

Advanced Program Structures

Scotland Yard Game

15/02/2020

Kobra Rahimi

Email: ite102770@fh-wedel.de

Field: Master of IT Engineering

Supervisor: Nils R. H. van Kan

Table of Contents

1. GENERAL INFORMATION	3
1.1 System Overview.....	3
1.2 Project Reference.....	3
1.3 Acronyms and Abbreviations	3
2. User manual	4
2.1 Requirements	4
A. Java 8 System Requirements	4
2.2 Program installation/start.....	6
2.3 Operating instructions.	6
A. Process of game	6
B. Rules of the game	10
C. End of game	11
2.4 Here are all error messages.....	11
3. Developer Manual.....	14
3.1 Development configuration	14
3.2 Problem analysis and realization	14
Problem analysis and realization of the GUI	14
A. GUI representation of game board.....	15
B. GUI representation of Stage for initial setting.....	16
C. GUI representation of Combo box for choosing the number of detectives	17

D.	GUI representation of menu bar	17
E.	GUI representation of the travel log	18
F.	GUI representation of shape for player's symbol	19
	Problem analysis and realization of logic:	20
A.	Logic representation of board class	20
B.	Logic representation of player class	22
C.	Logic representation of misterX class	22
D.	Logic representation of detective class	25
E.	Logic representation of game class	27
F.	Logic representation of IO class	28
G.	Logic representation of stations class	29
G.	Logic representation of station class	30
H.	Logic representation of log class	31
I.	Logic representation of save method	31
4.	Program organization plan	33
5.	Description of basic classes	35
6.	Program testing	37
7.	Summary	38
8.	Bibliography	39

1. GENERAL INFORMATION

1.1 System Overview

An intelligent aid for impaired individuals:

- A software system based on the Windows 7 Smartphone Platform.
- Project name or title: City Domino

1.2 Project Reference

https://docs.oracle.com/javafx/2/drag_drop/jfxpub-drag_drop.htm

1.3 Acronyms and Abbreviations

Provide a list of the acronyms and abbreviations used in this document and the meaning of each.

App: Application

MS: Microsoft

GUI: Graphic User Interface

Wiki: Wikipedia

RAM: Random Access Memory

2. User manual

2.1 Requirements

A. Java 8 System Requirements

Here is list of installing java 8 for this game according to [java Docs](#).

Windows

- Windows 10 (8u51 and above)
- Windows 8.x (Desktop)
- Windows 7 SP1
- Windows Vista SP2
- Windows Server 2008 R2 SP1 (64-bit)
- Windows Server 2012 and 2012 R2 (64-bit)
- RAM: 128 MB
- Disk space: 124 MB for JRE; 2 MB for Java Update
- Processor: Minimum Pentium 2 266 MHz processor
- Browsers: Internet Explorer 9 and above, Firefox

Mac OS X

- Intel-based Mac running Mac OS X 10.8.3+, 10.9+

- Administrator privileges for installation
- 64-bit browser

A 64-bit browser (Safari, for example) is required to run Oracle Java on Mac.

Linux

- Oracle Linux 5.5+¹
- Oracle Linux 6.x (32-bit), 6.x (64-bit)²
- Oracle Linux 7.x (64-bit)² (8u20 and above)
- Red Hat Enterprise Linux 5.5+¹, 6.x (32-bit), 6.x (64-bit)²
- Red Hat Enterprise Linux 7.x (64-bit)² (8u20 and above)
- Suse Linux Enterprise Server 10 SP2+, 11.x
- Suse Linux Enterprise Server 12.x (64-bit)² (8u31 and above)
- Ubuntu Linux 12.04 LTS, 13.x
- Ubuntu Linux 14.x (8u25 and above)
- Ubuntu Linux 15.04 (8u45 and above)
- Ubuntu Linux 15.10 (8u65 and above)
- Browsers: Firefox “

2.2 Program installation/start

For playing this game the only thing that user need is a jar file, java 8, and lib folder which contains the json library.

2.3 Operating instructions.

A. Process of game

First step, player open the jar file, two windows are opened, first windows contains three parts, dropdown list for selecting the number of detectives and two check box for choosing which player wants to play as AI mister x , detective or both of them. Then first window disappears when player clicks on the ok button.



Figure 1 first window to select number of detectives and AI player

The second window is the main board game that consists of some important options menu bar on the left top side includes new game, save, load and close. The player can see the map with 199 stations with different road color. Each color represents the rout of each means of transport.

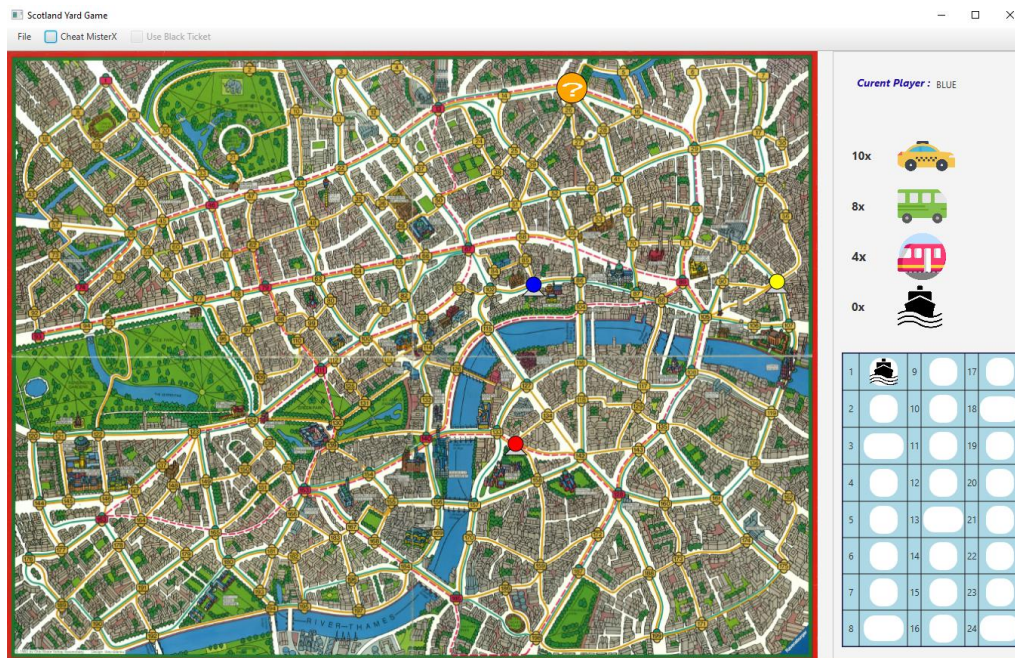


Figure 2 second window of the game

This window consists of some important options menu bar on the left top side includes new game, save, load and close.

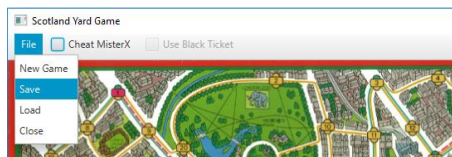


Figure 3 menu bar for saving, loading and closing the window

New game: When a player wants to start new a game, he clicks on the new game button.

Save: players can save their game by clicking on the save button, then a windows is opened, they can write the name of their saved file and also select the location for saving the game then click on save, finally a file with the type of json is saved.

Advanced Program Structures Projects (Scotland Yard Game)

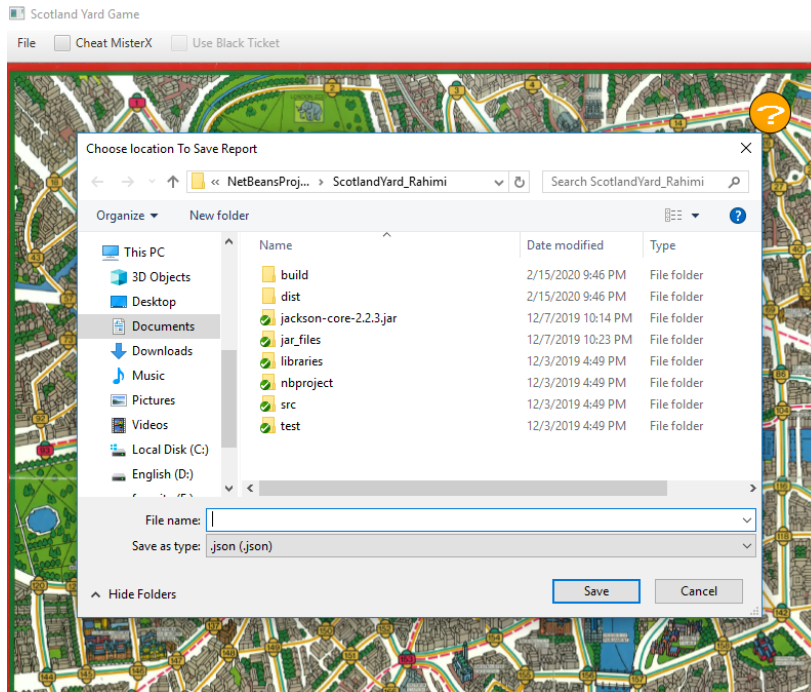


Figure 4 save the game in selected location with given name

Load: this button is used for loading the game. If players want to load the game from their saved file, click on this button. A window is opened, then select the file and click on load button. The format of the load should be json.

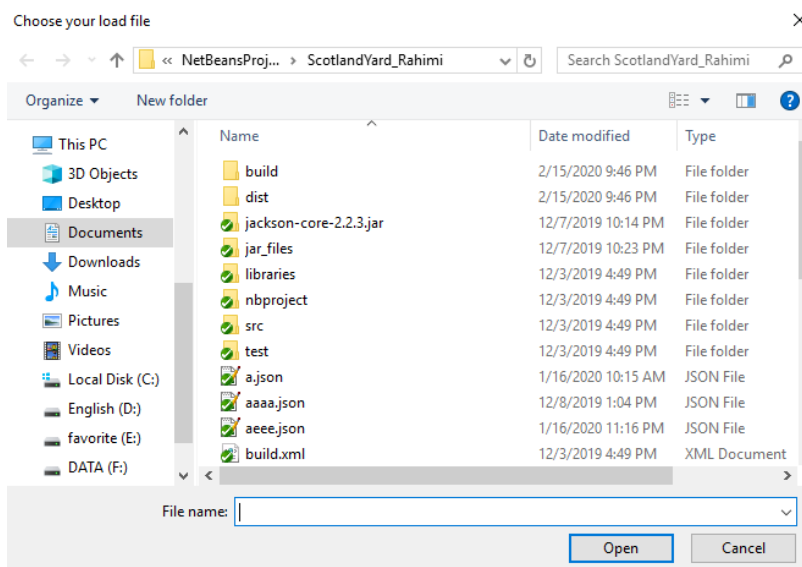


Figure 5 loading the game from the selected json file

Close: this one is used for ending the game and window is closed.

There are two check boxes on the menu bar, cheat mister x and use black ticket. When mister x is AI and second player wants to know the location of the mister x, he checks cheat mister x to show him.

The second check box (use black ticket) is used just for mister x, when he wants to use his black tickets.



Figure 6 cheat mister x or not, if player is mister x he can use black ticket with checking the second check box

When a player plays his turn, he/ she uses one specific ticket for moving from current station to clicked station, therefore the number of his used ticket is decreased and required to display the number of each tickets. This area represents the number of each ticket on the left side and the right side is the ticket type.

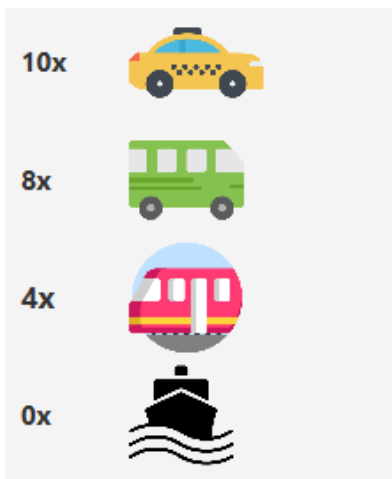


Figure 7 represent the number of each ticket for each player

This section is the log ticket of mister x, whenever mister x uses a ticket, the used ticket is displayed in this area.

























1		9		17	
2		10		18	
3		11		19	
4		12		20	
5		13		21	
6		14		22	
7		15		23	
8		16		24	

Figure 8 log ticket of mister x (travel log)

B. Rules of the game

Each player (Mr. X and the detectives) draws one of 18 possible cards which show where a player has to start, with Mr. X always drawing first. The locations on these cards are spaced far enough apart to ensure that Mr. X cannot be caught in the first round of play. There is a total of 199 locations on the board.

Each detective begins with a total of 22 tokens. Once each transport token is used by a detective, it is turned over to Mr. X, effectively giving him unlimited transport. He chooses the best means of transport and moves from his current station to clicked station, then used ticket is shown in the travel log. Mr. X also has a supply of black tokens that can be used for any mode of transport. Mr. X moves first on every turn, after which the detectives must move in the same order.[1]

The detectives should work together to catch the mister x. They use a ticket and move their game figures to the selected station. Mister x gets the used ticket.

Mister x must appear in the 3,8,13 and 18 stops. And the following rules:

- the number of detectives must be 3, 4 or 5
- Mister X does not have a colorless character, but is represented by a question mark (see chapter Representation)
- The players and Mister X do not "draw" any starting cards but are randomly assigned a starting position that has not yet been assigned. The possible starting positions are 13,26,29,34,50,53,91,94,103,112,117,132,138,141,155,174,197,198
- (Clarification) In addition to his starting stock, Mr. X can only use the tickets he has received from the detectives through their turns. For example, if the detectives never take the subway, then Mr. X will run out of subway tickets at some point (the rules of the game are contradictory here).
- there are no double turns for Mister X to simplify things[2]

C. End of game

Mister x wins the game if the detectives cannot play or mister x plays his all 24 turns.

All the detectives win, if one of them successfully catches the mister x.

2.4 Here are all error messages

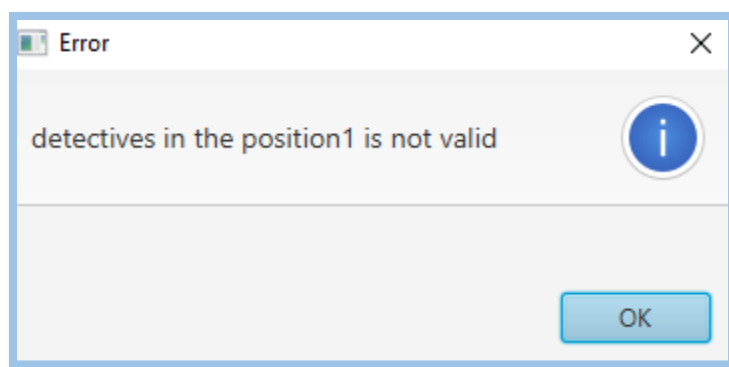


Figure 9 one of the detectives station number is not valid. The station number is less than one

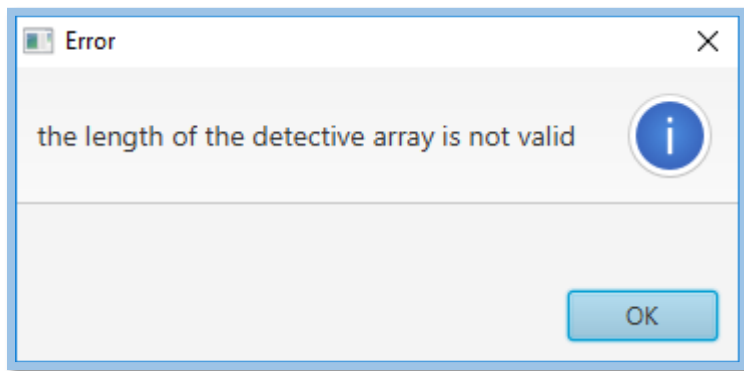


Figure 10 the number of detectives are more than 5

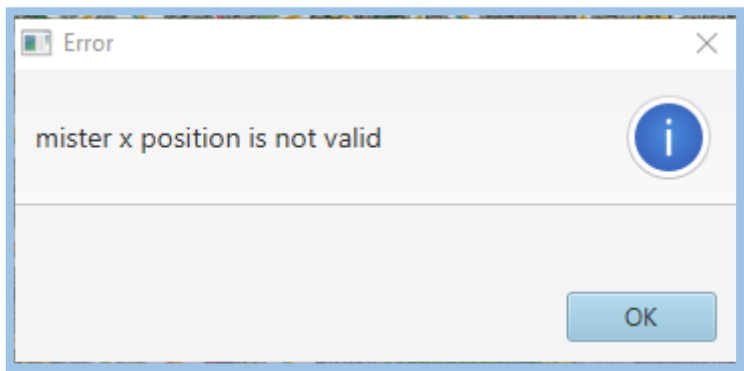


Figure 11 the station identifier of mister x is invalid. it is less than one.

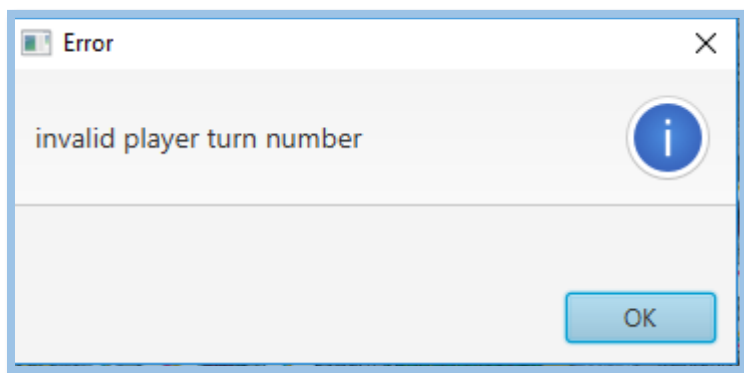


Figure 12 the turn number is invalid.

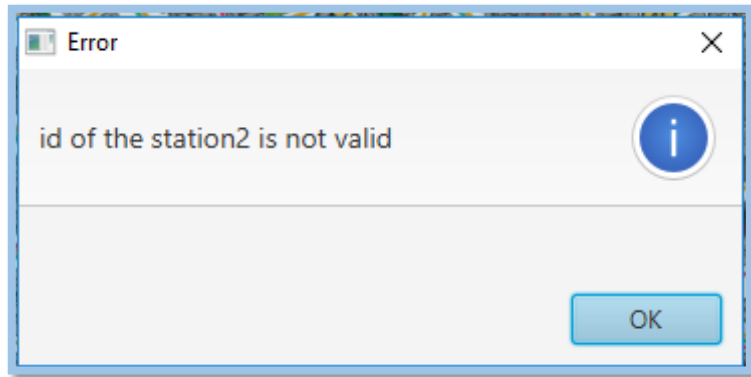


Figure 13 this error belongs to net.json which the id of the second station is invalid.

3. Developer Manual

3.1 Development configuration

Much like the hardware used for game development, the software used can vary based on your requirements.

Some software's which can have use in game development can include:

- A suitable IDE for developing the game, NetBeans 8.2 is one the flexible and very suitable software that I had used to develop my program
- A Java programming platform, Oracle's DK Java SE 8u181 is required for creating Java applications, including all Android apps.
- SVN- Client is another software which is necessary for push your project into repository. Simplify installation, automate upgrades, and manage code, instances, and users in a centralized, simple way by using this software.
- Scene-Builder: It is important to visual editing of FXML files and enabling the easy creation of Java FXML applications.
- UML diagram: I used this plugin to organize and design my organization plan of my project.

3.2 Problem analysis and realization

Problem analysis and realization of the GUI

The Scotland Yard game consists of two main packages, GUI and logic. Most of the important components that are used in GUI representation are listed below. Each component has three parts: problem analysis, realization analysis and realization description.

The following analysis and description get more information about every element functionality in Scotland Yard game. Here are all components in the GUI:

- Pane for the game board

- Stage for initial setting
- Combo box for choosing the number of detectives
- Menu bar
- Checkbox for black ticket usage and displaying the mister x
- Grid pane for showing the number of tickets and type of tickets
- Shapes for symbol of players
- Grid pane for travel log of mister x

A. GUI representation of game board

Problem analysis

The game board is a picture of all possible routs with 199 stations that it should be displayed in size 1081*814 pixels, if the window is resize, the playing board should be adjusted automatically.

When a player's turn, he clicks on a station then his symbol moves from the current station to the clicked station. If the clicked station has more than one available ticket, then a drop-down list is opened, and the player can select his transport ticket for moving. After relocation, the number of used ticket is decreased by one.

Realization analysis

A JavaFX pane is a layout component which lays picture inside that. It is easily resizable, flexible and adjusted automatically.

So, in this game grid pane is a suitable choice for showing the board picture. Another choice is anchor pane, but this component is not allowed to resizing the image.

Anchor pane:

Anchor pane allows you to position nodes in the top, bottom, left side, right side, or center of the pane.

Realization description

The playing board is represented by a pane and one image view inside of that to display the board picture. The initial size of the board is 1081x814 pixels. When a player clicks on the map, an Event

Handler method triggers processes in the logic and finally the corresponding symbol of the player is moved from the current position to the clicked position.

B. GUI representation of Stage for initial setting

Problem analysis

Before playing the game, the player should at first select the number of his detectives 3.5 or 6, choose if he wants to play against AI or not. For these setting we need a separate window. After running the game, two windows are displayed, the first window is a small with one drop down and two check boxes. Finally, when he clicks on the ok button, the first window is disappeared.

Realization analysis

The JavaFX Stage class is the top-level JavaFX container. The primary Stage is constructed by the platform. Additional Stage objects may be constructed by the application. (<https://docs.oracle.com/javase/8/javafx/api/javafx/stage/Stage.html>)

In this case, we need a separate window for some initial setting before stating the game, therefore the stage is the best choice. The second way is that to display in just one window on the left top side below the current player label, but this way of displaying needs more space.

Realization description

At the beginning of a new game two windows are displayed, the small window is initial setting before stating the game, then from the player is asked if he wants to control Mr. X and/or the detectives (otherwise the AI will do this) and how many detectives (3, 4 or 5) should play. The detectives are always completely controlled either by the AI or by the player. (TODO add reference)

After clicking the ok button, these data is sent from an action handler to the logic.

C. GUI representation of Combo box for choosing the number of detectives

Problem analysis

At the beginning of the new game in the first window, the player should select how many detectives (3, 4 or 5) play. These values are displayed in the combo box. The player clicks on the combo box and all required values are listed, then he can easily choose one of them.

Realization analysis

- Combo box

Is component of javafx that a popup list is shown to users providing them with a choice that they may select. In this case, this component is working very well because it is easily triggering between values in the list and takes less space.

- Buttons

Buttons are mostly suitable for triggering with an event. So, it is possible to use that but need a method for handling the selection and event listener.

Realization description

After clicking the combo box, a dropdown list is opened which includes the number of detectives. The player chooses one of them, the selected value is stored in a variable then sent it to the logic through game constructor.

D. GUI representation of menu bar

Problem analysis

For starting the game, saving and loading and closing the game, we need a component which includes all of them in an accessible place.

When a player clicks on this menu, three options are displayed, new game, save and load. Each of them has different functionality. If a player wants to start the game, he can click on new game button, if he wants to save his game, the save button does this action and if he wants to load his game he can click on the load button.

Realization analysis

There are some different ways to manage these buttons. Some components like bellow lists:

Buttons: we can use three buttons to these actions. Sets an event listener to each of them.

Label: is another choice like buttons sets an event handler.

Menu Bar: is the best choice for managing these buttons, because it needs less space to display and is easy to use and understandable.

Realization description

This menu bar consists of three submenus, that for each of them an event listener is defined. When one of these sub menus is clicked, an Event Handler method triggers processes in the logic. For example, when a player clicks on new game, an event handler method is called and the start Game method in logic is running. Same processes are for loading, saving and closing the game.

E. GUI representation of the travel log

Problem analysis

MisterX need an area to display the all used tickets that named as travel log. The travel log has 24 cells to show the used ticket and the number of round. Therefore, the best representation of the travel log id that using the grid pane with 24 labels inside that.

Realization analysis

Another approach is that using labels and every time when a mister x plays the background of the label is changed to the ticket image, but this way without using the grid pane is not functional.

Grid pane divide the area to how many cells we want with the same size and shape for all cells. This component is the best choice for the travel log.

Realization description

When mistex plays the game, the used ticket image is inserted into the first cell of the travel log table. Before inserting for each image an image view is created.

F. GUI representation of shape for player's symbol

Problem analysis

The detectives must be represented by a grey triangle with a black outer line and a colored head (also with a black outer line). The maximum of 5 detectives have the colors blue, yellow, red, green and black in this order. Mister X is displayed as an orange circle with a white question mark.

Realization analysis

There are some different solution for displaying the symbols. The best solution is using the shape elements of java fx.in this game two different shape are sued one circle for the head of the player and one polygon shape for the body of them. Another solution is that using the image for every player and displayed them on the board, but changing the attribute (color) of images are not possible.

Shape:

“The Shape class provides definitions of common properties for objects that represent some form of geometric shape. These properties include:

- The Paint to be applied to the fillable interior of the shape
- The Paint to be applied to stroke the outline of the shape
- The decorative properties of the stroke, including:
 - The width of the border stroke.
 - Whether the border is drawn as an exterior padding to the edges of the shape, as an interior edging that follows the inside of the border, or as a wide path that follows along the border straddling it equally both inside and outside
 - Decoration styles for the joins between path segments and the unclosed ends of paths.
 - Dashing attributes.”[3]

Realization description

When a new game is started, a method called display Symbol from logic calls a gui method. For each player one circle and one polygon is created. In the next step, all players are displayed on the board. When a player clicks on a station his/ her symbol removed from current station and displayed in the clicked station.

Problem analysis and realization of logic:

Logic package is the essential part of programming. All-important classes and methods are written in this package. It has 13 main classes that each of the classes has important rule in game.

Some of important classes and methods in logic are listed below:

- Board class
- Game class
- Player class
- Detective class
- MisterX class
- Ticket type class
- Player Type class
- GU Connector interface
- IO class
- JsonSaveLoad class
- JsonStation class
- Log class

A. Logic representation of board class

Problem analysis

This class is responsible for playing board of the game. All actions and behaviors of the board are written here. The board contains 199 station with available routes. Each path has four different tickets train, bus, taxi and boat. The stations are stored in a json file which IO class reads that and return the station with their properties and attributes. This also checks if two stations are neighbors or not. All 199 stations are stored in the linked list to get the specific station in an easy way.

There are some important attributes in this class including:

Array of object stations: for storing the all stations.

IO: this object returns all stations with all attributes of them.

List of stations: stores each station with all attributes.

Realization analysis

Linked List

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Delete and insert the element is easier than array. [4]

Array

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element of the array (generally denoted by the name of the array).[4]

Stack

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO (Last in First Out) or FILO (First In Last Out). [4]

Realization description

When a new game button is pressed, an object of game class is created and then inside the game class a board and other essential elements are created and initialized. Main idea of creating the field class is to better managing and handling some actions like, clear board, insert token in the board, finding the neighbors and check winner.

B. Logic representation of player class

Problem analysis

This game has two types of players mister x and detective. Each player has some different attributes which necessary during the game such as, id, type, Boolean is AI, station number, previous station number, weight of the selected station and four means of transport.

All tickets are stored in the hash map with the type of station and integer.

Realization analysis

HashMap

Hashmap is important data structures supported by Java collections framework. In this case we use the hash map for storing the tickets because of these reasons:

The purpose of a map is to store items based on a key that can be used to retrieve the item at a later point. Collection functionality some great utility functions are available for lists via the Collections class. HashMap is a fail-fast iterator. [5]

Array

The idea is to store multiple items of the same type together. For the ticket we need store two attribute type and number of each ticket. For this reason, array is not functional.

C. Logic representation of misterX class

This class represents the mister x player functionality and properties of that. This contains some important attributes and method. The attributes are including:

NumOfTurns: count the number of round which required for saving the game.

LastShowPos: this attribute store the last position of mister x which shown before, this is also required for the AI and save the game.

JourneyBoard: a list of all journey board of mister x with the type of integer. We used linked list because in every step, station number should all to this list, therefore, with list we can easily all an item.

NeighborsMisterX: store all neighbors of mister x and remove all nodes contains detectives that is needed for AI part.

Choose: is an object of choose class that store three attributes ticket, station and weight of the selected station.

One of the important method in this class is AI which is responsible for the AI part of the player. It calculates the each strategy and finally determines the optimal station for next move. Only moves to "neighboring" stations is possible and only means of transport for which a ticket is still available can be used.

The following rules are determined for choosing the optimal station:

- “Move to a possible target position (see below, circled light green in the picture on the right). So Mister X drove from the last pointing position at 116 out of 1x taxi). If there are more than one available, choose the one with the smallest station number --> the blue detective would go to station 118.
- Move to a directly adjacent subway station (to quickly get to other locations). If there are several accessible free stations, select the one with the lowest station number. --> The yellow detective would go to Station 185.
- Move towards the last position where Mister X was shown (see below). Use the shortest (currently) free path for this. --> The red detective would go to station 70 (and following to 87, 86, 116), because this is the first one on the shortest way and it has a smaller number than the 89 (see below).
- move to the directly adjacent free position with the smallest station number (quasi as fallback, if nothing else works)”[2]



Figure 14 tactic of the detectives when is handled by AI

“After each attempted move, the new game situation ("detective moves to station x") is assessed using the evaluation function described below. The best rated move is carried out (if there are several moves with the same rating, the station with the smaller number is taken). The evaluation function consists of these values and weights, which are added together. A larger value is better:

- How many possible target positions can be reached by all detectives in the next turn divided by the total number of all possible target positions (for the detectives not yet drawn, their old stations should still be taken into account) Weighting: Number of achievable target positions by total number of target positions * 10 --> after the blue detective has moved to station 118, there are still 3 target positions left (104, 117, 127), but all of them can not be reached --> $0 / 3 * 10 = 0$
- How far away is the " average possible target position " (see below) (so that the detectives who are further away from Mister X move roughly in his direction and don't run around stupidly in the area) Weighting: if 10 stations or more away = 0, otherwise (10 - number of stations on the

shortest way to the " average possible target position ") --> After the blue detective moved to station 118, 3 target positions remained (104, 117, 127). Their coordinates are 765/341, 848/447 and 693/447, averaged 769/412. The closest station is 116, which is exactly 1 station away from 118. --> $(10 - 1) = 9$

- How many stations can be reached from the current position with the existing tickets in one turn? Weighting: Number of stations divided by 13 (as these are the most different stations that can be reached at station 67) * 4 --> assuming the blue detective still has 3 subway tickets, 4 bus tickets and 3 taxi tickets, he can reach 4 other stations (all by taxi) from station 118 --> $4 / 13 * 4 = 1,23$
- What is the lowest number of tickets for a means of transport (if you don't have a ticket for one any more, this significantly restricts mobility) weighting: If there are more than 2 tickets, the rating is 3, otherwise number of tickets. --> The lowest number of tickets at the blue detective (see above) is 3. --> $= 3$ --> the total rating would be $0 + 9 + 1,23 + 3 = 13,23$ [2]

The first and second calculation are implemented in the choose neighbor method.

D. Logic representation of detective class

This class contains attributes and methods related to detective player. The important attributes are including:

- Choose: it is an object from choose class which contains the best station number with the weight of that and the available ticket.
- Board: a board of the detective
- Players: contains all detectives

The important methods here are AI and bestRoute which is finding the best route according to shortest path algorithm.

Dijkstra's algorithm

Is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. The algorithm exists in many variants. Dijkstra's original algorithm found the shortest path between two given nodes but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph. It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined. For example, if the nodes of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road (for simplicity, ignore red lights, stop signs, toll roads and other obstructions), Dijkstra's algorithm can be used to find the shortest route between one city and all other cities. Dijkstra's algorithm uses a data structure for storing and querying partial solutions sorted by distance from the start. The original algorithm uses a min-priority queue and runs in time $O(E + V \log V)$ (where $|V|$ is the number of nodes).[5]

Dijkstra algorithm is BFS (Breadth-First Search) because if we look at the working of the Dijkstra algorithm, then you will see that the source node (or any node, let's call it node X) first computes of the cost of all of its neighboring nodes (nodes which are reachable from the node X). Once that's done, it selects that neighboring node with the minimum cost. This process continues until we reach the target node.[6]

Shortest path finding of our game:

One solution is to solve in $O(VE)$ time using Bellman–Ford. If there are no negative weight cycles, then we can solve in $O(E + V \log V)$ time using Dijkstra's algorithm.

Since the graph is unweighted, we can solve this problem in $O(V + E)$ time. The idea is to use a modified version of Breadth-first search in which we keep storing a given vertex in route while doing the breadth first search.

Two stations, as source and destination, are provided and the source station is added to start of the route and it continuous to add to each child station to the separate route. Each route, which has been done and

added, will be in visited list added to avoid cycle in the graph. In this approach all the routes to destination are created and shortest one is compared and is chosen. The route should be free of detectives. It good in term of process but as it makes all the routes, more memory is used.

Another approach is to have predecessor map and each current node is predecessor of the child until it reaches to the destination. When destination is found, it will traverse back to the source station to find the start of route. This has better memory usage as the paths are not made beforehand but the process needs one more step to make the route.

Furthermore, instead of the simply travel from A to B, a tree complete tree can be used by adding all the routes from parent to leafs. In this approach, the cycle is not concerned but it cost to make the tree and find all route and compare them to choose shortest one.(according to Ali explanation)

E. Logic representation of game class

Problem analysis

Game class is the main class for this game. All objects and elements which are required for Scotland Yard game are created and initialized in this class. It has three necessary constructor first one is used for creating the game, second is for testing and the third one is used for loading the game. Game class organizes other classes and also the flow of the game. From start the game until the end of the game, this class knows in every step which method should be called and processed. For this reason, this class is required to implement. There are some necessary attributes such as:

- GuiConnector: connects the logic part with the gui of the game.
- Board: represent the playing board of the game.
- Starting point: is a list contains all stating station of the game.
- Idxplayer: is the id of the current player.
- GameIsWon: set to true when a player won the game
- Logger: is an object to carry out the log file.
- Target Positions: is a list of target positions.
- Players: is an array which stores all players with their attributes.

There are some essential methods which are necessary for the game logic. Display player just checks if the current position is the neighbor with the clicked position, if it is then calls change round method and a method in gui is triggered. Next player method at first check if we have winner if not then continue. With other condition, if current player is detective then disable the black ticket checkbox and increment x by one (we need x to check if all detectives cannot play). If current player is AI then bot turn method is called. Check winner method checks if current station of every detective is equal to mister x, then detectives are winner. If number of used ticket of mister x is 24 then mister x won the game.

Realization description

When player clicks on new game, an object of game with these parameters are created. New Game (3, 1, true, false, gui) first parameter is the number of detectives, second is the number of misterx, third is the handling detective by AI, forth is the handling mister x by AI and last is the gui connector. These constructor is created when player select the number of detectives and handling AI in the first small window, then game started and an event listener listens to the every click and event to the board. When a player is clicking in some station, a method display player is called and checks if the clicked position the neighbor of current player current position, then move to the clicked position. These process continues until one of the player won the game.

F. Logic representation of IO class

Problem analysis

This class is responsible for input and output, reading the json file and store it in an array of stations. it also saves the game with given parameters in a json file. Every professional java game has some options and functions to increase the efficiency of that. Save and load is one of the important facilities of a game. Moreover, when user want to save his game even the game is not finished yet and then later he wants to load it again, so, because of these reasons we need a separate input and output class for handling save and load the game. The purpose of IO class is that makes easier the process of saving and loading.

With using gson library technique, the flow of saving and loading the game are being applicable and simple.

Realization analysis

There are diverse approaches and techniques to read the json file like Jackson, normal json parsing and gson library. GSON is open source and standalone library which is used to convert JSON data into java objects and vice versa. Here using gson library has these advantages over the other techniques.

Facilities are provided in the following areas:

- Provide simple toJson () and fromJson () methods to convert Java objects to JSON and vice-versa.
- Support of generic objects.
- Allows to serialize a Collections of objects of the same type.
- It handles the null fields, by not including them in serialization output but during de-serialization it is initialized back to null.
- Allow custom representations for objects.[7]
- Easy Error handling

Realization description

After creating new game, IO object is called to read the json data and store them in an array of stations, but before storing the station it checks if the format of the json file is correct with using JsonValidation class. If everything is ok then stations passed to board, otherwise the error window will be shown with the type of error and game is closed.

G. Logic representation of stations class

This class provides the best way to create an object according to json element format. This just is used for the json. Where which station is located and which means of transport can be taken there and where, is stored in the net.json. This includes:

- the number of the station
- the pixel position on the map (0/0 is top left, 1/1 is bottom right)

- ascending the stations to be reached by subway
- ascending the stations to be reached by bus
- ascending the stations to be reached by taxi
- ascending the stations to be reached by boat.[2]

The attributes of this class is arranged according to the elements of the net.json (all 199 stations with their properties).

G. Logic representation of station class

Problem analysis

Each station in the board has these attributes, identifier of the station, position of each station, array of the stations which can be reached by tube ticket, array of the stations which can be reached by bus ticket, array of the stations which can be reached by cab ticket and the last array is for the boat ticket.

There are some applicable method which is required for the game such as `getAllneighbors`, which checks the neighbor's station of each station and store them into a list, `hasTrainNeighbors` checks if the current station has the train neighbor with clicked position.

Realization analysis

Each station can be represented as an instance of a station class. In the object the identifying number and the coordinates are stored.

Realization description

Each station is represented as an instance of a station class. The object stores the identifying number as an integer. Because there are only 199 stations at all the range of integer should be sufficient for this purpose. The x and y coordinates are stored as a reference to a position object. For each means of transport the object has an array with the references of all stations that can be reached by the corresponding means of transport starting from this station. An Array is here the best solution because it is fast to access and after the first initialization the connections never change, so there are never entries added to or removed from the array.

H. Logic representation of log class

For traceability, one log file is written per game (as a simple text file with the extension *.log) (there is exactly one, so it will be deleted or overwritten at the beginning of a new game). It contains the following information exactly in the given format:

- the number of detectives (integer)
- whether MisterX is controlled by the AI (true or false)
- whether the detectives are controlled by the AI (true or false)
- the starting position of MisterX
- the starting positions of all participating detectives --> all values up to this point are stored comma separated in one line
- all moves of the players or of Mister X comma separated in one line, as follows:
 - which player moves (integer starting from 1 for the detectives, 0 for MisterX)
 - previous station
 - new station
 - remaining tickets for subway, bus, taxi, boat (always 0 for detectives)
 - which of the tactics (starting from 1 in the above order) was chosen (for a human move = 0)
 - what is the rating (see above) for this (for a human move = 0.0) --> a decimal point should be used for better legibility
- if the game has ended, who won the game (0 for Mister X, 1 for the detectives)[2]

I. Logic representation of save method

Problem analysis

The current game situation is stored in a user-selectable file at any time and can be reloaded from a freely selectable file. Before saving the program asks the user then name of the file and location to save. A save file (also a JSON file) has the following entries:

- MisterX
 - whether it is controlled by the AI (true or false)

- the possible target positions
 - the last showing position (or 0 if there was none)
 - the current station
 - all remaining tickets he has (integer for subway, bus, taxi, boat)
 - the occupied entries of the travel log (ordinal values of the used tickets from 0 in the order subway, bus, taxi, boat)
- Detectives
 - the number of detectives (integer)
 - whether they are controlled by the AI (true or false)
 - for every detective
 - the current station
 - all remaining tickets (whole numbers for subway, bus, taxi)
- who is on the move (0 = Mister X, detectives starting from 1)
- the number of the round it is (starting with 1)
- whether the game has already been won (true or false) [2]

Realization analysis

For saving the file gson library is used because of these reasons:

- Provide simple toJson () and fromJson () methods to convert Java objects to JSON and vice-versa.
- Support of generic objects.

Realization description

When a user clicks on the save button, an object of jsonSaveLoad is made, then in the IO class, all given parameters are stored in this object, then write all these parameters to the stream. In the final steps the stream is closed.

4. Program organization plan

This is my program organization plan. The color coding makes it easy to assign the classes to the packages. Most of the arrow has label which shows the plan of that.

Scotland Yard game has two important packages gui and logic, which is distinguishable with different background color. They are separated for better understanding. Gui packages includes all methods and classes related to game graphic like, coloring , styling , playing filed, menu bar and buttons for saving and loading the game.

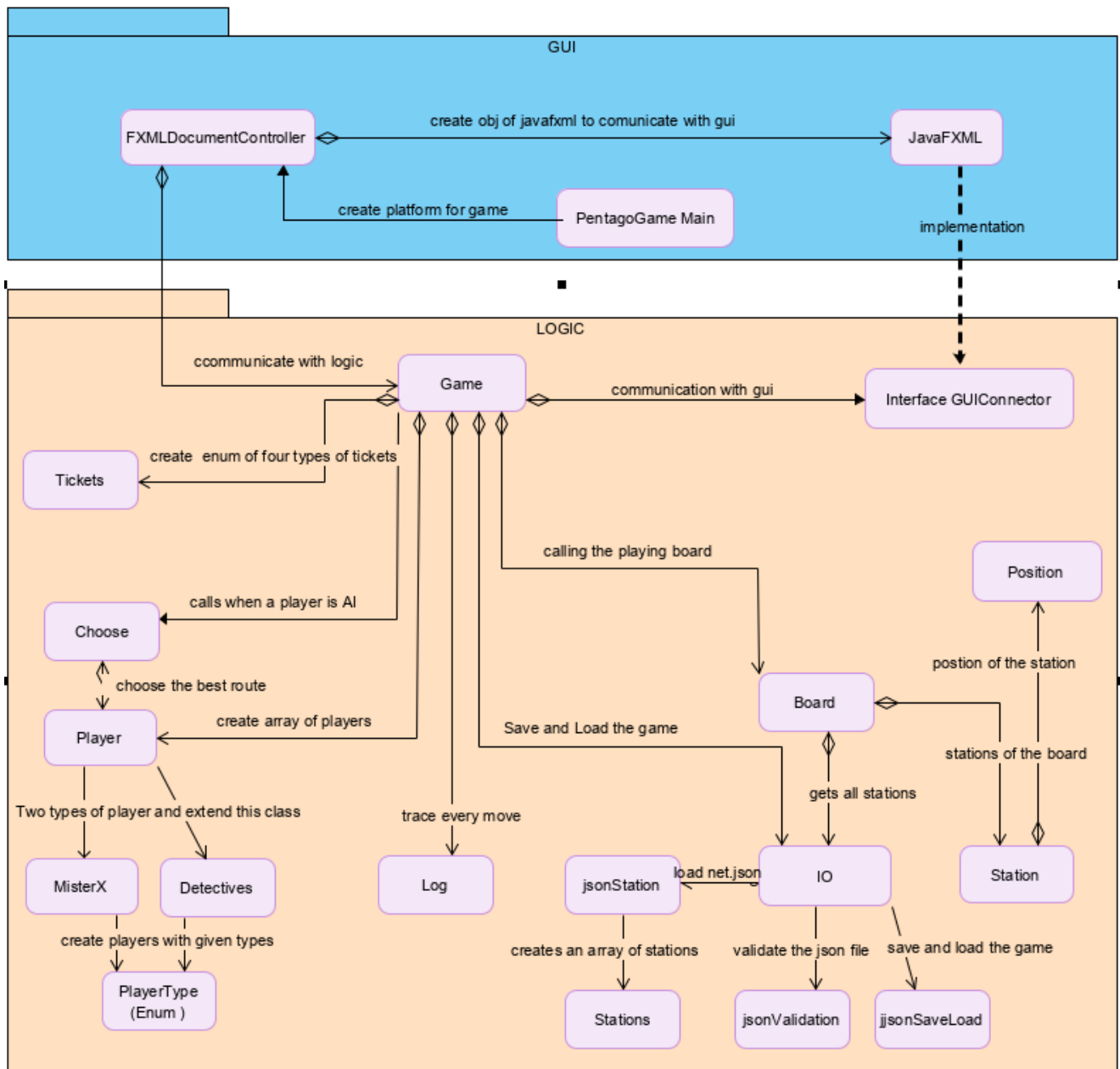


Figure 15 UML diagram of the game

5. Description of basic classes

Scotland Yard game is made of some diverse classes. Every class has different strategy and implementation. In the Problem analysis and realization I described every classes in detail. so in there I just described some of important classes with brief description.

Game class: this class is the heart of our game. It consists of some important methods to handle our game. Our game creates field for playing and every element that we need to play. In this class display player, change round, bot turn, next player, check winner, save and load have been written here.

Player class: This game has two type of player's misterx and detectives which are extending this class. each player has some attributes including id, is AI, weight for the chose station, type, number of taxi tickets, number of bus tickets, number of boat tickets and number of train tickets.

Board class: This class create a board for the game. The playing board contains 199 stations with available routes. Each path has four different tickets train, bus, taxi and boat. The stations are stored in a json file which IO class reads that and return the station with exact positions and tickets. This also checks if two stations are neighbors or not.

Station class: Each station in the board has these attributes, identifier of the station, position of each station, array of the stations which can be reached by tube ticket, array of the stations which can be reached by bus ticket, array of the stations which can be reached by cab ticket and the last array is for the boat ticket.

IO class: this class is responsible for input and output, reading the json file and store it in an array of stations. It also saves the game with given parameters in a json file. And last one is loading the game with given parameters and return the new game.

JavaFx GUI class: This class is implementing the method which are defined in the gui connector.

All method for visualization that are required are written here. Display symbol of player, create shape for every player, and show mister x travel log, show error messages are some important methods of this class.

GUI Connector interface: communicate logic with gui. These all methods here are required to display the status of the game.

Log class: For traceability, one log file should be written per game. Some important information are stored here such as, number of detectives, current position of each player and previous position, remaining tickets, starting points, AI value and so on.

6. Program testing

Error message	Cause	Correction action
Start of the game is not possible	Json file format in not valid and window is closed	Using the correct json format
Load the game is not possible	Format of the loaded file is not correct	Choose the correct load file json
Start of the game is not possible	Missed Taxi ticket	Change the taxi ticket name from lowercase to uppercase
Clicked on the station which is not neighbor with the current station	Cannot move the player to the clicked station	Click on the neighbor station
Use clicks on a station which does not have enough ticket	Cannot move player	Checks the player ticket before click
Clicks is not disabled after the game is finished	Click event is working even after the game is finished	Could not handle this error

7. Summary

A computer variant of the classic board game "Scotland Yard" is to be implemented, in which a group of detectives in the streets of London tries to catch the ominous Mr. X. The game is played on a computer. For a first impression this example program might be helpful (which needs the board and the net in the same folder). However, this task always remains binding in case of differences![2]

8. Bibliography

- [1] “Scotland Yard (board game).” [Online]. Available:
[https://en.wikipedia.org/wiki/Scotland_Yard_\(board_game\)](https://en.wikipedia.org/wiki/Scotland_Yard_(board_game)).
- [2] “APSP: Task: Scotland Yard.” [Online]. Available: <https://lms.rechnernetze.fh-wedel.de:888/mod/page/view.php?id=1443>. [Accessed: 15-Feb-2020].
- [3] “Shape (JavaFX 8).” [Online]. Available:
<https://docs.oracle.com/javase/8/javafx/api/javafx/scene/shape/Shape.html>. [Accessed: 29-Feb-2020].
- [4] “Array Data Structure - GeeksforGeeks.” [Online]. Available:
<https://www.geeksforgeeks.org/array-data-structure/>. [Accessed: 29-Feb-2020].
- [5] “Dijkstra’s algorithm,” *Wikipedia*. 21-Feb-2020.
- [6] “(1) Is Dijkstra BFS or DFS? - Quora.” [Online]. Available: <https://www.quora.com/Is-Dijkstra-BFS-or-DFS>. [Accessed: 01-Mar-2020].
- [7] A. Sinhal, “Parsing JSON with Gson library,” *Medium*, 17-Jan-2017. [Online]. Available:
<https://medium.com/@ankit.sinhal/parsing-json-with-gson-library-184d94a04dec>. [Accessed: 01-Mar-2020].