**CSE1142 Computer Programming II, Spring 2020**

**Term Project**

The Ball Game

Ahmet Hakan Beşel 150119007

Semir Tatlı 150119004

Date Submitted: May 10, 2020

**Problem Definition**

There is a game which is have a ball and tiles on a 4x4 board. The player must create a way for ball with dragging the tiles. There are 5 levels and each level are different from previous one. The player cannot play the next level without completing the current level. We need to create main menu for the player. In the main menu there should be some buttons for start the game, closing the game etc. When player clicked the start button level menu must open. In level menu player can choose only level 1. After completing levels, the other buttons must be able to click. Game window opens when player chooses a level.

When player decided and clicked a level the game board design must be like the text file of the level. The tiles shape and direction must be like the directives in the text files. After the board created the program should check the solution, if the solution is right the program should understand that. If the solution is right current level should mark as completed. And the next level should be available.

**Implementation Details**

***Main.java***

Application

Main

|  |  |
| --- | --- |
| **Main** | |
| +  +  +  +  +  +  -  - | GAME\_TITLE: String  VBOX\_WIDTH: double  WINDOW\_HEIGHT:double  WINDOW\_WIDTH:double  BACKGROUND\_COLOR: String  BUTTON\_COLOR: String  PRIMARY\_STAGE: Stage  player: Player |
| -  -  -  -  -  -  -  -  +  +  +  + | mainMenu(): void  howToPlay(): void  credits(): void  headVBox(): VBox  bodyVBox(): VBox  footerVBox(): VBox  defaultScene(sceneTitle: String, bodyText: String): void  changeScene(rootPane: StackPane): void  levelMenu():void  showLevel(level: int): void  getPlayer(): Player  setPlayer(player: Player): void |

The main menu is created by the Main class. GAME\_TITLE is for the top of the main menu. There will be several buttons in our game, so we decided to put them into VBox’s. We determined the size of all VBox’s. We did our game window close to resizing, and we determined the size of the game window. We gave different colours to our buttons and backscreen. The player cannot open the 2. Level without completing 1. Level so we created a Player object to take the information of completed levels. We designed the default screen with 3 VBox. One for the top, one for the body and one for the bottom. We created methods for top, body and middle VBox. The top VBox is for the label, as the name of the game or current level etc. Body VBox is for the buttons in the main menu and the level menu. For the game, we put the board in the VBox. The bottom VBox is for the back button in the level menu and the game. We did a method for creating the main menu. There is a VBox on the top for the game title, and a body VBox for the buttons. In the main menu we put buttons for learning about the authors, learning how to play the game, closing the game and start the game. If the player clicks to credits or how to play, there will be a default scene with a text in the center. If a player clicks the exit the game will close. If the player clicks the start button the levelMenu method will work with the same design but the level buttons will be in a gridpane. In the level menu, the buttons are for choosing the level. The player cannot play a level if the previous ones did not complete. If the player chooses a level which is open the game will open. Game window design is like the other windows but instead of buttons, there is a game board on the VBox. When a level chosen showLevel method creates a Board object and the game board is created by Board class.

***Board.java***

GridPane

Board

|  |  |
| --- | --- |
| **Board** | |
| -  -  -  -  -  -  -  -  -  +  + | level: int  clickedTiles: Tile[]  numberOfMoves: int  tiles: Tile[]  pipes: ArrayList<Tile>  a: int  b: int  ball: Circle  DURATION: int  currentLevel: Level  Main: Main |
| +  -  +  -  -  -  -  -  -  -  -  -  +  + | Board(level: int)  createBall(): void  findStartTile(): Tile  mousePressed(tile: Tile): void  resetClickedTiles(): void  swapTiles(): void  checkTileTypes(firstTile: Tile, secondTile: Tile): boolean  showAlert(): void  showInfo(): void  showNextLevel(): void  moveBall(): void  checkTileCoordinates(x1: int, y1:int, x2: int, y2:int): boolean  findWay(tile: Tile): boolean  getter/setter methods |

After the game object initialized in the main class, the constructor of the board is started. The board class extends grid pane, our board is a 4x4 grid pane. The level stands for a value of the current level. The currentLevel data field stands for getting the information about the current board from the text files. Clicked tiles array is for changing holding the two tiles who player wants to change. The numberofMoves stands for showing the number of moves to the player while the player plays the game. tiles array is holding the information of tiles for checking the solution. The pipes array list is for creating the way for the ball animation after the level completed. a and b values are for int values of the 2 indexes of tiles array, which are going to be changed. The ball is the ball which will be rolling. Main is for when the player completes the level automatic opening of next level. In the constructor of Board class with currentLevel the information’s about the current level is collecting and in a nested, for loop, every tile is created separately and goes to its place in the GridPane. tiles array is initializing, an if statement checks for a click for the changing two arrays. createBall puts the ball and sets its properties. If the player clicks to a tile which can be movable the mousePressed method works. This method changes the opacity of clicked tiles and puts it in the clickedTiles array if one of it is empty. If both clickedTiles aren’t empty it checks are they able to change with checkTileTypes method. If they can't change the program pop-ups and alert with showAlert method and makes clickedTiles null with resetClickedTiles method. If the tiles can change swapTiles method changes the tiles. This method also checks the coordinates of clickedTiles, if they are too far from each other program pop-ups and alert and sets clickedTiles to null with resetClickedTiles method. If they are 1 block away from each other then clickedTiles and tiles array changes. And the number of moves increases. Now the program checks for the solution with findWay method with the argument of starting tile. The findWay method works with a while loop. It has lastmove value for the direction of the last move, the boolean loop is for while loop, boolean solved is for the solution. In the while loop, there is a switch case for the tiles. It looks at the lastmove and the current tile, then checks if the next tile is appropriate and add current tile to the pipes array list. Every time it checks if the current tile is the end. If the solution is wrong it breaks the loop with assigning loop to false, if the solution is right it breaks the while loop with assigning the loop false and assigning the solved true. If the solution is false player keeps playing, if the solution is right playedLevels of the player object set as the current level is completed and the animation plays with moveBall method.

***Level.java***

|  |  |
| --- | --- |
| **Level** | |
| -  -  - | Levels: String[]  tiles: int[][]  rows:ArrayList<String> |
| +  -  -  +  + | Level(levelFile: String)  readRowList(): void  readLevelFile(filePath: String): void  addRow(row: String): void  getter/setter methods |

There is a string array for level text names. The multidimensional int array tiles takes the information of tiles properties: index value in Grid pane, type and direction. String array list rows holds the rows in the level text file. In the constructor with the levelname arguments, there are two methods for filling the tiles. readLevelFile with levelFile arguments reads the file and adds all the rows into the rows arraylist with addRow method. readRowsList fills the tiles array by assigning a rowData string array for holding the rows information.

***Player.java***

|  |  |
| --- | --- |
| **Player** | |
| -  - | id: int  completedLevels: int[] |
| +  +  +  +  + | Player()  isPlayable(levelNumber: int): boolean  isPlayed(levelNumber: int): boolean  addPlayedLevels(levelNumber: int): void  getter/setter methods |

Every player has a unique id. completedLevels int array is in the sizes of Levels string array and it is for holding the completed Levels for player. Player constructor is empty it just creates a player object. There are methods independent from the constructor. isPlayable checks is the player completes the previous level with int levelNumber argument which is the level questioning its playability. isPlayed is a boolean method for learning is the levelNumber in the arguments already played. addPlayedLevel marks the current level as completed.

***Tile.java***

Pane

Tile

|  |  |
| --- | --- |
| **Tile** | |
| -  -  -  -  -  -  - | tileId: int  tileType: int  tileDirection: int  PRIMARY\_COLOR: String  SECONDARY\_COLOR: String  RED\_COLOR: String  BLUE\_COLOR: String |
| +  +  +  +  + | Tile(tileId: int, tileType: int, tileDirection: int)  draw(): void  tileType(tileName: String): int  tileDirection(tileDirection: String): int  getter/setter methods |

Tile class is creating tiles in a pane, then in the Board class, this tile is added into the Grid pane. int tileId is for the index of the tiles in the Grid pane. tileType and tileDirection holds the int value of the current tiles properties. There are string data fields for setting the color and theme of the tiles. The constructor of Tile class is having three arguments which are tileId, tileType, tileDirection. In the constructor, those properties are setting with setter methods and the size of the pane setting. The draw method is for creating the tile with current properties. In the draw method program creates tiles with if and else statements. In the if and else statements program going into a switch case for the direction of the tiles. For some tiles, this method just sets the color, ads a rectangle to the pane, ads a rectangle and an arc, ads an arc.

The ball cannot roll in the curved tiles very good. We changed the colors and themes of the game, we tried to make a good design. We added the alert property, in a wrong move or if the player completes the messages the alerts pop-ups.

**Test Cases**

|  |  |
| --- | --- |
| **Main Menu**  We added a main menu to the game.  Player can start to play, learn how to play, see credits of the game or close the game. |  |
| **How to Play** Playing guide screen |  |
| **Credits**  Credits screen |  |
| **Levels Menu** Level choosing menu  Buttons with padlock icon is not playable because of previous levels have not completed yet. |  |
| **Level 1**  If you complete a level information alert will be appear on the screen. When you click the OK button, it opens the next level. |  |
| **Level 2**  If you try to move unmovable tiles (static tiles -blue-, empty -grey-, starter -blue-, end -red-) warning alert will appear on the screen. And it will ask you to try a new move. |  |
| **Level 3**  As you can see tiles can move to different locations. |  |
| **Level 4**  Number of your moves displayed bottom of the game board. The number increases when you make a valid move. |  |
| **Level 5** |  |
| **Level 6**  We created this level for our test purposes. This level includes all tile types with all directions. |  |