2048

在这个项目中，您需要使用Java编程语言实现一个名为2048的游戏。

I. 游戏介绍

2048是一款由意大利程序员Gabriele Cirulli在2014年创造的全球流行的数字拼图游戏。该游戏以其简洁的界面和强烈的逻辑性玩法而闻名，挑战玩家合并相同编号的瓷砖，直到达到或超过目标数字2048。

II. 游戏界面 & 基本元素

游戏屏幕由一个4x4的网格组成，初始时由随机数字2或4填充单元格。

玩家可以向上、向下、向左或向右滑动整个网格，导致所有瓷砖相应移动。

每次滑动后，一个新2或4的瓷砖随机出现在一个空单元格中。

III. 详细游戏规则

1. 滑动和合并：当玩家滑动网格时，所有瓷砖会向所选方向移动，直到它们撞到边缘或另一个不可移动的瓷砖。

如果两个相邻的瓷砖编号相同，它们将在停止移动后合并为一个瓷砖，其数值等于它们值的总和，

并且这个新合并的瓷砖也将沿着移动方向继续移动，直到不能再移动。

如果存在三个相邻的相同值的瓷砖，滑动方向末端的两个块将合并在一起。

1. 新瓷砖生成：在每次有效滑动（至少有一个瓷砖改变位置）之后，游戏将在网格上的任何空空间随机生成一个新的2或4瓷砖。

3. 游戏目标：玩家的目标是通过不断合并数字瓷砖来创建一个值为2048的瓷砖。虽然理论上可以继续玩以获得更高的分数，但达到2048通常被视为基本胜利条件。

4. 游戏结束：当没有更多的移动空间可用时，游戏结束；这发生在网格上的所有16个空间都被占据，并且没有相邻的瓷砖可以合并时。

IV. 项目要求

在这个项目中，您应该完成以下任务：

任务1：游戏初始化（10分）

1. 实现一个开始框架，玩家可以从各种初始游戏模式中选择，每种模式都允许自定义设置，如不同的起始棋盘布局或不同的目标数字。
2. 至少一种模式必须生成一个传统的4x4网格，初始状态包含一个随机放置的值为2的瓷砖和另一个随机放置的值为4的瓷砖。

3. 游戏应允许玩家在游戏过程中随时重新开始新游戏（不是退出程序并重新运行）。

4. 当开始新游戏时，游戏数据需要与新游戏一致。

5. 不同数字的网格应该有不同的颜色。（我的考虑是渐变）

任务2：多用户登录（15分）

1. 实现一个登录选择界面，供访客和注册用户使用。
2. 访客可以在不注册的情况下玩游戏，但没有保存游戏进度的功能。
3. 用户登录界面包括一个注册页面，并允许在输入账户凭证后登录。

4. 在程序退出并重新运行后，之前注册的用户仍然可以登录。

任务3：保存和加载游戏（15分）

1. 每个用户（除了访客）都有权加载他们之前保存的游戏；保存是一个单一的保存文件，再次保存将覆盖之前的保存（覆盖原始保存是基本要求。如果每个用户实现了多个保存槽，将不会获得额外的分数）。

2. 从游戏开始界面，玩家可以选择加载他们最后的保存，其中应包含有关已过去游戏时间、棋盘状态和迄今为止所做的移动次数的信息。

3. 每个用户的保存数据是唯一的。

4. 手动保存是基本要求；实现定时间隔或退出时自动保存可以增加分数。

5. 保存文件错误检查：当保存文件的格式或内容损坏时，损坏的保存将不会被加载，游戏将仍然运行而不是崩溃（如果您的游戏能够检测到由他人修改的保存文件，同时仍然保持保存数据的合法性，它将获得额外的分数）。

任务4：游戏玩法（30分）

1. 滑动和合并：当玩家滑动矩阵时，所有瓷砖将向滑动方向移动，直到它们撞到边界或一个不可移动的块。

滑动结束后，相同数字的相邻瓷砖将合并为一个瓷砖，其数值等于它们的总和。

2. 按钮控制：界面必须包括向上、向下、向左和向右的按钮，以促进不同方向的合并。

3. 键盘控制：键盘控制是必需的，用于在不同方向（上、下、左、右）合并。

4. 新瓷砖生成：在每次有效滑动（至少有一个瓷砖改变位置）之后，游戏将随机生成一个新的2或4瓷砖，放置在任何空空间上。

5. 游戏胜利：在经典模式中，目标是达到一个2048瓷砖，通过不断合并数字瓷砖。对于不同的模式，您可以设计自己的目标。一旦目标达成，显示胜利屏幕。

6. 游戏结束：当矩阵上没有更多的可移动空间时，即所有16个单元格都被占据，并且没有相邻的瓷砖可以合并时，游戏结束。

任务5：图形用户界面（GUI）（10分）

1. 使用JavaFX、Swing或任何其他Java图形框架实现游戏的图形界面。

2. 您将通过完成课程中提供的演示代码来为这一部分获得分数。

3. 独立创建GUI将视为高级分数。

4. 如果您的程序需要命令行输入，您将无法获得此任务的全部分数。

任务6：高级功能（20分）

任何超出上述基本要求的额外功能都将在这一高级类别中获得分数，包括但不限于：

1. 增强的图形和美学。

2. 实现AI以实现高分。

3. 添加动画合并效果。

4. 引入限时模式。

5. 在游戏板上加入障碍物。

6. 在游戏中添加道具。（撤销，自定义数字）

7.延迟加载

8.广告报考南科大

编写2048游戏的代码是一个涉及多个步骤的过程，以下是一些基本的指导步骤，帮助您开始：

理解游戏规则：确保您完全理解2048游戏的规则，包括如何移动瓷砖、合并瓷砖、游戏胜利和结束的条件。

设计游戏界面：决定您将使用哪种图形用户界面(GUI)库来创建游戏界面。JavaFX和Swing是Java中常用的GUI库。

搭建项目结构：在您选择的IDE（如IntelliJ IDEA、Eclipse等）中创建一个新的Java项目，并规划好您的包和类的结构。

创建类和接口：

Game：管理游戏逻辑，包括移动、合并、生成新瓷砖等。

Grid：表示游戏的4x4网格，存储每个单元格的瓷砖。

Tile：表示单个瓷砖，包含值和位置信息。

GUI：负责渲染游戏界面和处理用户输入。

编写核心游戏逻辑：

实现Grid类，它可以存储瓷砖，并提供方法来移动瓷砖和合并相邻的瓷砖。

实现Game类，它使用Grid对象来控制游戏状态，包括检测游戏是否结束。

用户输入处理：

在GUI类中，实现键盘和鼠标事件监听器，以响应用户的移动指令。

自动生成新瓷砖：在每次有效移动后，Game类应随机在空单元格中生成一个值为2或4的新瓷砖。

游戏胜利和结束条件：在Game类中实现检测游戏胜利和结束的逻辑。

保存和加载游戏：实现文件读写操作，以保存和加载游戏状态。

图形用户界面：

使用您选择的GUI库创建游戏的视觉界面，包括瓷砖的显示、按钮的创建等。

测试：在开发过程中不断测试游戏，确保所有功能按预期工作，修复任何发现的问题。

高级功能：如果时间允许，考虑添加一些高级功能，如动画效果、AI对手、新的游戏模式等。

文档和注释：为您的代码编写文档和注释，以提高可读性和可维护性。

发布和部署：在完成所有开发和测试后，您可以将游戏打包为可执行文件，并与他人分享。

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.Random;

public class Game2048 extends JFrame {

private static final int GRID\_SIZE = 4; // 定义网格大小

private Tile[][] grid; // 游戏网格

private boolean gameover; // 游戏结束标志

public Game2048() {

setTitle("2048 Game");

setSize(500, 500);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

grid = new Tile[GRID\_SIZE][GRID\_SIZE];

initializeGame();

renderGrid();

}

private void initializeGame() {

// 初始化游戏，设置随机起始瓷砖

Random random = new Random();

for (int i = 0; i < GRID\_SIZE; i++) {

for (int j = 0; j < GRID\_SIZE; j++) {

grid[i][j] = new Tile(i, j);

}

}

addRandomTile();

addRandomTile();

}

private void addRandomTile() {

// 添加一个随机的2或4瓷砖

int emptyCells = 0;

int[] emptyCellPositions = new int[GRID\_SIZE \* GRID\_SIZE];

for (int i = 0; i < GRID\_SIZE; i++) {

for (int j = 0; j < GRID\_SIZE; j++) {

if (grid[i][j].getValue() == 0) {

emptyCellPositions[emptyCells++] = (i \* GRID\_SIZE) + j;

}

}

}

if (emptyCells > 0) {

int randomIndex = random.nextInt(emptyCells);

int row = emptyCellPositions[randomIndex] / GRID\_SIZE;

int col = emptyCellPositions[randomIndex] % GRID\_SIZE;

grid[row][col] = new Tile(row, col, 2); // 随机生成2或4

}

}

private void renderGrid() {

// 渲染网格（这里只是一个框架，您需要实现具体的渲染逻辑）

getContentPane().setLayout(new GridLayout(GRID\_SIZE, GRID\_SIZE));

for (int i = 0; i < GRID\_SIZE; i++) {

for (int j = 0; j < GRID\_SIZE; j++) {

getContentPane().add(new JLabel(String.valueOf(grid[i][j].getValue())));

}

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

Game2048 game = new Game2048();

game.setVisible(true);

}

});

}

}

class Tile {

private int row;

private int col;

private int value;

public Tile(int row, int col) {

this.row = row;

this.col = col;

this.value = 0; // 默认值为0

}

public Tile(int row, int col, int value) {

this.row = row;

this.col = col;

this.value = value;

}

public int getRow() {

return row;

}

public int getCol() {

return col;

}

public int getValue() {

return value;

}

}