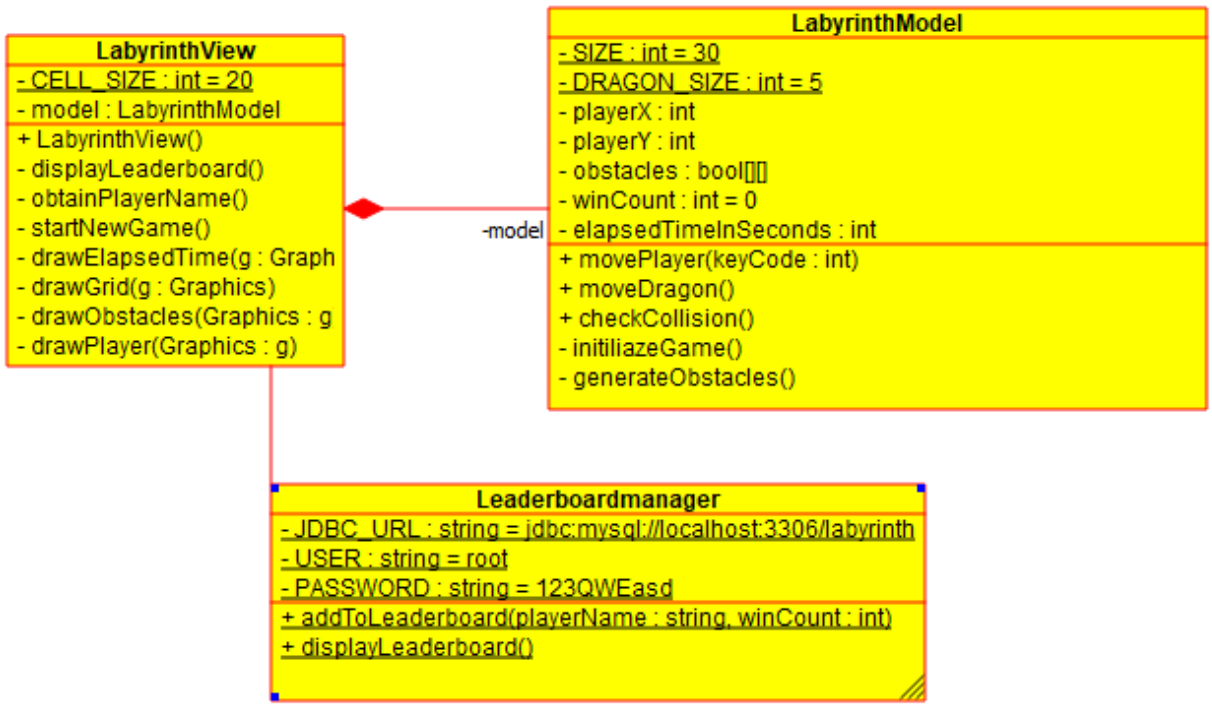
Aryan Shams Ansari, GTPHX8

Labyrinth

Create the Labyrinth game, where the objective of the player is to escape from this labyrinth. The player starts at the bottom left corner of the labyrinth. He must 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 chooses 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 3 units. Record the number of how many labyrinths the player solved, and if he loses his life, then save this number together with his name into the database. Create a menu item, which displays a high score 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.



The Program consists of a view(GUI) class, model class, leaderboard class.

Implementation:

This Java class, **LabyrinthModel,** represents the model for a labyrinth game. The labyrinth is a grid of a specified size (30x30), where the player and a dragon move around. The goal is for the player to reach a specific location in the labyrinth while avoiding collisions with a randomly moving dragon and obstacles.

Fields:

playerX, playerY: Current coordinates of the player.

dragonX, dragonY: Current coordinates of the dragon.

obstacles: 2D array indicating the presence of obstacles in the labyrinth.

winCount: Number of times the player has successfully reached the goal.

elapsedTimeInSeconds: Time elapsed in seconds.

playerName: Name of the player.

Methods:

movePlayer(int keyCode): Moves the player based on the specified key code (arrow keys).

moveDragon(): Moves the dragon randomly within the labyrinth.

checkCollision(): Checks for collisions between the player and the dragon, as well as if the player reaches the goal.

initializeGame(): Initializes the game by setting initial positions of the player and dragon, generating obstacles, and resetting the timer.

generateObstacles(): Populates the labyrinth with obstacles, avoiding certain predefined locations.

The class encapsulates the game's logic and state, providing methods to interact with and update the game's elements.

The **LabyrinthView** class extends JFrame and serves as the graphical user interface (GUI) for the labyrinth game. It interacts with the LabyrinthModel to visually represent the game state and handle user input. Here's a brief overview:

Fields:

CELL_SIZE: Constant representing the size of each cell in the labyrinth grid.

model: Instance of LabyrinthModel representing the game state.

nameObtained: Boolean flag to track if the player's name has been obtained.

Constructor:

Sets up the JFrame with the title "Labyrinth Game" and initializes the LabyrinthModel.

Obtains the player's name if it hasn't been obtained yet.

Configures key event handling for player movement and a timer for dragon movement.

Sets up a menu bar with options for displaying the leaderboard, starting a new game, and exiting the application.

Creates a JPanel for drawing the game elements and sets its preferred size based on the labyrinth size.

Methods:

displayLeaderboard(): Calls Leaderboardmanager to display the leaderboard.

obtainPlayerName(): Prompts the user to enter their name and sets the player's name in the model.

startNewGame(): Resets the game by creating a new LabyrinthModel instance and obtaining the player's name.

drawElapsedTime(Graphics g): Draws the elapsed time on the GUI.

drawGrid(Graphics g): Draws the grid around the player's current position.

drawObstacles(Graphics g): Draws obstacles within the grid based on the player's vicinity.

drawPlayer(Graphics g): Draws the player on the GUI.

drawDragon(Graphics g): Draws the dragon on the GUI.

drawExit(Graphics g): Draws the exit point on the GUI.

The class combines Swing components and custom drawing to create a visual representation of the labyrinth game. The GUI updates based on user input and the game's state, providing a dynamic and interactive experience.

The Leaderboardmanager class manages interactions with a MySQL database for the leaderboard functionality in the labyrinth game. Here's a brief overview:

Fields:

JDBC_URL: JDBC connection URL for the MySQL database.

USER: Username for the database connection.

PASSWORD: Password for the database connection.

Methods:

addToLeaderboard(String playerName, int winCount): Adds a new entry to the leaderboard with the specified player name and win count. Uses a try-with-resources statement to ensure proper resource closure.

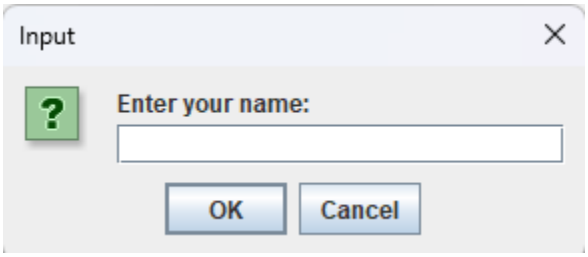
displayLeaderboard(): Retrieves and displays the leaderboard entries from the database, ordered by win count in descending order. Uses try-with-resources statements to manage database resources.

Database Interaction:

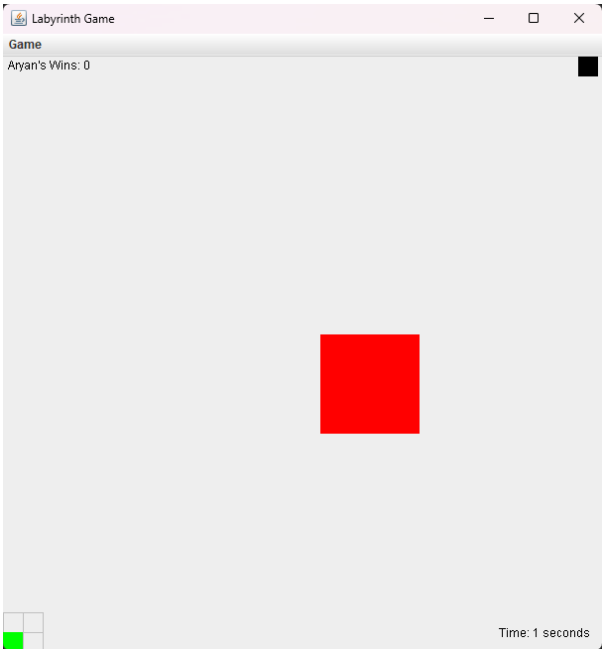
The addToLeaderboard method prepares and executes an SQL INSERT statement to add a new record to the "leaderboard" table, including the player's name and win count.

The displayLeaderboard method prepares and executes an SQL SELECT statement to retrieve all records from the "leaderboard" table, ordered by win count in descending order. It then prints the leaderboard entries to the console.

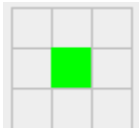
Testing:



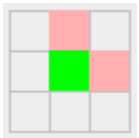
The prompt we get when we start the game, if the user inputs nothing the default name will be “Player”.



The main UI of the game is like above, the red block represents the dragon, the green block is the player and black block is the exit.

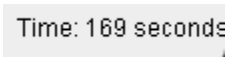


The player’s vision is limited, they can only see one block ahead of them.

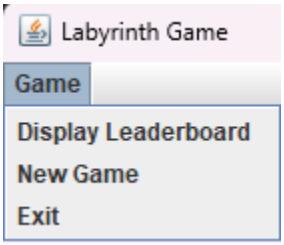


There are walls implemented in the game, the player cannot pass through walls.

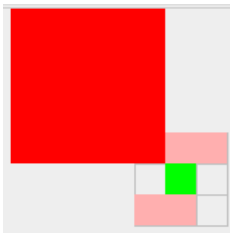
This is the counter for how many times the player has escaped the labyrinth, this will be added to the leaderboard later.



Time spent in labyrinth is also recorded.



We have three menu bar items, Display Leaderboard, New Game, Exit.



In this case the dragon is right next to the player, if the dragon moves one more block to the right and one more to bottom the player dies.

If the player moves to the black block the game is over. And the 1 point will be incremented to their victories.

```
Leaderboard:
Player Name          Win Count
------------------------------
Aryan                    3
Ruslan                   2
Aryan                    2
```

The leaderboard is like this, it stores the winCounter sorted from maximum to minimum.