大作业1:2048小游戏

时间：4.21-4:30

一、人员分工：

思路设计、需要分析—王超

概要设计、详细设计—修玥

代码编写—石雨宁 周学士

测试—全体组员

二、各阶段简要规划：

1.明确项目内容，查询资料

2.总体概要设计，详细设计

3.编写程序，调试分析

4.文档编写，提交

三、设计思路阐述：

1.设计游戏时，要充分考虑到2048游戏的特性，以及实现

2.使用Java语言进行开发

3.如何实现界面

4.绘制方格和数字

5.移动方格且让数值发生变化

6.数字的相加

7.输赢的判断

四、游戏规则：

1.2048是一款益智类小游戏，刚开始随机出现两个数字，可以上下左右控制数字的移动。

2.当选择一个方向移动后，所有数字都会沿该方向移动到表格尽头，并且空余表格会随机出现2或4，当碰到相同的两个数字时，该两个数字会合并相加成一个数字，直到最大的数字变成2048游戏成功

3.否则当数字填满表格且不能再移动时游戏失败。

五：游戏需求分析

1.需求分析

现代社会对休闲小游戏的需求是：提高游戏的操作可行性，降低游戏的操作难度，降低游戏的上手难度，降低游戏的竞争需求，对使用设备的要求趋于简洁化和多样化，游戏界面更加人性化功能更加贴合使用。

2.可行性分析

（1）技术的可行性

根据游戏设计的需要，可以通过Java进行源代码的编辑，通过Windows界面进行用户界面的编辑和优化，结合时间和目前学习水平等各项因素，项目所要求功能和性能完全可以实现。

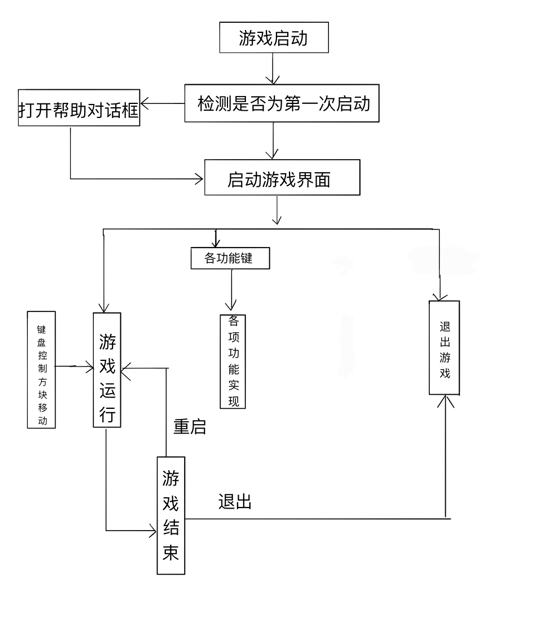
（2）运行可行性

游戏要求在大部分的用户设备上可以实现，并且游戏运行硬件几乎无要求，项目运行可以得到保证。

（3）可行性综合分析

综合以上两点，项目开发可以得到技术和运行的保证，证明项目可行。

六、程序概要设计

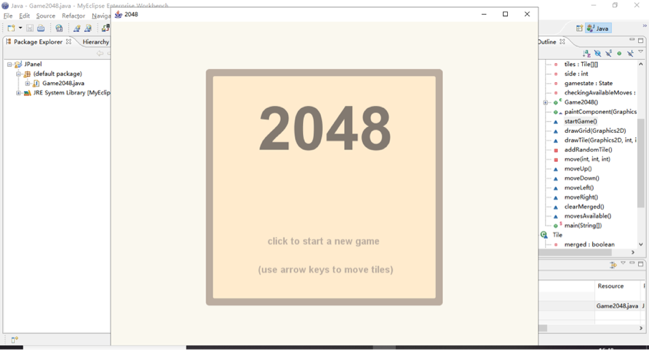


七、相关代码

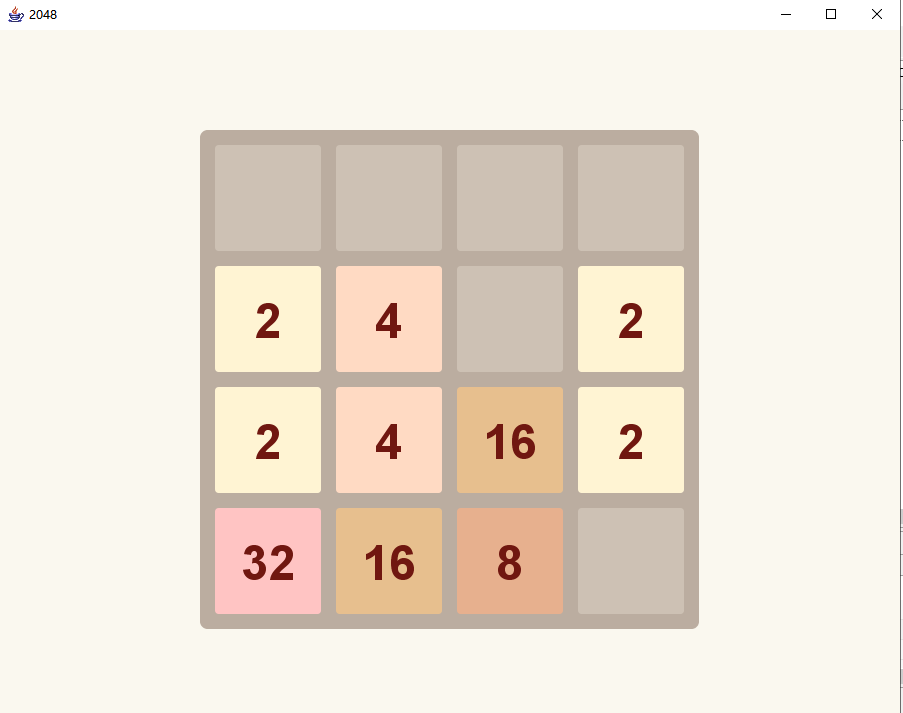
import java.awt.\*;import java.awt.event.\*;import java.util.Random;import javax.swing.\*;public class Game2048 extends JPanel { enum State { start, won, running, over } final Color[] colorTable = { new Color(0x701710), new Color(0xFFE4C3), new Color(0xfff4d3), new Color(0xffdac3), new Color(0xe7b08e), new Color(0xe7bf8e), new Color(0xffc4c3), new Color(0xE7948e), new Color(0xbe7e56), new Color(0xbe5e56), new Color(0x9c3931), new Color(0x701710) }; final static int target = 2048; static int highest; static int score; private Color gridColor = new Color(0xBBADA0); private Color emptyColor = new Color(0xCDC1B4); private Color startColor = new Color(0xFFEBCD); private Random rand = new Random(); private Tile[][] tiles; private int side = 4; private State gamestate = State.start; private boolean checkingAvailableMoves; public Game2048() { setPreferredSize(new Dimension(900, 700)); setBackground(new Color(0xFAF8EF)); setFont(new Font("SansSerif", Font.BOLD, 48)); setFocusable(true); addMouseListener(new MouseAdapter() { @Override public void mousePressed(MouseEvent e) { startGame(); repaint(); } }); addKeyListener(new KeyAdapter() { @Override public void keyPressed(KeyEvent e) { switch (e.getKeyCode()) { case KeyEvent.VK\_UP: moveUp(); break; case KeyEvent.VK\_DOWN: moveDown(); break; case KeyEvent.VK\_LEFT: moveLeft(); break; case KeyEvent.VK\_RIGHT: moveRight(); break; } repaint(); } }); } @Override public void paintComponent(Graphics gg) { super.paintComponent(gg); Graphics2D g = (Graphics2D) gg; g.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON); drawGrid(g); } void startGame() { if (gamestate != State.running) { score = 0; highest = 0; gamestate = State.running; tiles = new Tile[side][side]; addRandomTile(); addRandomTile(); } } void drawGrid(Graphics2D g) { g.setColor(gridColor); g.fillRoundRect(200, 100, 499, 499, 15, 15); if (gamestate == State.running) { for (int r = 0; r < side; r++) { for (int c = 0; c < side; c++) { if (tiles[r][c] == null) { g.setColor(emptyColor); g.fillRoundRect(215 + c \* 121, 115 + r \* 121, 106, 106, 7, 7); } else { drawTile(g, r, c); } } } } else { g.setColor(startColor); g.fillRoundRect(215, 115, 469, 469, 7, 7); g.setColor(gridColor.darker()); g.setFont(new Font("SansSerif", Font.BOLD, 128)); g.drawString("2048", 310, 270); g.setFont(new Font("SansSerif", Font.BOLD, 20)); if (gamestate == State.won) { g.drawString("you made it!", 390, 350); } else if (gamestate == State.over) g.drawString("game over", 400, 350); g.setColor(gridColor); g.drawString("click to start a new game", 330, 470); g.drawString("(use arrow keys to move tiles)", 310, 530); } } void drawTile(Graphics2D g, int r, int c) { int value = tiles[r][c].getValue(); g.setColor(colorTable[(int) (Math.log(value) / Math.log(2)) + 1]); g.fillRoundRect(215 + c \* 121, 115 + r \* 121, 106, 106, 7, 7); String s = String.valueOf(value); g.setColor(value < 128 ? colorTable[0] : colorTable[1]); FontMetrics fm = g.getFontMetrics(); int asc = fm.getAscent(); int dec = fm.getDescent(); int x = 215 + c \* 121 + (106 - fm.stringWidth(s)) / 2; int y = 115 + r \* 121 + (asc + (106 - (asc + dec)) / 2); g.drawString(s, x, y); } private void addRandomTile() { int pos = rand.nextInt(side \* side); int row, col; do { pos = (pos + 1) % (side \* side); row = pos / side; col = pos % side; } while (tiles[row][col] != null); int val = rand.nextInt(10) == 0 ? 4 : 2; tiles[row][col] = new Tile(val); } private boolean move(int countDownFrom, int yIncr, int xIncr) { boolean moved = false; for (int i = 0; i < side \* side; i++) { int j = Math.abs(countDownFrom - i); int r = j / side; int c = j % side; if (tiles[r][c] == null) continue; int nextR = r + yIncr; int nextC = c + xIncr; while (nextR >= 0 && nextR < side && nextC >= 0 && nextC < side) { Tile next = tiles[nextR][nextC]; Tile curr = tiles[r][c]; if (next == null) { if (checkingAvailableMoves) return true; tiles[nextR][nextC] = curr; tiles[r][c] = null; r = nextR; c = nextC; nextR += yIncr; nextC += xIncr; moved = true; } else if (next.canMergeWith(curr)) { if (checkingAvailableMoves) return true; int value = next.mergeWith(curr); if (value > highest) highest = value; score += value; tiles[r][c] = null; moved = true; break; } else break; } } if (moved) { if (highest < target) { clearMerged(); addRandomTile(); if (!movesAvailable()) { gamestate = State.over; } } else if (highest == target) gamestate = State.won; } return moved; } boolean moveUp() { return move(0, -1, 0); } boolean moveDown() { return move(side \* side - 1, 1, 0); } boolean moveLeft() { return move(0, 0, -1); } boolean moveRight() { return move(side \* side - 1, 0, 1); } void clearMerged() { for (Tile[] row : tiles) for (Tile tile : row) if (tile != null) tile.setMerged(false); } boolean movesAvailable() { checkingAvailableMoves = true; boolean hasMoves = moveUp() || moveDown() || moveLeft() || moveRight(); checkingAvailableMoves = false; return hasMoves; } public static void main(String[] args) { { JFrame f = new JFrame(); f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); f.setTitle("2048"); f.setResizable(true); f.add(new Game2048(), BorderLayout.CENTER); f.pack(); f.setLocationRelativeTo(null); f.setVisible(true); }; }}class Tile { private boolean merged; private int value; Tile(int val) { value = val; } int getValue() { return value; } void setMerged(boolean m) { merged = m; } boolean canMergeWith(Tile other) { return !merged && other != null && !other.merged && value == other.getValue(); } int mergeWith(Tile other) { if (canMergeWith(other)) { value \*= 2; merged = true; return value; } return -1; }}

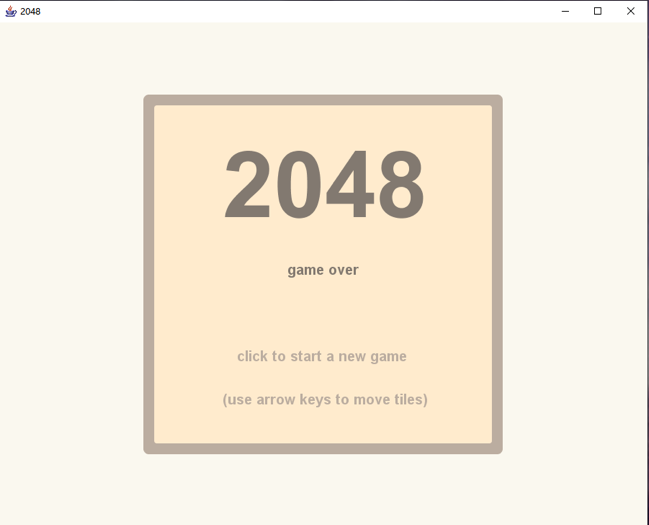
八、测试及测试用例

**程序编写完成后，用Java软件运行，检测程序设计结果，执行目标程序后得到如图所示结果：**



游戏中界面：



游戏结束界面：

经测试，程序正常运行没有发现什么错误。