Snakes and Ladders

Integrator Project 2

Jean Carlo Alvarez Lopez

Systems engineering department, Universidad ICESI.

Juan Reyes

May 10th, 2021

Functional requirements

The program should be able to:

RF1. Display a menu with 3 options.

RF1.1 Create a new game with a size, number of snakes and ladders determined by the user's input after the first option is selected.

RF1.1.1 Add players into the game randomly or by a given input when the game is being created.

RF1.1.2 Display the enumerated game cells after the game is created.

RF1.1.3 Allow the user to switch between four more options while a game is currently being played.

RF1.1.3.1 Move the first player and so on after the user presses the enter key (this being the first option) and displaying the dice's roll.

RF1.1.3.1.1 Display the cells with their players and letting know the user whether they're a snake or ladder.

RF1.1.3.1.2 Let the user that which player has won after reaching the last cell.

RF1.1.3.2 Display the enumerated board after the user writes "enum".

RF1.1.3.3 Return to the main menu if the user writes "menu".

RF1.1.3.4 Simulate how the game would continue if the user writes "simul", making a move automatically every 2 seconds until someone wins.

RF1.2 Display the ranking score for all the winners of the previous games.

RF1.3 Close the game.

The class diagram in the next page…

model

Ranking
- counter : int

+ addScore(p : int, n : string, pi : char) : void
+ addScore(current : Score, newScore : Score) : void
+ inOrder(score : Score) : void
+ inOrder() : void
+ getRoot() : Score
+ Ranking()

Score
- points : int
- name : string
- piece : char

+ Score(name : string, points : int, pi : char)
+ getLeft() : Score
+ getRight() : Score
+ setLeft(left : Score) : void
+ setRight(right : Score) : void
+ setPoints(points : int) : void
+ getName() : string
+ getPiece() : char

Board
- snChar : char = 65
- ldChar : char = 106
- plChar : char = 945
- rows : int
- cols : int
- snakes : int
- ladders : int
- cellSize : int
- nPlayers : int

+ Board(n : int, m : int, s : int, l : int)
+ createBoard() : void
+ createRow(r : int, c : int, firstRowCell : Cell) : void
+ createCol(r : int, c : int, prev : Cell) : void
+ enumerateBoard() : void
+ enumerateBoard(current : Cell, id : int) : void
+ movePlayer() : boolean
+ addPlayersByNumber(n : int) : void
+ addPlayersByChar(chars : String, current : int) : void
+ addPlayersByChar(chars : String) : void
+ addPlayer(toAdd : Player, toAddB : Player) : void
+ searchByCellID(i : int) : Cell
+ searchByCellID(i : int, current : Cell, r : int) : Cell
+ initialSetup() : void
+ printGameCell(current : Cell) : string
+ printEnumeratedCell(current : Cell) : string
+ printBoard(current : Cell, b : String, boe : boolean) : string
+ fixCell(c : String, ul : int) : string
+ printGameBoard() : string
+ printEnumeratedBoard() : string
+ searchRow(r : int) : Cell
+ searchRow(current : Cell, r : int) : Cell
+ searchCol(current : Cell, c : int) : Cell
+ addSnakes(sol : boolean) : void
+ addSnakesAndLadders() : void
+ getActualPlayer() : Player
+ getRows() : int
+ getCols() : int

Cell
- row : int
- col : int
- cellChar : char
- id : int

+ Cell(r : int, c : int)
+ getCellChar() : char
+ setCellChar(cellChar : char) : void
+ getId() : int
+ setId(id : int) : void
+ getRow() : int
+ getCol() : int
+ getLeft() : Cell
+ getRight() : Cell
+ getUp() : Cell
+ getDown() : Cell
+ setLeft(left : Cell) : void
+ setRight(right : Cell) : void
+ setUp(up : Cell) : void
+ setDown(down : Cell) : void
+ addPlayer(toAdd : Player) : void
+ searchPlayer(c : char, current : Player) : Player
+ searchPlayer(c : char) : Player
+ removePlayer(c : char) : void
+ playersPieces() : string
+ playersPieces(current : Player, pieces : string) : ...

Player
- piece : char
- moves : int

+ Player(p : char, c : Cell)
+ getCurrentCell() : Cell
+ setCurrentCell(currentCell : Cell) : void
+ setMoves(moves : int) : void
+ getMoves() : int
+ getPiece() : char
+ getNext() : Player
+ getPrevious() : Player
+ setNext(nx : Player)
+ setPrevious(p : Player)

ui

Menu
+ Menu()
+ menuOptions() : void
+ mainMenu(i : int) : void
+ play(order : string, si : boolean)
+ simulation() : void
+ validateBoard(settings : string[]) : void

Main
+ main(args : string[]) : void

Associations:
-right 0..1, -root 0..1, -left 0..1, -ranking 1, -gameBoard 0..1, -menu 1, -swapsTo 0..1, -firstCell 0..1, -currentCell 1, -up 0..1, -down 0..1, -firstBoardPlayer 0..1, -firstPlayer 0..*, -next 1, -previous 1