



City Domino

Advanced Program Structures Project

Kobra Rahimi

IT Engineering

Ite102770

Third Semester

Email: ite102770@fh-wedel.de

Supervisor: Nils R. H. van Kan




Table of Contents

1. GENERAL INFORMATION.....	1
1.1 System Overview	2
1.2 Project Reference.....	2
1.3 Acronyms and Abbreviations.....	2
2 User manual	3
2.1 Requierments.....	3
2.1.1 Java 8 System Requirements	3
2.2 Program installation/start.....	4
2.3 Operating instructions.	4
2.3.1 Process of game.....	4
2.3.2 Rules of the game	7
2.3.3 End of game	7
2.4 Here are all error messages.....	8
2.4.1 Renaming the package.....	8
2.4.2 save/load	9
3 Developer Manual.....	9
3.1 Development configuration	9
3.2 Problem analysis and realization	10
3.2.1 GUI representation of playing field.....	10
3.2.2 GUI representation of next box	12
3.2.3 GUI representation of current box	12
3.2.4 GUI representation of selection box.....	13
3.2.5 GUI representation of rotate button	13
3.2.6 GUI representation of menu bar	14
4 Program organization plan	15
5 Description of basic classes	16
6 Program testing	18
Bibliography.....	18

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 Requierments

2.1.1 Java 8 System Requirements

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

2.1.1.1 „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

2.1.1.2 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.

2.1.1.3 *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“ (WebSite, kein Datum)

2.2 Program installation/start

For playing this game the only thing that user need is jar file and java 8 .

2.3 Operating instructions.

2.3.1 Process of game

First step, player open the jar file, he sees the windows which made of some parts . on the top left side of the windows, there is a menu bar which contains three buttons: new game, save and load.

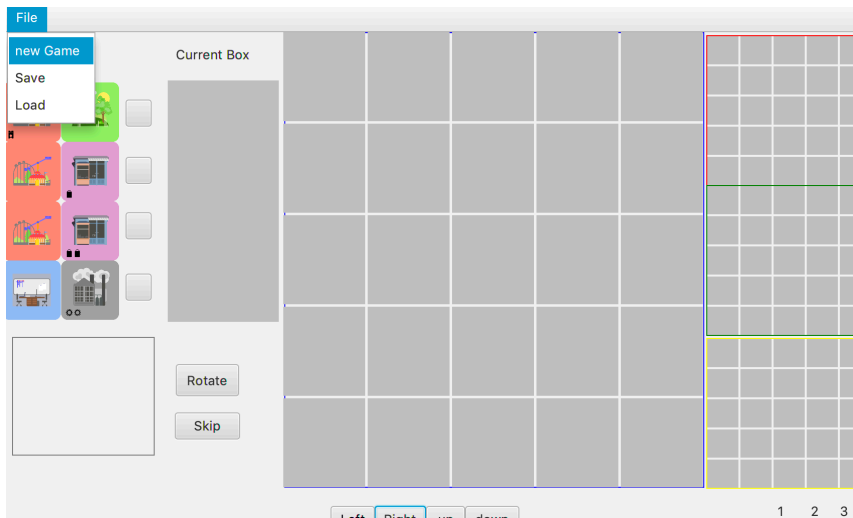


Abbildung 1 : gui representation(new Game)

If he clicks on new game, the new game is starting. All fields and all boxes are empty. If player clicks on save, a new window will open then he can write the file name and the location for storing his game file. If he clicks on load button, a new window is opened and then he can select his file which he wants to load game from that.

As you can see, on the left side there three boxes, next box , current box and selection box. The next box is filled randomly with ascending order of dominos ordinal. The current box shows the current player domino and also there are a label next to current box which which shows the current player is. Player id 0 is used for human, player id 1 is used for bot1, player id 2 is used for bot2 and last one is used for bot3.

In a round, first a card from the second selection area is marked and is then selected for the next round. So the more valuable his marked card has in this round, the later the player's turn comes and the less choice he has for the next round. Then the player tries to place his card for the current round on the game field.(copy from descrition of the game)

The small box in the bottom is the selection box that is displaying the current domino. When domino is here, player can drag it and put in his field. The two buttons rotate and skip are used for rotating the domino in clockwise, in every click the domino is rotated by one. When player cannot lay his domino on the field or he does not want to play and skip his round he can clicks on skip button and skip his round.

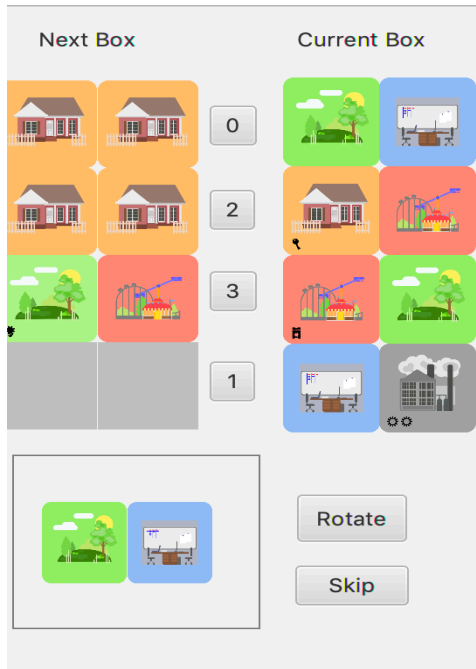


Abbildung 2 : gui representation (next box , current Box)

On the right side of the window, there are four fields, the biggest field with gray background color is the human field, the field with red background color is for Bot1, the field with green background color is for Bot2 and the field with yellow background color is for Bot3.

There are also four buttons on the bottom of the windows. Up, left, right and down buttons, these buttons are used for changing the location of town hall or city center.

If player clicks on up, city domino is moving one cell to top of the field. If he clicks on right it moves to the rights side. The functionality of left and down buttons are similar to two previous buttons, but moves to left or bottom side. when a card is placed on the field it cannot be moved individually, only together with the city center.

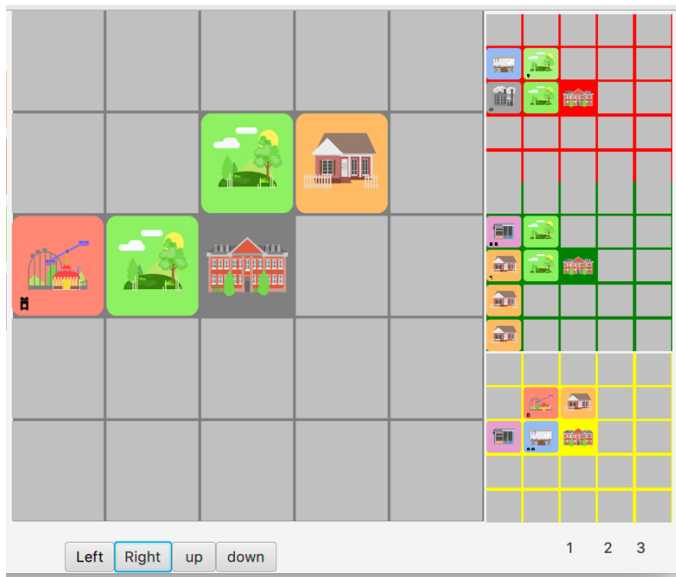


Abbildung 3 : Gui representation(Four fields)

2.3.2 Rules of the game

“The first card must border the city centre. The city centre may be bordered by any part of the city. Every following card must border either the city centre or an other card on the field. If you place a card next to another card, at least one half with one side must border on an identical district type of the lying card.

If the card for the current round does not match either the city centre or a card that is already on display, it is discarded.

All playing cards must fit into the 5*5 hex, no half may protrude. However, the city centre does not have to be in the middle, but can be moved during the game, which means that all cards that have already been placed are also moved.

Once a card is placed on the field it cannot be moved individually, only together with the city centre.

2.3.3 End of game

If all playing cards have been drawn out of the bag and placed or discarded by the selection areas on the playing fields, the points are determined.

- Each city consists of several districts. A district consists of horizontally and/or vertically connected cells of the same district type. The city centre does not belong to any district.
- The points of a district result from the number of its cells multiplied by the number of prestige symbols it contains.
- Within a city there can be several separate districts of the same type. Each district must be evaluated individually.
- Districts without prestige symbols do not score points.

For the evaluation, the sum of the points for each player's areas is determined. The player with the most points wins. If there is a tie, the player with the largest single area wins. If there is also a tie, both players win equally.“ (Nills, 2018)

2.4 Here are all error messages

2.4.1 Renaming the package

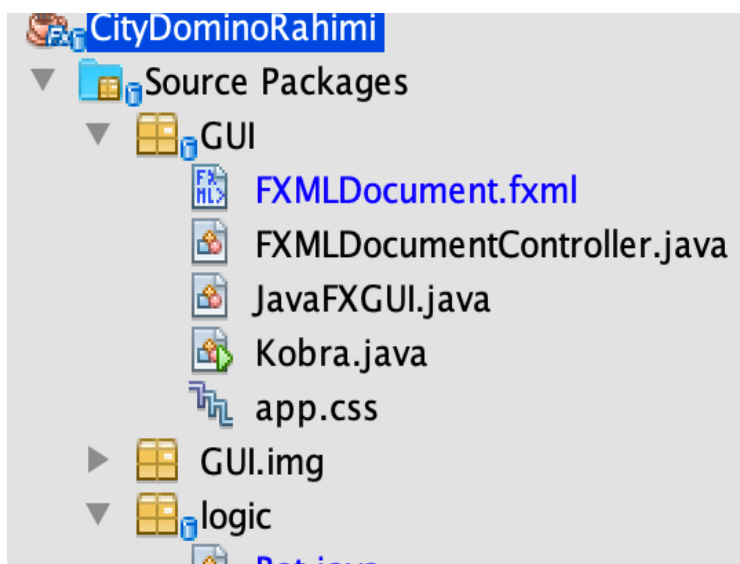


Abbildung 4 : error in package name

Error occurs when you run program or debug that. We have many different kinds of error in java programming. For example, semantic error and syntactic error.

Incorrect naming of the Java file: this error occurs when your program does not have correct name here all package name should be lower case, for first time I put the all package names to uppercase and later try to change them to lower case, but GUI package does not be changed to lowercase and when I tried to change it I have got error in running program. I tried many ways but could not resolve this problem.

2.4.2 save/load

Error message	cause	Corrective action
The type of chosen file is not valid	The type of chosen file for loading is not valid	File with correct format should be selected.
Save without any format. Don't write the file format	When loading the myfile.txt It is problem,	Do not load a file with format type like myFile.txt Just load myfile

Abbildung 5 : table of showing the error

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 softwares 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 JavaFXML applications.
- UMLdiagram: I used this plugin to organize and design my organization plan of my project.

3.2 Problem analysis and realization

This part consists of components which needs for gui representation. The following analysis and description get more information about every element that is used in City domino game.

The City domino game some following javaFx components:

- Field
- Next box
- current box
- selection box
- rotate button
- menu bar

3.2.1 GUI representation of playing field

Problem analysis

The field for City domino should be 5*5 size which first 5 represent the number of rows and the second represent the number of columns. So, in this game we need four fields for four players. Every cell is filled with a domino with type of image. In the middle of field, we have City Center image. it is located exactly in the middle of the field. When human is turn, he or she drag his tiles from selection area and drop it to his or her field. If other player's turn, their tiles will be shown in their fields. In every cell just one tile should be placed.

Realization analysis

A JavaFX GridPane is a layout component which lays out its child components in a grid. The size of the cells in the grid depends on the components displayed in the GridPane, but there are some rules. All cells in the same row will have the same height, and all cells in the same column will have the same width.

So, in this game gridpane is a suitable choice for showing the field in rows and columns.

The javaFx components that used inside the gridpane is image view.

imageView

The ImageView is a Node used for painting images loaded with Image class. it is a component which helps you to display images on JavaFX. You can also apply effects to

display images such as rotate, zoom in and out,... for this game, we need image view for showing the image inside each cell.

Realization description

The playing field is represented by a GridPane with five rows/columns each. In city domino game we have four fields for four players. All fields have the same size which is 5*5.

When a player drags his cards (with a method that handle the event) to the her / his field, cards should be shown in every cell of the field according to logic in game class. The two dimensional image view is added to every field by addImagesToGrid(GridPane grid) method in FXMLDocumentController.java class. If size of window is changed only the human field is resizing. Other fields is not changed during resizing the windows.

3.2.2 GUI representation of next box

Problem analysis

We have another area called next box area, when a new game starts this box is filled with random domino getting from stack. Each player selects one of the dominos from this area.

Realization analysis

The GridPane is also used for this area, that GridPane clearly represents the functionality of next box. So, for this purpose I have used 2*4 size(Two columns and four rows).

Inside the GridPane imageview is used. This component is suitable for showing the images.

Realization description

The size of this next box is 2*4 which two is representing the two parts of our dominos and four is according to number of players. In this game we have four players , so the second dimension of next box and also current box is four.

When player is selecting his domino, an even listener is calling a method from logic called `clickOnNextBox(int index)` which is storing the selected domino in an array with the type of choose. In first round human is the first player who plays. In sequence other players play.

3.2.3 GUI representation of current box

Problem analysis

When players select their dominos form next box, the selected dominos and their orders are stored in current box. Current domino and current player are picked from current box.

Realization analysis

Realization analysis of this box is the same as next box. The GridPane is used for this box with the size of 2*4 because the Gridpane is a good component for representing the content of an array. ImageView is another suitable javaFX component inside the gridpane. There is another component that is used to display the current player id nearby the current box. This buttons trigger with the current box.

Realization description

When each player selects his dominos from next box, their choices are stored in an array called nextChoose and then after a one round is finished the nextChoose is cloned to the currChoose(for keeping the turn of players) array. In next step, currChoose is sent to gui and displayed in the current box. These processes are done in logic in next player, tryDominosFromNextBox, clickOnNextBox and botsTurn methods in game class.

3.2.4 GUI representation of selection box

Problem analysis

The current domino is shown be in this box. When a new round is started the current domino will be displayed here. For displaying we only need 1*2 rows / columns image view because current domino consists of two images, so the selection box size should be the same as domino image size. When player clicks on rotate button, the current domino is rotating clockwise.

Realization analysis

For this box gridpane is an excellent choice for displaying the cells. ImageView inside GridPane is also necessary.

Realization description

Selection box always shows the current domino which is selected by current player. With set an event listener in gui, when current domino is displayed in this box, human can drag it into his board and then set this box domino to null. This process happens with setToChooseBox(Domino domino) , next player , setOnBorad and clickOnNextBox methods in game class. The setTochooseBox method triggers with a method in gui showInChooseBox for showing the current domino.

3.2.5 GUI representation of rotate button

Problem analysis

For putting the current domino into the borad/field, sometimes the current rotation of domino is not fit to the field, for example horizontal domino is not fit and may be vertical domino is

fitted into the field. So, we need a button to rotate the current domino in clockwise by adding one number to rotation.

Realization analysis

Button is a best choice for rotating the domino. There are different javafx components such as: label, pane. Without button it also possible when a player clicks on domino, just set the event listener to selected pane. With label is also possible but it is not a suitable component.

Realization description

When a player clicks on rotate button, an event handler triggers with a method in logic, whenever player clicks, the rotation value is increased by one in clockwise direction.

3.2.6 GUI representation of menu bar

Problem analysis

when a player clicks on this menu, three option is displayed, new game, save and load. Each of them have different functionality. if player wants to start the game he can clicks 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 clicks on load button.

Realization analysis

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

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 also is easy to use and understandable.

Realization description

This menu bar consists of three submenus, that for each of them an event listener is set. When one of these sub menus is clicked, an EventHandler method triggers processes in the logic.

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.

I have two important packages `gui` and `logic`, which is distinguishable with different background color.

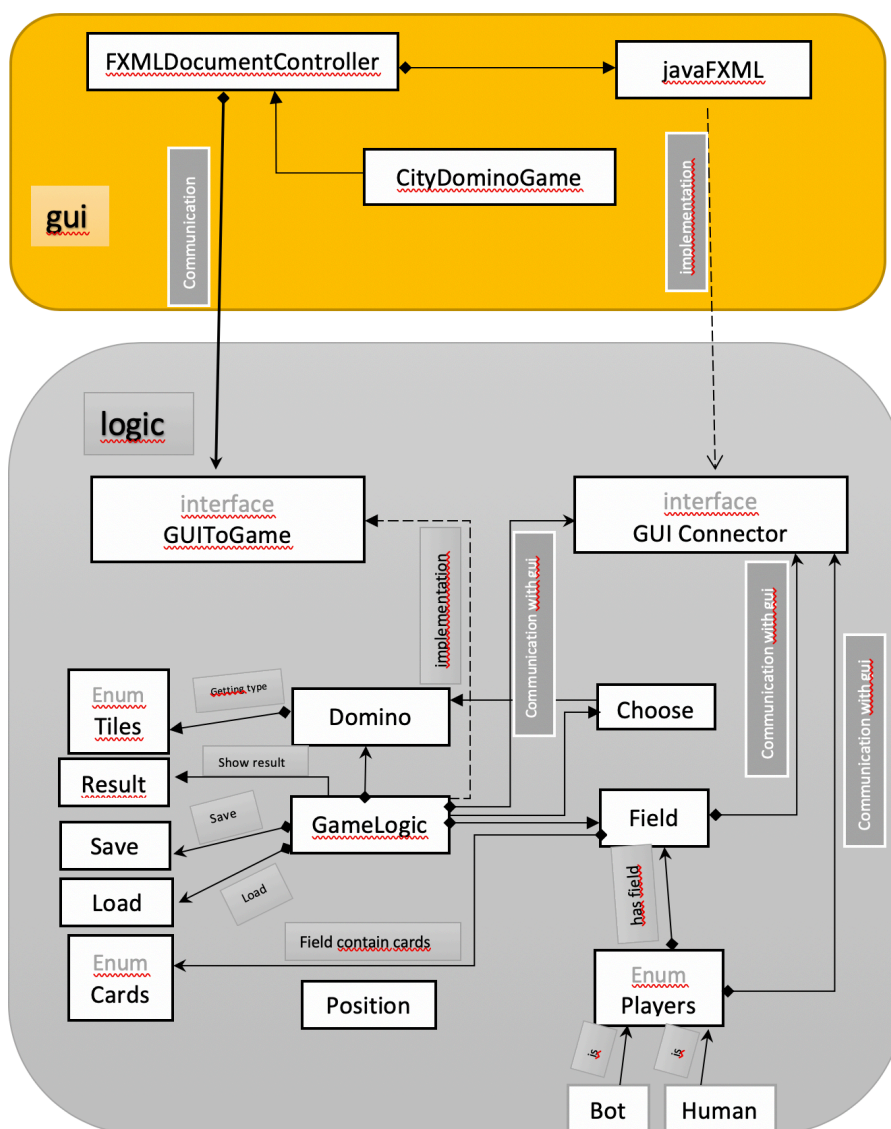


Abbildung 6 : Program organization plan

5 Description of basic classes

Here is description of important classes that I used to develop my program. In gui package two essential classes are javaFXML and FXMLDocumentController.

javaFXML: Concrete implementation of the GUIConnector used for the actual communication of the logic with the real gui.

FXMLDocumentController: define some event functionality for buttons and panes. This class has main role in gui representation.

In the logic package , the logic of game defined here in some important classes lilke, Game, Field, Players, Choose, Domino, Position, Tiles and Cards. Cards are single part of tiles , that fields are filled with these Tiles. We also have two important interfaces for triggering with gui, GUI Connector and GUIToGame.

Tiles: A game bag for all players contains 48 playing cards in the size of two cells, each of which displays one (possibly the same) district type on its two halves. The district types differ in image and background color. Each playing card has a defined value.

Cards: this class is responsible for type. We have two enum first one create enum with two parameter, symbol and type of cards. the second enum just stores the type of cards. We have eight types of cards.

Choose: This class returns the chosen domino by players and current player who chosen the domino.

GUI Connector: method the logic needs to display current game status on gui. Defined methods in here trigger with gui.

GUIToGame: defines the methods needed by the FXMLController for reacting to the users input.in here we have two important method save and load, which save method saves your program into a file and load method load your method form a file and show them in the gui.

City Domino Project

Players: creates the players with two type , human and bot. Each player has name, id, tiles and field. there is abstract method isBot, if the player is bot then returns true.

Position: Here is the position of dominos, each position has two values x and y value. There is another method getNeighbors which finds the neighbors of given position and return them. Each position has four neighbor.

Field: a field to place the domino on. Usually it is set to size 5*5. Each cell contains a value of a half domino or a value meaning it is empty. The first domino is city center which is set to the middle of the field.

Game : the main functionality of game is implemented in this class. process of starting the game , filling the stack and then next box, selecting the domino form next box, laying on the board, and the when game ends the count points is called and show in log.

Players are created with their fields. I have an array of players which stores the players of the game. The fields are created when players are created. NextChoose is an array which keeps the chosen domino of players. This array is copied to currChoose array when a new round is started. CurrChoose array keeps the chosen domino and also the round of players.

Save: save the all field values, players id, next box values, current box values into a file.

Load: load all fields cells, bag cards, banks cards from a file. AllFileString is the main method inside the load class, which is reading the give string file and convert every part of that string to related type. For example, change all fields to Field type, change bag value to domino and store them into a link list. I have used link list because it is easier and more comfortable to add cards to that. For testing there are two files that you can find them on my files, loadFileTest1 and loadFileTest2, that you could test my load function with these files.

6 Program testing

Here is some important test cases with result of them.

TEST CASE	RESULT
clicks on up button. excepted to move city center to up direction	working as excepted. if there is no empty row on top , moving is not working
clicks on down button. excepted to move city center to down direction	working as excepted. if there is no empty row on down , moving is not working
clicks on left button. excepted to move city center to left direction	working as excepted. if there is no empty row on left side , moving is not working
clicks on right button. excepted to move city center to right direction	working as excepted. if there is no empty row on right side , moving is not working
click on recycle bin picture . current player skip his round and others players are playing	working as excepted. other players play their round
drag the current domino from selection area. this domino only drag into human filed	shown domino in human field. in other field do not drag and drag is disable.
open window when game is finished	show result in new windows and see the name of winner.

Abbildung 7 : table of test cases and results

Bibliography

- 1) Nills. (2018, September 20). *Advanced Programming Structures*.
- 2) WebSite, o. (n.d.). Retrieved from <https://www.java.com/en/download/help/sysreq.xml>

City Domino Project