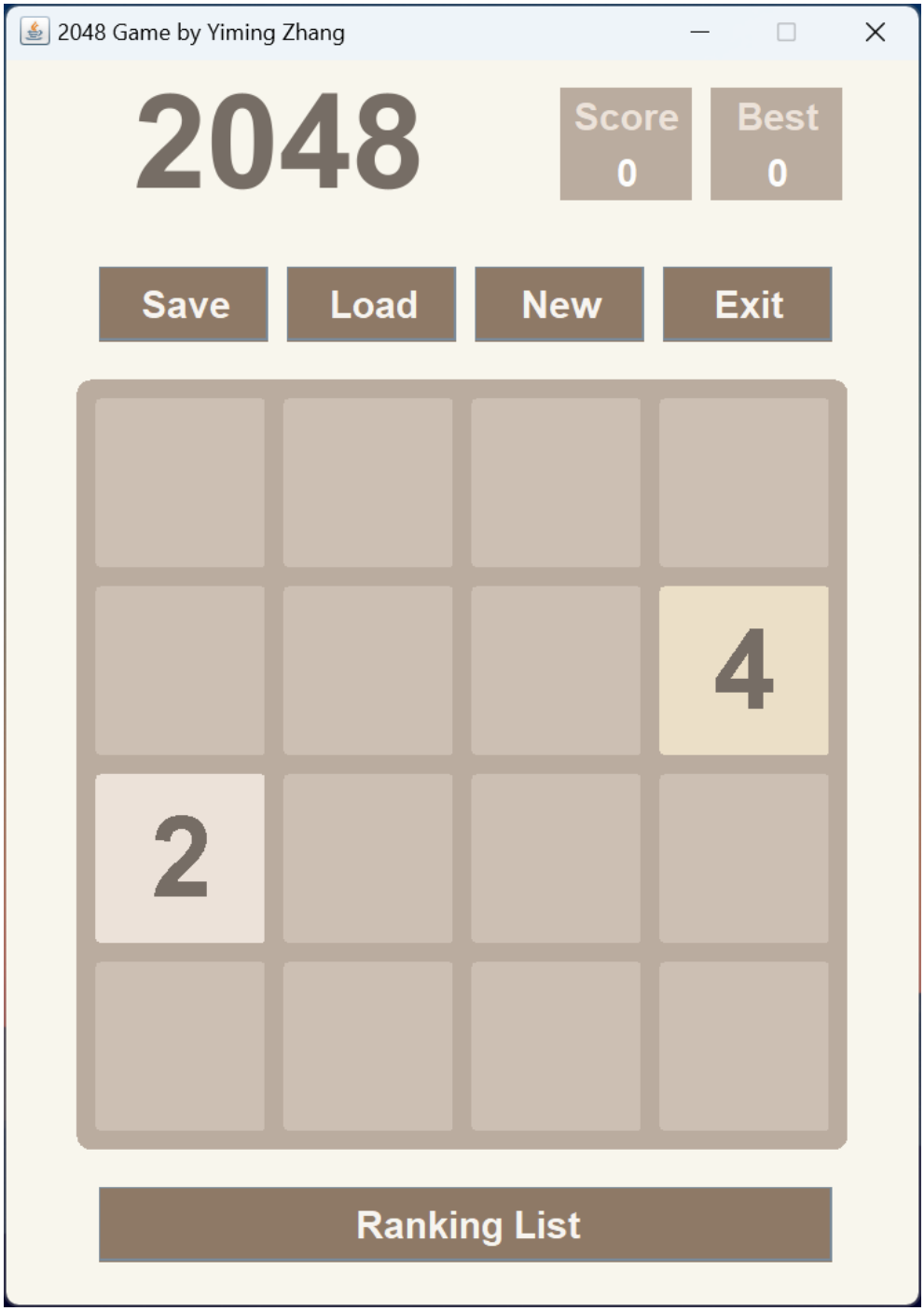
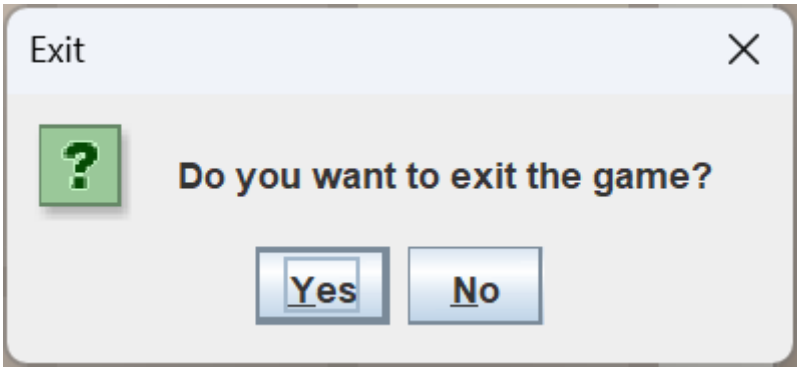
实验报告

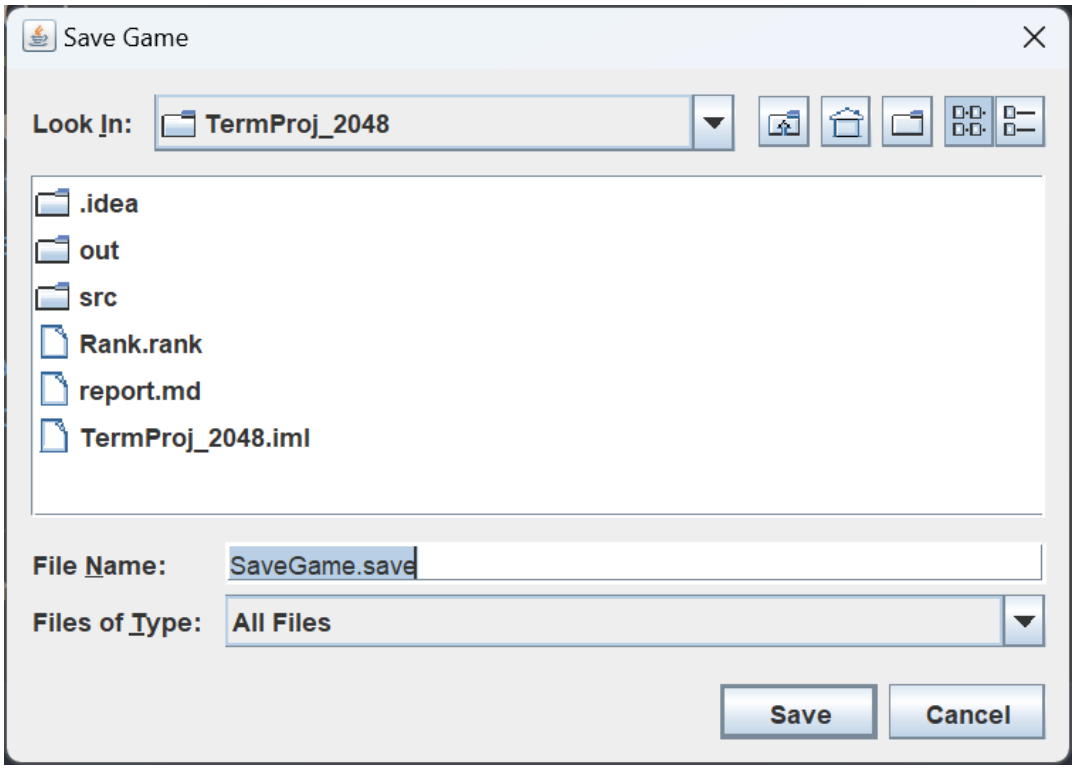
操作方式



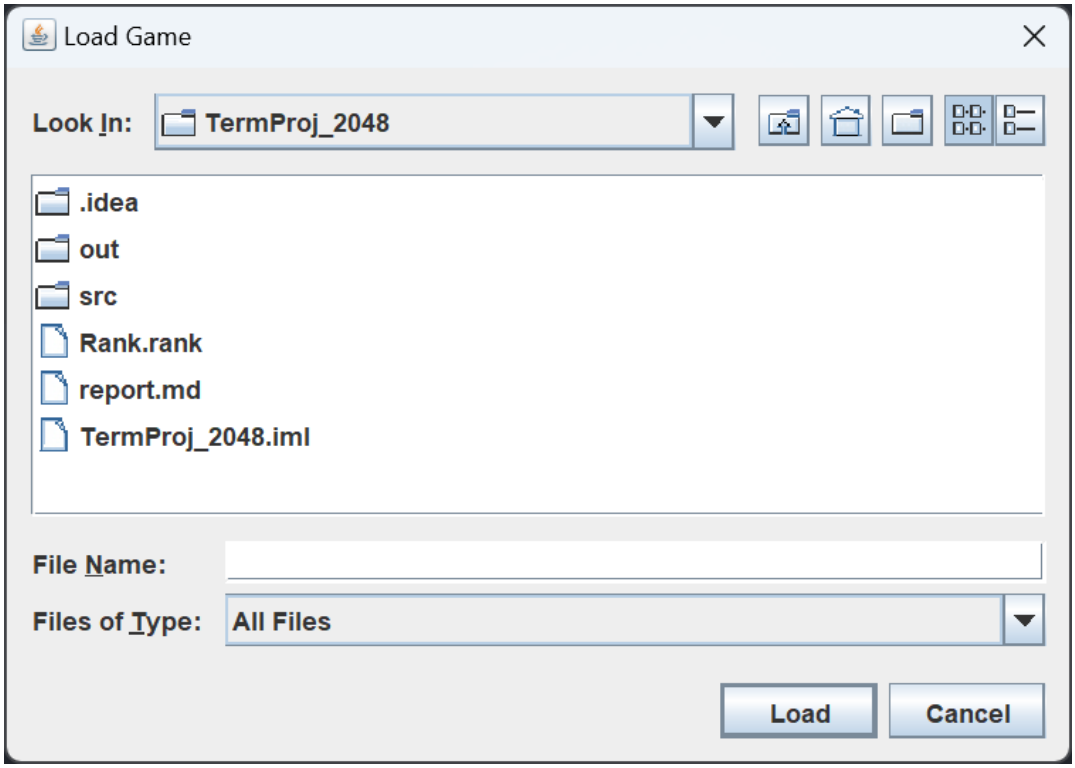
- 游戏主界面如上，尺寸为500px × 700px，有5个按钮、两个得分面板以及游戏主面板。
- 主要玩法为使用键盘控制游戏主面板中方块的移动，使相同的得以合并，当有一个方块的值为2048时游戏胜利，当所有面板均有数字并且不能通过上下左右操作使之合并时，游戏失败。
- 游戏有三种操作方式，上下左右操作分别对应于`WSAD`，`↑↓←→`，`KJHL`~~最后一种给vim玩家使用~~。
- 按下`ESC`按键可以退出游戏，并且弹出对话框询问是否退出，以防误触。



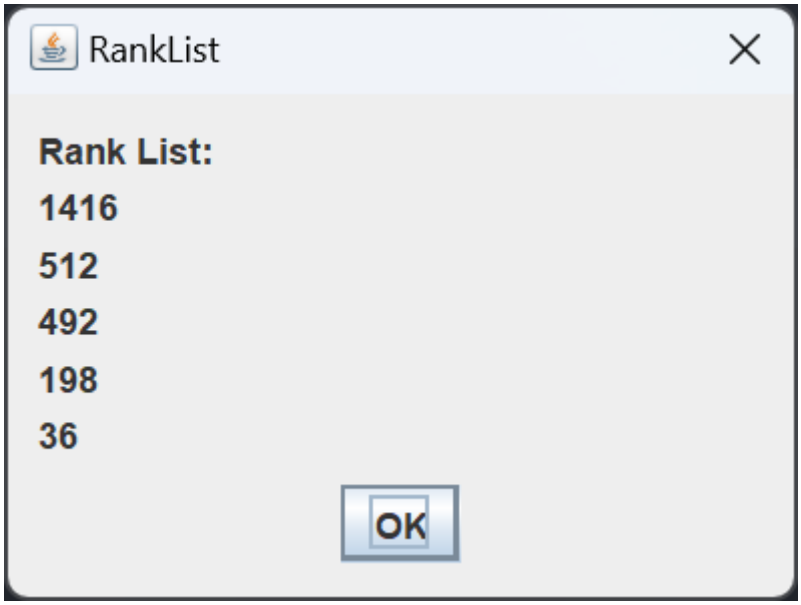
- 游戏主面板上方四个按钮分别对应`保存游戏`，`加载游戏`，`重新开始`以及`退出游戏`：
  - 保存游戏
    - 点击`Save`按钮后会弹出对话框让用户选择保存位置，默认位置为当前项目下，默认文件名为`SaveGame.save`，生成的文件内容为17个数字，在一行内用空格隔开。
    - 前16个数字对应从左向右，从上到下每一个方格的数值，最后一个数字为此时游戏得分。



  - 加载游戏
    - 点击`Load`按钮后会弹出对话框，让用户选择想要加载文件的位置，最好为由游戏生成的存档，若想手工输入，须保持`保存游戏`中介绍的存储内容要求。
    - 默认路径为当前目录，格式要求为`.save .txt .dat`等纯文本文件。



    - 重新开始
        - 点击`New`按钮后，游戏会清空当前游戏面板，并随机添加两个方格，数值为2或4，并将得分`Score`一栏重置为0。
    - 退出游戏
        - 点击`Exit`按钮后，游戏会自动保存并推出，与按`Esc`按键不同的是，不会弹出询问是否退出窗口。
- 游戏主面板下方的按钮为显示游戏排行榜，存有5个最高分，当游戏失败、游戏胜利、退出游戏、新建游戏时都会比较记录最高分数



- 游戏右上方的两个面板`Score`和`Best`下方数值分别对应当前分数和最佳分数，每一次移动后二者均会被更新。
- Java版本`18.0.2.1`

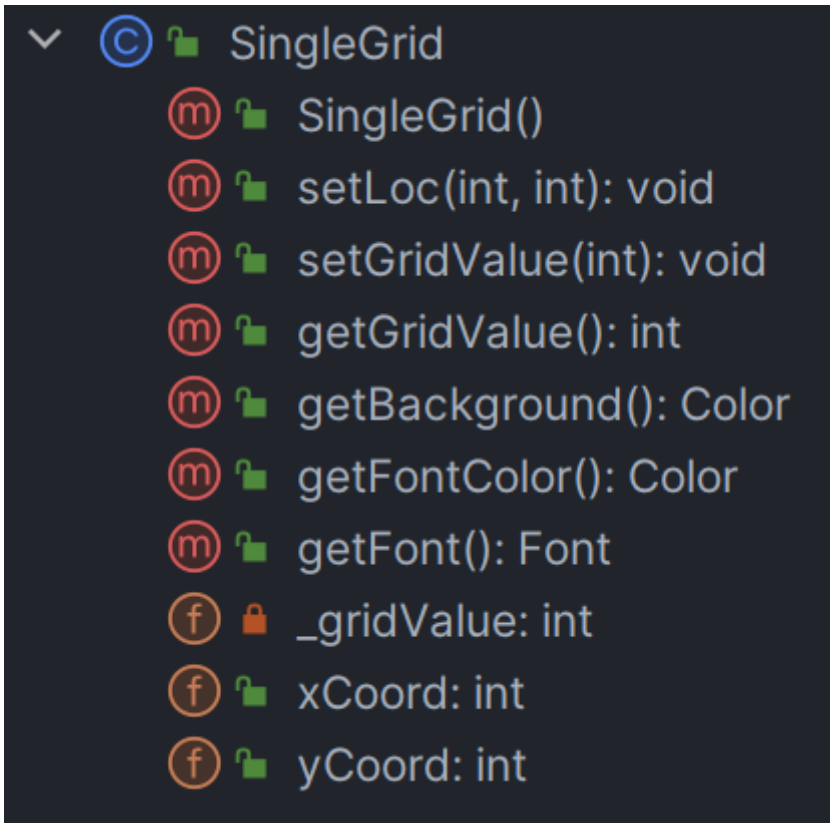
```
$ java --version
java 18.0.2.1 2022-08-18
Java(TM) SE Runtime Environment (build 18.0.2.1+1-1)
Java HotSpot(TM) 64-Bit Server VM (build 18.0.2.1+1-1, mixed mode, sharing)
```

架构设计

项目源代码由五个java文件构成，分别为`SingleGrid.java`,`MainView.java`,`Game.java`,`ButtonController.java`,`Test.java`，实现了所有**基本功能**以及提高功能中的**实现保存游戏与排行榜**。

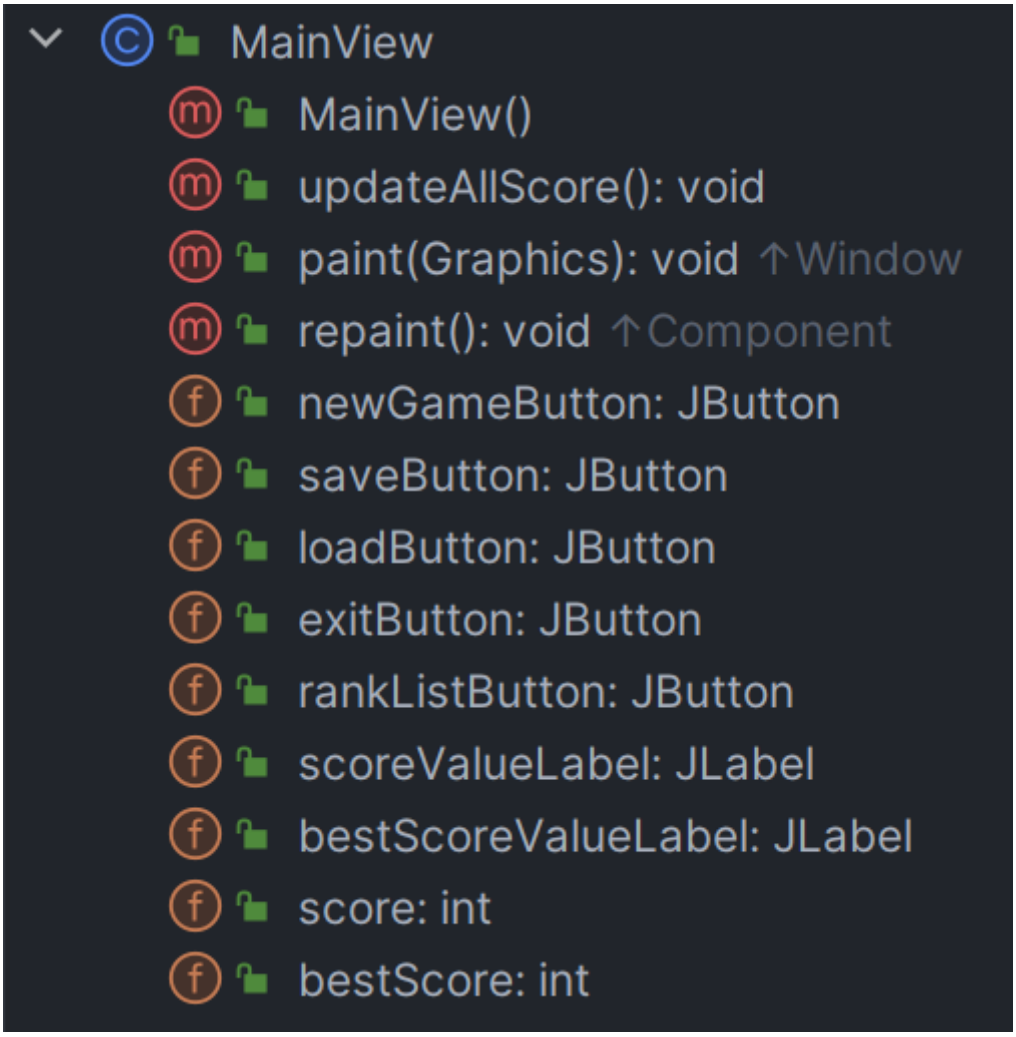
主要内容如下：

SingleGrid.java



- 该文件主要为每个单元格生成背景颜色，数值字体，数字颜色。其中0的数值字体颜色和数字颜色均为背景色，实现"隐身"的效果。
- 初始化时通过`void setLoc(int, int)`函数赋予每个单元格的绝对坐标，`void setGridValue(int)`赋予单元格数值。

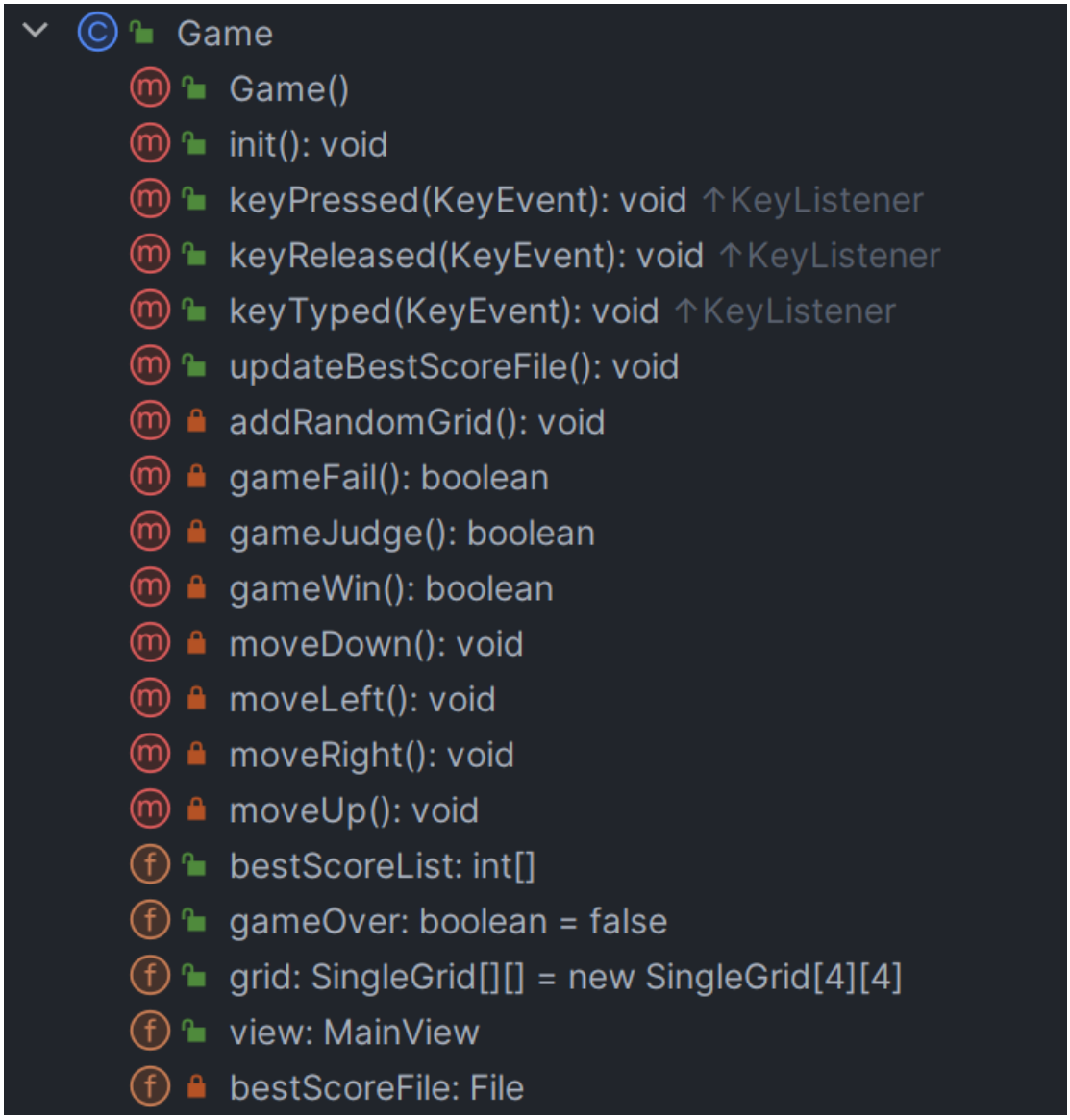
MainView.java



- 该文件主要为构建游戏框架，绘制基本图形。其中游戏面板的16个方格是重写`paint`函数绘制，其余部分是继承`JFrame`绘制。
- 该文件还包括了五个按钮以及两个得分面板，均为`public static`类型，便于修改。
- 为消除`repaint()`函数带来的闪烁，我重写了`repaint()`函数，并在每一调用的时候更新当前分数以及最佳分数的值。

```
public void repaint() {
    updateAllScore();
    update(getGraphics());
}
```

Game.java



- 该文件为游戏的主体部分，实现键盘监听、判断游戏是否结束、管理最高分数文件等重要功能。
- 判断游戏是否结束位于119行，针对成功还是失败弹出不同对话框，选择`Try Again`会重新开始游戏，选择`Exit`会退出游戏。

```
private boolean gameJudge() {
    if (gameWin() || gameFail()) {
        gameOver = true;

        String[] options = {"Try Again", "Exit"};
        int userChoice = JOptionPane.showOptionDialog(null,
                gameWin() ? "You Win!" : "You Lose!",
                gameWin() ? "Congratulations!" : "Game Over!",
                JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE,
                null, options, options[0]);
        if (userChoice == JOptionPane.YES_OPTION)
            ButtonController.newGame();
        else
            ButtonController.exitGame();
        return true;
    }
    return false;
}
```

- 键盘监听实现了`KeyListener`接口，实现方法在该文件末尾。

```
@Override
public void keyPressed(KeyEvent keyEvent) {
    switch (keyEvent.getKeyCode()) {
        case KeyEvent.VK_LEFT, KeyEvent.VK_A, KeyEvent.VK_H -> moveLeft();
        case KeyEvent.VK_RIGHT, KeyEvent.VK_D, KeyEvent.VK_L -> moveRight();
        case KeyEvent.VK_UP, KeyEvent.VK_W, KeyEvent.VK_K -> moveUp();
        case KeyEvent.VK_DOWN, KeyEvent.VK_S, KeyEvent.VK_J -> moveDown();
        case KeyEvent.VK_ESCAPE -> {
            int result = JOptionPane.showConfirmDialog(
                    null,
                    "Do you want to exit the game?",
                    "Exit", JOptionPane.YES_NO_OPTION);
            if (result == JOptionPane.YES_OPTION)
                ButtonController.exitGame();
        }
    }
}
```

- 移动操作从153行开始，以向左移动为例介绍，其余三个方向的移动类似

```
private void moveLeft() {
    for (int i = 0; i < 4; ++i)
        for (int j = 1; j < 4; ++j) {
            if (grid[i][j].getGridValue() == 0)
                continue;
            int k = j;
            while (k > 0 && grid[i][k - 1].getGridValue() == 0) {
                grid[i][k - 1].setGridValue(grid[i][k].getGridValue());
                grid[i][k].setGridValue(0);
                k--;
            }
            if (k > 0 && grid[i][k - 1].getGridValue() == grid[i]
[k].getGridValue()) {
                MainView.score += grid[i][k].getGridValue();
                MainView.bestScore = Math.max(MainView.bestScore,
MainView.score);
                grid[i][k - 1].setGridValue(grid[i][k - 1].getGridValue() *
2);
                grid[i][k].setGridValue(0);
            }
        }

    if (gameJudge()) {
        addRandomGrid();
        addRandomGrid();
    } else
        addRandomGrid();
    view.repaint();
}
```

  - 依次遍历每个方格，遇到0则跳过，当当前值不为0且左侧值为0时，直接将当前方格左移，直到不能移动；
  - 之后再比较与左侧方格数值是否相同，是则合并，并记录下此次移动的得分；
  - 最后随机添加一个一个值为2或4的方格，并更新游戏页面。

- 随机添加方格位于69行

```
private static void addRandomGrid() {
    if (gameOver)
        return;
    int cnt = 0;
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            if (grid[i][j].getGridValue() == 0)
                cnt++;
    if (cnt == 0)
        return;
    int x = (int) (Math.random() * 4);
    int y = (int) (Math.random() * 4);
    int value = (int) (Math.random() * 2 + 1) * 2;
    if (grid[x][y].getGridValue() == 0)
        grid[x][y].setGridValue(value);
    else
        addRandomGrid();
}
```

  - 若游戏结束或无剩余位置，无法创建，return；
  - 否则随机生成坐标及数值，若此位置已经被占，递归调用直到找到新位置。
- 更新最高分文件函数在138行
  - 游戏运行后会自动创建`Rank.rank`文件，内容为五个数字，用空格分开，存储五个最高分。
  - 若调用该函数时刻的`score`大于排行榜中5个分数的最小值，则将排行榜最小的分数改为`score`，并排序输出到`Rank.rank`中。

ButtonController.java

该文件为五个按钮添加监听，构造函数如下

```
public ButtonController() {
    MainView.saveButton.addActionListener(e -> saveGame());
    MainView.loadButton.addActionListener(e -> loadGame());
    MainView.newGameButton.addActionListener(e -> newGame());
    MainView.rankListButton.addActionListener(e -> RankList());
    MainView.exitButton.addActionListener(e -> exitGame());
}
```

Test.java

main函数在该文件中，负责启动游戏

```
public class Test {
    public static void main(String[] args) {
        new Game();
    }
}
```