Data Structures Project Report

Hakan Kocaman 2321021003
Ahmet Emin Uğurlu 2321021019

Game part:

After running the program, “Main menu” frame appears. It expects an username that is not blank. There are two buttons “Play” and “Scoreboard”. When we clicked into “Play” button game starts with Level 1. Game screen have: Cells 1 to 30, result console, score board, “Roll”,”Restart”,”Back” buttons. Game starts by first cell that is “ST” cell.

By Roll button we can roll a dice 1 to 6 and we can move according to dice result. When we move to another cell, cell’s button becomes enabled and a label indicates it so we can know where we are. Most of cells have meaning that : “+10 point” , “-5 point”, ”Big prize”,”Go forward or backward”. Sometimes they dont have a meaning so they do nothing.

Level 1 contains only score risers and reducers. Level 2 also includes movers. All these rolling dice, moving by movers and getting points appears on the console so we can follow it. At Level 1 when it reaches on “FN” cell game asks if we want to continue by level 2 or stop. Stop button ends the game, shows stats, writes scores at some .txt file and returns to main menu. Continue button writes scores at some .txt file , opens Level 2 screen and scoreboard resets itself.

Level 2 screen has mover indicators, unlike Level 1. When it reaches on “FN” game shows stats, writes scores at some .txt file and returns to main menu. “Restart” button restarts the game from Level 1 and resets scoreboard. “Back” button returns back to main menu.

Cell node contains an action, next cell, prev cell, alt cell, jbutton and jlabel. Game got a Cell list in roll actions it moves by next cell or prev cell. In sophisticated moves(Level 2 movers) it moves by alt cell field. Cell’s action field determines the action(“+50”,”+10”,”-5”)

Score Board part:

The scoreboard opens after clicking the “Scoreboard” button. It creates a new, empty Binary Search Tree. It opens a frame, and after entering a username and clicking the “List” button, it reads lines from the score.txt file(formatted as username, level, score), checks if the line contains the name that the user searches, and splits them by comma

“,”. Then, it creates a Player object with username, level, and score data, then creates a Binary Search Tree Node and gives that Player object as an argument to the insert() method. Then, with the insert() method, it adds the new Node by comparing their score values and decides whether to locate it on the left node or right node. After the Binary Search Tree creation is completed, the code starts to navigate through it using the traverseInOrder() method (comparing by the score value of the Player object in the Node's data), and takes each Node's data, which contains the Player object, during every recursive iteration, and adds each Player object to a new CellList (linked list). In this way, Player objects are added sequentially to the linked list. After that, the linked list is read with a for loop, which takes all values from the Player objects and writes them to a JLabel. After that, to find the player's worst and best scores, the findMin() and findMax() methods are used, which traverse all the way to the left and right respectively, and the results are written to JLabels.