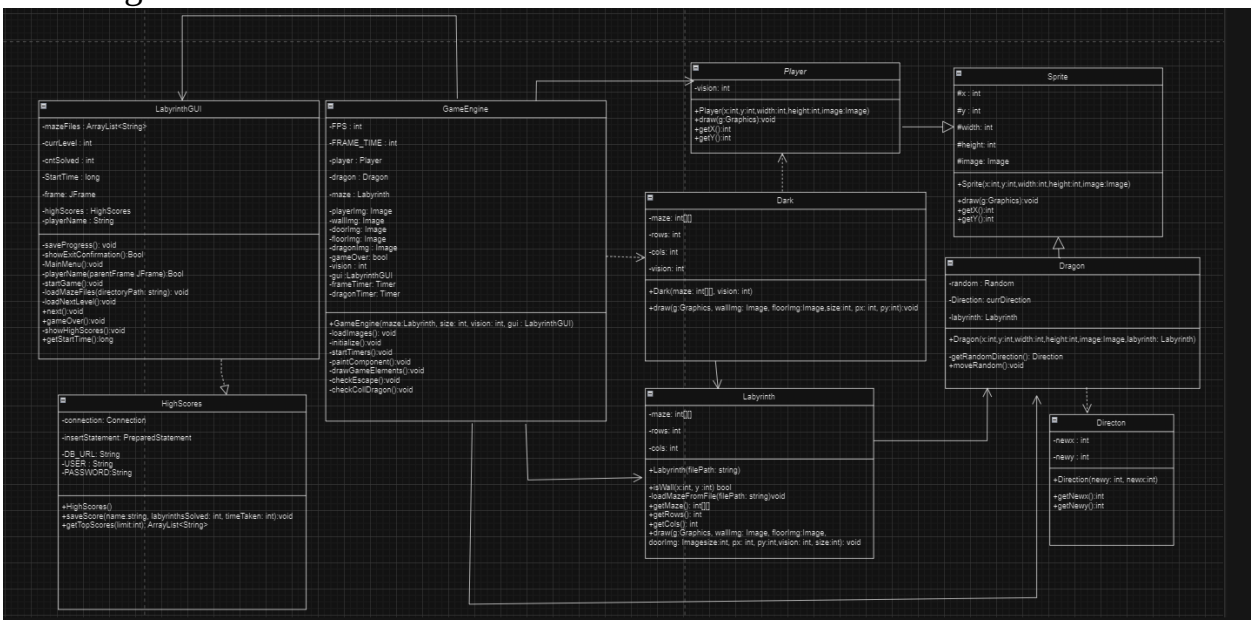
Mahammad Handamzada JBPPYN Jbppyn@inf.elte.hu	**3. assignment/3rd. Task**	**12.09.2024**

Task

Create the Labyrinth game, where objective of the player is to escape from this labyrinth. The player starts at the bottom left corner of the labyrinth. He has to get to the top right corner of the labyrinth as fast he can, avoiding a meeting with the evil dragon. The player can move only in four directions: left, right, up or down. There are several escape paths in all labyrinths. The dragon starts off from a randomly chosen position, and moves randomly in the labyrinth so that it choose a direction and goes in that direction until it reaches a wall. Then it chooses randomly a different direction. If the dragon gets to a neighboring field of the player, then the player dies. Because it is dark in the labyrinth, the player can see only the neighboring fields at a distance of 3 units. Record the number of how many labyrinths did the player solve, and if he loses his life, then save this number together with his name into the database. Create a menu item, which displays a highscore table of the players for the 10 best scores. Also, create a menu item which restarts the game. Take care that the player and the dragon cannot start off on walls.

UML Class Diagram

This diagram was made with Draw.io



Short Description of each method

Player Class

1. **Player(int x, int y, int width, int height, Image image, int visionRange)**
   Initializes the player with position, size, image, and vision range.
2. **int getRange()**
   Returns the player's vision range.
3. **void move(int dx, int dy, int[][] maze)**
   Moves the player within the maze if the target position is not a wall.

Dragon Class

1. **Dragon(int x, int y, int width, int height, Image image, Labyrinth labyrinth)**
   Initializes the dragon with its position, size, image, and reference to the labyrinth.
2. **void moveRandomly()**
   Moves the dragon to a random valid position within the labyrinth.

Dark Class

1. **Dark(int[][] maze, int visionRange)**
   Initializes the fog-of-war effect with the maze and vision range.
2. **void draw(Graphics g, Image wallImage, Image floorImage, int cellSize, int playerX, int playerY)**
   Draws the fog-of-war effect, only revealing tiles within the player's vision range.

Labyrinth Class

1. **Labyrinth(String filePath)**

Loads a labyrinth from a file.

2. **boolean isWall(int x, int y)**
   Checks if the given position is a wall or out-of-bounds.
3. **int[][] getMaze()**
   Returns the 2D array representing the maze layout.
4. **int getRows()**
   Returns the number of rows in the maze.
5. **int getCols()**
   Returns the number of columns in the maze.
6. **void draw(Graphics g, Image wallImage, Image floorImage, Image doorImage, int cellSize, int playerX, int playerY, int visionRange)**
   Draws the maze with appropriate images and fog-of-war.

HighScores Class

1. **HighScores()**
   Initializes the database connection and prepares the statement for saving scores.
2. **void saveScore(String name, int labyrinthsSolved, int timeTaken)**
   Saves a player's score to the database.
3. **ArrayList<String> getTopScores(int limit)**
   Retrieves the top scores from the database.

GameEngine Class

1. **GameEngine(Labyrinth maze, int cellSize, int visionRange, LabyrinthGUI gui)**
   Initializes the game engine with the maze, cell size, vision range, and GUI reference.
2. **void checkEscape ()**
   Checks if the player has reached the exit tile.
3. **void checkCollDragon ()**
   Checks if the dragon has caught the player.
4. **protected void paintComponent(Graphics g)**
   Draws the game elements and FPS on the screen.
5. **void addKeyListener(KeyInputHandler handler)**

Listens for player input to handle movement.

LabyrinthGUI Class

1. **LabyrinthGUI()**
   Initializes the main menu and connects to the high scores database.
2. **void mainMenu()**
   Displays the main menu with options to start the game or view high scores.
3. **boolean playerName(JFrame parentFrame)**
   Prompts the player to enter their name.
4. **void initializeGame()**
   Prepares the game by loading the maze files and starting the first level.
5. **void loadMazeFiles(String directoryPath)**
   Loads maze file paths from the specified directory.
6. **void loadNextLevel()**
   Advances to the next level or prompts the player when all levels are completed.
7. **void nextLevel()**
   Marks the current level as completed and moves to the next one.
8. **void gameOver()**
   Handles the game-over screen, offering options to restart or return to the main menu.
9. **void showHighScores()**
   Displays the top high scores from the database.
10. **void saveProgress()**
    Saves the player's progress to the database.
11. **long getStartTime()**
    gets time in the game.

Event-Handler Connections

Player Movement
Event: Keyboard key press (W, A, S, D).
Handler: KeyInputHandler class (in GameEngine).
Method: keyPressed(KeyEvent e)
Handles directional movement based on key input.

Dragon Movement
Event: Timer triggers every 200ms.
Handler: ActionListener in the dragonTimer (in GameEngine).
Method: dragon.moveRandomly()
Moves the dragon to a new position within the maze.

Game Rendering (FPS Updates)
Event: Timer triggers every 16ms (for 60 FPS).
Handler: ActionListener in the gameTimer (in GameEngine).
Method: paintComponent(Graphics g)
Redraws all game elements

Player Reaches Exit
Event: Player steps on the exit tile (2 in the maze).
Handler: checkPlayerAtExit() (in GameEngine).
Method: gui.nextLevel()
Signals LabyrinthGUI to load the next level.

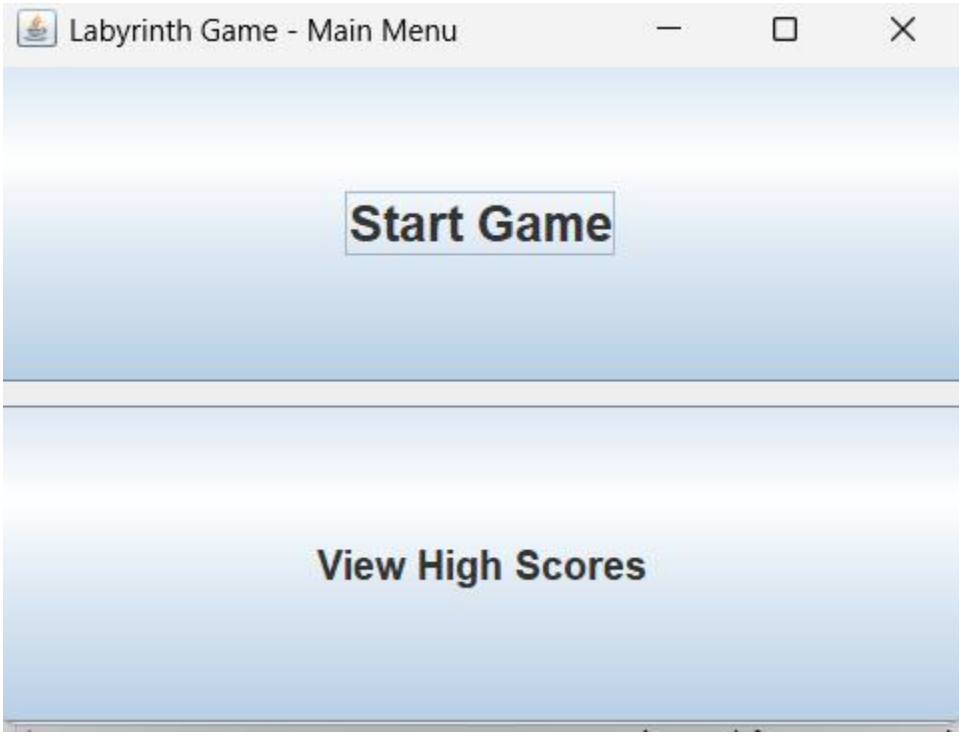
Player Dies (Collision with Dragon)
Event: Player is adjacent to the dragon.
Handler: checkCollisionWithDragon() (in GameEngine).
Method: gui.gameOver()
Signals LabyrinthGUI to handle the game-over logic

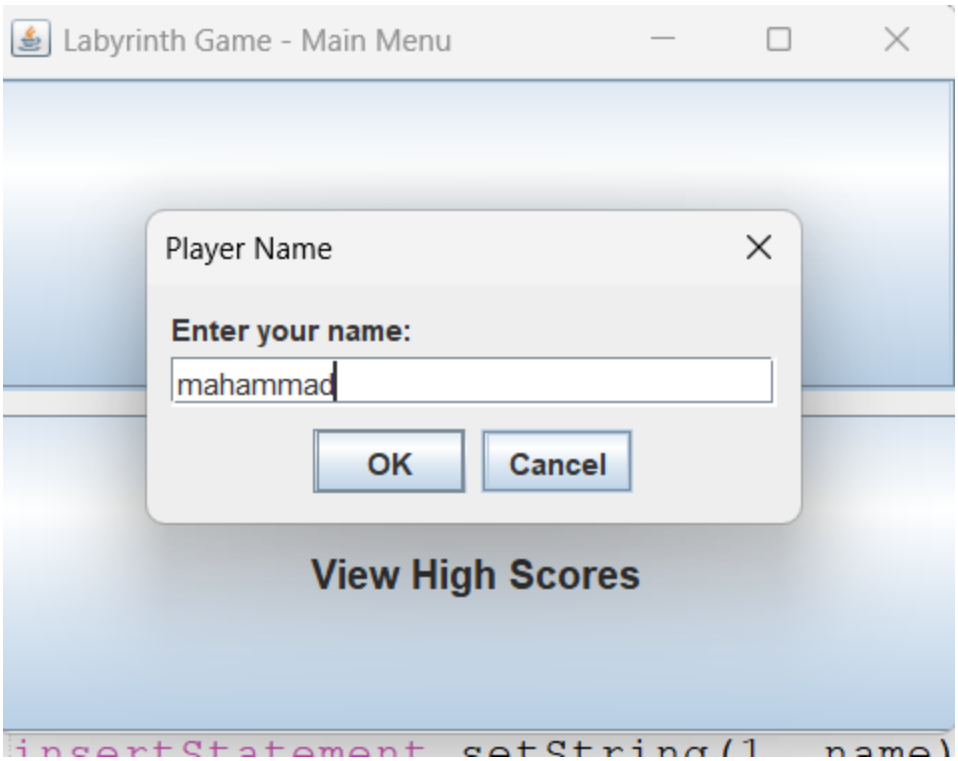
Game Initialization
Event: Player clicks "Start Game" in the main menu.
Handler: startButton.addActionListener() (in LabyrinthGUI).

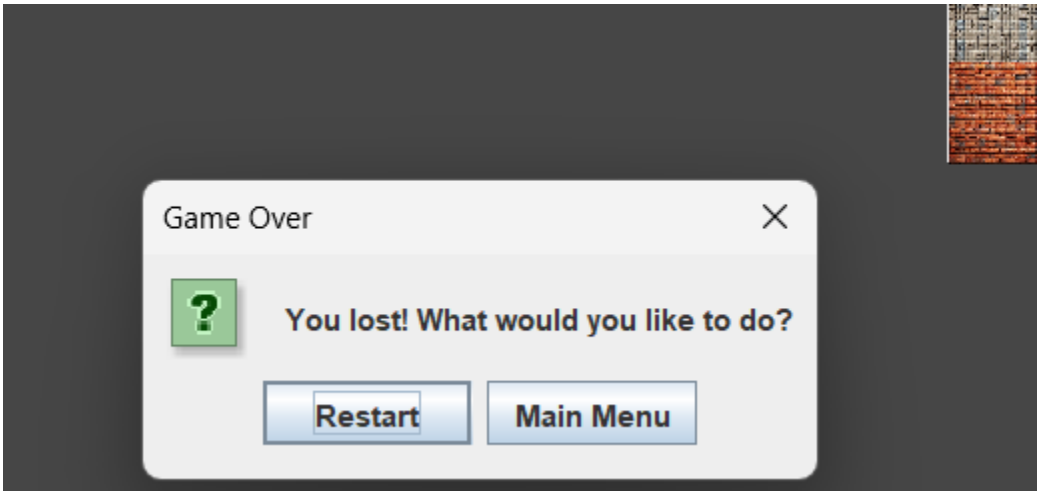
Method: initializeGame()
Loads maze files, resets progress, and starts the first level.

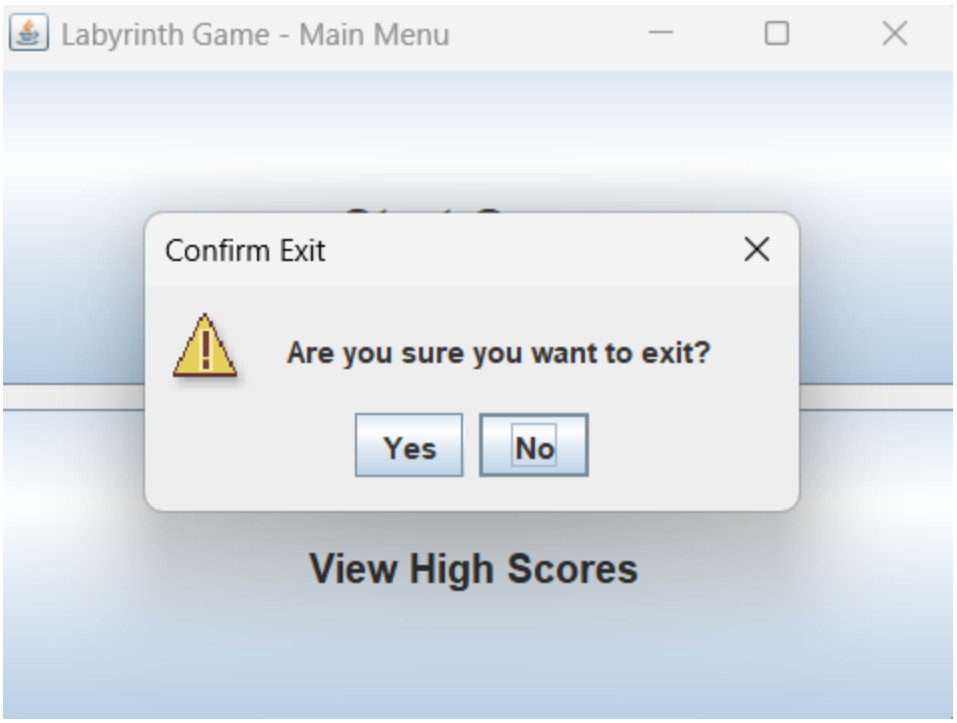
Save Progress
Event: Player exits the game or transitions to the main menu after completing some levels.
Handler: saveProgress() (in LabyrinthGUI).
Saves the player's progress

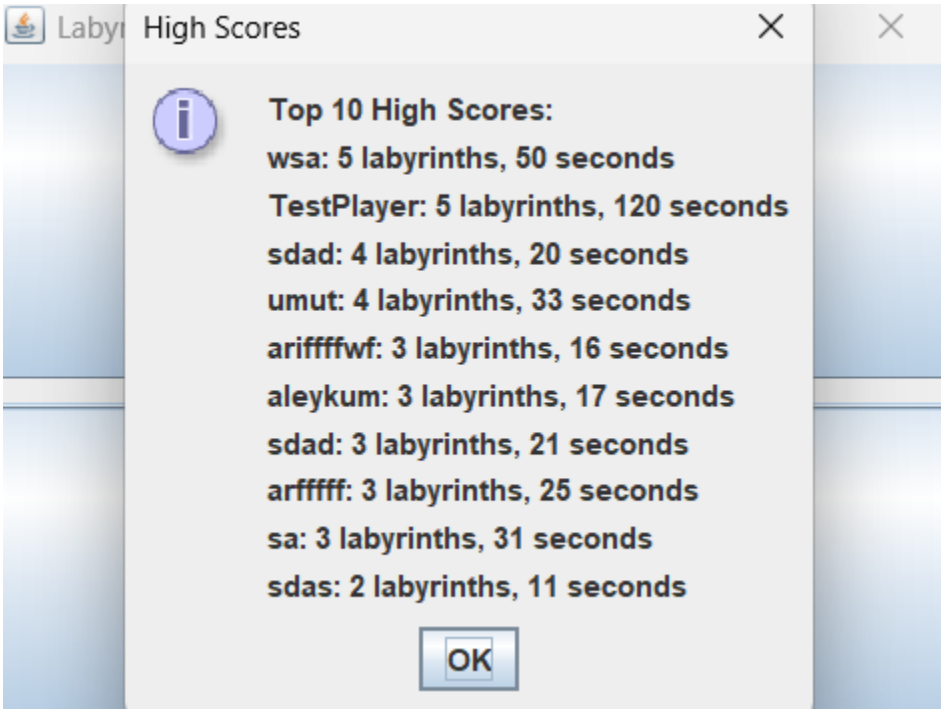
Exit Confirmation
Event: User closes the game window or selects "Exit" from menu.
Handler: showExitConfirmation() (in LabyrinthGUI).
Method: Prompts the user to confirm exiting the game.




Labyrinth Game - Main Menu
Player Name
Enter your name:
mahammad
OK
Cancel
View High Scores


Game Over
You lost! What would you like to do?
Restart
Main Menu


Labyrinth Game - Main Menu
Confirm Exit
Are you sure you want to exit?
Yes
No
View High Scores


High Scores
Top 10 High Scores:
wsa: 5 labyrinths, 50 seconds
TestPlayer: 5 labyrinths, 120 seconds
sdad: 4 labyrinths, 20 seconds
umut: 4 labyrinths, 33 seconds
ariffffwf: 3 labyrinths, 16 seconds
aleykum: 3 labyrinths, 17 seconds
sdad: 3 labyrinths, 21 seconds
arffffff: 3 labyrinths, 25 seconds
sa: 3 labyrinths, 31 seconds
sdas: 2 labyrinths, 11 seconds
OK