目录

综合训练实验报告：扫雷游戏

一、题目及任务

本次实验内容的完成分成基础部分和扩展部分，基础部分全部完成最多可以获得综合训练的 80 分；在基础部分全部实现的基础上对扩展部分的实现实行加分制，且由每位同学自行填写分值表。

下面将从基础部分和扩展部分描述具体的实验要求。

- 基础部分
  - 能自动生成地雷的布局，而不能像预备知识中源文件那样是固定的地雷布局。
  - 能够让玩家在进入新游戏之前可以选择游戏的难度（Easy、Intermediate 和 Difficult），难度可以从“棋盘”的大小和地雷的数量两个维度进行设定，具体规则自定义。
  - 创建一个“File”菜单，该菜单具有如下菜单项：New Game、Reset Game 和 Exit。提示：可以使用 JMenuBar、JMenu 和 JMenuItem 类类型。
  - 游戏能够正常运行。
- 扩展部分
  - 在主界面上创建一个状态条（可以使用 JTextField 充当，并且在布局管理器中将其放置到合适的位置处），该状态条可以动态显示当前还有多少地雷没有被发现。（该项满分 7 分）
  - 设置一个计时器，记录玩家赢得游戏的时间。（该项满分 7 分）
  - 美化界面。（该项满分 6 分）

二、题目分析

通读题目和代码后，可知本题目分为基础部分和扩展部分，分条目分析如下

2.1 基础部分

1. 自动生成布局。采用 Java 自带的 Random 类在 MineMap 中编写一个 Randomize 函数，并将其嵌入构造函数即可。该函数需要将 isMined 全部改为 false 并随机选

中给定数量的位置为 true. 值得注意的是，随机选中过程中需要避免重复随机到同一个位置的可能。

2. 难度选择。

   - 难度选择需要一个提示框，采用 JOptionPane.showOptionDialog 的自定义构造方法即可，根据其返回值确定玩家的选择，生成地图。
   - 对于不同的难度，需要生成不同的 MineMap, 我的参数选择参考自 微软官方的在线游戏 并以 static 的形式存储在 MineMap 类中。为便于引用，新建一个 ValueModel 类作为存储单元，该类需要包含地图的行，列和地雷数。
   - 从美观的角度，在提供不同的难度后，需要对原代码中 CELLSIZE 适当修改；修改后需要重新调整 cell 之间的间隔。

3. 创建 File 菜单该菜单具有如下菜单项：New Game、Reset Game 和 Exit。

   - 使用 JMenuBar 创建目录框并添加一个“File”JMenu，在 File 中添加 New Game、Reset Game 和 Exit 三个 JMenuItem，最后把 JMenuBar 添加到面板中，这样 GUI 部分就完成了
   - 对于不同的 JMenuItem 需要设置不同的事件监听器，并添加监听后的动作

4. 游戏正常运行。基本代码中游戏逻辑缺失，需要补全

   - 对于每一次点击，分为左键和右键两种。
   - 左键表示踩雷。如果踩到雷（isMined = true），则游戏失败；否则递归展示雷数
   - 右键表示插旗。需要更改 isFlagged 并更改对应 cell 的显示标号
   - 点击后判断游戏是否胜利：即是否所有的地雷都被插旗或踩到地雷，该逻辑在代码中实现为用一个整型变量 RevealMines 记录，当 RevealMines = numMines 时游戏胜利；
   - 已经展示过雷数的位置不能再插旗
   - 失败/成功要弹出对应的窗口提示

2.2 扩展部分

1. 状态条。状态条采用 JTextField 类，题目要求动态显示还有多少地雷没有被发现这和游戏规则不符，所以这里改用剩余旗子数来表示。最后将状态条和 File 菜单集成为一个 Container 并放入面板的上部。

2. 计时器。计时器可以嵌入到状态条内，为了实现其更新，参考第二次作业的实现方式和相关资料，新建一个 Thread 维护状态条的更新。

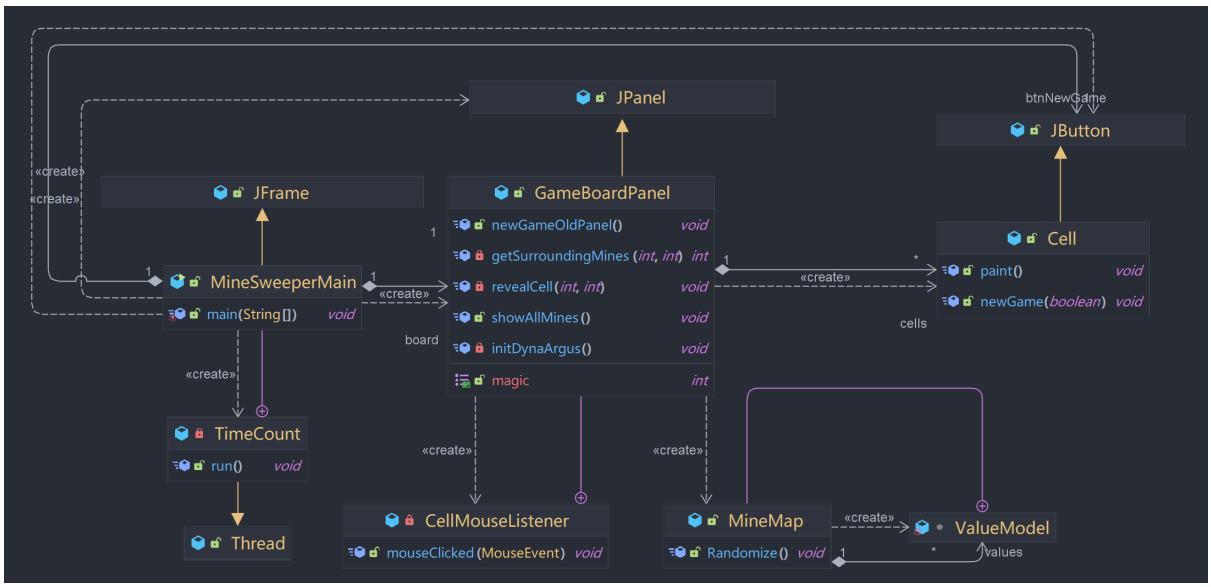
3. 美化界面。除了上文提到过的美化外，为了更贴近原版游戏，对地雷和旗子采用图标展现；修改字体大小和填充颜色等，具体可以看后文和视频中的效果展示。

三、数据设计

本次实验内容并无复杂数据结构设计，除了框架中已给的数据结构外在 MineMap 中新建了一个存储地图参数的内部类 ValueModel 包括三个整型变量 ROW,COL 和 MINES

对用以维护更新状态条的 Thread ，考虑人体手指的速度，设置为 0.05s 更新一次。

代码的整体 uml 框架如下



四、算法设计

本次实验内容主要需要完善代码逻辑和优化 GUI，没有算法上的设计需要。

五、主干代码说明

5.1 游戏逻辑补全

Listing 1: CellMouseListener

```
1 private class CellMouseListener extends MouseAdapter {
```

```
2       public void mouseClicked(MouseEvent e) {
3           // 获得触发此次鼠标事件的Cell对象
4           Cell sourceCell = (Cell) e.getSource();
5           if (sourceCell.isRevealed) return;
6           // 获得鼠标事件的类型，MouseEvent.BUTTON1为单击鼠标左键
7           if (e.getButton() == MouseEvent.BUTTON1) {
8               // [TODO 5] 如果当前Cell对象里面有地雷，则游戏结束；否则对该
                    Cell对象执行挖雷操作
9               if (sourceCell.isMined) {
10                  sourceCell.setIcon(MineIcon);
11                  System.out.println("Game Over");
12                  sourceCell.isRevealed = true;
13                  showAllMines();
14                  int result_of_dialog = JOptionPane.showConfirmDialog(((Cell
                        ) e.getSource()).getRootPane(), "Game Over!\n\nDo you
                        want a new game?", "", JOptionPane.YES_NO_OPTION);
15                  if (result_of_dialog == 0) newGameOldPanel();
16              } else {
17                  if (sourceCell.isFlagged) {
18                      sourceCell.isFlagged = false;
19                      flag_number++;
20                  }
21                  revealCell(sourceCell.row, sourceCell.col);
22              }
23          } else if (e.getButton() == MouseEvent.BUTTON3) { //MouseEvent.
                BUTTON3为单击鼠标右键
24              // [TODO 6] 如果该Cell对象上插了旗子，那么就去掉旗子；否则将该
                    Cell对象打上旗子的标记。
25              if (sourceCell.isFlagged) {
26                  sourceCell.setIcon(null);
27                  sourceCell.isFlagged = false;
28                  if (sourceCell.isMined) RevealMines--;
29                  flag_number++;
30              } else if (flag_number == 0) {
31                  JOptionPane.showConfirmDialog(((Cell) e.getSource()).
                        getRootPane(),"ERROR!!!\n\nYou have run out of flags",
```

```
                    "", JOptionPane.DEFAULT_OPTION);
32              } else {
33                  flag_number--;
34                  if (sourceCell.isMined) RevealMines++;
35                  sourceCell.setIcon(FlagIcon);
36                  sourceCell.isFlagged = true;
37                  // [TODO 7] 当对Cell单元格对象执行了挖雷操作之后判断玩家是
                        否赢得该游戏
38                  if (RevealMines == numMines) {
39                      JOptionPane.showConfirmDialog(((Cell) e.getSource()).
                            getRootPane(),"WIN!!!\n\nYou have found all " +
                            numMines + " mines in " + UsedTime + "seconds.\n" +
                             "Do you want a new game?", "WIN", JOptionPane.
                            DEFAULT_OPTION);
40                      newGameOldPanel();
41                  }
42              }
43          }
44      }
45  }
```

5.2 难度选择

Listing 2: void getMagic()

```
1  public int getMagic() {
2      Object[] options = {"easy", "intermediate", "hard"};  //自定义按钮上的
           文字
3      return JOptionPane.showOptionDialog(null, "请选择难度", "New Game",
           JOptionPane.YES_NO_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE,
           null, options, options[0]);
4  }
```

5.3 随机排布地雷

Listing 3: void Randomize()

```
public void Randomize(){
    for(int i = 0;i<row;i++){
        for (int j = 0;j<col;j++){
            isMined[i][j] = false;
        }
    }
    Random ran = new Random();
    for (int i = 0; i < numMines; i++) {
        int j = ran.nextInt(0, row);
        int k = ran.nextInt(0, col);
        if (isMined[j][k]) i--;
        else isMined[j][k] = true;
    }
}
```

5.4 地图参数

Listing 4: ValueModel

```
static class ValueModel {
    int ROW;
    int COL;
    int MINES;

    public ValueModel(int r, int c, int m) {
        ROW = r;
        COL = c;
        MINES = m;
    }
}

public static ValueModel[] values = {
    new ValueModel(9, 9, 10),
    new ValueModel(16, 16, 40),
    new ValueModel(16, 30, 99)
};
```

5.5 菜单栏监听器

Listing 5: addActionListener

```
MenuNewGame.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent event) {
        board.newGameOldPanel();
    }
});
MenuExit.addActionListener(e -> System.exit(0));
MenuResetGame.addActionListener(e -> {
    dispose();
    MineSweeperMain newMS = new MineSweeperMain();
});
```

六、运行结果展示

根据本次实验要求，运行结果通过附件以视频的形式展示

七、总结和收获

本次实验依托老师给的框架，完善了游戏的基本逻辑。基于官方游戏优化了交互界面，并新增了难度选择。

本次实验让我更深入地了解了 Java GUI 类的构建思路和主要的 API。

对于 Thread 的探索为我揭开了 Java 多线程编程神秘的面纱，期待在后续学习中能更加深入了解。

本次实验也让我初步了解到监听器的妙用，其背后的思想和实现机制让我很感兴趣。

扩展任务自我打分表		
功能列表	打分说明	分值
扩展 1	能够实时显示游戏的状态信息为 7 分	7
扩展 2	可以正确将时间显示在扩展 1 的状态条中显示为 7 分	7
扩展 3	和运行原型的界面效果有变化即可加 2 分， 在此基础上视觉效果更赏心悦目加 1-4 分。 （可以通过同学为你的界面打 Call 侧面证明界面的效果）	5
填表人：杨豪		

附录：源代码汇总

Cell.java

Listing 6: Cell.java

```
1  package minesweeper;
2
3  import javax.swing.*;
4  import java.awt.*;
5
6  /**
7   * 类Cell是对JButton的类定制（也就是JButton的一个子类，其目的是表示扫雷游戏
        的一个单元格；
8   * 该类中定义了row/column属性以及相关状态函数。
9   */
10 public class Cell extends JButton{
11     // 为Cell单元格定义若干颜色和字体常量
12     // 这些常量将随着Cell单元格的状态变化而被使用
13     private static final long serialVersionUID = 1L;  // to prevent serial
           warning
14     public static final Color BG_NOT_REVEALED = new Color(0,191,255);
15     public static final Color FG_NOT_REVEALED = Color.RED;     // flag,
           mines
16     public static final Color BG_REVEALED = Color.DARK_GRAY;
17     public static final Color FG_REVEALED = Color.YELLOW;      // number of
           mines
18     public static final Font FONT_NUMBERS = new Font("Monospaced", Font.
           BOLD, 15);
19
20     // 定义Cell对象的属性，比如row和col值用来表示单元格在最终“棋盘”上的位
           置定位
21     int row, col;
22     boolean isRevealed; // 标记是否已经被挖出？
23     boolean isMined;    // 标记是否是地雷？
```

```
24      boolean isFlagged;  // 标记是否被玩家插上了一个小红旗
25
26      public Cell(int row, int col){
27          this.row = row;
28          this.col = col;
29          super.setFont(FONT_NUMBERS);
30          setBorder(null);
31      }
32
33      public void newGame(boolean b){
34          this.isRevealed = false;
35          this.isFlagged = false;
36          this.isMined = b;
37          super.setEnabled(true);
38          super.setText("");
39          paint();
40      }
41      /** 基于单元格的状态进行绘制 */
42      public void paint(){
43          super.setIcon(null);
44          super.setForeground(isRevealed? FG_REVEALED: FG_NOT_REVEALED);
45          super.setBackground(isRevealed? BG_REVEALED: BG_NOT_REVEALED);
46      }
47  }
```

MineMap.java

Listing 7: MineMap.java

```
1   package minesweeper;
2
3   import java.util.Random;
4
5   /** 这个类主要用来存储地雷在单元格中的位置，目前这个类只是一个示意，所以地
        雷都是固定位置。 */
6   public class MineMap {
```

```
7
8       static class ValueModel {
9           int ROW;
10          int COL;
11          int MINES;
12
13          public ValueModel(int r, int c, int m) {
14              ROW = r;
15              COL = c;
16              MINES = m;
17          }
18      }
19
20      public static ValueModel[] values = {
21              new ValueModel(9, 9, 10),
22              new ValueModel(16, 16, 40),
23              new ValueModel(16, 30, 99)
24      };
25
26
27      int numMines;
28      int row;
29      int col;
30      boolean[][] isMined;
31
32      public MineMap(int MineMagicNum) {
33          this.numMines = values[MineMagicNum].MINES;
34          this.row = values[MineMagicNum].ROW;
35          this.col = values[MineMagicNum].COL;
36          isMined = new boolean[row][col];
37          Randomize();
38      }
39
40      public void Randomize(){
41          for(int i = 0;i<row;i++){
42              for (int j = 0;j<col;j++){
```

```
43                  isMined[i][j] = false;
44              }
45          }
46          Random ran = new Random();
47          for (int i = 0; i < numMines; i++) {
48              int j = ran.nextInt(0, row);
49              int k = ran.nextInt(0, col);
50              if (isMined[j][k]) i--;
51              else isMined[j][k] = true;
52          }
53      }
54  }
```

GameBoardPanel.java

Listing 8: GameBoardPanel.java

```
1  package minesweeper;
2
3  import javax.swing.*;
4  import java.awt.*;
5  import java.awt.event.MouseAdapter;
6  import java.awt.event.MouseEvent;
7
8  //这个类的作用就是充当“棋盘”
9  public class GameBoardPanel extends JPanel {
10
11     // some redefined MAGIC number
12     private static final long serialVersionUID = 1L;
13     public static final int CELL_SIZE = 30;
14
15     final int numMines;
16     int flag_number, UsedTime , RevealMines, MineMagicNum;
17     final int ROWS, COLS, CANVAS_WIDTH, CANVAS_HEIGHT;
18
19     // some Icons
```

```
20      public static final ImageIcon MineIcon = new ImageIcon(new ImageIcon("
            ./src/minesweeper/icon/th.jpg").
21              getImage().getScaledInstance(CELL_SIZE, CELL_SIZE, Image.
                    SCALE_DEFAULT));
22      public static final ImageIcon FlagIcon = new ImageIcon(new ImageIcon("
            ./src/minesweeper/icon/flag.jpg").
23              getImage().getScaledInstance(CELL_SIZE, CELL_SIZE, Image.
                    SCALE_DEFAULT));
24      Cell[][] cells;
25
26      private void initDynaArgus(){
27          flag_number = numMines;
28          UsedTime = 0;
29          RevealMines = 0;
30      }
31
32
33      public GameBoardPanel() {
34          MineMagicNum = getMagic();
35          numMines = MineMap.values[MineMagicNum].MINES;
36          initDynaArgus();
37          ROWS = MineMap.values[MineMagicNum].ROW;
38          COLS = MineMap.values[MineMagicNum].COL;
39          CANVAS_HEIGHT = CELL_SIZE * ROWS;
40          CANVAS_WIDTH = CELL_SIZE * COLS;
41          cells = new Cell[ROWS][COLS];
42          super.setLayout(new GridLayout(ROWS, COLS, 1, 1));
43          for (int row = 0; row < ROWS; ++row) {
44              for (int col = 0; col < COLS; ++col) {
45                  cells[row][col] = new Cell(row, col);
46                  super.add(cells[row][col]);
47              }
48          }
49          // [TODO 3] 为所有的Cell单元对象创建一个共享的鼠标事件监听器
50          CellMouseListener listener = new CellMouseListener();
51          // [TODO 4] 通过下面的循环，将每个Cell对象的鼠标事件监听器对象设为
```

```
            listener
52          for (int row = 0; row < ROWS; ++row) {
53              for (int col = 0; col < COLS; ++col) {
54                  cells[row][col].addMouseListener(listener);
55              }
56          }
57          super.setPreferredSize(new Dimension(CANVAS_WIDTH, CANVAS_HEIGHT));
58      }
59
60      // get Magic num with temporal dialog.
61      public int getMagic() {
62          Object[] options = {"easy", "intermediate", "hard"};  //自定义按钮
                上的文字
63          return JOptionPane.showOptionDialog(null, "请选择难度",
64                  "New Game", JOptionPane.YES_NO_CANCEL_OPTION, JOptionPane.
                       QUESTION_MESSAGE,
65                  null, options, options[0]);
66      }
67
68      // 初始化一个新的游戏
69      public void newGameOldPanel() {
70          // 通过MineMap获得新游戏中的地雷数据的布局
71          MineMap mineMap = new MineMap(MineMagicNum);
72          initDynaArgus();
73          // 根据mineMap中的数据初始化每个Cell单元对象
74          for (int row = 0; row < ROWS; ++row) {
75              for (int col = 0; col < COLS; ++col) {
76                  cells[row][col].newGame(mineMap.isMined[row][col]);
77              }
78          }
79      }
80
81      // 获得[srcRow, srcCol]Cell单元对象周围的8个邻居的地雷总数
82      private int getSurroundingMines(int srcRow, int srcCol) {
83          int numMines = 0;
84          for (int row = srcRow - 1; row <= srcRow + 1; ++row) {
```

```
85              for (int col = srcCol - 1; col <= srcCol + 1; ++col) {
86                  if (row >= 0 && row < ROWS && col >= 0 && col < COLS)
87                      if (cells[row][col].isMined) numMines++;
88              }
89          }
90          return numMines;
91      }
92
93      // 对[srcRow, srcCol]Cell单元对象执行挖雷操作
94      // 如果该单元格对象中的标记的雷的数量为0，那么就自动递归对其周围8个邻居
            执行挖雷操作
95      private void revealCell(int srcRow, int srcCol) {
96          if (cells[srcRow][srcCol].isFlagged) flag_number++;
97          int resideMine = getSurroundingMines(srcRow, srcCol);
98          cells[srcRow][srcCol].setText(resideMine + "");
99          cells[srcRow][srcCol].isRevealed = true;
100         cells[srcRow][srcCol].paint();
101         if (resideMine == 0) {
102             for (int row = srcRow - 1; row <= srcRow + 1; ++row) {
103                 for (int col = srcCol - 1; col <= srcCol + 1; ++col) {
104                     if (row >= 0 && row < ROWS && col >= 0 && col < COLS)
105                         if (!cells[row][col].isRevealed) revealCell(row,
                                col);
106                 }
107             }
108         }
109     }
110
111     public void showAllMines() {
112         for (int i = 0; i < ROWS; i++)
113             for (int j = 0; j < COLS; j++) if (cells[i][j].isMined) cells[i
                    ][j].setIcon(MineIcon);
114     }
115
116     // [TODO 2] 定义一个内部类，该类的作用为鼠标事件监听器
117     private class CellMouseListener extends MouseAdapter {
```

```
118         public void mouseClicked(MouseEvent e) {
119             // 获得触发此次鼠标事件的Cell对象
120             Cell sourceCell = (Cell) e.getSource();
121             if (sourceCell.isRevealed) return;
122             // 获得鼠标事件的类型，MouseEvent.BUTTON1为单击鼠标左键
123             if (e.getButton() == MouseEvent.BUTTON1) {
124                 // [TODO 5] 如果当前Cell对象里面有地雷，则游戏结束；否则对
                        该Cell对象执行挖雷操作
125                 if (sourceCell.isMined) {
126                     sourceCell.setIcon(MineIcon);
127                     System.out.println("Game Over");
128                     sourceCell.isRevealed = true;
129                     showAllMines();
130                     int result_of_dialog = JOptionPane.showConfirmDialog(((
                            Cell) e.getSource()).getRootPane(),
131                                 "Game Over!\n\nDo you want a new game?", "",
                                        JOptionPane.YES_NO_OPTION);
132                     if (result_of_dialog == 0) newGameOldPanel();
133                 } else {
134                     if (sourceCell.isFlagged) {
135                         sourceCell.isFlagged = false;
136                         flag_number++;
137                     }
138                     revealCell(sourceCell.row, sourceCell.col);
139                 }
140             } else if (e.getButton() == MouseEvent.BUTTON3) { //MouseEvent.
                    BUTTON3为单击鼠标右键
141                 // [TODO 6] 如果该Cell对象上插了旗子，那么就去掉旗子；否则
                        将该Cell对象打上旗子的标记。
142                 if (sourceCell.isFlagged) {
143                     sourceCell.setIcon(null);
144                     sourceCell.isFlagged = false;
145                     if (sourceCell.isMined) RevealMines--;
146                     flag_number++;
147                 } else if (flag_number == 0) {
148                     JOptionPane.showConfirmDialog(((Cell) e.getSource()).
```

```
                        getRootPane(),
149                         "ERROR!!!\n\nYou have run out of flags", "",
                               JOptionPane.DEFAULT_OPTION);
150                 } else {
151                     flag_number--;
152                     if (sourceCell.isMined) RevealMines++;
153                     sourceCell.setIcon(FlagIcon);
154                     sourceCell.isFlagged = true;
155                     // [TODO 7] 当对Cell单元格对象执行了挖雷操作之后判断玩
                           家是否赢得该游戏
156                     if (RevealMines == numMines) {
157                         JOptionPane.showConfirmDialog(((Cell) e.getSource()
                               ).getRootPane(),
158                             "WIN!!!\n\nYou have found all " + numMines
                                   + " mines in " + UsedTime + "seconds.\n
                                   " +
159                                 "Do you want a new game?", "WIN",
                                       JOptionPane.DEFAULT_OPTION);
160                         newGameOldPanel();
161                     }
162                 }
163             }
164         }
165     }
166 }
```

MineSweeperMain.java

Listing 9: MineSweeperMain.java

```
1 package minesweeper;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.ActionEvent;
6 import java.awt.event.ActionListener;
```

```
7
8  /**
9   * 挖雷游戏的主程序。
10  * 单击鼠标左键对单元格执行挖雷操作。
11  * 单击鼠标右键用来对单元格执行添加标记，或者移除标记，标记疑似有地雷的单元
        格。
12  * 如果所有没有地雷的单元格都执行了挖雷操作，那么玩家赢得游戏。
13  * 如果对某个有地雷的单元格执行了挖雷操作，那么玩家输。
14  */
15 public class MineSweeperMain extends JFrame {
16     private static final long serialVersionUID = 1L;
17     JTextField gameTXT = new JTextField();
18     GameBoardPanel board = new GameBoardPanel();
19     JButton btnNewGame = new JButton("New Game");
20     JMenuBar GameMenu = new JMenuBar();
21     JMenu MenuFile = new JMenu("File");
22     JMenuItem MenuNewGame = new JMenuItem("New Game");
23     JMenuItem MenuResetGame = new JMenuItem("Reset Game");
24     JMenuItem MenuExit = new JMenuItem("Exit");
25
26     public MineSweeperMain() {
27         GameMenu.add(MenuFile);
28         JPanel top = new JPanel();
29         top.add(GameMenu, BorderLayout.WEST);
30         top.add(gameTXT, BorderLayout.EAST);
31         Container cp = this.getContentPane();
32         gameTXT.setEditable(false);
33         gameTXT.setAlignmentX(CENTER_ALIGNMENT);
34         gameTXT.setAlignmentY(CENTER_ALIGNMENT);
35
36         MenuFile.add(MenuNewGame);
37         MenuFile.add(MenuResetGame);
38         MenuFile.add(MenuExit);
39
40         cp.setLayout(new BorderLayout());
41         cp.add(top, BorderLayout.NORTH);
```

```
42          cp.add(board, BorderLayout.CENTER);
43          cp.add(btnNewGame, BorderLayout.SOUTH);
44          // 使用匿名类的方式为 btnNewGame 按钮添加 Action 事件监听器
45          btnNewGame.addActionListener(new ActionListener() {
46              public void actionPerformed(ActionEvent event) {
47                  board.newGameOldPanel();
48              }
49          });
50          MenuNewGame.addActionListener(new ActionListener() {
51              public void actionPerformed(ActionEvent event) {
52                  board.newGameOldPanel();
53              }
54          });
55          MenuExit.addActionListener(e -> System.exit(0));
56          MenuResetGame.addActionListener(e -> {
57              dispose();
58              MineSweeperMain newMS = new MineSweeperMain();
59          });
60
61          board.newGameOldPanel();
62          TimeCount tc = new TimeCount();
63          tc.start();
64          pack(); // Pack the UI components, instead of setSize()
65          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
66          setTitle("Minesweeper");
67          setVisible(true);
68      }
69
70      // a Thread to maintain the text field.
71      private class TimeCount extends Thread {
72          public void run() {
73              while (true) {
74                  try {
75                      ++board.UsedTime;
76                      for (int i = 0; i < 20; ++i) {
77                          gameTXT.setText("time : " + board.UsedTime + "s\t\
```

```
                        tleft flags : " + board.flag_number);
78                      Thread.sleep(50);
79                  }
80              } catch (Exception e) {
81                  System.out.println(e);
82              }
83          }
84      }
85  }
86
87  public static void main(String[] args) {
88      // [TODO 1] 使用安全的方式启动下面的构造函数
89      javax.swing.SwingUtilities.invokeLater(new Runnable() {
90          @Override
91          public void run() {
92              new MineSweeperMain();
93          }
94      });
95  }
96  }
```

参考文献

A. 解题内容参考

a. Java® Platform, Standard Edition & Java Development Kit Version 17 API Specification [DB/OL]
https://docs.oracle.com/en/java/javase/17/docs/api/index.html, 2022.

b. MicroSoft Online Game. [DB/OL].
https://www.msn.com/en-us/play/microsoft-minesweeper/cg-msminesweeper, 2022.

B. LaTeX 代码参考

c. 【LaTeX】自用简洁模板（六）：学校作业

d. LaTeX 里「添加程序代码」的完美解决方案