2020

DOCUMENTATION

LASER MAZE

JUAN MANUEL PALTA CORTES

FUNCTIONAL REQUERIMENTS:

FR#1THE CELLS MUST BE IDENTIFIABLE: Each of the cells of the grid can be identified through a nomenclature in which the row is given by a whole number (the first row is assigned the number 1, the second row the number 2, and so on), and the column It is given by an uppercase letter (the first column is A, the second B, and so on). Therefore, there will be a maximum of 26 columns (for the 26 letters of the English alphabet).

FR#2 SHOOT LASER: You can shoot a laser beam horizontally or vertically from any cell on the edge of the grid**. You only can shoot from a border.**

The program requests the cell coordinate to perform the shot, if you shoot from a corner, you need put the direction of the shoot (V:VERTICAL OR H:HORIZONTAL).

FR#3 MENU: the program must have 1 menu with three options, play, see score and exit.

FR#4 CREATE GRID AND SHOW GRID: The user must enter the number of mirrors, rows, and columns. With these data the grid will be created, and it will be shown in console.

FR#5 PUT MIRRORS: The mirrors must be randomly placed on the grid also the orientation of them.

FR#6 SHOW END & START: When shoot, a grid is displayed in console which will indicate where the shot entered and where it left using S OR E.

FR#7 LOCATE A CELL WITH A MIRROR: The user indicates the cell where he thinks there is a mirror, he will indicate the orientation of the mirror and if he is right it will be revealed. a command will be used for this function example: L2AR.

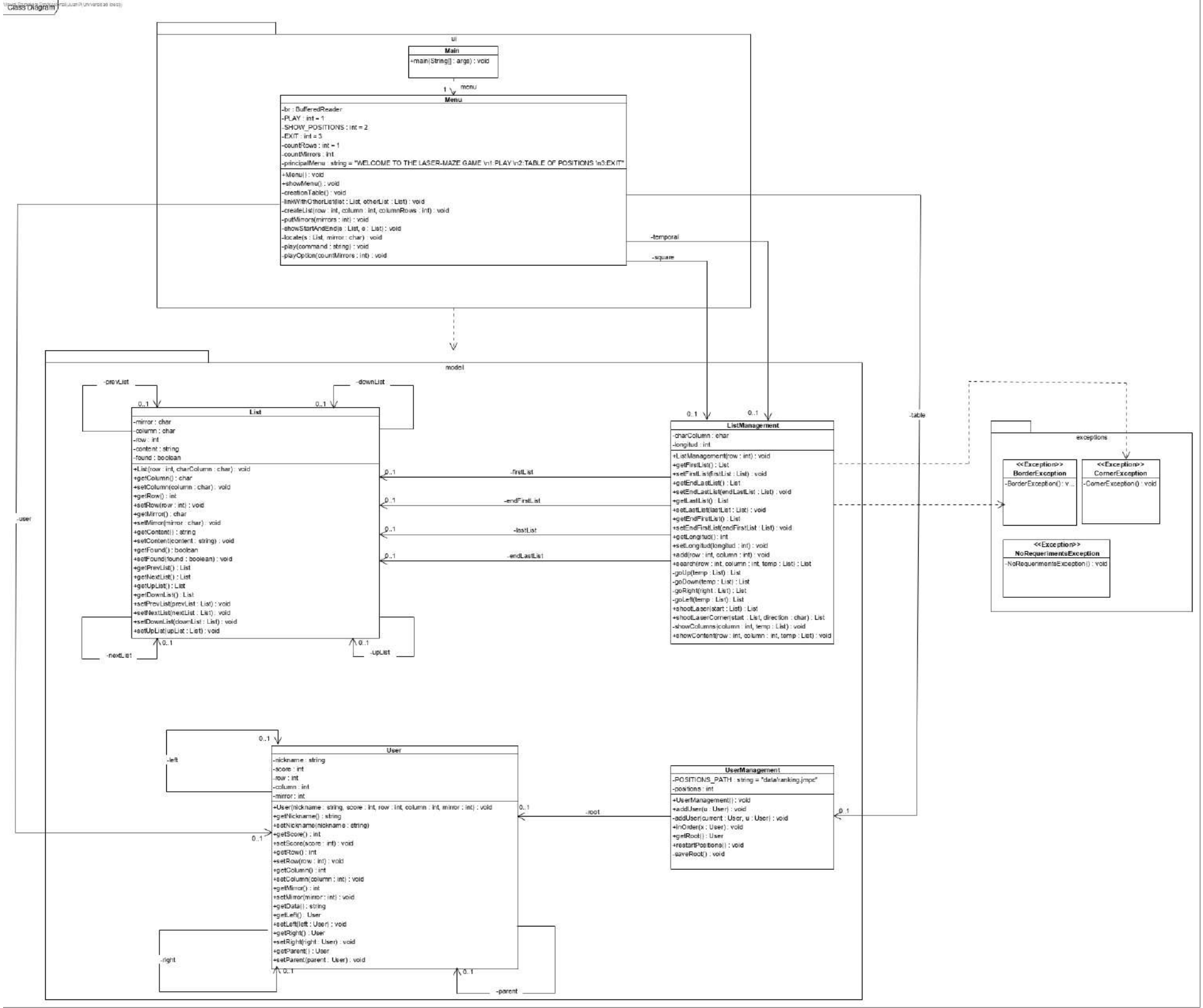
FR#8 SHOW MIRRORS REMAINING: The remaining mirrors and the user's nickname will be displayed every time they make a move.

FR#9 RETURN TO MENU: the user can only return to the menu if they win the game or enter the menu command.

FR10# CALCULATE SCORE: When you return to the menu, a score is calculated for the user that must be stored in a binary search tree ordered by score.

FR#11 SHOW SCORE TABLE: When selecting option 2 from the menu, the binary tree is scrolled in order and a ranking by score will be displayed in console showing the users data.

FR#12 SAVE SCORE TABLE: The score table must be persistent.


Class Diagram
ui
Main
+main(String[] : args) : void
1
menu
Menu
-br : BufferedReader
-PLAY : int = 1
-SHOW_POSITIONS : int = 2
-EXIT : int = 3
-countRows : int = 1
-countMirrors : int
-principalMenu : string = "WELCOME TO THE LASER-MAZE GAME \n1:PLAY\n2:TABLE OF POSITIONS \n3:EXIT"
+Menu() : void
+showMenu() : void
-creationTable() : void
-linkWithOtherList(list : List, otherList : List) : void
-createList(row : int, column : int, columnRows : int) : void
-putMirrors(mirrors : int) : void
-showStartAndEnd(a : List, e : List) : void
-locate(s : List, mirror : char) : void
-play(command : string) : void
-playOption(countMirrors : int) : void
-temporal
-square
model
-prevList
-downList
0..1
List
-mirror : char
-column : char
-row : int
-content : string
-found : boolean
+List(row : int, charColumn : char) : void
+getColumn() : char
+setColumn(column : char) : void
+getRow() : int
+setRow(row : int) : void
+getMirror() : char
+setMirror(mirror : char) : void
+getContent() : string
+setContent(content : string) : void
+getFound() : boolean
+setFound(found : boolean) : void
+getPrevList() : List
+getNextList() : List
+getUpList() : List
+getDownList() : List
+setPrevList(prevList : List) : void
+setNextList(nextList : List) : void
+setDownList(downList : List) : void
+setUpList(upList : List) : void
-nextList
-upList
-firstList
-endFirstList
-lastList
-endLastList
ListManagement
-charColumn : char
-longitud : int
+ListManagement(row : int) : void
+getFirstList() : List
+setFirstList(firstList : List) : void
+getEndLastList() : List
+setEndLastList(endLastList : List) : void
+getLastList() : List
+setLastList(lastList : List) : void
+getEndFirstList() : List
+setEndFirstList(endFirstList : List) : void
+getLongitud() : int
+setLongitud(longitud : int) : void
+add(row : int, column : int) : void
+search(row : int, column : int, temp : List) : List
-goUp(temp : List) : List
-goDown(temp : List) : List
-goRight(right : List) : List
-goLeft(temp : List) : List
+shootLaser(start : List) : List
+shootLaserCorner(start : List, direction : char) : List
-showColumns(column : int, temp : List) : void
+showContent(row : int, column : int, temp : List) : void
-table
exceptions
<<Exception>>
BorderException
-BorderException() : v...
<<Exception>>
CornerException
-CornerException() : void
<<Exception>>
NoRequerimentsException
-NoRequerimentsException() : void
-user
User
-nickname : string
-score : int
-row : int
-column : int
-mirror : int
+User(nickname : string, score : int, row : int, column : int, mirror : int) : void
+getNickname() : string
+setNickname(nickname : string)
+getScore() : int
+setScore(score : int) : void
+getRow() : int
+setRow(row : int) : void
+getColumn() : int
+setColumn(column : int) : void
+getMirror() : int
+setMirror(mirror : int) : void
+getData() : string
+getLeft() : User
+setLeft(left : User) : void
+getRight() : User
+setRight(right : User) : void
+getParent() : User
+setParent(parent : User) : void
-left
-right
-parent
-root
UserManagement
-POSITIONS_PATH : string = "data/ranking.jmpc"
-positions : int
+UserManagement() : void
+addUser(u : User) : void
-addUser(current : User, u : User) : void
+inOrder(x : User) : void
+getRoot() : User
+restartPositions() : void
-saveRoot() : void