

GitHub Stuff\project-5-minewalker-wildestPantaloons\MineWalkerPanel.java

```
1  import java.util.ArrayList;
2  import java.util.Random;
3  import java.awt.BorderLayout;
4  import java.awt.Color;
5  import java.awt.GridLayout;
6  import java.awt.Point;
7  import java.awt.event.ActionEvent;
8  import java.awt.event.ActionListener;
9
10 import javax.swing.JButton;
11 import javax.swing.JLabel;
12 import javax.swing.JOptionPane;
13 import javax.swing.JPanel;
14 import javax.swing.JTextField;
15
16 /**
17  * Program that runs and plays the game "Mine walker"
18  *
19  * @author Everett Wilcox
20  */
21 public class MineWalkerPanel extends JPanel {
22     private TileButton[][] buttons;
23     private int lives = 5;
24     private JLabel livesLabel = new JLabel("lives = " + lives);
25     private int hints = 8;
26     private JLabel hintsLabel = new JLabel("Hints = " + hints + " |");
27     private int gridHeight;
28     private int gridWidth;
29     private Color mine0 = Color.GREEN;
30     private Color mine1 = Color.YELLOW;
31     private Color mine2 = Color.ORANGE;
32     private Color mine3 = Color.RED;
33     private int playerX = 0;
34     private int playerY = 0;
35     private ArrayList<Point> mineLocations = new ArrayList<>();
36     private ArrayList<Point> visitedMineLocations = new ArrayList<>();
37     JTextField minePercentageTextField;
38
39     /**
40     * This constructor is called when the panel is created
41     *
42     * @param width
43     * @param height
44     */
45     public MineWalkerPanel(int width, int height) {
46
47         setLayout(new BorderLayout());
48
49         JPanel controlsPanel = new JPanel();
50         JButton newGameButton = new JButton("New Game");
51         newGameButton.addActionListener(new ActionListener() {
52             @Override
53             public void actionPerformed(ActionEvent e) {
```

```

54         newGame();
55     }
56 });
57 JButton hintButton = new JButton("Hint");
58 hintButton.addActionListener(new ActionListener() {
59     @Override
60     public void actionPerformed(ActionEvent e) {
61         Hint();
62     }
63 });
64 JLabel minePercentLabel = new JLabel("Percentage of Mines: ");
65 minePercentageTextField = new JTextField("30", 2);
66 controlsPanel.add(livesLabel);
67
68 controlsPanel.add(newGameButton);
69 controlsPanel.add(hintButton);
70 controlsPanel.add(hintsLabel);
71 controlsPanel.add(minePercentLabel);
72 controlsPanel.add(minePercentageTextField);
73 this.add(controlsPanel, BorderLayout.NORTH);
74 gridHeight = height;
75 gridWidth = width;
76
77 JPanel gridPanel = new JPanel();
78 this.buttons = new TileButton[width][height];
79
80 for (int i = 0; i < width; i++) {
81     for (int j = 0; j < height; j++) {
82
83         TileButton peg = new TileButton(i, j);
84
85         this.buttons[i][j] = peg;
86
87         peg.setEnabled(false);
88         peg.setPath(false);
89         peg.setMine(false);
90         peg.setNumMineNeighbors(0);
91         peg.setText("");
92         peg.setHidden(true);
93         peg.setBackground(Color.LIGHT_GRAY);
94
95         if (i == playerY && j == playerX) {
96             peg.setHidden(false);
97             peg.setBackground(mine0);
98             peg.setText("X");
99         }
100
101         if (playerY < gridHeight - 1) {
102             if (i == playerY + 1 && j == playerX) {
103                 peg.setEnabled(true);
104                 peg.setText("v");
105                 peg.addActionListener(new ActionListener() {
106                     @Override
107                     public void actionPerformed(ActionEvent e) {
108                         moveButtonsDown();
109                     }

```

```

110         });
111     }
112 }
113
114     if (i == playerY - 1 && j == playerX) {
115         peg.setEnabled(true);
116         peg.setText("^");
117         peg.addActionListener(new ActionListener() {
118             @Override
119             public void actionPerformed(ActionEvent e) {
120                 moveButtonsUp();
121             }
122         });
123     }
124
125     if (i == playerY && j == playerX + 1) {
126         peg.setEnabled(true);
127         peg.setText(">");
128         peg.addActionListener(new ActionListener() {
129             @Override
130             public void actionPerformed(ActionEvent e) {
131                 moveButtonsRight();
132             }
133         });
134     }
135 }
136
137     if (i == playerY && j == playerX - 1) {
138         peg.setEnabled(true);
139         peg.setText("<");
140         peg.addActionListener(new ActionListener() {
141             @Override
142             public void actionPerformed(ActionEvent e) {
143                 moveButtonsLeft();
144             }
145         });
146     }
147     gridPanel.add(peg);
148 }
149 }
150
151 // Call the method to print the mine locations
152 gridPanel.setLayout(new GridLayout(width, height));
153 this.add(gridPanel, BorderLayout.CENTER);
154
155 JPanel colorKeyPanel = new JPanel();
156 colorKeyPanel.setLayout(new GridLayout(0, 1));
157 JButton mine0Button = new JButton("0 Mine Neighbors");
158 JButton mine1Button = new JButton("1 Mine Neighbor");
159 JButton mine2Button = new JButton("2 Mine Neighbors");
160 JButton mine3Button = new JButton("3 Mine Neighbors");
161 JButton explodedMine = new JButton("Exploded Mine");
162 JButton xIsYou = new JButton("X <- You");
163 mine0Button.setBackground(Color.GREEN);
164 mine0Button.setBorderPainted(false);
165 mine0Button.setFocusPainted(false);

```

```

166     mine1Button.setBackground(Color.YELLOW);
167     mine1Button.setBorderPainted(false);
168     mine1Button.setFocusPainted(false);
169     mine2Button.setBackground(new Color(255, 130, 0));
170     mine2Button.setBorderPainted(false);
171     mine2Button.setFocusPainted(false);
172     mine3Button.setBackground(Color.RED);
173     mine3Button.setBorderPainted(false);
174     mine3Button.setFocusPainted(false);
175     explodedMine.setBackground(Color.BLACK);
176     explodedMine.setForeground(Color.WHITE);
177     explodedMine.setBorderPainted(false);
178     explodedMine.setFocusPainted(false);
179     xIsYou.setBackground(Color.WHITE);
180     xIsYou.setBorderPainted(false);
181     xIsYou.setFocusPainted(false);
182     colorKeyPanel.add(mine0Button);
183     colorKeyPanel.add(mine1Button);
184     colorKeyPanel.add(mine2Button);
185     colorKeyPanel.add(mine3Button);
186     colorKeyPanel.add(explodedMine);
187     colorKeyPanel.add(xIsYou);
188     this.add(colorKeyPanel, BorderLayout.WEST);
189
190     ArrayList<Point> randomPath = RandomPath.getPath(width);
191     for (Point point : randomPath) {
192         int x = point.x;
193         int y = point.y;
194         TileButton tileButton = buttons[x][y];
195         buttons[0][1].setPath(true);
196         buttons[1][0].setPath(true);
197         tileButton.setPath(true);
198     }
199
200     int numTiles = width * height;
201     int pathLength = randomPath.size();
202     int minePercent = Integer.parseInt(minePercentageTextField.getText());
203
204     if (minePercent > 50) {
205         minePercent = 50;
206         minePercentageTextField.setText("50");
207     }
208
209     int numMinesToPlace = (numTiles - pathLength) * minePercent / 100;
210
211
212
213     // Place the mines
214     Random rand = new Random();
215     while (numMinesToPlace > 0) {
216         int x = rand.nextInt(width);
217         int y = rand.nextInt(height);
218         TileButton tileButton = buttons[x][y];
219
220         if (!tileButton.isPath() && !tileButton.isMine()) {
221             tileButton.setHidden(true);

```

```

222         tileButton.setMine(true);
223         numMinesToPlace--;
224         mineLocations.add(new Point(x, y));
225     }
226 }
227 }
228
229 /**
230  * Removes the action listeners from the list of buttons
231  *
232  * @param button
233  */
234 private void removeListeners(JButton button) {
235     ActionListener[] listeners = button.getActionListeners();
236     for (ActionListener listener : listeners) {
237         button.removeActionListener(listener);
238     }
239 }
240
241 /**
242  * Function that moves the player position to the right
243  */
244 private void moveButtonsRight() {
245     System.out.println("New Player Position: X = " + (playerX + 1) + ", Y = " + playerY);
246 ;
247     removePreviousArrows();
248
249     playerX++;
250
251     if (buttons[playerY][playerX].isMine() && !visitedMineLocations.contains(new
Point(playerX, playerY)) {
252         removeLife();
253         if (playerX < gridWidth - 1) {
254             buttons[playerY][playerX].setHidden(false);
255             colorStuff();
256         }
257         if (playerX == gridWidth - 1) {
258             buttons[playerY][playerX].setHidden(false);
259             buttons[playerY][playerX].setBackground(Color.BLACK);
260         }
261         playerX--;
262
263         if (playerX < gridWidth - 1) {
264             buttons[playerY][playerX + 1].setText(">");
265
266             buttons[playerY][playerX + 1].setEnabled(false);
267             buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
268                 @Override
269                 public void actionPerformed(ActionEvent e) {
270                     moveButtonsRight();
271                 }
272             });
273         }
274     } else if (playerX < 0) {
275

```

```

276         if (buttons[playerY][playerX].isMine() && visitedMineLocations.contains(new
Point(playerX, playerY))) {
277             playerX--;
278
279             if (playerX < gridWidth - 1) {
280                 buttons[playerY][playerX + 1].setText(">");
281                 buttons[playerY][playerX + 1].setEnabled(false);
282                 buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
283                     @Override
284                     public void actionPerformed(ActionEvent e) {
285                         moveButtonsRight();
286                     }
287                 });
288             }
289         }
290
291     } else if (playerX < 0) {
292
293         if (buttons[playerY][playerX + 1].isMine()
294             && visitedMineLocations.contains(new Point(playerX + 1, playerY))) {
295
296             if (playerX < gridWidth - 1) {
297                 buttons[playerY][playerX + 1].setText(">");
298                 buttons[playerY][playerX + 1].setEnabled(false);
299                 buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
300                     @Override
301                     public void actionPerformed(ActionEvent e) {
302                         moveButtonsRight();
303                     }
304                 });
305             }
306         }
307
308     } else if (!buttons[playerY][playerX].isMine()) {
309         if (playerX < gridWidth - 1) {
310             if (!visitedMineLocations.contains(new Point(playerX + 1, playerY))) {
311                 buttons[playerY][playerX + 1].setText(">");
312                 buttons[playerY][playerX + 1].setEnabled(true);
313                 buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
314                     @Override
315                     public void actionPerformed(ActionEvent e) {
316                         moveButtonsRight();
317                     }
318                 });
319             } else {
320                 buttons[playerY][playerX + 1].setText(">");
321                 buttons[playerY][playerX + 1].setEnabled(false);
322                 buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
323                     @Override
324                     public void actionPerformed(ActionEvent e) {
325                         moveButtonsRight();
326                     }
327                 });
328             }
329         }
330     }

```

```

331         removeActionListenersFromNonAdjacentButtons();
332     }
333
334     buttons[playerY][playerX].setText("X");
335     buttons[playerY][playerX].setHidden(false);
336     colorStuff();
337     buttons[playerY][playerX].setEnabled(false);
338
339     if ((playerX > 0)) {
340         if (!visitedMineLocations.contains(new Point(playerX - 1, playerY))) {
341             buttons[playerY][playerX - 1].setText("<");
342             buttons[playerY][playerX - 1].setEnabled(true);
343             buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
344                 @Override
345                 public void actionPerformed(ActionEvent e) {
346                     moveButtonsLeft();
347                 }
348             });
349         } else {
350             buttons[playerY][playerX - 1].setText("<");
351             buttons[playerY][playerX - 1].setEnabled(false);
352             buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
353                 @Override
354                 public void actionPerformed(ActionEvent e) {
355                     moveButtonsLeft();
356                 }
357             });
358         }
359     }
360
361     if ((playerY > 0)) {
362         if (!visitedMineLocations.contains(new Point(playerX, playerY - 1))) {
363             buttons[playerY - 1][playerX].setText("^");
364             buttons[playerY - 1][playerX].setEnabled(true);
365             buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
366                 @Override
367                 public void actionPerformed(ActionEvent e) {
368                     moveButtonsUp();
369                     System.out.println("Up");
370                 }
371             });
372         } else {
373             buttons[playerY - 1][playerX].setText("^");
374             buttons[playerY - 1][playerX].setEnabled(false);
375             buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
376                 @Override
377                 public void actionPerformed(ActionEvent e) {
378                     moveButtonsUp();
379                     System.out.println("Up");
380                 }
381             });
382         }
383     }
384
385     if ((playerY < gridHeight - 1)) {
386         if (!visitedMineLocations.contains(new Point(playerX, playerY + 1))) {

```

```

387         buttons[playerY + 1][playerX].setText("v");
388         buttons[playerY + 1][playerX].setEnabled(true);
389         buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
390             @Override
391             public void actionPerformed(ActionEvent e) {
392                 moveButtonsDown();
393                 System.out.println("Down");
394             }
395         });
396     } else {
397         buttons[playerY + 1][playerX].setText("v");
398         buttons[playerY + 1][playerX].setEnabled(false);
399         buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
400             @Override
401             public void actionPerformed(ActionEvent e) {
402                 moveButtonsDown();
403                 System.out.println("Down");
404             }
405         });
406     }
407 }
408 lose();
409 win();
410 removeActionListenersFromNonAdjacentButtons();
411 }
412
413 /**
414  * Moves the player position down
415  */
416 private void moveButtonsDown() {
417
418     System.out.println("New Player Position: X = " + playerX + ", Y = " + (playerY + 1))
;
419
420     removePreviousArrows();
421     playerY++;
422
423     if (buttons[playerY][playerX].isMine() && !visitedMineLocations.contains(new
Point(playerX, playerY)) {
424         removeLife();
425         if (playerY < gridHeight - 1) {
426             buttons[playerY][playerX].setHidden(false);
427             colorStuff();
428         }
429         if (playerY == gridHeight - 1) {
430             buttons[playerY][playerX].setHidden(false);
431             buttons[playerY][playerX].setBackground(Color.BLACK);
432         }
433         playerY--;
434
435         if (playerY < gridHeight - 1) {
436             buttons[playerY + 1][playerX].setText("v");
437
438             buttons[playerY + 1][playerX].setEnabled(false);
439             buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
440                 @Override
441                 public void actionPerformed(ActionEvent e) {

```



```

442         moveButtonsDown();
443         System.out.println("Down");
444     }
445 }
446 });
447 }
448
449 } else if (playerY < 0) {
450
451     if (buttons[playerY][playerX].isMine() && visitedMineLocations.contains(new
Point(playerX, playerY))) {
452
453         playerY--;
454
455         if ((playerY < gridHeight - 1)) {
456             buttons[playerY + 1][playerX].setText("v");
457             buttons[playerY + 1][playerX].setEnabled(false);
458             buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
459                 @Override
460                 public void actionPerformed(ActionEvent e) {
461                     moveButtonsDown();
462                     System.out.println("Down");
463                 }
464             });
465         }
466     }
467 } else if (playerY < 0) {
468
469     if (buttons[playerY + 1][playerX].isMine()
&& visitedMineLocations.contains(new Point(playerX, playerY + 1))) {
470
471         if ((playerY < gridHeight - 1)) {
472             buttons[playerY + 1][playerX].setText("v");
473             buttons[playerY + 1][playerX].setEnabled(false);
474             buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
475                 @Override
476                 public void actionPerformed(ActionEvent e) {
477                     moveButtonsDown();
478                     System.out.println("Down");
479                 }
480             });
481         }
482     }
483 }
484 } else if (!buttons[playerY][playerX].isMine()) {
485
486     if (playerY < gridHeight - 1) {
487         if (!visitedMineLocations.contains(new Point(playerX, playerY + 1))) {
488             buttons[playerY + 1][playerX].setText("v");
489             buttons[playerY + 1][playerX].setEnabled(true);
490             buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
491                 @Override
492                 public void actionPerformed(ActionEvent e) {
493                     moveButtonsDown();
494                     System.out.println("Down");
495                 }
496             });

```

```

497     } else {
498         buttons[playerY + 1][playerX].setText("v");
499         buttons[playerY + 1][playerX].setEnabled(false);
500         buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
501             @Override
502             public void actionPerformed(ActionEvent e) {
503                 moveButtonsDown();
504                 System.out.println("Down");
505             }
506         });
507     }
508 }
509
510 }
511
512 buttons[playerY][playerX].setText("X");
513 buttons[playerY][playerX].setHidden(false);
514 colorStuff();
515 buttons[playerY][playerX].setEnabled(false);
516
517 if ((playerX > 0)) {
518     if (!visitedMineLocations.contains(new Point(playerX - 1, playerY))) {
519         buttons[playerY][playerX - 1].setText("<");
520         buttons[playerY][playerX - 1].setEnabled(true);
521         buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
522             @Override
523             public void actionPerformed(ActionEvent e) {
524                 moveButtonsLeft();
525             }
526         });
527     } else {
528         buttons[playerY][playerX - 1].setText("<");
529         buttons[playerY][playerX - 1].setEnabled(false);
530         buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
531             @Override
532             public void actionPerformed(ActionEvent e) {
533                 moveButtonsLeft();
534             }
535         });
536     }
537 }
538
539 if (playerX < gridWidth - 1) {
540     if (!visitedMineLocations.contains(new Point(playerX + 1, playerY))) {
541         buttons[playerY][playerX + 1].setText(">");
542         buttons[playerY][playerX + 1].setEnabled(true);
543         buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
544             @Override
545             public void actionPerformed(ActionEvent e) {
546                 moveButtonsRight();
547             }
548         });
549     } else {
550
551         buttons[playerY][playerX + 1].setText(">");
552         buttons[playerY][playerX + 1].setEnabled(false);

```

```

553         buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
554             @Override
555             public void actionPerformed(ActionEvent e) {
556                 moveButtonsRight();
557             }
558         });
559     }
560 }
561
562 if (playerY > 0) {
563     if (!visitedMineLocations.contains(new Point(playerX, playerY - 1))) {
564         buttons[playerY - 1][playerX].setText("^");
565         buttons[playerY - 1][playerX].setEnabled(true);
566         buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
567             @Override
568             public void actionPerformed(ActionEvent e) {
569                 moveButtonsUp();
570             }
571         });
572     } else {
573         buttons[playerY - 1][playerX].setText("^");
574         buttons[playerY - 1][playerX].setEnabled(false);
575         buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
576             @Override
577             public void actionPerformed(ActionEvent e) {
578                 moveButtonsUp();
579             }
580         });
581     }
582 }
583 }
584 }
585 lose();
586 win();
587 removeActionListenersFromNonAdjacentButtons();
588 }
589
590 /**
591  * Moves the player position to the left
592  */
593 private void moveButtonsLeft() {
594     removePreviousArrows();
595     playerX--;
596     if (buttons[playerY][playerX].isMine() && !visitedMineLocations.contains(new
Point(playerX, playerY))) {
597         removeLife();
598         if (playerX > 0) {
599             buttons[playerY][playerX].setHidden(false);
600             colorStuff();
601         }
602         if (playerX == 0) {
603             buttons[playerY][playerX].setHidden(false);
604             buttons[playerY][playerX].setBackground(Color.BLACK);
605         }
606         playerX++;
607     }

```

```

608     if (playerX > 0) {
609         buttons[playerY][playerX - 1].setText("<");
610         buttons[playerY][playerX - 1].setEnabled(false);
611         buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
612             @Override
613             public void actionPerformed(ActionEvent e) {
614                 moveButtonsLeft();
615             }
616         });
617     }
618
619     } else if (playerX > gridWidth - 1) {
620         if (buttons[playerY][playerX].isMine() && visitedMineLocations.contains(new
Point(playerX, playerY))) {
621             playerX++;
622
623             if (playerX > 0) {
624                 buttons[playerY][playerX - 1].setText("<");
625                 buttons[playerY][playerX - 1].setEnabled(false);
626                 buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
627                     @Override
628                     public void actionPerformed(ActionEvent e) {
629                         moveButtonsLeft();
630                     }
631                 });
632             }
633         }
634     } else if (playerX > gridWidth - 1) {
635         if (buttons[playerY][playerX - 1].isMine()
&& visitedMineLocations.contains(new Point(playerX - 1, playerY))) {
636
637             if (playerX > 0) {
638                 buttons[playerY][playerX - 1].setText("<");
639                 buttons[playerY][playerX - 1].setEnabled(false);
640                 buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
641                     @Override
642                     public void actionPerformed(ActionEvent e) {
643                         moveButtonsLeft();
644                     }
645                 });
646             }
647         }
648     }
649     } else if (!buttons[playerY][playerX].isMine()) {
650         if (playerX > 0) {
651             if (!visitedMineLocations.contains(new Point(playerX - 1, playerY))) {
652                 buttons[playerY][playerX - 1].setText("<");
653                 buttons[playerY][playerX - 1].setEnabled(true);
654                 buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
655                     @Override
656                     public void actionPerformed(ActionEvent e) {
657                         moveButtonsLeft();
658                     }
659                 });
660             } else {
661                 buttons[playerY][playerX - 1].setText("<");
662                 buttons[playerY][playerX - 1].setEnabled(false);

```

```

663         buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
664             @Override
665             public void actionPerformed(ActionEvent e) {
666                 moveButtonsLeft();
667             }
668         });
669     }
670 }
671 }
672
673 buttons[playerY][playerX].setText("X");
674 buttons[playerY][playerX].setHidden(false);
675 colorStuff();
676 buttons[playerY][playerX].setEnabled(false);
677
678 if (playerX < gridWidth - 1) {
679     if (!visitedMineLocations.contains(new Point(playerX + 1, playerY))) {
680         buttons[playerY][playerX + 1].setText(">");
681         buttons[playerY][playerX + 1].setEnabled(true);
682         buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
683             @Override
684             public void actionPerformed(ActionEvent e) {
685                 moveButtonsRight();
686             }
687         });
688     } else {
689         buttons[playerY][playerX + 1].setText(">");
690         buttons[playerY][playerX + 1].setEnabled(false);
691         buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
692             @Override
693             public void actionPerformed(ActionEvent e) {
694                 moveButtonsRight();
695             }
696         });
697     }
698 }
699
700 if (playerY < gridHeight - 1) {
701     if (!visitedMineLocations.contains(new Point(playerX, playerY + 1))) {
702         buttons[playerY + 1][playerX].setText("v");
703         buttons[playerY + 1][playerX].setEnabled(true);
704         buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
705             @Override
706             public void actionPerformed(ActionEvent e) {
707                 moveButtonsDown();
708             }
709         });
710     } else {
711         buttons[playerY + 1][playerX].setText("v");
712         buttons[playerY + 1][playerX].setEnabled(false);
713         buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
714             @Override
715             public void actionPerformed(ActionEvent e) {
716                 moveButtonsDown();
717             }
718         });

```

```

719     }
720 }
721
722 if (playerY > 0) {
723     if (!visitedMineLocations.contains(new Point(playerX, playerY - 1))) {
724         buttons[playerY - 1][playerX].setText("^");
725         buttons[playerY - 1][playerX].setEnabled(true);
726         buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
727             @Override
728             public void actionPerformed(ActionEvent e) {
729                 moveButtonsUp();
730             }
731         });
732     } else {
733         buttons[playerY - 1][playerX].setText("^");
734         buttons[playerY - 1][playerX].setEnabled(false);
735         buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
736             @Override
737             public void actionPerformed(ActionEvent e) {
738                 moveButtonsUp();
739             }
740         });
741     }
742 }
743 lose();
744 win();
745 removeActionListenersFromNonAdjacentButtons();
746 }
747
748 /**
749  * Moves the player position upwards
750  */
751 private void moveButtonsUp() {
752     removePreviousArrows();
753     playerY--;
754     if (buttons[playerY][playerX].isMine() && !visitedMineLocations.contains(new
Point(playerX, playerY))) {
755         removeLife();
756         if (playerY > 0) {
757             buttons[playerY][playerX].setHidden(false);
758             colorStuff();
759         }
760         if (playerY == 0) {
761             buttons[playerY][playerX].setHidden(false);
762             buttons[playerY][playerX].setBackground(Color.BLACK);
763         }
764         playerY++;
765         if (playerY > 0) {
766
767             buttons[playerY - 1][playerX].setText("^");
768
769             buttons[playerY - 1][playerX].setEnabled(false);
770             buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
771                 @Override
772                 public void actionPerformed(ActionEvent e) {
773                     moveButtonsUp();

```

```

774         }
775     });
776 }
777 } else if (playerY > gridHeight - 1) {
778     if (buttons[playerY][playerX].isMine() && visitedMineLocations.contains(new
Point(playerX, playerY))) {
780         playerY++;
781
782         if (playerY > 0) {
783             buttons[playerY - 1][playerX].setText("^");
784             buttons[playerY - 1][playerX].setEnabled(false);
785             buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
786                 @Override
787                 public void actionPerformed(ActionEvent e) {
788                     moveButtonsUp();
789                 }
790             });
791         }
792     }
793
794 } else if (playerY > gridHeight - 1) {
795     if (buttons[playerY - 1][playerX].isMine()
&& visitedMineLocations.contains(new Point(playerX, playerY - 1))) {
796
797
798         if (playerY > 0) {
799             buttons[playerY - 1][playerX].setText("^");
800             buttons[playerY - 1][playerX].setHidden(false);
801             buttons[playerY - 1][playerX].setEnabled(false);
802             buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
803                 @Override
804                 public void actionPerformed(ActionEvent e) {
805                     moveButtonsUp();
806                 }
807             });
808         }
809     }
810 } else if (!buttons[playerY][playerX].isMine()) {
811     if (playerY > 0) {
812         if (!visitedMineLocations.contains(new Point(playerX, playerY - 1))) {
813             buttons[playerY - 1][playerX].setText("^");
814             buttons[playerY - 1][playerX].setEnabled(true);
815             buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
816                 @Override
817                 public void actionPerformed(ActionEvent e) {
818                     moveButtonsUp();
819                 }
820             });
821         } else {
822             buttons[playerY - 1][playerX].setText("^");
823             buttons[playerY - 1][playerX].setEnabled(false);
824             buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
825                 @Override
826                 public void actionPerformed(ActionEvent e) {
827                     moveButtonsUp();
828                 }

```

```

829         });
830     }
831 }
832 }
833
834 buttons[playerY][playerX].setText("X");
835 buttons[playerY][playerX].setHidden(false);
836 colorStuff();
837 buttons[playerY][playerX].setEnabled(false);
838
839 if (playerX > 0) {
840     if (!visitedMineLocations.contains(new Point(playerX - 1, playerY))) {
841         buttons[playerY][playerX - 1].setText("<");
842         buttons[playerY][playerX - 1].setEnabled(true);
843         buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
844             @Override
845             public void actionPerformed(ActionEvent e) {
846                 moveButtonsLeft();
847             }
848         });
849     } else {
850         buttons[playerY][playerX - 1].setText("<");
851         buttons[playerY][playerX - 1].setEnabled(false);
852         buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
853             @Override
854             public void actionPerformed(ActionEvent e) {
855                 moveButtonsLeft();
856             }
857         });
858     }
859 }
860
861 if (playerX < gridWidth - 1) {
862     if (!visitedMineLocations.contains(new Point(playerX + 1, playerY))) {
863         buttons[playerY][playerX + 1].setText(">");
864         buttons[playerY][playerX + 1].setEnabled(true);
865         buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
866             @Override
867             public void actionPerformed(ActionEvent e) {
868                 moveButtonsRight();
869             }
870         });
871     } else {
872         buttons[playerY][playerX + 1].setText(">");
873         buttons[playerY][playerX + 1].setEnabled(false);
874         buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
875             @Override
876             public void actionPerformed(ActionEvent e) {
877                 moveButtonsRight();
878             }
879         });
880     }
881 }
882
883 if (playerY < gridHeight - 1) {
884     if (!visitedMineLocations.contains(new Point(playerX, playerY + 1))) {

```



```

885         buttons[playerY + 1][playerX].setText("v");
886         buttons[playerY + 1][playerX].setEnabled(true);
887         buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
888             @Override
889             public void actionPerformed(ActionEvent e) {
890                 moveButtonsDown();
891             }
892         });
893     } else {
894         buttons[playerY + 1][playerX].setText("v");
895         buttons[playerY + 1][playerX].setEnabled(false);
896         buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
897             @Override
898             public void actionPerformed(ActionEvent e) {
899                 moveButtonsDown();
900             }
901         });
902     }
903 }
904 lose();
905 win();
906 removeActionListenersFromNonAdjacentButtons();
907 }
908
909 /**
910  * Removes the action listeners from the buttons above, below, to the right and left of
the player position
911  */
912 private void removeActionListenersFromNonAdjacentButtons() {
913     for (int i = 0; i < buttons.length; i++) {
914         for (int j = 0; j < buttons[0].length; j++) {
915             if (!isAdjacentToPlayer(i, j)) {
916                 removeListeners(buttons[i][j]);
917             }
918         }
919     }
920 }
921
922 /**
923  * Checks if the button is adjacent to the player
924  *
925  * @param i
926  * @param j
927  * @return
928  */
929 private boolean isAdjacentToPlayer(int i, int j) {
930     return (Math.abs(i - playerY) == 1 && j == playerX) || (i == playerY && Math.abs(j -
playerX) == 1);
931 }
932
933 /**
934  * Removes a life
935  */
936 private void removeLife() {
937     visitedMineLocations.add(new Point(playerX, playerY));
938     System.out.println(visitedMineLocations);
939     lives--;

```

```

940     livesLabel.setText("Lives = " + lives);
941     removeActionListenersFromNonAdjacentButtons();
942 }
943
944 /**
945  * Function that checks if you lose
946  */
947 public void lose() {
948     if (lives == 0) {
949         JOptionPane.showMessageDialog(this, "Aw Shucks you suck");
950         if (playerY > 0) {
951             buttons[playerY - 1][playerX].setEnabled(false);
952         }
953         if (playerY < buttons.length) {
954             buttons[playerY + 1][playerX].setEnabled(false);
955         }
956         if (playerX > 0) {
957             buttons[playerY][playerX - 1].setEnabled(false);
958         }
959         if (playerX < buttons.length) {
960             buttons[playerY][playerX + 1].setEnabled(false);
961         }
962         unhideAllButtons();
963     }
964
965     if (hints == 0) {
966         JOptionPane.showMessageDialog(this, "You used all them???");
967         if (playerY > 0) {
968             buttons[playerY - 1][playerX].setEnabled(false);
969         }
970         if (playerY < buttons.length) {
971             buttons[playerY + 1][playerX].setEnabled(false);
972         }
973         if (playerX > 0) {
974             buttons[playerY][playerX - 1].setEnabled(false);
975         }
976         if (playerX < buttons.length) {
977             buttons[playerY][playerX + 1].setEnabled(false);
978         }
979         unhideAllButtons();
980     }
981 }
982
983 /**
984  * Function that checks if you win
985  */
986 private void win() {
987     if (playerY == 9 && playerX == 9) {
988         JOptionPane.showMessageDialog(this, "Congratuations! You wone!");
989         buttons[playerY - 1][playerX].setEnabled(false);
990         buttons[playerY][playerX - 1].setEnabled(false);
991         unhideAllButtons();
992     }
993 }
994
995 /**

```

```

996     * Unhides all buttons on the grid
997     */
998     public void unhideAllButtons() {
999         for (int i = 0; i < gridWidth; i++) {
1000             for (int j = 0; j < gridHeight; j++) {
1001                 TileButton peg = buttons[i][j];
1002                 peg.setHidden(false);
1003                 colorStuff();
1004             }
1005         }
1006     }
1007
1008     /**
1009     * Hides all buttons on the grid
1010     */
1011     public void hideAllButtons() {
1012         for (int i = 0; i < gridWidth; i++) {
1013             for (int j = 0; j < gridHeight; j++) {
1014                 TileButton peg = buttons[i][j];
1015                 peg.setHidden(true);
1016             }
1017         }
1018     }
1019
1020     /**
1021     * Function that removes the previous arrows so that it doesnt print them twice
1022     */
1023     private void removePreviousArrows() {
1024         buttons[playerY][playerX].setText("");
1025         buttons[playerY][playerX].setHidden(false);
1026         buttons[playerY][playerX].setEnabled(false);
1027
1028         if (playerX > 0) {
1029             buttons[playerY][playerX - 1].setText("");
1030             buttons[playerY][playerX - 1].setEnabled(false);
1031         }
1032
1033         if (playerX < gridWidth - 1) {
1034             buttons[playerY][playerX + 1].setText("");
1035             buttons[playerY][playerX + 1].setEnabled(false);
1036         }
1037
1038         if (playerY > 0) {
1039             buttons[playerY - 1][playerX].setText("");
1040             buttons[playerY - 1][playerX].setEnabled(false);
1041         }
1042
1043         if (playerY < gridHeight - 1) {
1044             buttons[playerY + 1][playerX].setText("");
1045             buttons[playerY + 1][playerX].setEnabled(false);
1046         }
1047     }
1048
1049     /**
1050     * Function that creates a new game
1051     */

```

```

1052 public void newGame() {
1053     hideAllButtons();
1054     lives = 5;
1055     livesLabel.setText("Lives = " + lives);
1056     hints = 8;
1057     hintsLabel.setText("Hints = " + hints + " |");
1058     playerX = 0;
1059     playerY = 0;
1060
1061     for (int i = 0; i < gridWidth; i++) {
1062         for (int j = 0; j < gridHeight; j++) {
1063             TileButton peg = buttons[i][j];
1064             ActionListener[] listeners = peg.getActionListeners();
1065
1066             for (ActionListener listener : listeners) {
1067                 peg.removeActionListener(listener);
1068             }
1069
1070             peg.setEnabled(false);
1071             peg.setPath(false);
1072             peg.setMine(false);
1073             peg.setNumMineNeighbors(0);
1074             peg.setText("");
1075             peg.setHidden(true);
1076             peg.setBackground(Color.LIGHT_GRAY);
1077
1078             if (i == playerY && j == playerX) {
1079                 peg.setHidden(false);
1080                 colorStuff();
1081                 peg.setText("X");
1082             }
1083             if (i == playerY && j == playerX) {
1084                 peg.setHidden(false);
1085                 peg.setText("X");
1086             }
1087
1088             if (playerY < gridHeight - 1) {
1089                 if (i == playerY + 1 && j == playerX) {
1090                     peg.setEnabled(true);
1091                     peg.setText("v");
1092                     peg.addActionListener(new ActionListener() {
1093                         @Override
1094                         public void actionPerformed(ActionEvent e) {
1095                             moveButtonsDown();
1096                         }
1097                     });
1098                 }
1099             }
1100
1101             if (i == playerY - 1 && j == playerX) {
1102                 peg.setEnabled(true);
1103                 peg.setText("^");
1104                 peg.addActionListener(new ActionListener() {
1105                     @Override
1106                     public void actionPerformed(ActionEvent e) {
1107                         moveButtonsUp();

```

```

1108     }
1109     });
1110 }
1111
1112     if (i == playerY && j == playerX + 1) {
1113         peg.setEnabled(true);
1114         peg.setText(">");
1115         peg.addActionListener(new ActionListener() {
1116             @Override
1117             public void actionPerformed(ActionEvent e) {
1118                 moveButtonsRight();
1119             }
1120         });
1121     }
1122
1123     if (i == playerY && j == playerX - 1) {
1124         peg.setEnabled(true);
1125         peg.setText("<");
1126         peg.addActionListener(new ActionListener() {
1127             @Override
1128             public void actionPerformed(ActionEvent e) {
1129                 moveButtonsLeft();
1130             }
1131         });
1132     }
1133 }
1134 }
1135
1136 ArrayList<Point> randomPath = RandomPath.getPath(gridWidth);
1137 for (Point point : randomPath) {
1138     int x = point.x;
1139     int y = point.y;
1140     TileButton tileButton = buttons[x][y];
1141     buttons[0][1].setPath(true);
1142     buttons[1][0].setPath(true);
1143     tileButton.setPath(true);
1144 }
1145
1146 // Get the current mine percentage from the text field
1147 int numTiles = gridWidth * gridHeight;
1148 int pathLength = randomPath.size();
1149 int minePercent = Integer.parseInt(minePercentageTextField.getText());
1150 // Cap mine percentage at 50 if it exceeds 50
1151 if (minePercent > 50) {
1152     minePercent = 50;
1153     minePercentageTextField.setText("50"); // Optionally update the text field to
reflect the change
1154 }
1155 if (minePercent < 10) {
1156     minePercent = 10;
1157     minePercentageTextField.setText("10");
1158 }
1159 int maxMines = (numTiles - pathLength) * 50 / 100;
1160 int numMinesToPlace = Math.min((numTiles - pathLength) * minePercent / 100,
maxMines);
1161
1162 // Clear existing mine locations

```



```

1217         if (playerX + 1 < buttons.length && buttons[playerY][playerX + 1]
.isHidden()) {
1218             buttons[playerY][playerX + 1].setHidden(false);
1219             buttonUnhidden = true;
1220             hints--;
1221             hintsLabel.setText("Hints = " + hints + " |");
1222         }
1223         break;
1224     case 2:
1225         if (playerY + 1 < buttons.length && buttons[playerY + 1][playerX]
.isHidden()) {
1226             buttons[playerY + 1][playerX].setHidden(false);
1227             buttonUnhidden = true;
1228             hints--;
1229             hintsLabel.setText("Hints = " + hints + " |");
1230         }
1231         break;
1232     case 1:
1233         if (playerX - 1 >= 0 && buttons[playerY][playerX - 1].isHidden()) {
1234             buttons[playerY][playerX - 1].setHidden(false);
1235             buttonUnhidden = true;
1236             hints--;
1237             hintsLabel.setText("Hints = " + hints + " |");
1238         }
1239         break;
1240     }
1241     } while (!buttonUnhidden);
1242     lose();
1243
1244     colorStuff();
1245 }
1246
1247 }
1248
1249 /**
1250  * Colors in all the buttons
1251  */
1252 private void colorStuff() {
1253     for (int x = 0; x < buttons.length; x++) {
1254         for (int y = 0; y < buttons.length; y++) {
1255             TileButton tileButton = buttons[x][y];
1256
1257             if (tileButton.isMine() && tileButton.isHidden()) {
1258                 tileButton.setBackground(Color.LIGHT_GRAY);
1259             }
1260             if (tileButton.isMine() && !tileButton.isHidden()) {
1261                 tileButton.setBackground(Color.BLACK);
1262             }
1263
1264             if (!tileButton.isMine() && !tileButton.isHidden()) {
1265                 int neighboringMines = 0;
1266
1267                 if (x > 0 && buttons[x - 1][y].isMine()) {
1268                     neighboringMines++;
1269                 }
1270                 if (x < buttons.length - 1 && buttons[x + 1][y].isMine()) {
1271                     neighboringMines++;

```

```
1272     }
1273     if (y > 0 && buttons[x][y - 1].isMine()) {
1274         neighboringMines++;
1275     }
1276     if (y < buttons.length - 1 && buttons[x][y + 1].isMine()) {
1277         neighboringMines++;
1278     }
1279
1280     tileButton.setNumMineNeighbors(neighboringMines);
1281
1282     if (neighboringMines == 0) {
1283         tileButton.setBackground(mine0);
1284     } else if (neighboringMines == 1) {
1285         tileButton.setBackground(mine1);
1286     } else if (neighboringMines == 2) {
1287         tileButton.setBackground(mine2);
1288     } else if (neighboringMines == 3) {
1289         tileButton.setBackground(mine3);
1290     } else if (neighboringMines == 4) {
1291         tileButton.setBackground(Color.PINK);
1292     }
1293 }
1294 }
1295 }
1296 }
1297 }
```