Klotski Puzzle

Project Overview

In this project, you will implement the classic **Klotski Puzzle** game using the Java programming language. Players must move sliding blocks to guide a specific block to the target position.

1. Game Introduction

Klotski is a traditional Chinese puzzle game in which players need to move the "Cao Cao" block out of the board's exit.

The game is typically played on a rectangular board with blocks of different sizes and shapes. Players must carefully plan each move to complete the task successfully.

2. Project Requirements

For this project, you need to complete the following tasks:

Task 1: Game Initialization (10 points)

1. **Game Start Interface**: Implement a game start interface.
2. **Board Initialization**: For the basic requirements, the board should be initialized as a 4 x 5 grid with blocks of different sizes:
    - Cao Cao block: 2 x 2 size, 1 piece
    - Guan Yu block: 2 x 1 size, 1 piece
    - Other general blocks: 1 x 2 size, 4 pieces
    - Soldier blocks: 1 x 1 size, 4 pieces
3. **Block Color Differentiation**: Blocks of different sizes should be distinguished by different colors.
4. **Restart Function**: Players can restart the game at any time, and the board should reset to its initial state.

Task 2: Block Movement (20 points)

1. **Block Movement**: Players can control the movement of blocks (up, down, left, right). Blocks can only move along empty spaces on the board and cannot overlap with other blocks.
2. **Button Controls**: The interface should include up, down, left, and right buttons for controlling block movement.
3. **Keyboard Controls**: The game must also support keyboard controls for movement.
4. **Boundary Detection**: Blocks cannot move beyond the board's boundaries.
5. **Collision Detection**: Blocks cannot overlap with other blocks during movement.
6. **Information Recording**: At the beginning of a new game, record the number of moves and other necessary information.
7. **Movement Recording**: Record each move and provide an undo function (completing this function will earn advanced points).

Task 3: Victory Condition (5 points)

1. **Victory Condition**: The game is won when the "Cao Cao" block reaches the exit position, and a victory prompt need to be displayed.
2. **Victory Interface**: The victory interface should display the number of moves taken by the player.

Task 4: Multi-User Login (15 points)

1. Implement a login selection interface for both guests and registered users.
2. Guests can play without registration but do not have the functionality to save game progress.
3. The user login interface includes a registration page and allows login after entering account credentials.
4. After the program exits and is run again, previously registered users can still log in

Task 5: Save and Load Game (20 points)

1. Each user (except guests) can load previously saved games. The save file should be a single file, and subsequent saves will overwrite the previous one (overwriting is the basic requirement. Additional points would not be given if multiple save slots are implemented per user).
2. From the game start interface, players can choose to load the last saved game. The file should include the game board's status, the number of moves made so far, and other necessary information.
3. Each user's save data is unique.
4. Manual saving is the basic requirement; implementing timed auto-save or auto-save on exit will earn advanced points.
5. Save File Error Handling: If the save file is corrupted in format or content, the corrupted file will not be loaded, and the game should continue running without crashing. (If your game is capable of detecting save files that have been modified by others while still maintaining the legitimacy of the save data， it will earn the advanced points.)

Task 6: Graphical User Interface (GUI) (10 points)

1. Implement a graphical interface for the game using JavaFX, Swing, or any other **Java** graphical framework.
2. You can earn points for this task by completing the demo code provided in the course.
3. Independently creating a unique GUI will earn advanced points.
4. If your program relies on command-line input, you cannot earn full points for this task.

Task 7: Advanced Features (20 points)

Any features beyond the basic requirements can earn advanced points, including but not limited to:

1. **Interface Beautification**: Enhanced graphics and visual effects.
2. **Multi-Level Design**: Design multiple levels of varying difficulty, with different board layouts or block configurations for each level.
    - Required blocks: Cao Cao and Guan Yu blocks.
    - If multi-level functionality is added, provide corresponding buttons on the start interface to access different levels.

3. **Artificial Intelligence**: Implement an AI algorithm to automatically solve the Klotski puzzle. Different levels of intelligence will earn varying points.
4. **Animation Effects**: Add smooth animations for block movements.
5. **Sound Effects and Background Music**: Enhance the gaming experience with sound effects and background music.
6. **Time Attack Mode**: Introduce a timed mode where players must complete the game within a set time.
    - If a timed mode is added, the save and load operations must include the player's time usage.
    - The victory interface should also display the time taken.
7. **Props and Obstacles**: Add props and obstacles to enrich gameplay.
    - If props and obstacles are added, the save and load operations must include this information.
8. **Online Spectating**: Allow multiple users to log in simultaneously and spectate the game or perform other operations online.

3. Notes

1. **Code Standards**: The code should be well-structured and commented for readability and maintainability.
2. **Compatibility**: Ensure the code runs smoothly on different operating systems.
3. **Testing**: Thoroughly test all functionalities before submission to avoid major bugs. Program crashes will result in point deductions.

Wishing everyone success in completing the project!