it checks if the day is the last of the month it does that by looking at the Boolean value which was passed through. If true it takes 10% off the price off of the tickets. The functions get the prices by accessing the multi-dimensional array and then calling a function from one of the instances in it. Then the text is written to the text area.

Timing()

This simply takes the time variable from the route which we are looking at and displays it in the text area

Routing ()

This function takes all the values from a certain objects extraRoutes[] array and shows them in the text area

Sorting ()

This takes all the values from a certain objects extraRoutes[] and sorts them alphabetically and returns them to the text area.

Admining()

This opens up the admin gui

Admin Class

This is the final class and is where all the admin options comes under

Admin()

This creates the gui with two drop down menus for picking the stations you want to use and has a lot of buttons which you can press to do many things

ActionPerformed()

If a button is pressed this function is called where using a switch case it is determined what function should be fired.

InputRoute()

So, input routes start with a 10 time for loop the reason for this is the size of the arrays for stations inbetween stations is only 10. So, the first thing I do is try to find the first open space in the array and I do this by checking what is returned when I look at the array when I find that null is returned I know its an empty space so I can put the new route in there.

SaveRoute()

I start this by creating a file with the destination of the current directory with /Saving.txt on the end as this will be where I’m going to be saving stuff. Next I create a file writer which gives me thje ability to write to a file. I then start up two for loops which go through the entirety of the multidimensional array known as routesArray looking for routes in between the stations. The best way to explain how a file is added is a diagram

Lets say we find a route between Loughborough and Leicester and its called norm

First we do is write the station letters to the file which in this case would be 1 for the first station and 0 for the second station. So at the moment the file has just “10”

Next we add the name of the station we found so now the file says “10norm”

Now we keep checking this route to find anymore stations lets say theres one more called filde so we add that to the file “10norm,filde” if we find no more we add a semi colon “10norm,filde;” then we keep searching for more routes and repeat the process.

RetrieveROUTE()

This function start with taking the value which the user inputted into the user interface. This value is needed to find the file that the user Is using for retrieving the route from. Next, we create a file which is where we will be reading from. We use what the user inputted earlier plus the words “System.getProperty(“user.dir”) what this keyword do is give us the directory of which the program is in which allows the software to be used on anyone’s computer no matter the file structure. Now we create a file reader and a buffered reader which allows me to easily read In the file. Using br.readline() I get the first line of the word document. I only use this command once so the whole input must be one word with no breaks. Before I start breaking up the program I check to see if there is anything in the file. The first stage of converting the input into something coherent is to split it by the “;” semi colons are used to signify the end of talking about a certain route. Next I start a for loop to go through all the parts I’ve created. The first letter of the part will signify the first station and the second character is the second station. I then split this up again based on “,” these signify the end of a certain station in between the routes. So I’m taking these one by one and add them to the program I think its easier to show this in a diagram.

Input : 21chelly,philly,hems;34home,town;

Split based on ;

Gives array[0] = 21chelly,philly,hems array[1] = 34home,town

For array[0] firststation = 2 , secondStation = 1

Split based on , for array [0]

Array[0] = chelly array[1] = philly array[2] = hems

Routes.routesArray[Character.getNumericValue(firstStation)][Character.getNumericValue(secondStation)].setArray(j,tips[j]);

Would equal for The town of chelly

Routes.routesArray[2][1].setArray(0,tips[0]);

Hopefully this explained it.

Testing

|  |  |  |
| --- | --- | --- |
| Test | What will I do | Result |
| What happens if I set both stations to the same thing | Set stations both to Loughborough |  |
| Put the day in wrong | Try the 31st of February |  |
| Make the day the last of the month and check the price | Take before and after of the 3rd of jan and 31st of jan |  |
| See if I can successfully add a route | Ill add “Jamestown” between derby and Loughborough |  |
| Add a second route and sort them in alphabetical order | I will add “Amarillo” between derby and Loughborough |  |
| I can save the results to a file | Im going to press save route and check the results |  |
| I can retrieve results from a file | Im going to add the towns of “ Cherry” and “Philly” and “Holm” to derby and Loughborough |  |