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

```java
import java.util.ArrayList;
import java.util.Random;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.GridLayout;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

/**
 * Program that runs and plays the game "Mine walker"
 *
 * @author Everett Wilcox
 */
public class MineWalkerPanel extends JPanel {
    private TileButton[][] buttons;
    private int lives = 5;
    private JLabel livesLabel = new JLabel("lives = " + lives);
    private int hints = 8;
    private JLabel hintsLabel = new JLabel("Hints = " + hints + " |");
    private int gridHeight;
    private int gridWidth;
    private Color mine0 = Color.GREEN;
    private Color mine1 = Color.YELLOW;
    private Color mine2 = Color.ORANGE;
    private Color mine3 = Color.RED;
    private int playerX = 0;
    private int playerY = 0;
    private ArrayList<Point> mineLocations = new ArrayList<>();
    private ArrayList<Point> visitedMineLocations = new ArrayList<>();
    JTextField minePercentageTextField;

    /**
     * This constructor is called when the panel is created
     *
     * @param width
     * @param height
     */
    public MineWalkerPanel(int width, int height) {

        setLayout(new BorderLayout());

        JPanel controlsPanel = new JPanel();
        JButton newGameButton = new JButton("New Game");
        newGameButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
```

```
                newGame();
            }
        });
        JButton hintButton = new JButton("Hint");
        hintButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                Hint();
            }
        });
        JLabel minePercentLabel = new JLabel("Percentage of Mines: ");
        minePercentageTextField = new JTextField("30", 2);
        controlsPanel.add(livesLabel);

        controlsPanel.add(newGameButton);
        controlsPanel.add(hintButton);
        controlsPanel.add(hintsLabel);
        controlsPanel.add(minePercentLabel);
        controlsPanel.add(minePercentageTextField);
        this.add(controlsPanel, BorderLayout.NORTH);
        gridHeight = height;
        gridWidth = width;

        JPanel gridPanel = new JPanel();
        this.buttons = new TileButton[width][height];

        for (int i = 0; i < width; i++) {
            for (int j = 0; j < height; j++) {

                TileButton peg = new TileButton(i, j);

                this.buttons[i][j] = peg;

                peg.setEnabled(false);
                peg.setPath(false);
                peg.setMine(false);
                peg.setNumMineNeighbors(0);
                peg.setText("");
                peg.setHidden(true);
                peg.setBackground(Color.LIGHT_GRAY);

                if (i == playerY && j == playerX) {
                    peg.setHidden(false);
                    peg.setBackground(mine0);
                    peg.setText("X");
                }

                if (playerY < gridHeight - 1) {
                    if (i == playerY + 1 && j == playerX) {
                        peg.setEnabled(true);
                        peg.setText("v");
                        peg.addActionListener(new ActionListener() {
                            @Override
                            public void actionPerformed(ActionEvent e) {
                                moveButtonsDown();
                            }
```

```
                    });
                }
            }

            if (i == playerY - 1 && j == playerX) {
                peg.setEnabled(true);
                peg.setText("^");
                peg.addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsUp();
                    }
                });
            }

            if (i == playerY && j == playerX + 1) {
                peg.setEnabled(true);
                peg.setText(">");
                peg.addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsRight();
                    }
                });

            }

            if (i == playerY && j == playerX - 1) {
                peg.setEnabled(true);
                peg.setText("<");
                peg.addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsLeft();
                    }
                });
            }
            gridPanel.add(peg);
        }
    }

    // Call the method to print the mine locations
    gridPanel.setLayout(new GridLayout(width, height));
    this.add(gridPanel, BorderLayout.CENTER);

    JPanel colorKeyPanel = new JPanel();
    colorKeyPanel.setLayout(new GridLayout(0, 1));
    JButton mine0Button = new JButton("0 Mine Neighbors");
    JButton mine1Button = new JButton("1 Mine Neighbor");
    JButton mine2Button = new JButton("2 Mine Neighbors");
    JButton mine3Button = new JButton("3 Mine Neighbors");
    JButton explodedMine = new JButton("Exploded Mine");
    JButton xIsYou = new JButton("X <- You");
    mine0Button.setBackground(Color.GREEN);
    mine0Button.setBorderPainted(false);
    mine0Button.setFocusPainted(false);
```

```
        mine1Button.setBackground(Color.YELLOW);
        mine1Button.setBorderPainted(false);
        mine1Button.setFocusPainted(false);
        mine2Button.setBackground(new Color(255, 130, 0));
        mine2Button.setBorderPainted(false);
        mine2Button.setFocusPainted(false);
        mine3Button.setBackground(Color.RED);
        mine3Button.setBorderPainted(false);
        mine3Button.setFocusPainted(false);
        explodedMine.setBackground(Color.BLACK);
        explodedMine.setForeground(Color.WHITE);
        explodedMine.setBorderPainted(false);
        explodedMine.setFocusPainted(false);
        xIsYou.setBackground(Color.WHITE);
        xIsYou.setBorderPainted(false);
        xIsYou.setFocusPainted(false);
        colorKeyPanel.add(mine0Button);
        colorKeyPanel.add(mine1Button);
        colorKeyPanel.add(mine2Button);
        colorKeyPanel.add(mine3Button);
        colorKeyPanel.add(explodedMine);
        colorKeyPanel.add(xIsYou);
        this.add(colorKeyPanel, BorderLayout.WEST);

        ArrayList<Point> randomPath = RandomPath.getPath(width);
        for (Point point : randomPath) {
            int x = point.x;
            int y = point.y;
            TileButton tileButton = buttons[x][y];
            buttons[0][1].setPath(true);
            buttons[1][0].setPath(true);
            tileButton.setPath(true);
        }

        int numTiles = width * height;
        int pathLength = randomPath.size();
        int minePercent = Integer.parseInt(minePercentageTextField.getText());

        if (minePercent > 50) {
            minePercent = 50;
            minePercentageTextField.setText("50");
        }

        int numMinesToPlace = (numTiles - pathLength) * minePercent / 100;



        // Place the mines
        Random rand = new Random();
        while (numMinesToPlace > 0) {
            int x = rand.nextInt(width);
            int y = rand.nextInt(height);
            TileButton tileButton = buttons[x][y];

            if (!tileButton.isPath() && !tileButton.isMine()) {
                tileButton.setHidden(true);
```

```
                tileButton.setMine(true);
                numMinesToPlace--;
                mineLocations.add(new Point(x, y));
            }
        }
    }

    /**
     * Removes the action listeners from the list of buttons
     *
     * @param button
     */
    private void removeListeners(JButton button) {
        ActionListener[] listeners = button.getActionListeners();
        for (ActionListener listener : listeners) {
            button.removeActionListener(listener);
        }
    }

    /**
     * Function that moves the player position to the right
     */
    private void moveButtonsRight() {
        System.out.println("New Player Position: X = " + (playerX + 1) + ", Y = " + playerY)
;

        removePreviousArrows();

        playerX++;

        if (buttons[playerY][playerX].isMine() && !visitedMineLocations.contains(new
Point(playerX, playerY))) {
            removeLife();
            if (playerX < gridWidth - 1) {
                buttons[playerY][playerX].setHidden(false);
                colorStuff();
            }
            if (playerX == gridWidth - 1) {
                buttons[playerY][playerX].setHidden(false);
                buttons[playerY][playerX].setBackground(Color.BLACK);
            }
            playerX--;

            if (playerX < gridWidth - 1) {
                buttons[playerY][playerX + 1].setText(">");

                buttons[playerY][playerX + 1].setEnabled(false);
                buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsRight();
                    }
                });
            }
        } else if (playerX < 0) {

```

```
            if (buttons[playerY][playerX].isMine() && visitedMineLocations.contains(new Point(playerX, playerY))) {
                playerX--;

                if (playerX < gridWidth - 1) {
                    buttons[playerY][playerX + 1].setText(">");
                    buttons[playerY][playerX + 1].setEnabled(false);
                    buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            moveButtonsRight();
                        }
                    });
                }
            }

        } else if (playerX < 0) {

            if (buttons[playerY][playerX + 1].isMine()
                    && visitedMineLocations.contains(new Point(playerX + 1, playerY))) {

                if (playerX < gridWidth - 1) {
                    buttons[playerY][playerX + 1].setText(">");
                    buttons[playerY][playerX + 1].setEnabled(false);
                    buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            moveButtonsRight();
                        }
                    });
                }
            }

        } else if (!buttons[playerY][playerX].isMine()) {
            if (playerX < gridWidth - 1) {
                if (!visitedMineLocations.contains(new Point(playerX + 1, playerY))) {
                    buttons[playerY][playerX + 1].setText(">");
                    buttons[playerY][playerX + 1].setEnabled(true);
                    buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            moveButtonsRight();
                        }
                    });
                } else {
                    buttons[playerY][playerX + 1].setText(">");
                    buttons[playerY][playerX + 1].setEnabled(false);
                    buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            moveButtonsRight();
                        }
                    });
                }
            }

```

```
            removeActionListenersFromNonAdjacentButtons();
        }

        buttons[playerY][playerX].setText("X");
        buttons[playerY][playerX].setHidden(false);
        colorStuff();
        buttons[playerY][playerX].setEnabled(false);

        if ((playerX > 0)) {
            if (!visitedMineLocations.contains(new Point(playerX - 1, playerY))) {
                buttons[playerY][playerX - 1].setText("<");
                buttons[playerY][playerX - 1].setEnabled(true);
                buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsLeft();
                    }
                });
            } else {
                buttons[playerY][playerX - 1].setText("<");
                buttons[playerY][playerX - 1].setEnabled(false);
                buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsLeft();
                    }
                });
            }
        }

        if ((playerY > 0)) {
            if (!visitedMineLocations.contains(new Point(playerX, playerY - 1))) {
                buttons[playerY - 1][playerX].setText("^");
                buttons[playerY - 1][playerX].setEnabled(true);
                buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsUp();
                        System.out.println("Up");
                    }
                });
            } else {
                buttons[playerY - 1][playerX].setText("^");
                buttons[playerY - 1][playerX].setEnabled(false);
                buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsUp();
                        System.out.println("Up");
                    }
                });
            }
        }

        if ((playerY < gridHeight - 1)) {
            if (!visitedMineLocations.contains(new Point(playerX, playerY + 1))) {
```

```
                buttons[playerY + 1][playerX].setText("v");
                buttons[playerY + 1][playerX].setEnabled(true);
                buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsDown();
                        System.out.println("Down");
                    }
                });
            } else {
                buttons[playerY + 1][playerX].setText("v");
                buttons[playerY + 1][playerX].setEnabled(false);
                buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsDown();
                        System.out.println("Down");
                    }
                });
            }
        }
        lose();
        win();
        removeActionListenersFromNonAdjacentButtons();
    }

    /**
     * Moves the player position down
     */
    private void moveButtonsDown() {

        System.out.println("New Player Position: X = " + playerX + ", Y = " + (playerY + 1))
;

        removePreviousArrows();
        playerY++;

        if (buttons[playerY][playerX].isMine() && !visitedMineLocations.contains(new
Point(playerX, playerY))) {
            removeLife();
            if (playerY < gridHeight - 1) {
                buttons[playerY][playerX].setHidden(false);
                colorStuff();
            }
            if (playerY == gridHeight - 1) {
                buttons[playerY][playerX].setHidden(false);
                buttons[playerY][playerX].setBackground(Color.BLACK);
            }
            playerY--;

            if (playerY < gridHeight - 1) {
                buttons[playerY + 1][playerX].setText("v");

                buttons[playerY + 1][playerX].setEnabled(false);
                buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
```

```
                        moveButtonsDown();
                        System.out.println("Down");

                    }
                });
            }

        } else if (playerY < 0) {

            if (buttons[playerY][playerX].isMine() && visitedMineLocations.contains(new Point(playerX, playerY))) {

                playerY--;

                if ((playerY < gridHeight - 1)) {
                    buttons[playerY + 1][playerX].setText("v");
                    buttons[playerY + 1][playerX].setEnabled(false);
                    buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            moveButtonsDown();
                            System.out.println("Down");
                        }
                    });
                }
            }
        } else if (playerY < 0) {

            if (buttons[playerY + 1][playerX].isMine()
                    && visitedMineLocations.contains(new Point(playerX, playerY + 1))) {

                if ((playerY < gridHeight - 1)) {
                    buttons[playerY + 1][playerX].setText("v");
                    buttons[playerY + 1][playerX].setEnabled(false);
                    buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            moveButtonsDown();
                            System.out.println("Down");
                        }
                    });
                }
            }
        } else if (!buttons[playerY][playerX].isMine()) {

            if (playerY < gridHeight - 1) {
                if (!visitedMineLocations.contains(new Point(playerX, playerY + 1))) {
                    buttons[playerY + 1][playerX].setText("v");
                    buttons[playerY + 1][playerX].setEnabled(true);
                    buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            moveButtonsDown();
                            System.out.println("Down");
                        }
                    });
```

```
                } else {
                    buttons[playerY + 1][playerX].setText("v");
                    buttons[playerY + 1][playerX].setEnabled(false);
                    buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            moveButtonsDown();
                            System.out.println("Down");
                        }
                    });
                }
            }

        }

        buttons[playerY][playerX].setText("X");
        buttons[playerY][playerX].setHidden(false);
        colorStuff();
        buttons[playerY][playerX].setEnabled(false);

        if ((playerX > 0)) {
            if (!visitedMineLocations.contains(new Point(playerX - 1, playerY))) {
                buttons[playerY][playerX - 1].setText("<");
                buttons[playerY][playerX - 1].setEnabled(true);
                buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsLeft();
                    }
                });
            } else {
                buttons[playerY][playerX - 1].setText("<");
                buttons[playerY][playerX - 1].setEnabled(false);
                buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsLeft();
                    }
                });
            }
        }

        if (playerX < gridWidth - 1) {
            if (!visitedMineLocations.contains(new Point(playerX + 1, playerY))) {
                buttons[playerY][playerX + 1].setText(">");
                buttons[playerY][playerX + 1].setEnabled(true);
                buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsRight();
                    }
                });
            } else {

                buttons[playerY][playerX + 1].setText(">");
                buttons[playerY][playerX + 1].setEnabled(false);
```

```
                buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsRight();
                    }
                });
            }
        }

        if (playerY > 0) {
            if (!visitedMineLocations.contains(new Point(playerX, playerY - 1))) {
                buttons[playerY - 1][playerX].setText("^");
                buttons[playerY - 1][playerX].setEnabled(true);
                buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsUp();

                    }
                });
            } else {
                buttons[playerY - 1][playerX].setText("^");
                buttons[playerY - 1][playerX].setEnabled(false);
                buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsUp();

                    }
                });
            }
        }
        lose();
        win();
        removeActionListenersFromNonAdjacentButtons();
    }

    /**
     * Moves the player position to the left
     */
    private void moveButtonsLeft() {
        removePreviousArrows();
        playerX--;
        if (buttons[playerY][playerX].isMine() && !visitedMineLocations.contains(new
Point(playerX, playerY))) {
            removeLife();
            if (playerX > 0) {
                buttons[playerY][playerX].setHidden(false);
                colorStuff();
            }
            if (playerX == 0) {
                buttons[playerY][playerX].setHidden(false);
                buttons[playerY][playerX].setBackground(Color.BLACK);
            }
            playerX++;

```

```
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()
636                     && visitedMineLocations.contains(new Point(playerX - 1, playerY))) {
637 
638                 if (playerX > 0) {
639                     buttons[playerY][playerX - 1].setText("<");
640                     buttons[playerY][playerX - 1].setEnabled(false);
641                     buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
642                         @Override
643                         public void actionPerformed(ActionEvent e) {
644                             moveButtonsLeft();
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);
```

```
                    buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            moveButtonsLeft();
                        }
                    });
                }
            }
        }

        buttons[playerY][playerX].setText("X");
        buttons[playerY][playerX].setHidden(false);
        colorStuff();
        buttons[playerY][playerX].setEnabled(false);

        if (playerX < gridWidth - 1) {
            if (!visitedMineLocations.contains(new Point(playerX + 1, playerY))) {
                buttons[playerY][playerX + 1].setText(">");
                buttons[playerY][playerX + 1].setEnabled(true);
                buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsRight();
                    }
                });
            } else {
                buttons[playerY][playerX + 1].setText(">");
                buttons[playerY][playerX + 1].setEnabled(false);
                buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsRight();
                    }
                });
            }
        }

        if (playerY < gridHeight - 1) {
            if (!visitedMineLocations.contains(new Point(playerX, playerY + 1))) {
                buttons[playerY + 1][playerX].setText("v");
                buttons[playerY + 1][playerX].setEnabled(true);
                buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsDown();
                    }
                });
            } else {
                buttons[playerY + 1][playerX].setText("v");
                buttons[playerY + 1][playerX].setEnabled(false);
                buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsDown();
                    }
                });
```

```
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();
```

```
                    }
                });
            }
        } else if (playerY > gridHeight - 1) {

            if (buttons[playerY][playerX].isMine() && visitedMineLocations.contains(new Point(playerX, playerY))) {
                playerY++;

                if (playerY > 0) {
                    buttons[playerY - 1][playerX].setText("^");
                    buttons[playerY - 1][playerX].setEnabled(false);
                    buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            moveButtonsUp();
                        }
                    });
                }
            }

        } else if (playerY > gridHeight - 1) {
            if (buttons[playerY - 1][playerX].isMine()
                    && visitedMineLocations.contains(new Point(playerX, playerY - 1))) {

                if (playerY > 0) {
                    buttons[playerY - 1][playerX].setText("^");
                    buttons[playerY - 1][playerX].setHidden(false);
                    buttons[playerY - 1][playerX].setEnabled(false);
                    buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            moveButtonsUp();
                        }
                    });
                }
            }
        } else if (!buttons[playerY][playerX].isMine()) {
            if (playerY > 0) {
                if (!visitedMineLocations.contains(new Point(playerX, playerY - 1))) {
                    buttons[playerY - 1][playerX].setText("^");
                    buttons[playerY - 1][playerX].setEnabled(true);
                    buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            moveButtonsUp();
                        }
                    });
                } else {
                    buttons[playerY - 1][playerX].setText("^");
                    buttons[playerY - 1][playerX].setEnabled(false);
                    buttons[playerY - 1][playerX].addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            moveButtonsUp();
                        }
```

```
                    });
                }
            }
        }

        buttons[playerY][playerX].setText("X");
        buttons[playerY][playerX].setHidden(false);
        colorStuff();
        buttons[playerY][playerX].setEnabled(false);

        if (playerX > 0) {
            if (!visitedMineLocations.contains(new Point(playerX - 1, playerY))) {
                buttons[playerY][playerX - 1].setText("<");
                buttons[playerY][playerX - 1].setEnabled(true);
                buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsLeft();
                    }
                });
            } else {
                buttons[playerY][playerX - 1].setText("<");
                buttons[playerY][playerX - 1].setEnabled(false);
                buttons[playerY][playerX - 1].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsLeft();
                    }
                });
            }
        }

        if (playerX < gridWidth - 1) {
            if (!visitedMineLocations.contains(new Point(playerX + 1, playerY))) {
                buttons[playerY][playerX + 1].setText(">");
                buttons[playerY][playerX + 1].setEnabled(true);
                buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsRight();
                    }
                });
            } else {
                buttons[playerY][playerX + 1].setText(">");
                buttons[playerY][playerX + 1].setEnabled(false);
                buttons[playerY][playerX + 1].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsRight();
                    }
                });
            }
        }

        if (playerY < gridHeight - 1) {
            if (!visitedMineLocations.contains(new Point(playerX, playerY + 1))) {
```

```
                buttons[playerY + 1][playerX].setText("v");
                buttons[playerY + 1][playerX].setEnabled(true);
                buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsDown();
                    }
                });
            } else {
                buttons[playerY + 1][playerX].setText("v");
                buttons[playerY + 1][playerX].setEnabled(false);
                buttons[playerY + 1][playerX].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        moveButtonsDown();
                    }
                });
            }
        }
        lose();
        win();
        removeActionListenersFromNonAdjacentButtons();
    }

    /**
     * Removes the action listeners from the buttons above, below, to the right and left of the player position
     */
    private void removeActionListenersFromNonAdjacentButtons() {
        for (int i = 0; i < buttons.length; i++) {
            for (int j = 0; j < buttons[0].length; j++) {
                if (!isAdjacentToPlayer(i, j)) {
                    removeListeners(buttons[i][j]);
                }
            }
        }
    }

    /**
     * Checks if the button is adjacent to the player
     *
     * @param i
     * @param j
     * @return
     */
    private boolean isAdjacentToPlayer(int i, int j) {
        return (Math.abs(i - playerY) == 1 && j == playerX) || (i == playerY && Math.abs(j - playerX) == 1);
    }

    /**
     * Removes a life
     */
    private void removeLife() {
        visitedMineLocations.add(new Point(playerX, playerY));
        System.out.println(visitedMineLocations);
        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     /**
```

```
     * Unhides all buttons on the grid
     */
    public void unhideAllButtons() {
        for (int i = 0; i < gridWidth; i++) {
            for (int j = 0; j < gridHeight; j++) {
                TileButton peg = buttons[i][j];
                peg.setHidden(false);
                colorStuff();
            }
        }
    }

    /**
     * Hides all buttons on the grid
     */
    public void hideAllButtons() {
        for (int i = 0; i < gridWidth; i++) {
            for (int j = 0; j < gridHeight; j++) {
                TileButton peg = buttons[i][j];
                peg.setHidden(true);
            }
        }
    }

    /**
     * Function that removes the previous arrows so that it doesnt print them twice
     */
    private void removePreviousArrows() {
        buttons[playerY][playerX].setText("");
        buttons[playerY][playerX].setHidden(false);
        buttons[playerY][playerX].setEnabled(false);

        if (playerX > 0) {
            buttons[playerY][playerX - 1].setText("");
            buttons[playerY][playerX - 1].setEnabled(false);
        }

        if (playerX < gridWidth - 1) {
            buttons[playerY][playerX + 1].setText("");
            buttons[playerY][playerX + 1].setEnabled(false);
        }

        if (playerY > 0) {
            buttons[playerY - 1][playerX].setText("");
            buttons[playerY - 1][playerX].setEnabled(false);
        }

        if (playerY < gridHeight - 1) {
            buttons[playerY + 1][playerX].setText("");
            buttons[playerY + 1][playerX].setEnabled(false);
        }
    }

    /**
     * Function that creates a new game
     */
```

```
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();
```

```
                        }
                    });
                }

                if (i == playerY && j == playerX + 1) {
                    peg.setEnabled(true);
                    peg.setText(">");
                    peg.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            moveButtonsRight();
                        }
                    });
                }

                if (i == playerY && j == playerX - 1) {
                    peg.setEnabled(true);
                    peg.setText("<");
                    peg.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            moveButtonsLeft();
                        }
                    });
                }
            }
        }

        ArrayList<Point> randomPath = RandomPath.getPath(gridWidth);
        for (Point point : randomPath) {
            int x = point.x;
            int y = point.y;
            TileButton tileButton = buttons[x][y];
            buttons[0][1].setPath(true);
            buttons[1][0].setPath(true);
            tileButton.setPath(true);
        }

        // Get the current mine percentage from the text field
        int numTiles = gridWidth * gridHeight;
        int pathLength = randomPath.size();
        int minePercent = Integer.parseInt(minePercentageTextField.getText());
        // Cap mine percentage at 50 if it exceeds 50
        if (minePercent > 50) {
            minePercent = 50;
            minePercentageTextField.setText("50"); // Optionally update the text field to
reflect the change
        }
        if (minePercent < 10) {
            minePercent = 10;
            minePercentageTextField.setText("10");
        }
        int maxMines = (numTiles - pathLength) * 50 / 100;
        int numMinesToPlace = Math.min((numTiles - pathLength) * minePercent / 100,
maxMines);

        // Clear existing mine locations
```

```
        mineLocations.clear();
        visitedMineLocations.clear();

        // Place the mines
        Random rand = new Random();
        while (numMinesToPlace > 0) {
            int x = rand.nextInt(gridWidth);
            int y = rand.nextInt(gridHeight);
            TileButton tileButton = buttons[x][y];

            if (!tileButton.isPath() && !tileButton.isMine()) {
                tileButton.setHidden(true);
                tileButton.setMine(true);
                numMinesToPlace--;
                mineLocations.add(new Point(x, y));
            }
        }
    }

    /**
     * Function that lets the player use a hint
     */
    private void Hint() {
        Random ran = new Random();
        int r;
        boolean buttonUnhidden = false;


        // Check if there are hidden buttons around the player
        boolean hiddenButtonsExist = false;
        if (playerY - 1 >= 0 && buttons[playerY - 1][playerX].isHidden()) {
            hiddenButtonsExist = true;
        } else if (playerX + 1 < buttons.length && buttons[playerY][playerX + 1].isHidden())
{
            hiddenButtonsExist = true;
        } else if (playerY + 1 < buttons.length && buttons[playerY + 1][playerX].isHidden())
{
            hiddenButtonsExist = true;
        } else if (playerX - 1 >= 0 && buttons[playerY][playerX - 1].isHidden()) {
            hiddenButtonsExist = true;
        }

        if (hiddenButtonsExist) {
            // Generate a random number only if there are hidden buttons
            do {
                r = ran.nextInt(4) + 1;
                switch (r) {
                    case 4:
                        if (playerY - 1 >= 0 && buttons[playerY - 1][playerX].isHidden()) {
                            buttons[playerY - 1][playerX].setHidden(false);
                            buttonUnhidden = true;
                            hints--;
                            hintsLabel.setText("Hints = " + hints + " |");
                        }
                        break;
                    case 3:
```

```
                        if (playerX + 1 < buttons.length && buttons[playerY][playerX + 1]
.isHidden()) {
                            buttons[playerY][playerX + 1].setHidden(false);
                            buttonUnhidden = true;
                            hints--;
                            hintsLabel.setText("Hints = " + hints + " |");
                        }
                        break;
                    case 2:
                        if (playerY + 1 < buttons.length && buttons[playerY + 1][playerX]
.isHidden()) {
                            buttons[playerY + 1][playerX].setHidden(false);
                            buttonUnhidden = true;
                            hints--;
                            hintsLabel.setText("Hints = " + hints + " |");
                        }
                        break;
                    case 1:
                        if (playerX - 1 >= 0 && buttons[playerY][playerX - 1].isHidden()) {
                            buttons[playerY][playerX - 1].setHidden(false);
                            buttonUnhidden = true;
                            hints--;
                            hintsLabel.setText("Hints = " + hints + " |");
                        }
                        break;
                }
            } while (!buttonUnhidden);
            lose();

            colorStuff();
        }

    }

    /**
     * Colors in all the buttons
     */
    private void colorStuff() {
        for (int x = 0; x < buttons.length; x++) {
            for (int y = 0; y < buttons.length; y++) {
                TileButton tileButton = buttons[x][y];

                if (tileButton.isMine() && tileButton.isHidden()) {
                    tileButton.setBackground(Color.LIGHT_GRAY);
                }
                if (tileButton.isMine() && !tileButton.isHidden()) {
                    tileButton.setBackground(Color.BLACK);
                }

                if (!tileButton.isMine() && !tileButton.isHidden()) {
                    int neighboringMines = 0;

                    if (x > 0 && buttons[x - 1][y].isMine()) {
                        neighboringMines++;
                    }
                    if (x < buttons.length - 1 && buttons[x + 1][y].isMine()) {
                        neighboringMines++;
```

```
                    }
                    if (y > 0 && buttons[x][y - 1].isMine()) {
                        neighboringMines++;
                    }
                    if (y < buttons.length - 1 && buttons[x][y + 1].isMine()) {
                        neighboringMines++;
                    }

                    tileButton.setNumMineNeighbors(neighboringMines);

                    if (neighboringMines == 0) {
                        tileButton.setBackground(mine0);
                    } else if (neighboringMines == 1) {
                        tileButton.setBackground(mine1);
                    } else if (neighboringMines == 2) {
                        tileButton.setBackground(mine2);
                    } else if (neighboringMines == 3) {
                        tileButton.setBackground(mine3);
                    } else if (neighboringMines == 4) {
                        tileButton.setBackground(Color.PINK);
                    }
                }
            }
        }
    }
}
```