Maze Runner [Assignment 01]

CSCS 290: Introduction to JAVA Programming

> ❗ **Deadline: 1st August 2023**
>
> ➔ Plagiarism is absolutely forbidden.
> ➔ Do not use any programming constructs that are not covered in class.
> ➔ Use only what has been covered in class (conditional statements, loops, arrays and methods)
> ➔ Submit the repository link of your assignment (pushed on Github) along with the actual assignment on classroom.
> ➔ **Your program must not crash**. If the user enters wrong input, your program should handle it and ask for the input again.
> → Your submission should only contain ONE **MazeRunner.java** file

Introduction

Welcome to the "Maze Runner" game! This Java assignment will challenge your programming skills and problem-solving abilities. In this game, you will develop a text-based maze navigation application using Java. You will utilize arrays, loops, and methods to represent the maze, move the player through the maze, and implement game logic.

Game Description

The "Maze Runner" game is a text-based adventure where you, the player, are placed in a maze represented by a two-dimensional grid. Your mission is to find your way from the starting position (P) to the exit point (E) while avoiding walls (#) and dead ends. You can navigate through the maze by entering the following moves:

- W: Move up.
- A: Move left.
- S: Move down.
- D: Move right.

Game Rules

1. The maze will be represented by a two-dimensional character array.

2. The maze will have walls (#) that are impassable obstacles, open paths (.) that you can move through, your starting position (P), and the exit point (E).
3. Your moves will be considered valid if they do not hit any walls (#) or go outside the maze boundaries.
4. The game will end only when you reach the exit (E).
5. Keep a `timer`, `numberOfSteps`, `score` and `highscore` and display all **four** of them at the end of the game.
6. Ask the user if he/she wants to play the game again.

Requirements

To complete this assignment, you will need to:

1. Your Game should have a Main Menu with the following Options:
    a. Play Game
    b. Instructions
    c. Credits
    d. High Score
    e. Exit
2. Create a two-dimensional character array to represent the maze.
3. Initialize the maze with the given layout, including walls, open paths, starting position, and the exit point.
4. Implement a method to display the maze on the console.
5. Implement a method to move the player based on the user's input (W/A/S/D).
6. Handle user input and update the player's position on the maze.
7. Check for valid moves to avoid walls and dead ends.
8. Track the number of steps taken.
9. Display appropriate messages when the player wins or loses the game if the **timer goes out.**

Methods/Functions to Consider

When developing your Maze Runner game, consider implementing these methods:

1. **initializeMaze()**: Initializes the maze with the given layout and identifies the player's starting position.
2. **printMaze()**: Prints the current state of the maze.
3. **isValidMove(int newX, int newY)**: Checks whether the player's proposed move is valid (i.e., within the maze boundaries and not into a wall).
4. **movePlayer(char direction)**: Updates the player's position based on the input direction (W/A/S/D).
5. **hasPlayerWon()**: Checks whether the player has reached the exit.
6. **playGame()**: Contains the main game loop. This method requests user input for moves, updates the player's position, and checks whether the game has been won.
7. **displayResult()**: Displays the game result, number of steps taken, and the score at the end of the game.
8. **updateScore()**: Calculates and updates the player's score based on the rules you decide.
9. **startNewGame()**: Resets the game, allowing the player to start a new game.

Additional Functionalities

To make your game more interactive and user-friendly, consider implementing the following functionalities:

1. **showInstructions()**: This method should display the game rules and instructions when the user selects the 'Instructions' option from the main menu.
2. **showCredits()**: This method should display game development credits (like who developed the game, the game version, etc.) when the user selects the 'Credits' option from the main menu.
3. **showHighScore()**: This method should display the highest score achieved so far in the game when the user selects the 'High Score' option from the main menu.

4. **exitGame()**: This method should terminate the game application when the user selects the 'Exit' option from the main menu.

Remember, writing clear and concise code is the key to a successful project. You are advised to follow the principles of clean coding, and keep your functions small and dedicated to a single task.

Puzzle Pattern Layout

Here is a simple 7x7 grid layout for a maze to get you started. You can modify it according to your requirements.

```
# # # # # # #
# P . . . . #
# # # # . # #
# . . . . . #
# # # . # # #
# . . . . E #
# # # # # # #

In this layout:

# represents walls that are impassable.
. represents open paths that you can move through.
P represents the starting position of the player.
E represents the exit point that the player needs to reach.
```

Tips for Success

1. Carefully plan your approach before starting to code. Break down the problem into smaller tasks and implement them one by one.
2. Test your code incrementally to ensure each function works correctly before moving on to the next one.
3. Use loops and conditional statements effectively to control the flow of the game.
4. Utilize methods to encapsulate different functionalities and enhance the readability of your code.

Have Fun!

Enjoy the process of developing the "Maze Runner" game!

Good luck, and happy coding!

Tutorial for git/github

https://www.youtube.com/watch?v=uaeKhfhYE0U&t=252s


DON'T WORRY
YOU GOT THIS!
imgflip.com