2048

In this project, you are required to implement a game `2048` using the Java programming language.

I. Game Introduction

2048 is a globally popular number puzzle game created by Italian programmer Gabriele Cirulli in 2014. The game boasts a clean interface and a strong logical gameplay that challenges players to merge identical numbered tiles until they reach or surpass the titular goal of 2048.

II. Game Interface & Basic Elements

The game screen consists of a 4x4 grid where random numbers 2 or 4 initially populate the cells. Players slide the entire grid up, down, left, or right, causing all tiles to move accordingly. After each slide, a new 2 or 4 tile randomly appears in an empty cell.

III. Detailed Game Rules

1. **Slide and Merge**: When sliding the grid, all tiles shift towards the direction chosen until they hit the edge or another immovable tile. If two adjacent tiles have the same number, they will merge into one tile with the sum of their values after the movement stops and this new merged tile will also shift along the movement direction until it cannot move further. If there exist three identically valued tiles that are adjacent to one another, the two blocks nearest to the end of the sliding direction will merge together.
2. **New Tile Generation**: Following every valid slide (where at least one tile changes position), the game will generate a new 2 or 4 tile randomly in any empty space on the grid.
3. **Game Objective**: The player's aim is to continuously combine number tiles to create a tile with a value of 2048. While it is theoretically possible to continue playing for higher scores, reaching 2048 is typically considered the base victory condition.
4. **Game Over**: The game ends when there are no more moves available; this occurs when all 16 spaces on the grid are occupied and no adjacent tiles can be merged.

IV. Project Requirements

In this project, you should finish the following tasks:

Task 1: Game Initialization (10 points)

1. Realize a start frame for the game where players can choose from various initial game modes, each allowing customizable settings such as different starting board layouts or distinct target numbers to achieve.
2. At least one mode must generate a traditional 4x4 grid, with an initial state containing a randomly placed tile with a value of 2 and another random tile with a value of 4.
3. The game should allow players to restart a new game at any time during gameplay. (Not exiting the program and run it again.)
4. When starting a new game, the game data needs to be consistent with the new game.
5. Grids with different numbers should be in different colors.

Task 2: 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 3: Save and Load Games (15 points)

1. Each user (except guests) has the option to load their previous saved game; the save is a single save file, and saving again will overwrite the previous save (Overwriting the original save 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 their last save which should contain information about the elapsed game time, the game board's status, and the number of moves made so far.
3. Each user's save data is unique.
4. Manual saving is a basic requirement; implementing automatic saving at timed intervals or upon exit can earn points in the advanced section.
5. Save File Error Check: When a save file's format or contents are corrupted, the damaged save will not be loaded, and the game will still run rather than crash. (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 4: Gameplay (30 points)

1. Sliding and Merging: When a player slides the matrix, all tiles will move towards the slide direction until they hit a boundary or an immovable block. Adjacent tiles with the same number will merge into a single tile with their sum after the slide ends.
2. Button control: The interface must include up, down, left, and right buttons to facilitate merging in different directions.
3. Keyboard control: Keyboard control are required for merging (up, down, left, right) in different directions.
4. New Tile Generation: After every valid slide (where at least one tile changes position), the game will randomly generate a new tile with a value of either 2 or 4 on any empty space.
5. Game Victory: In classic mode, the goal is to reach a 2048 tile through continuous merging of number tiles. For different modes, you can design your own objectives. Once the objective is met, display a victory screen.
6. Game Over: The game ends when there are no more movable spaces on the matrix, i.e., all 16 cells are occupied, and no adjacent tiles can merge.

Task 5: 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 will earn points for this section by completing the code based on the demo provided in the course.
3. Independently creating a GUI will count as Advanced points.
4. If your program need to input into command line， you can not get full points of this task.

Task 6: Advanced Features (20 points)

Any additional features beyond the basic requirements described above will earn points in this advanced category, including but not limited to:

1. Enhanced graphics and aesthetics
2. Implementation of AI to achieve high scores
3. Adding animated merging effects
4. Introducing a time-limited mode
5. Incorporating obstacles on the game board
6. Adding props in the game