8 Puzzle

Documentation

Zander Labuschagne
23585137

Table of contents

1. Introduction

1.1. Intended Readership

This document is intended for the end-users or gamers of the 8 Puzzle game developed by Zander Labuschagne. The user guide section is focussed on the UNIX version of the game.

1.2. Applicability

This user guide applies only to the 8 Puzzle game developed by Zander Labuschagne.

1.3. Purpose

The aim of this document is to aid the end-user in the understanding of this 8 Puzzle game and to provide assistance if needed.

1.4. Motivation and Background

This software is part of an Artificial Intelligence project presented by Dr. Tiny Du Toit at the North-West University, Potchefstroom Campus. The goal of this project is to provide students with practical experience in the subject of artificial intelligence, this will help them to understand the algorithms better and to enable them to implement new algorithms learnt at a level of comfort and without struggle. This software is the first part of two, the first part is only the game where a human player plays the game. The second part adds an additional artificial intelligent part to the game where an artificial intelligent algorithm will be implemented by the student to solve the puzzle in the least number of moves. (du Toit, 2016).

2. Literature Study

The 8-puzzle first started out as the 15-puzzle designed in the 1870's by a mathematician, Sam Loyd. Sam Loyd designed many puzzles, but the 15-puzzle had a winning prize of a $1,000 for the first person to solve the puzzle. Up to date, nobody has ever claimed the prize because the 15-puzzle is actually unsolvable. As a result this caused a simplified and solvable version of the 15 puzzle namely the 8 puzzle. (Archer, 2006).

3. User Guide

3.1. Installation

No official installation is necessary, the program or game can be started by clicking or double clicking(depending on your operating system and settings) on the 8Puzzle_UNIX.jar file if you are using a UNIX-like system. If you are a Windows user, then you must double click on the 8Puzzle_Windows.bat file which will open in a terminal window.

3.2. System Requirements

3.2.1. Operating Systems

This game functions on any UNIX-like or Windows system. However if you are using Microsoft Windows, you will not be able to run the game with graphical capabilities, thus you will be using a command line interface but all functions are the same of the UNIX version.

8Puzzle_UNIX runs successful with graphical interface on:

- Apple MacOS X El Capitan 10.11.3
- Elementary OS Freya 0.3.2
- Kali Linux 2016.1
- Oracle Linux 7.2
- Red Hat Enterprise Server 7.0
- Ubuntu 14.10

8Puzzle_Windows runs successful without graphical interface on:

- Microsoft Windows 10
- Microsoft Windows 7

3.2.2. Hardware Requirements

- Keyboard
- Mouse or Trackpad(UNIX version)
- Computer Monitor
- Speaker or Headphones Recommended

3.2.3. Additional Software Requirements

- Java 1.8
- PDF Document Viewer to view the help

3.4. Game Overview

3.4.1. 8 Puzzle Game Rules

The puzzle becomes solved when the user has moved all tiles into some meaningful order, the order is not fixed, however the default is starting from 1 at the top left and ending at 8 at the bottom middle with a blank space to the right of the 8.

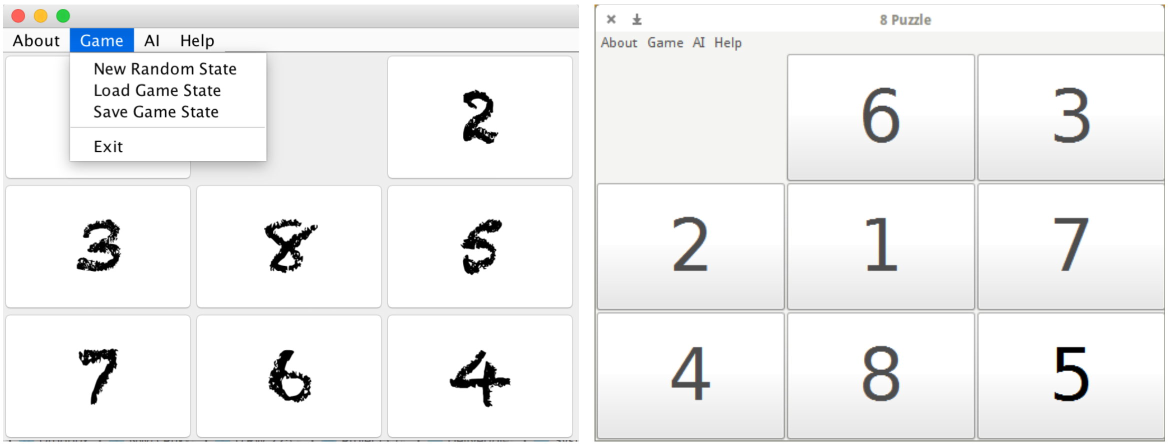
The user can manipulate the order of the tiles by clicking on them, the tile will only move if the attempted move is a valid move. A valid move can only be if the tile clicked on is adjacent to the blank space, if so the tile will move into the blank space leaving a new blank space in the tile's previous position.

3.4.2. Features

- **Game -> New Random State**

This feature can be used by clicking on the "Game" dropdown list in the menu bar, followed by clicking on the "New Random State" menu item in the "Game" dropdown list. This will generate a new pseudo random game state which one can start with to play the game like in the example of *Figure 1*.

Figure 1 Left: MacOS X
Figure 1 Right: Linux



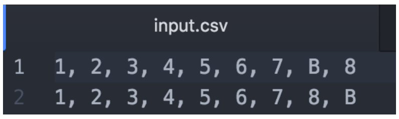
- **Game -> Load Game State**

This feature can be used by clicking on the "Game" dropdown list in the menu bar, followed by clicking on the "Load Game State" menu item in the "Game" dropdown list. This will open a dialog window where the end-user can browse for a specific comma separated value(.csv) file which contains a game state. After the user has loaded a .csv file which's contents is in the appropriate format, the game will reset and start in the state contained in the .csv file. Refer to *Figure 3.*

The .csv file should contain 9 values which is separated by commas (",") and contain one "B" letter which denotes the blank tile. The "B" should always be uppercase. Refer to *Figure 4.*

Examples: 1, 2, 3, 4, 5, 6, 7, 8, B
2,4,6,8,B,3,5,7

This file should also contain a second line of comma separated values which indicates a winning state.

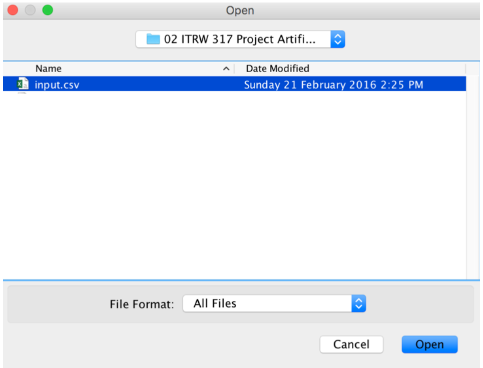


input.csv

```
1, 2, 3, 4, 5, 6, 7, B, 8
1, 2, 3, 4, 5, 6, 7, 8, B
```

Figure 3: Example of a .csv file opened in a text editor.

Figure 2: Load game state.



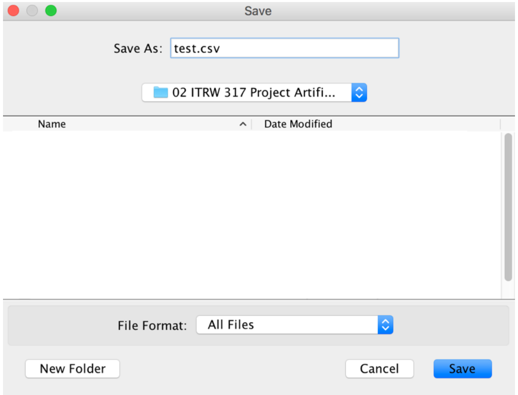
input.csv Sunday 21 February 2016 2:25 PM

File Format: All Files

- **Game -> Save Game State**

This feature can be used by clicking on the "Game" dropdown list in the menu bar, followed by clicking on the "Save Game State" menu item in the "Game" dropdown list. This will open a dialog window where the end-user can browse to a specific location and save the current state of the game as a .csv file. The user must enter the .csv file extension at the end of the file name. Refer to *Figure 5.*

Figure 4: Save game state.



Save As: test.csv

File Format: All Files

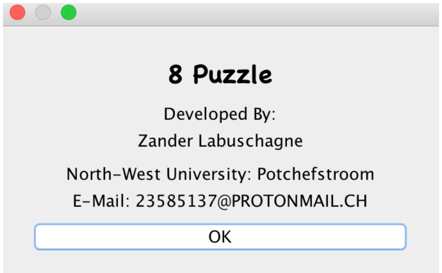
- **Exit**

This is a mandatory feature included in all software applications. This exits the game without saving.

- **About**

This is just a menu item on the menu bar to view the details of the developer.

Figure 6: About.



8 Puzzle

Developed By:
Zander Labuschagne

North-West University: Potchefstroom
E-Mail: 23585137@PROTONMAIL.CH

- **Help**

Another mandatory feature to improve the user experience by answering the user's questions about the game.

- **AI -> Animated Solve**

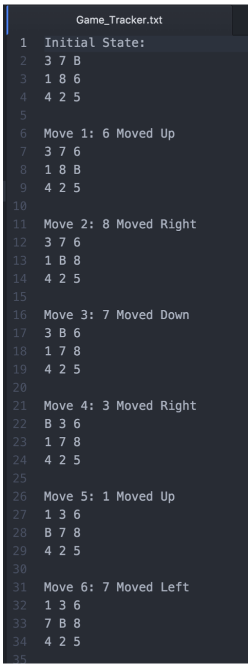
This feature is not active yet and will be included in part 2. The feature can be found under the AI menu item inside the menu bar. When this is activated the current game state will be solved in the least possible moves with the move animation effect. This is done by using artificial intelligence.

- **AI -> Quick Solve**

This feature is not active yet and will be included in part 2. The feature can be found under the AI menu item inside the menu bar. When this is activated the current game state will be solved in the least possible moves without the move animation effect to spare time.

- **Game Tracker**

8 Puzzle keeps track of all the moves made in the current game, the moves made are saved in a text(.txt) file inside the game root folder. This may be used to analyse the game play. Refer to *Figure 7. Warning: Make a backup of the game tracker file if you want keep the history before loading/starting new game, otherwise the game tracker file will get overwritten and the information will be lost forever. Do not try to view the file while the game is running, rather make a backup when the game has exited and BEFORE the game is launched again.*



```
Initial State:
3 7 B
1 8 6
4 2 5

Move 1: 6 Moved Up
3 7 6
1 8 B
4 2 5

Move 2: 8 Moved Right
3 7 6
1 B 8
4 2 5

Move 3: 7 Moved Down
3 B 6
1 7 8
4 2 5

Move 4: 3 Moved Right
B 3 6
1 7 8
4 2 5

Move 5: 1 Moved Up
1 3 6
B 7 8
4 2 5

Move 6: 7 Moved Left
1 3 6
7 B 8
4 2 5
```

Figure 7: Game Tracker.

3.3. Technical Support

Support is only provided for UNIX-like systems. The developer can be contacted at: 23585137@PROTONMAIL.CH

The North-West University: Potchefstroom Campus holds ownership rights over this software.

3.4. End User Licence Agreement

This software is open source and free to modify but may not be sold commercially. North-West University, Potchefstroom Campus holds ownership rights to this software. This software was developed by Zander Labuschagne, student at North-West University, Potchefstroom Campus. The developer may be contacted for support on UNIX-like systems, do not bother him with Microsoft problems.

This software was made possible by the following five rules:

1. Never give up on anything.
2. Do what you love no matter who, no matter what.

3. Don’t stick with what you know, move on to the unknown.
4. It’s not the result/outcome that matter, it’s the process that makes the difference.
5. Always do what you are afraid to do. – Ralph Waldo Emerson.

References

Archer, A. 2006. The 15 Puzzle: How it Drove the World Crazy by Jerry Slocum and Dic Sonneveld. Florham Park, NJ: AT&T Shannon Research Laboratory.

du Doit, J.V. 2016. Artificial Intelligence. Potchefstroom: NWU, Potchefstroom Campus. (Study Guide ITRW 317 PEC).

Appendix A: Source Code

```
/**

 * @author Zander Labuschagne 23585137
 * Class to manage file I/O

 */

//I/O
import javax.swing.*;

import java.awt.*;
import java.io.File;

import java.io.FileNotFoundException;
import java.io.IOException;

import java.io.PrintWriter;

public class IO //implements Runnable
{

    //Instance Variables
    /**

     * Name of file to be used, including directory/path
     */

    private String fileName;
    /**

     * object of File
     */

    private File file;
    /**

     * object of PrintWriter to write to files
     */

    private PrintWriter writer;

    /**
     * Default Constructor

     */
    public IO() throws FileNotFoundException

    {
        this("");

    }

    /**
     * Overloaded Constructor #1

     * @param fileName name of file to be used
     */

    public IO(String fileName) throws FileNotFoundException
    {

        setFileName(fileName);
        setFile(new File(getFileName()));

    }

    //Mutators
    /**

     * set the file name
     * @param fileName name of file

     */
    private void setFileName(String fileName)

    {
        this.fileName = fileName;

    }

    /**
     * set the PrintWriter object

     * @param writer object of PrntWriter
     */

    private void setWriter(PrintWriter writer)
    {

        this.writer = writer;
    }

    /**

     * set the File object
     * @param file file to be used

     */
    public void setFile(File file)

    {
        this.file = file;

    }

    //Accessors
    /**

     * get the file name
     * @return the file name

     */
    private String getFileName()

    {
        return fileName;

    }

    /**
     * get the PrintWriter object

     * @return PrintWriter object
     */

    private PrintWriter getWriter()
    {

        return writer;
    }

    /**

     * get the File object
     * @return the File object

     */
    public File getFile()

    {
        return file;

    }

    /**
     * Create a file and overwrite any existing file with identical file name followed by opening of the file

     */
    public void createFile()

    {
        try

        {
            file = new File(getFileName());

            if (file.exists())
            {

                file.delete();
            }

            setWriter(new PrintWriter(file));
        }

        catch(FileNotFoundException fnfex)
        {

            //Exception handled somewhere else
        }

        catch(Exception ex)
        {

            //Exception handled somewhere else
        }

    }

    public void help()
    {

        if(Desktop.isDesktopSupported())
        {

            try
            {

                File help = new File("Help.pdf");
                Desktop.getDesktop().open(help);

            }
            catch (IOException ioex)

            {
                final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/alert-red.png"));

                JOptionPane.showMessageDialog(null, ioex.toString() + "\nPlease contact the developer immediately.", "Error: void help()", JOptionPane.ERROR_MESSAGE, icon);
            }

            catch (Exception ex)
            {

                final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/alert-red.png"));
                JOptionPane.showMessageDialog(null, ex.toString() + "\nPlease contact the developer immediately.", "Error: void help()", JOptionPane.ERROR_MESSAGE, icon);

            }
        }

    }

    /**
     * append a text into a file

     * @param text text to append into file
     */

    public void printToFile(String text)
    {

        getWriter().append(text);
    }

    /**

     * Close the file so other processes can use it
     */

    public void close()
    {

        getWriter().close();
    }

    /**

     * Destructor to close the opened file and free memory
     * @throws Throwable

     */
    @Override

    protected void finalize() throws Throwable
    {

        super.finalize();
        close();

    }
}

/**
 * @author Zander Labuschagne 23585137

 * Main class to be used with Windows Operating Systems
 */

import javax.swing.*;

import java.util.Scanner;

public class Main
{

    public static void main(String[] args)
    {

        Scanner scanner = new Scanner(System.in);
        JFileChooser fileChooser = new JFileChooser();

        String sInput = "";
        int iInput = 0;

        NumberMap map = new NumberMap();

        System.out.println("Welcome Microsoft Windows user. To start playing, enter the number you want to move.");
        System.out.println("Enter any of the following at any time, or just continue playing:");

        System.out.println("\t\'about\'\t - Displays About Window with information on this software.");
        System.out.println("\t\'new\'\t - Generates a new random state and restart the game.");

        System.out.println("\t\'load\'\t - Loads a predefined game state from .csv file.");
        System.out.println("\t\'save\'\t - Saves the current game state to .csv file.");

        System.out.println("\t\'exit\'\t - Exits the game without saving.");
        System.out.println("\t\'help\'\t - Opens help document.");

        System.out.print("\n\n");
        map.printActiveState();

        try

        {

            while (true)
            {

                //Continue Game
                if (scanner.hasNextInt())

                {
                    iInput = scanner.nextInt();

                    if(iInput == 1)

                    {
                        map.move(1);

                        map.printActiveState();
                    }

                    else if(iInput == 2)
                    {

                        map.move(2);
                        map.printActiveState();

                    }
                    else if(iInput == 3)

                    {
                        map.move(3);

                        map.printActiveState();
                    }

                    else if(iInput == 4)
                    {

                        map.move(4);
                        map.printActiveState();

                    }
                    else if(iInput == 5)

                    {
                        map.move(5);

                        map.printActiveState();
                    }

                    else if(iInput == 6)
                    {

                        map.move(6);
                        map.printActiveState();

                    }
                    else if(iInput == 7)

                    {
                        map.move(7);

                        map.printActiveState();
                    }

                    else if(iInput == 8)
                    {

                        map.move(8);
                        map.printActiveState();

                    }
                    else if(iInput == 9)

                    {
                        map.move(9);

                        map.printActiveState();
                    }

                }
                //Menu

                else
                {

                    sInput = scanner.next();
                    if (sInput.equals("exit"))

                    {
                        map.finalize();

                        System.exit(0);
                    }

                    else if (sInput.equals("about"))
                    {

                        About about = new About();

                        about.pack();
                        about.setLocationRelativeTo(null);
                        about.setAlwaysOnTop(true);
                        about.setVisible(true);

                    }
                    else if (sInput.equals("new"))

                    {
                        map.finalize();

                        map = new NumberMap();
                        System.out.print(map.toString());

                    }
                    else if (sInput.equals("load"))

                    {
                        map.finalize();

                        int result = fileChooser.showOpenDialog(null);
                        if (result == JFileChooser.APPROVE_OPTION)

                        {
                            map.loadState(fileChooser.getSelectedFile().getPath());

                            map.printActiveState();
                        }

                    }
                    else if (sInput.equals("save"))

                    {
                        int result = fileChooser.showSaveDialog(null);

                        if (result == JFileChooser.APPROVE_OPTION)
                            map.saveState(fileChooser.getSelectedFile().getPath());

                        map.printActiveState();
                    }

                    else if (sInput.equals("help"))
                    {

                        IO io = new IO();
                        io.help();

                    }
                    else

                    {
                        final ImageIcon icon = new ImageIcon(Main.class.getResource("icons/error-red.png"));

                        JOptionPane.showMessageDialog(null, "Command or number not recognized.\nPlease enter either a number to continue playing the game or a valid command from the menu.", "Command/Number not Recognized", JOptionPane.ERROR_MESSAGE, icon);
                        System.out.println("\t\'about\'\t - Displays About Window with information on this software.");

                        System.out.println("\t\'new\'\t - Generates a new random state and restart the game.");
                        System.out.println("\t\'load\'\t - Loads a predefined game state from .csv file.");

                        System.out.println("\t\'save\'\t - Saves the current game state to .csv file.");
                        System.out.println("\t\'exit\'\t - Exits the game without saving.");

                        System.out.println("\t\'help\'\t - Opens help document.");
                        System.out.print("\n\n");

                        map.printActiveState();
                    }

                }
            }

        }
        catch(Exception ex)

        {
            final ImageIcon icon = new ImageIcon(Main.class.getClass().getResource("icons/alert-red.png"));

            JOptionPane.showMessageDialog(null, ex.toString() + "\nPlease contact the developer immediately.", "Error in main()", JOptionPane.ERROR_MESSAGE, icon);
        }

        catch (Throwable throwable)
        {

            throwable.printStackTrace();
        }

    }

}
/**

 * @author Zander Labuschagne 23585137
 * Main GUI

 * This GUI interacts with game in background
 * This GUI is designed to be used with only 9/8 Tiles

 * This GUI is onlu compatable with UNIX systems
 */

import javax.swing.Timer;

//GUI
import java.awt.*;

import java.awt.event.*;
import com.jgoodies.forms.factories.*;

import com.jgoodies.forms.layout.*;
//IO

import java.io.IOException;
//Audio

import javax.sound.sampled.*;
import javax.swing.*;

public class Puzzle extends JFrame

{
    /**

     * Object of the game
     */

    private NumberMap map;

    /**
     * Default Constructor

     */
    public Puzzle()

    {
        initComponents();

        map = new NumberMap();
        setButtons();

    }

    /**
     * Plays the Click sound

     * Execute when illegal move is attempted
     * Runs in own thread

     */
    public static synchronized void playClick()

    {
        new Thread(new Runnable()

        {
            public void run()

            {
                try

                {
                    Clip clip = AudioSystem.getClip();

                    AudioInputStream inputStream = AudioSystem.getAudioInputStream((this.getClass().getResource("sounds/click.wav")));
                    //AudioInputStream inputStream = AudioSystem.getAudioInputStream(new File("sounds/click.wav"));

                    clip.open(inputStream);

                    clip.start();
                }

                catch(LineUnavailableException luex)
                {

                    final ImageIcon icon = new ImageIcon((this.getClass().getResource("icons/error-red.png")));
                    JOptionPane.showMessageDialog(null, luex.toString() + "\nPlease upgrade your audio hardware to be able to play 24-bit sound.", "Error: void playClick()", JOptionPane.ERROR_MESSAGE, icon);

                }
                catch (Exception ex)

                {
                    final ImageIcon icon = new ImageIcon((this.getClass().getResource("icons/alert-red.png")));

                    JOptionPane.showMessageDialog(null, ex.toString() + "\nPlease contact the developer immediately.", "Error: void playClick()", JOptionPane.ERROR_MESSAGE, icon);
                }

            }
        }).start();

    }

    /**
     * Play Swipe sound

     * Executed when successful move is made
     * Runs in own thread

     */
    public static synchronized void playSwipe()

    {
        new Thread(new Runnable()

        {
            // The wrapper thread is unnecessary, unless it blocks on the

            // Clip finishing; see comments.
            public void run()

            {
                try

                {
                    Clip clip = AudioSystem.getClip();

                    AudioInputStream inputStream = AudioSystem.getAudioInputStream((this.getClass().getResource("sounds/whoosh.wav")));
                    clip.open(inputStream);

                    clip.start();
                }

                catch(Exception ex)
                {

                    final ImageIcon icon = new ImageIcon((this.getClass().getResource("icons/alert-red.png")));
                    JOptionPane.showMessageDialog(null, ex.toString() + "\nPlease contact the developer immediately.", "Error: void playSwipe()", JOptionPane.ERROR_MESSAGE, icon);

                }
            }

        }).start();
    }

    /**

     * Sets the new location for moved tile
     * @param btn JButton that's moved

     * @param x x-coordinate
     * @param y y-coordinate

     */
    public void setLocation(JButton btn, int x, int y)

    {
        int lat, longi;

        if(x == 0)

            lat = 0;
        else if(x == 1)

            lat = 170;
        else if(x == 2)

            lat = 340;
        else

            lat = 0;

        if(y == 0)
            longi = 230;

        else if(y == 1)
            longi = 115;

        else if(y == 2)
            longi = 0;

        else
            longi = 0;

        btn.setSize(170, 115);

        btn.setLocation(lat, longi);
    }

    /**

     * Set the JButtons on their correct positions
     */

    public void setButtons()
    {

        int x = map.getNode(1).getX();
        int y = map.getNode(1).getY();

        setLocation(btn1, x, y);
        btn1.setText("1");

        x = map.getNode(2).getX();

        y = map.getNode(2).getY();
        setLocation(btn2, x, y);

        btn2.setText("2");

        x = map.getNode(3).getX();
        y = map.getNode(3).getY();

        setLocation(btn3, x, y);
        btn3.setText("3");

        x = map.getNode(4).getX();

        y = map.getNode(4).getY();
        setLocation(btn4, x, y);

        btn4.setText("4");

        x = map.getNode(5).getX();
        y = map.getNode(5).getY();

        setLocation(btn5, x, y);
        btn5.setText("5");

        x = map.getNode(6).getX();

        y = map.getNode(6).getY();
        setLocation(btn6, x, y);

        btn6.setText("6");

        x = map.getNode(7).getX();
        y = map.getNode(7).getY();

        setLocation(btn7, x, y);
        btn7.setText("7");

        x = map.getNode(8).getX();

        y = map.getNode(8).getY();
        setLocation(btn8, x, y);

        btn8.setText("8");

        this.requestFocus();
    }

    /**

     * Move a Tile
     * @param btn JButton that is moved

     */
    public void moveButton(JButton btn)

    {
        if(String.format("%d", map.getLastMove()).equals(btn.getText()) /*|| map.getCountMoves() == 1*/)

        {
            playSwipe();

            if (map.getLastMoveVector() == "left")
                moveButtonLeft(btn);

            else if (map.getLastMoveVector() == "right")
                moveButtonRight(btn);

            else if (map.getLastMoveVector() == "up")
                moveButtonUp(btn);

            else if (map.getLastMoveVector() == "down")
                moveButtonDown(btn);

        }
        else

            playClick();
    }

    /**

     * Move a Tile to the Left
     * @param btn JButton to be moved

     */
    private void moveButtonLeft(JButton btn)
    {

        final Timer timer = new Timer(1, null);

        int origin = btn.getLocation().x;

        timer.addActionListener(new ActionListener()
        {

            @Override
            public void actionPerformed(ActionEvent e)

            {
                Point loc = btn.getLocation();

                int x = loc.x;

                int y = loc.y;

                btn.setLocation(x -= 1, y);
                if(btn.getLocation().x == origin - 168)

                {
                    setButtons();

                    timer.stop();
                }

                btn.setLocation(x -= 1, y);
                if(btn.getLocation().x == origin - 168)

                {
                    setButtons();

                    timer.stop();
                }

                btn.setLocation(x -= 1, y);
                if(btn.getLocation().x == origin - 168)

                {
                    setButtons();

                    timer.stop();
                }

            }
        });

        timer.start();
    }

    /**

     * Move a Tile to the Right
     * @param btn JButton to be moved

     */
    private void moveButtonRight(JButton btn)//////////////Toets

    {
        final Timer timer = new Timer(1, null);

        int origin = btn.getLocation().x;
        timer.addActionListener(new ActionListener()

        {
            @Override

            public void actionPerformed(ActionEvent e)
            {

                Point loc = btn.getLocation();

                int x = loc.x;
                int y = loc.y;

                btn.setLocation(x += 1, y);

                if(btn.getLocation().x == origin + 168)
                {

                    setButtons();
                    timer.stop();

                }
                btn.setLocation(x += 1, y);

                if(btn.getLocation().x == origin + 168)
                {

                    setButtons();
                    timer.stop();

                }
                btn.setLocation(x += 1, y);

                if(btn.getLocation().x == origin + 168)
                {

                    setButtons();
                    timer.stop();

                }
            }

        });
        timer.start();

    }

    /**
     * Moves a Tile Upwards

     * @param btn JButton to be moved
     */

    private void moveButtonUp(JButton btn)//////////////Toets
    {

        final Timer timer = new Timer(1, null);
        int origin = btn.getLocation().y;

        timer.addActionListener(new ActionListener()
        {

            @Override
            public void actionPerformed(ActionEvent e)

            {
                Point loc = btn.getLocation();

                int x = loc.x;

                int y = loc.y;

                btn.setLocation(x, y -= 1);
                if(btn.getLocation().y == origin - 114)

                {
                    setButtons();

                    timer.stop();
                }

                btn.setLocation(x, y -= 1);
                if(btn.getLocation().y == origin - 114)

                {
                    setButtons();

                    timer.stop();
                }

                btn.setLocation(x, y -= 1);
                if(btn.getLocation().y == origin - 114)

                {
                    setButtons();

                    timer.stop();
                }

            }
        });

        timer.start();
    }

    /**

     * Moves a Tile Downwards
     * @param btn JButton to be moved

     */
    private void moveButtonDown(JButton btn)

    {
        final Timer timer = new Timer(1, null);

        int origin = btn.getLocation().y;
        timer.addActionListener(new ActionListener()

        {
            @Override

            public void actionPerformed(ActionEvent e)
            {

                Point loc = btn.getLocation();

                int x = loc.x;
                int y = loc.y;

                btn.setLocation(x, y += 1);

                if(btn.getLocation().y == origin + 114)
                {

                    setButtons();
                    timer.stop();

                }
                btn.setLocation(x, y += 1);

                if(btn.getLocation().y == origin + 114)
                {

                    setButtons();
                    timer.stop();

                }
                btn.setLocation(x, y += 1);

                if(btn.getLocation().y == origin + 114)
                {

                    setButtons();
                    timer.stop();

                }
            }

        });
        timer.start();

    }

    /**
     * New Random Game State Menu Item

     * @param e
     * @throws Throwable

     */
    private void mnuFile_NewRandomStateActionPerformed(ActionEvent e) throws Throwable

    {
        map.finalize();

        map = new NumberMap();
        setButtons();

    }

    private void menuItem4ActionPerformed(ActionEvent e) {
        // TODO add your code here

    }

    /**
     * Set Tiles when window is focussed

     * @param e
     */

    private void thisWindowGainedFocus(WindowEvent e)
    {

        setButtons();
    }

    /**

     * Closes the current game state and load a new state from file
     * @param e

     * @throws Throwable
     */

    private void mnuFile_LoadGameStateActionPerformed(ActionEvent e) throws Throwable
    {

        map.finalize();
        int result = fileChooser.showOpenDialog(this);

        if(result == JFileChooser.APPROVE_OPTION)
        {

            map.loadState(fileChooser.getSelectedFile().getPath());
            setButtons();

        }
    }

    private void menuItem1ActionPerformed(ActionEvent e) throws IOException

    {
    }

    /**
     * Saves the current game state to file

     * @param e
     */

    private void mnuFile_SaveGameStateActionPerformed(ActionEvent e)
    {

        int result = fileChooser.showSaveDialog(this);
        if(result == JFileChooser.APPROVE_OPTION)

            map.saveState(fileChooser.getSelectedFile().getPath());
    }

    /**

     * Closes the current game state and Exit
     * @param e

     * @throws Throwable
     */

    private void mnuFile_ExitActionPerformed(ActionEvent e) throws Throwable
    {

        map.finalize();
        System.exit(0);

    }

    /**
     * Attempt Move Tile 1

     * @param e
     * @throws InterruptedException

     */
    private void btn1ActionPerformed(ActionEvent e) throws InterruptedException

    {
        map.move(1);

        moveButton(btn1);
    }

    /**

     * Attempt Move Tile 2
     * @param e

     */
    private void btn2ActionPerformed(ActionEvent e)

    {
        map.move(2);

        moveButton(btn2);
    }

    /**

     * Attempt Move Tile 3
     * @param e

     */
    private void btn3ActionPerformed(ActionEvent e)

    {
        map.move(3);

        moveButton(btn3);
    }

    /**

     * Attempt Move Tile 4
     * @param e

     */
    private void btn4ActionPerformed(ActionEvent e)

    {
        map.move(4);

        moveButton(btn4);
    }

    /**

     * Attempt Move Tile 5
     * @param e

     */
    private void btn5ActionPerformed(ActionEvent e)

    {
        map.move(5);

        moveButton(btn5);
    }

    /**

     * Attempt Move Tile 6
     * @param e

     */
    private void btn6ActionPerformed(ActionEvent e)

    {
        map.move(6);

        moveButton(btn6);
    }

    /**

     * Attempt Move Tile 7
     * @param e

     */
    private void btn7ActionPerformed(ActionEvent e)

    {
        map.move(7);

        moveButton(btn7);
    }

    /**

     * Attempt Move Tile 8
     * @param e

     */
    private void btn8ActionPerformed(ActionEvent e)

    {
        map.move(8);

        moveButton(btn8);
    }

    /**

     * Closes current game state and Exit
     * @param e

     * @throws Throwable
     */

    private void thisWindowClosing(WindowEvent e) throws Throwable
    {

        map.finalize();
        System.exit(0);

    }

    private void mnuAboutActionPerformed(ActionEvent e) throws IOException
    {

    }

    /**

     * Opens About Dialog
     * @param e

     * @throws IOException
     */

    private void mnuAboutMouseClicked(MouseEvent e) throws IOException
    {

        About about = new About(this);

        about.pack();
        about.setLocationRelativeTo(null);

        about.setAlwaysOnTop(true);
        about.setVisible(true);

    }

    private void mnuHelpMouseClicked(MouseEvent e)
    {

        new Thread(new Runnable()
        {

            public void run()
            {

                try
                {

                    IO io = new IO();
                    io.help();

                }
                catch (Exception ex)

                {
                    final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/alert-red.png"));

                    JOptionPane.showMessageDialog(null, ex.toString() + "\nPlease contact the developer immediately.", "Error: Help", JOptionPane.ERROR_MESSAGE, icon);
                }

            }
        }).start();

    }

    private void mnuAI_AnimatedSolveActionPerformed(ActionEvent e) {
        // TODO add your code here

    }

    private void mnuAI_QuickSolveActionPerformed(ActionEvent e) {
        // TODO add your code here

    }

    /**
     * Initialize GUI

     * Generated
     */

    private void initComponents()
    {

        // JFormDesigner - Component initialization - DO NOT MODIFY  //GEN-BEGIN:initComponents
        // Generated using JFormDesigner Evaluation license - Zander Labuschagne

        menuBar = new JMenuBar();
        mnuAbout = new JMenu();

        mnuFile = new JMenu();
        mnuFile_NewRandomState = new JMenuItem();

        mnuFile_LoadGameState = new JMenuItem();
        mnuFile_SaveGameState = new JMenuItem();

        separator1 = new JSeparator();
        mnuFile_Exit = new JMenuItem();

        mnuAI = new JMenu();
        mnuAI_AnimatedSolve = new JMenuItem();

        mnuAI_QuickSolve = new JMenuItem();
        mnuHelp = new JMenu();

        btn6 = new JButton();
        btn7 = new JButton();

        btn8 = new JButton();
        btn3 = new JButton();

        btn4 = new JButton();
        btn5 = new JButton();

        btn1 = new JButton();
        btn2 = new JButton();

        fileChooser = new JFileChooser();

        //======== this ========
        setTitle("8 Puzzle");

        setIconImage(new ImageIcon("/Users/ZanderMac/Downloads/WaterPlain0017_3_M.jpg").getImage());
        setResizable(false);

        addWindowFocusListener(new WindowAdapter() {
            @Override

            public void windowGainedFocus(WindowEvent e) {
                thisWindowGainedFocus(e);

            }
        });

        addWindowListener(new WindowAdapter() {
            @Override

            public void windowClosing(WindowEvent e) {
                try

                {
                    thisWindowClosing(e);

                } catch (Throwable throwable)
                {

                    throwable.printStackTrace();
                }

            }
        });

        Container contentPane = getContentPane();
        contentPane.setLayout(new FormLayout(

            "88dlu, $lcgap, 85dlu, $lcgap, 92dlu",
            "70dlu, $lgap, 65dlu, $lgap, 70dlu"));

        //======== menuBar ========

        {

            //======== mnuAbout ========
            {

                mnuAbout.setText("About");
                mnuAbout.addActionListener(e -> {

                    try
                    {

                        mnuAboutActionPerformed(e);
                    } catch (IOException e1)

                    {

                        e1.printStackTrace();

                    }
                    try

                    {

                        mnuAboutActionPerformed(e);

                    } catch (IOException e1)
                    {

                        e1.printStackTrace();
                    }

                    try
                    {

                        mnuAboutActionPerformed(e);
                    } catch (IOException e1)

                    {

                        e1.printStackTrace();

                    }
                });

                mnuAbout.addMouseListener(new MouseAdapter() {
                    @Override

                    public void mouseClicked(MouseEvent e) {
                        try

                        {
                            mnuAboutMouseClicked(e);

                        } catch (IOException e1)
                        {

                            e1.printStackTrace();
                        }

                    }
                });

            }
            menuBar.add(mnuAbout);

            //======== mnuFile ========

            {
                mnuFile.setText("Game");

                //---- mnuFile_NewRandomState ----

                mnuFile_NewRandomState.setText("New Random State");
                mnuFile_NewRandomState.addActionListener(e -> {

                                                                                                                                  menuItem4ActionPerformed(e);
                    try

                    {
                        mnuFile_NewRandomStateActionPerformed(e);

                    } catch (Throwable throwable)
                    {

                        throwable.printStackTrace();
                    }

                });
                mnuFile.add(mnuFile_NewRandomState);

                //---- mnuFile_LoadGameState ----

                mnuFile_LoadGameState.setText("Load Game State");
                mnuFile_LoadGameState.addActionListener(e -> {

                    try
                    {

                        menuItem1ActionPerformed(e);
                    } catch (IOException e1)

                    {

                        e1.printStackTrace();

                    }
                    try

                    {

                        mnuFile_LoadGameStateActionPerformed(e);

                    } catch (Throwable throwable)
                    {

                        throwable.printStackTrace();
                    }

                });
                mnuFile.add(mnuFile_LoadGameState);

                //---- mnuFile_SaveGameState ----

                mnuFile_SaveGameState.setText("Save Game State");
                mnuFile_SaveGameState.addActionListener(e -> mnuFile_SaveGameStateActionPerformed(e));

                mnuFile.add(mnuFile_SaveGameState);
                mnuFile.add(separator1);

                //---- mnuFile_Exit ----

                mnuFile_Exit.setText("Exit");
                mnuFile_Exit.addActionListener(e -> {

                    try
                    {

                        mnuFile_ExitActionPerformed(e);
                    } catch (Throwable throwable)

                    {

                        throwable.printStackTrace();

                    }
                });

                mnuFile.add(mnuFile_Exit);
            }

            menuBar.add(mnuFile);

            //======== mnuAI ========
            {

                mnuAI.setText("AI");

                //---- mnuAI_AnimatedSolve ----
                mnuAI_AnimatedSolve.setText("Animated Solve");

                mnuAI_AnimatedSolve.addActionListener(e -> mnuAI_AnimatedSolveActionPerformed(e));
                mnuAI.add(mnuAI_AnimatedSolve);

                //---- mnuAI_QuickSolve ----

                mnuAI_QuickSolve.setText("Quick Solve");
                mnuAI_QuickSolve.addActionListener(e -> mnuAI_QuickSolveActionPerformed(e));

                mnuAI.add(mnuAI_QuickSolve);
            }

            menuBar.add(mnuAI);

            //======== mnuHelp ========
            {

                mnuHelp.setText("Help");
                mnuHelp.addMouseListener(new MouseAdapter() {

                    @Override
                    public void mouseClicked(MouseEvent e) {

                        mnuHelpMouseClicked(e);
                        mnuHelpMouseClicked(e);

                    }
                });

            }
            menuBar.add(mnuHelp);

        }
        setJMenuBar(menuBar);

        //---- btn6 ----

        btn6.setFont(new Font("Chalkduster", Font.PLAIN, 64));
        btn6.addActionListener(e -> btn6ActionPerformed(e));

        contentPane.add(btn6, CC.xy(1, 1, CC.FILL, CC.FILL));

        //---- btn7 ----
        btn7.setFont(new Font("Chalkduster", Font.PLAIN, 64));

        btn7.addActionListener(e -> btn7ActionPerformed(e));
        contentPane.add(btn7, CC.xy(3, 1, CC.FILL, CC.FILL));

        //---- btn8 ----

        btn8.setFont(new Font("Chalkduster", Font.PLAIN, 64));
        btn8.addActionListener(e -> btn8ActionPerformed(e));

        contentPane.add(btn8, CC.xy(5, 1, CC.FILL, CC.FILL));

        //---- btn3 ----
        btn3.setFont(new Font("Chalkduster", Font.PLAIN, 64));

        btn3.addActionListener(e -> btn3ActionPerformed(e));
        contentPane.add(btn3, CC.xy(1, 3, CC.FILL, CC.FILL));

        //---- btn4 ----

        btn4.setFont(new Font("Chalkduster", Font.PLAIN, 64));
        btn4.addActionListener(e -> btn4ActionPerformed(e));

        contentPane.add(btn4, CC.xy(3, 3, CC.FILL, CC.FILL));

        //---- btn5 ----
        btn5.setFont(new Font("Chalkduster", Font.PLAIN, 64));

        btn5.setForeground(Color.black);
        btn5.setBackground(new Color(0, 204, 255));

        btn5.setOpaque(false);
        btn5.addActionListener(e -> btn5ActionPerformed(e));

        contentPane.add(btn5, CC.xy(5, 3, CC.FILL, CC.FILL));

        //---- btn1 ----
        btn1.setFont(new Font("Chalkduster", Font.PLAIN, 64));

        btn1.addActionListener(e -> {
            try

            {
                btn1ActionPerformed(e);

            } catch (InterruptedException e1)
            {

                e1.printStackTrace();
            }

        });
        contentPane.add(btn1, CC.xy(1, 5, CC.FILL, CC.FILL));

        //---- btn2 ----

        btn2.setFont(new Font("Chalkduster", Font.PLAIN, 64));
        btn2.addActionListener(e -> btn2ActionPerformed(e));

        contentPane.add(btn2, CC.xy(3, 5, CC.FILL, CC.FILL));
        pack();

        setLocationRelativeTo(getOwner());

        //---- fileChooser ----
        fileChooser.setCurrentDirectory(null);

        // JFormDesigner - End of component initialization  //GEN-END:initComponents
    }

    // JFormDesigner - Variables declaration - DO NOT MODIFY  //GEN-BEGIN:variables

    // Generated using JFormDesigner Evaluation license - Zander Labuschagne
    private JMenuBar menuBar;

    private JMenu mnuAbout;
    private JMenu mnuFile;

    private JMenuItem mnuFile_NewRandomState;
    private JMenuItem mnuFile_LoadGameState;

    private JMenuItem mnuFile_SaveGameState;
    private JSeparator separator1;

    private JMenuItem mnuFile_Exit;
    private JMenu mnuAI;

    private JMenuItem mnuAI_AnimatedSolve;
    private JMenuItem mnuAI_QuickSolve;

    private JMenu mnuHelp;
    private JButton btn6;

    private JButton btn7;
    private JButton btn8;

    private JButton btn3;
    private JButton btn4;

    private JButton btn5;
    private JButton btn1;

    private JButton btn2;
    private JFileChooser fileChooser;

    // JFormDesigner - End of variables declaration  //GEN-END:variables

    /**
     * Main Method

     * @param args Arguments loaded from Command
     */

    public static void main(String[] args)
    {

        javax.swing.SwingUtilities.invokeLater(new Runnable()
        {

            public void run()
            {

                Puzzle GUI = new Puzzle();

                GUI.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                GUI.setVisible(true);

            }

        });

    }
}

/**
 * @author Zander Labuschagne

 * About Form Class
 */

//GUI

import java.awt.*;
import java.awt.event.*;

import javax.swing.*;
import com.jgoodies.forms.factories.*;

import com.jgoodies.forms.layout.*;
//I/O
```

```
import java.io.IOException;

public class About extends JDialog
{
    public About() throws IOException
    {
        initComponents();
    }

    /**
     * Overloaded Constructor

     * Generated
     * @param owner

     * @throws IOException
     */
    public About(Frame owner) throws IOException
    {

        super(owner);

        initComponents();
    }

    /**

     * OK Button closes dialog
     * @param e

     */
    private void okButtonActionPerformed(ActionEvent e)

    {
        dispose();

    }

    private void thisWindowGainedFocus(WindowEvent e) {
        // TODO add your code here

    }

    /**
     * Initialize the GUI

     * Generated
     */

    private void initComponents()
    {

        // JFormDesigner - Component initialization - DO NOT MODIFY  //GEN-BEGIN:initComponents
        // Generated using JFormDesigner Evaluation license - Zander Labuschagne

        dialogPane = new JPanel();
        contentPanel = new JPanel();

        label1 = new JLabel();
        label2 = new JLabel();

        label6 = new JLabel();
        panel2 = new JPanel();

        label4 = new JLabel();
        label5 = new JLabel();

        buttonBar = new JPanel();
        okButton = new JButton();

        //======== this ========

        setAlwaysOnTop(true);
        addWindowFocusListener(new WindowAdapter() {

            @Override
            public void windowGainedFocus(WindowEvent e) {

                thisWindowGainedFocus(e);
            }

        });
        Container contentPane = getContentPane();

        contentPane.setLayout(new BorderLayout());

        //======== dialogPane ========
        {

            dialogPane.setBorder(Borders.createEmptyBorder("9dlu, 9dlu, 9dlu, 9dlu"));

            // JFormDesigner evaluation mark
            dialogPane.setBorder(new javax.swing.border.CompoundBorder(

                new javax.swing.border.TitledBorder(new javax.swing.border.EmptyBorder(0, 0, 0, 0),
                    "JFormDesigner Evaluation", javax.swing.border.TitledBorder.CENTER,

                    javax.swing.border.TitledBorder.BOTTOM, new java.awt.Font("Dialog", java.awt.Font.BOLD, 12),
                    java.awt.Color.red), dialogPane.getBorder())); dialogPane.addPropertyChangeListener(new java.beans.PropertyChangeListener(){public void propertyChange(java.beans.PropertyChangeEvent e){if("border".equals(e.getPropertyName()))throw new RuntimeException();}});

            dialogPane.setLayout(new BorderLayout());

            //======== contentPanel ========

            {
                contentPanel.setLayout(new FormLayout(

                    "4*(default, $lcgap), 149dlu, 4*($lcgap, default)",
                    "7*(default, $lgap), default"));

                //---- label1 ----

                label1.setText("8 Puzzle");
                label1.setFont(new Font("Chalkboard", Font.BOLD, 20));

                contentPanel.add(label1, CC.xywh(1, 3, 17, 1, CC.CENTER, CC.DEFAULT));

                //---- label2 ----
                label2.setText("Developed By:");

                contentPanel.add(label2, CC.xywh(1, 7, 17, 1, CC.CENTER, CC.DEFAULT));

                //---- label6 ----
                label6.setText("Zander Labuschagne");

                contentPanel.add(label6, CC.xywh(1, 9, 17, 1, CC.CENTER, CC.DEFAULT));

                //======== panel2 ========
                {

                    panel2.setLayout(new FormLayout(
                        "73dlu",

                        "default"));
                }

                contentPanel.add(panel2, CC.xy(9, 9));

                //---- label4 ----
                label4.setText("North-West University: Potchefstroom");

                contentPanel.add(label4, CC.xywh(1, 13, 17, 1, CC.CENTER, CC.DEFAULT));

                //---- label5 ----
                label5.setText("E-Mail: 23585137@PROTONMAIL.CH");

                contentPanel.add(label5, CC.xywh(1, 15, 17, 1, CC.CENTER, CC.DEFAULT));
            }

            dialogPane.add(contentPanel, BorderLayout.CENTER);

            //======== buttonBar ========
            {

                buttonBar.setBorder(Borders.createEmptyBorder("4dlu, 0dlu, 0dlu, 0dlu"));
                buttonBar.setLayout(new FormLayout(

                    "$glue, $button",
                    "pref"));

                //---- okButton ----

                okButton.setText("OK");
                okButton.addActionListener(e -> okButtonActionPerformed(e));

                buttonBar.add(okButton, CC.xywh(1, 1, 2, 1));
            }

            dialogPane.add(buttonBar, BorderLayout.SOUTH);
        }

        contentPane.add(dialogPane, BorderLayout.CENTER);
        pack();

        setLocationRelativeTo(null);
        // JFormDesigner - End of component initialization  //GEN-END:initComponents

    }

    // JFormDesigner - Variables declaration - DO NOT MODIFY  //GEN-BEGIN:variables
    // Generated using JFormDesigner Evaluation license - Zander Labuschagne

    private JPanel dialogPane;
    private JPanel contentPanel;

    private JLabel label1;
    private JLabel label2;

    private JLabel label6;
    private JPanel panel2;

    private JLabel label4;
    private JLabel label5;

    private JPanel buttonBar;
    private JButton okButton;

    // JFormDesigner - End of variables declaration  //GEN-END:variables
}

/**
 * @author Zander Labuschagne 23585137

 * Class to manage Number Nodes(Game Engine)
 * This software was designed to be independent of GUI and other software

 */

import java.util.Random;
//GUI

import javax.swing.JOptionPane;
import javax.swing.ImageIcon;

//Audio
import javax.sound.sampled.AudioInputStream;

import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;

//I/O
import java.util.Scanner;

//Exceptions
import java.io.FileNotFoundException;

import java.nio.file.FileSystemException;

public class NumberMap
{

    //Instance Variables
    /**

     * the starting state of the game presented as a matrix
     */

    private int[][] stateIn;
    /**

     * the current state of the game
     * should be updated after every move

     */
    private int[][] activeState;

    /**
     * node which is presented as the origin node

     * see mutator for more details
     */

    private NumberNode origin;
    /**

     * node which is presented as the empty space
     */

    private NumberNode nullNode;
    /**

     * integer which contains the amount of moves made
     * updated after every move

     */
    private int countMoves;

    /**
     * Matrix to be used to determine end game

     */
    private int[][] winState;

    /**
     * object of IO class to handle all file io

     */
    private IO gameTracker;

    /**
     * string containing last move vector(left;right;up;down)

     */
    private String lastMoveVector;

    /**
     * int containing last number moved

     */
    private int lastMove;

    /**
     * Number of nodes/blocks contained in the x-axis

     */
    private int y_axisNodes;

    /**
     * Number of nodes/blocks contained in the y-axis

     */
    private int x_axisNodes;

    /**

     * Default Constructor
     */

    public NumberMap()
    {

        this(null, null);
    }

    /**

     * Overloaded Constructor
     * @param stateIn The state in which the game should start, presented in matrix format

     * @param winState The state in which the game is won, presented in matrix format
     */

    public NumberMap(int[][] stateIn, int[][] winState)
    {

        setNullNode(null);
        setOrigin(null);

        setY_axis(3);
        setX_axis(3);

        if(stateIn == null)

        {
            setRandomState();

        }
        else

            setStateIn(stateIn);
        if(winState == null)

            setWinState(new int[][]{{7, 8, 0}, {4, 5, 6}, {1, 2, 3}});
        else

            setWinState(winState);
        countMoves = 0;

    }

    //Mutators
    /**

     * set the origin node
     * Multi Dimensional Doubly Linked List Data Structure Presented in the form of a Cartesian Pane

     * @param origin the node object which should be set/considered as the origin node
     */

    public void setOrigin(NumberNode origin)
    {

        this.origin = origin;
    }

    /**

     * sets the null node
     * The null node is the node occupying the empty block

     * @param nullNode the node object which should be set/considered as the empty block/node
     */

    public void setNullNode(NumberNode nullNode)
    {

        this.nullNode = nullNode;
    }

    private void setX_axis(int x)

    {
        x_axisNodes = x;

    }

    private void setY_axis(int y)
    {

        y_axisNodes = y;
    }

    /**

     * Awesome Goed Gebeur
     * Builds the Multi Dimensional Doubly Linked List Data Structure

     * Configure the starting game state
     * @param stateIn 2 Dimensional Array that represents the starting state of the game

     */
    public void setStateIn(int[][] stateIn)

    {
        try

        {
            this.stateIn = stateIn;

            countMoves = 0;
            //Create Datastructure

            NumberNode node = new NumberNode(getStateIn()[0][0], 0, getY_axis() - 1);
            NumberNode prev;

            for (int x = 0; x < getX_axis(); x++)
            {

                int y = 0;

                if (x != 0)
                {

                    prev = getNodeAt(x - 1, getY_axis() - 1);
                    prev.setRight(new NumberNode(getStateIn()[y][x], x, (getY_axis() - 1 - y)));

                    prev.getRight().setLeft(prev);
                    node = prev.getRight();

                }
                if (getStateIn()[y][x] == 0)

                    setNullNode(node);

                for (y = getY_axis() - 2; y >= 0; y--)
                {

                    node.setDown(new NumberNode(getStateIn()[getY_axis() - 1 - y][x], x, y));
                    node.getDown().setUp(node);

                    node = node.getDown();
                    if (getStateIn()[getY_axis() - 1 - y][x] == 0)

                        setNullNode(node);

                    if (x > 0)
                    {

                        NumberNode ptr = getNodeAt(x - 1, y);
                        node.setLeft(ptr);

                        node.getLeft().setRight(node);
                    }

                    if (x == 0 && y == 0)

                        setOrigin(node);
                }

            }
            setActiveState();

            gameTracker = new IO("Game_Tracker.txt");

            gameTracker.createFile();
            gameTracker.printToFile("Initial State:\n" + toString() + "\n");

        }
        catch(Exception ex)

        {
            final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/alert-red.png"));

            JOptionPane.showMessageDialog(null, ex.toString() + "\nPlease contact the developer immediately.", "Error: void setStateIn(int[][])", JOptionPane.ERROR_MESSAGE, icon);
        }

    }

    /**
     * sets a randomized state to be built as a data structure

     * used as initial game state
     */

    public void setRandomState()
    {

        try
        {

            Random ra = new Random();
            int[][] a = new int[getY_axis()][getX_axis()];

            int random_number = ra.nextInt(getY_axis() * getX_axis() + 1);
            int iii = 0;

            for (int i = 0; i < getX_axis(); i++)

                for (int ii = 0; ii < getY_axis(); ii++)
                {

                    while (contains(a, random_number) && iii < (getX_axis() * getY_axis()))
                    {

                        random_number = ra.nextInt(getY_axis() * getX_axis() + 1);
                    }

                    a[ii][i] = random_number;

                    iii++;
                }

            for (int i = 0; i < getX_axis(); i++)

                for (int ii = 0; ii < getY_axis(); ii++)
                    if (a[ii][i] == getY_axis() * getX_axis())

                        a[ii][i] = 0;

            setActiveState(a);
        }

        catch(Exception ex)
        {

            final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/alert-red.png"));
            JOptionPane.showMessageDialog(null, ex.toString() + "\nPlease contact the developer immediately.", "Error: void setRandomState()", JOptionPane.ERROR_MESSAGE, icon);

        }
    }

    /**

     * Updates the matrix to be used as a reference throughout the game
     * Should be called after every move

     */
    public void setActiveState()

    {
        int[][] a = new int[getY_axis()][getX_axis()];

        for(int i = 0; i < getY_axis(); i++)
            for(int ii = 0; ii < getX_axis(); ii++)

                a[i][ii] = getNodeAt(ii, i).getInfo();

        activeState = a;
    }

    /**

     * sets the current state of the game after move or initialization
     * @param activeState the matrix to configure or update the data structure

     */
    public void setActiveState(int[][] activeState)// set up a state according to the array

    {
        setStateIn(activeState);

    }

    /**
     * sets the winning state of the game to determine end game

     * @param winState the matrix to be used to compare with the active game state to determine end game
     */

    public void setWinState(int[][] winState)
    {

        this.winState = winState;
    }

    /**

     * sets the lastmove
     * @param lastMove the last move

     */
    public void setLastMoveVector(String lastMove)

    {
        this.lastMoveVector = lastMove;

    }

    /**
     * sets the node that made the last move

     * used by GUI
     * @param move

     */
    public void setLastMove(int move)

    {
        lastMove = move;

    }

    //Accessors
    /**

     * get the amount of moves made
     * @return number of moves made

     */
    public int getCountMoves()

    {
        return countMoves;

    }

    /**
     * get the origin node reference to find or reference other nodes

     * @return the origin node reference
     */

    public NumberNode getOrigin()
    {

        return origin;
    }

    /**

     * get the empty space node
     * @return the empty space node reference

     */
    public NumberNode getNullNode()

    {
        return nullNode;

    }

    /**
     * Request a node by value

     * No node should ever hold the same value
     * @param value the value that the requested node sshould hold

     * @return node reference congaing the requested value
     */
```

```
public NumberNode getNode(int value)
{

    NumberNode ptr = getOrigin();

    try
    {

        for(int x = 0; x < getX_axis() && ptr.getInfo() != value; x++)
            for(int y = 0; y < getY_axis() && ptr.getInfo() != value; y++)

                ptr = getNodeAt(x, y);

        if(ptr == null)
            throw new NullPointerException("The node with the queried value does not exist.\ngetNode(int value) returned null");

    }
    catch(NullPointerException npex)

    {
        final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/alert-red.png"));

        JOptionPane.showMessageDialog(null, npex.getMessage() + "\nPlease contact the developer immediately.", "Non existent node value", JOptionPane.ERROR_MESSAGE, icon);
    }

    catch(Exception ex)
    {

        final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/alert-red.png"));
        JOptionPane.showMessageDialog(null, ex.getMessage() + "\nPlease contact the developer immediately.", "Error: NumberNode getNode(int value)", JOptionPane.ERROR_MESSAGE, icon);

    }

    return ptr;
}

/**

 * Request a node by coordinates
 * No node should ever hold the same coordinates

 * @param x latitude(x-axis) coordinate to find requested node
 * @param y longitude(y-axis) coordinate to find requested node

 * @return reference to the requested node
 */
public NumberNode getNodeAt(int x, int y)
{

    NumberNode ptr = getOrigin();

    try
    {

        if(x == 0 && y == 0)
            return ptr;

        if(x > (getX_axis() - 1) || y > (getY_axis() - 1) || x < 0 || y < 0)

            throw new NullPointerException("The node at the queried coordinates does not exist in the given range.\nNumberNode getNodeAt(int x, int y) returned null");

        int lat = 0, longi = 0;
        while(x != lat || y != longi)

        {
            if(lat < x)

            {
                ptr = ptr.getRight();

                lat++;
            }

            if(longi < y)
            {

                ptr = ptr.getUp();
                longi++;

            }
        }

    }
    catch(NullPointerException npex)

    {
        final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/alert-red.png"));

        JOptionPane.showMessageDialog(null, npex.getMessage() + "\nPlease contact the developer immediately.", "Node coordinates out of range", JOptionPane.ERROR_MESSAGE, icon);
    }

    catch(Exception ex)
    {

        final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/alert-red.png"));
        JOptionPane.showMessageDialog(null, ex.getMessage() + "\nPlease contact the developer immediately.", "Error: NumberNode getNodeAt(int x, int y)", JOptionPane.ERROR_MESSAGE, icon);

    }

    return ptr;

}

/**
 * get the winning state

 * @return matrix representing the end game
 */

public int[][] getWinState()
{

    return winState;
}

/**

 * get the initial/start state of the game
 * @return matrix which represent the initial state

 */
public int[][] getStateIn()

{
    return stateIn;

}

/**
 * get the current game state

 * @return matrix which represent the current game state
 */

public int[][] getActiveState()
{

    return activeState;
}

/**

 * get the last move made's vector
 * @return string containing last move vector

 */
public String getLastMoveVector()

{
    return lastMoveVector;

}

/**
 * get the last move made's number

 * @return ineger containing the last move's number
 */

public int getLastMove()
{

    return lastMove;
}

/**

 * get the x-axis nodes
 * @return

 */
private int getX_axis()

{
    return x_axisNodes;

}

/**
 * get the y-axis nodes

 * @return
 */

private int getY_axis()
{

    return y_axisNodes;
}

/**
 * Matrix representation of the current game state in string format

 * @return string formatted matrix representation of the game
 */

public String toString()
{

    String state = "";
    for(int i = 0; i < getY_axis(); i++)

    {
        for(int ii = 0; ii < getX_axis(); ii++)

            state += (getActiveState()[getY_axis() - 1 - i][ii] == 0 ? "B " : String.format("%d ", getActiveState()[getY_axis() - 1 - i][ii]));
        state += "\n";

    }

    return state;
}

/**

 * Comma separated value formatted representation of the current game state to be used to write/output to .csv file
 * @return string formatted comma separated value representation of current game state

 */
public String toCSV()

{
    String state = "";

    for(int i = 0; i < getY_axis(); i++)
    {

        for(int ii = 0; ii < getX_axis(); ii++)
            state += (getActiveState()[getY_axis() - 1 - i][ii] == 0 ? 'B' : String.format("%d", getActiveState()[getY_axis() - 1 - i][ii])) + (i == (getY_axis() - 1) && ii == (getX_axis() - 1) ? "" : ", ");

    }

    return state;
}

/**

 * Comma separated value formatted representation of the end game state to be used to write/output to .csv file
 * @return string formatted comma separated value representation of end game state

 */
public String toCSVWin()

{
    String state = "";

    for(int i = 0; i < getY_axis(); i++)
    {

        for(int ii = 0; ii < getX_axis(); ii++)
            state += (getWinState()[getY_axis() - 1 - i][ii] == 0 ? 'B' : String.format("%d", getWinState()[getY_axis() - 1 - i][ii])) + (i == (getY_axis() - 1) && ii == (getX_axis() - 1) ? "" : ", ");

    }

    return state;
}

/**

 * Print the current game state to console in matrix format
 */

public void printActiveState()
{

    System.out.print(toString() + '\n');
}

/**

 * Determine if game won
 * @return true if game won

 * play win sound in new thread
 */

public boolean gameStatus()
{

    boolean b = true;

    for(int i = 0; i < getY_axis(); i++)
        for(int ii = 0; ii < getX_axis(); ii++)

            if(getActiveState()[i][ii] != getWinState()[i][ii])
                b = false;

    if(b)

    {
        new Thread(new Runnable()

        {
            public void run()

            {
                try

                {
                    Clip clip = AudioSystem.getClip();

                    AudioInputStream inputStream = AudioSystem.getAudioInputStream(this.getClass().getResource("sounds/win2.wav"));
                    clip.open(inputStream);

                    clip.start();
                }

                catch(Exception ex)
                {

                    final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/alert-red.png"));
                    JOptionPane.showMessageDialog(null, ex.toString() + "\nPlease contact the developer immediately.", "Error: boolean gameStatus()", JOptionPane.ERROR_MESSAGE, icon);

                }
            }

        }).start();
        final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/info-green.png"));

        JOptionPane.showMessageDialog(null, String.format("Congratulations!\nYou have completed the puzzle in %d moves.\nTime for some coffee...", getCountMoves()), "Puzzle Completed!", JOptionPane.INFORMATION_MESSAGE, icon);
        gameTracker.printToFile(String.format("GAME WON! in %d moves\n\n\n", getCountMoves()));

        countMoves = 0;
        setLastMove(0);

        setLastMoveVector("None");
        return true;

    }
    return false;

}

/**
 * Calls the correct function that moves the node holding the number

 * Only a node adjacent to the null node can be moved into the null node's position, by swapping this node with the null node
 * @param number value of the node to be moved

 */
public void move(int number)

{
    NumberNode node = getNode(number);

    if(node.getLeft() != null && node.getLeft().isNullNode())

        moveLeft(node);
    else if(node.getRight() != null && node.getRight().isNullNode())

        moveRight(node);
    else if(node.getUp() != null && node.getUp().isNullNode())

        moveUp(node);
    else if(node.getDown() != null && node.getDown().isNullNode())

        moveDown(node);
}

/**

 * Move the node a block left, by swapping this node with the null node
 * Will only activate if the null node is left adjacent to this node

 * @param node node which is requested to move
 */

private void moveLeft(NumberNode node)
{

    try
    {

        if (node.getUp() != null)
        {

            node.setUp(node.getUp().getLeft());
            node.getUp().setDown(node);

            node.getUp().getRight().setDown(getNullNode());

            getNullNode().setUp(node.getUp().getRight());
        }

        if (node.getDown() != null)
        {

            node.setDown(node.getDown().getLeft());
            node.getDown().setUp(node);

            node.getDown().getRight().setUp(getNullNode());

            getNullNode().setDown(node.getDown().getRight());
        }

        if (node.getLeft() != null && node.getLeft().getLeft() != null)//Kyk of node in derde kolom is
        {

            node.setLeft(node.getLeft().getLeft());
            node.getLeft().setRight(node);

        }
        if (node.getRight() != null)

        {
            node.getRight().setLeft(getNullNode());

            getNullNode().setRight(node.getRight());
        }

        if (node.getRight() == null)
            getNullNode().setRight(null);

        if (getNullNode().getLeft() == null)
            node.setLeft(null);

        node.setRight(getNullNode());

        getNullNode().setLeft(node);

        if (getOrigin().isNullNode())
            setOrigin(node);

        node.setX(node.getX() - 1);

        getNullNode().setX(getNullNode().getX() + 1);

        setActiveState();

        countMoves++;
        addMove(node.getInfo(), "Left");

        setLastMoveVector("left");
        setLastMove(node.getInfo());

        gameStatus();
    }

    catch(Exception ex)
    {

        final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/alert-red.png"));
        JOptionPane.showMessageDialog(null, ex.toString() + "\nPlease contact the developer immediately.", "Error: void moveLeft(NumberNode node)", JOptionPane.ERROR_MESSAGE, icon);

    }
}

/**

 * Move the node a block right, by swapping it with the null node
 * Will only activate if the null node is right adjacent to this node

 * @param node node which is requested to move
 */

private void moveRight(NumberNode node)
{

    try
    {

        if (getOrigin().getInfo() == node.getInfo())
        {

            setOrigin(getNullNode());
            getNullNode().setX(0);

            getNullNode().setY(0);
        }

        if (node.getUp() != null)

        {
            node.setUp(node.getUp().getRight());

            node.getUp().setDown(node);

            node.getUp().getLeft().setDown(getNullNode());
            getNullNode().setUp(node.getUp().getLeft());

        }
        if (node.getDown() != null)

        {
            node.setDown(node.getDown().getRight());

            node.getDown().setUp(node);

            node.getDown().getLeft().setUp(getNullNode());
            getNullNode().setDown(node.getDown().getLeft());

        }
        if (node.getRight() != null && node.getRight().getRight() != null)//Kyk of node in derde kolom is

        {
            node.setRight(node.getRight().getRight());

            node.getRight().setLeft(node);
        }

        if (node.getLeft() != null)
        {

            node.getLeft().setRight(getNullNode());
            getNullNode().setLeft(node.getLeft());

        }
        if (node.getLeft() == null)

            getNullNode().setLeft(null);
        if (getNullNode().getRight() == null)

            node.setRight(null);

        node.setLeft(getNullNode());
        getNullNode().setRight(node);

        node.setX(node.getX() + 1);

        getNullNode().setX(getNullNode().getX() - 1);

        setActiveState();
        countMoves++;

        addMove(node.getInfo(), "Right");
        setLastMoveVector("right");

        setLastMove(node.getInfo());
        gameStatus();

    }
    catch(Exception ex)

    {
        final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/alert-red.png"));

        JOptionPane.showMessageDialog(null, ex.toString() + "\nPlease contact the developer immediately.", "Error: void moveRight(NumberNode node)", JOptionPane.ERROR_MESSAGE, icon);
    }

}

/**
 * Move the node a block up, by swapping it with the null node

 * Will only activate if the null node is upwards adjacent to this node
 * @param node node which is requested to move

 */
private void moveUp(NumberNode node)

{
    try

    {
        if (node.getLeft() != null)

        {
            node.setLeft(node.getLeft().getUp());

            node.getLeft().setRight(node);

            node.getLeft().getDown().setRight(getNullNode());
            getNullNode().setLeft(node.getLeft().getDown());

        }
        if (node.getRight() != null)

        {
            node.setRight(node.getRight().getUp());

            node.getRight().setLeft(node);

            node.getRight().getDown().setLeft(getNullNode());
            getNullNode().setRight(node.getRight().getDown());

        }
        if (node.getUp() != null && node.getUp().getUp() != null)//Kyk of node in derde kolom is

        {
            node.setUp(node.getUp().getUp());

            node.getUp().setDown(node);
        }

        if (node.getDown() != null)
        {

            node.getDown().setUp(getNullNode());
            getNullNode().setDown(node.getDown());

        }
        if (node.getDown() == null)

            getNullNode().setDown(null);
        if (getNullNode().getUp() == null)

            node.setUp(null);

        node.setDown(getNullNode());
        getNullNode().setUp(node);

        node.setY(node.getY() + 1);

        getNullNode().setY(getNullNode().getY() - 1);

        if (getOrigin().getInfo() == node.getInfo())
        {

            setOrigin(getNullNode());
            getNullNode().setX(0);

            getNullNode().setY(0);
        }

        setActiveState();

        countMoves++;
        addMove(node.getInfo(), "Up");

        setLastMoveVector("up");
        setLastMove(node.getInfo());

        gameStatus();
    }

    catch(Exception ex)
    {

        final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/alert-red.png"));
        JOptionPane.showMessageDialog(null, ex.toString() + "\nPlease contact the developer immediately.", "Error: void moveUp(NumberNode node)", JOptionPane.ERROR_MESSAGE, icon);

    }
}

/**

 * Move the node a block down, by swapping it with the null node
 * Will only activate if the null node is downwards adjacent to this node

 * @param node node which is requested to move
 */

private void moveDown(NumberNode node)
{

    try
    {
            if (node.getLeft() != null)
            {

                node.setLeft(node.getLeft().getDown());
                node.getLeft().setRight(node);

                node.getLeft().getUp().setRight(getNullNode());

                getNullNode().setLeft(node.getLeft().getUp());
            }

            if (node.getRight() != null)
            {

                node.setRight(node.getRight().getDown());
                node.getRight().setLeft(node);

                node.getRight().getUp().setLeft(getNullNode());

                getNullNode().setRight(node.getRight().getUp());
            }

            if (node.getDown() != null && node.getDown().getDown() != null)//Kyk of node in derde kolom is
            {

                node.setDown(node.getDown().getDown());
                node.getDown().setUp(node);

            }
            if (node.getUp() != null)

            {
                node.getUp().setDown(getNullNode());

                getNullNode().setUp(node.getUp());
            }

            if (node.getUp() == null)
                getNullNode().setUp(null);

            if (getNullNode().getDown() == null)
                node.setDown(null);

            node.setUp(getNullNode());

            getNullNode().setDown(node);

            if (getOrigin().isNullNode())
                setOrigin(node);

            node.setY(node.getY() - 1);

            getNullNode().setY(getNullNode().getY() + 1);

            setActiveState();
            countMoves++;

            addMove(node.getInfo(), "Down");
            setLastMoveVector("down");

            setLastMove(node.getInfo());
            gameStatus();

        }

        catch(Exception ex)
        {

            final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/alert-red.png"));
            JOptionPane.showMessageDialog(null, ex.toString() + "\nPlease contact the developer immediately.", "Error: void moveDown(NumberNode node)", JOptionPane.ERROR_MESSAGE, icon);

        }
    }

    /**

     * Adds the move to the current game history
     * Will only record successful moves

     * Writes the move details to a text file
     * @param number move number

     * @param direction move direction
     */

    private void addMove(int number, String direction)
    {

        gameTracker.printToFile(String.format("Move %d: %d Moved %s\n", getCountMoves(), number, direction));
        gameTracker.printToFile(toString() + "\n");

    }

    /**
     * Saves the current game state if requested

     * Writes the current game state in comma separated value format into a .csv file to be loaded in future
     * Prompts user to enter the directory in which the .csv will be saved

     * @param fileName file name to save game as
     */

    public void saveState(String fileName)
    {

        /*new Thread(new Runnable()
        {

            public void run()
            {*/

                try
                {

                    if(!(fileName.contains(".")) || (!fileName.substring(fileName.lastIndexOf('.'), fileName.length()).equals(".csv")))
                        throw new Exception("Please add the extension \".csv\" after the filename");

                    IO save = new IO(fileName);
                    save.createFile();

                    save.printToFile(toCSV());
                    save.printToFile("\n");

                    save.printToFile(toCSVWin());
                    save.close();

                    final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/info-blue.png"));
                    JOptionPane.showMessageDialog(null, "Game saved at  " + fileName, "Game Save Successful", JOptionPane.INFORMATION_MESSAGE, icon);

                }
                catch(Exception ex)

                {
                    final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/error-red.png"));

                    JOptionPane.showMessageDialog(null, ex.getMessage(), "Error: Game Save Failed", JOptionPane.ERROR_MESSAGE, icon);
                }

            /*}
        }).start();*/

    }

    /**
     * Loads a requested game state from a .csv in comma separated value format

     * Reads the comma separated values from .csv file into current game state
     * Prompts user to enter the directory of the .csv file to load

     * if no state is loaded a randomized state will be generated
     * @param fileName name of file to save game as

     */
    public void loadState(String fileName)

    {
        new Thread(new Runnable()

        {
            public void run()

            {
                try

                {
                    IO io = new IO(fileName);

                    Scanner input = new Scanner(io.getFile());
                    String sNumbers = input.nextLine();

                    String sWinNumbers = input.nextLine();
                    String[] arrsNumbers;

                    String[] arrsWinNumbers;

                    CharSequence cs1 = ", ";
                    CharSequence cs2 = ",";

                    if(sNumbers.contains(cs1))
                        arrsNumbers = sNumbers.split(", ");

                    else if(sNumbers.contains(cs2))
                        arrsNumbers = sNumbers.split(",");

                    else
                        throw new FileSystemException("The format of the .csv file is incorrect\nPlease use as follows:\n1, 2, 3, 4, 5, 6, 7, 8, B\n1,2,3,4,5,6,7,8,B\nWhere B denote the Blank block.");

                    if(sWinNumbers.contains(cs1))

                        arrsWinNumbers = sWinNumbers.split(", ");
                    else if(sWinNumbers.contains(cs2))

                        arrsWinNumbers = sWinNumbers.split(",");
                    else

                        throw new FileSystemException("The format of the .csv file is incorrect\nPlease use as follows:\n1, 2, 3, 4, 5, 6, 7, 8, B\n1,2,3,4,5,6,7,8,B\nWhere B denote the Blank block.");

                    for(int i = 0; i < arrsNumbers.length; i++)
                        if(arrsNumbers[i].equals("B"))

                            arrsNumbers[i] = "0";

                    for(int i = 0; i < arrsWinNumbers.length; i++)
                        if(arrsWinNumbers[i].equals("B"))

                            arrsWinNumbers[i] = "0";

                    int[][] arrIn = new int[getY_axis()][getX_axis()];
                    int iii = 0;

                    for(int i = 0; i < getY_axis(); i++)
                        for(int ii = 0; ii < getX_axis(); ii++)

                            arrIn[i][ii] = Integer.parseInt(arrsNumbers[iii++]);

                    int[][] arrWinIn = new int[getY_axis()][getX_axis()];
                    int iv = 0;

                    for(int i = getY_axis() - 1; i >= 0; i--)
                        for(int ii = 0; ii < getX_axis(); ii++)

                            arrWinIn[i][ii] = Integer.parseInt(arrsWinNumbers[iv++]);

                    input.close();
                    setActiveState(arrIn);

                    setWinState(arrWinIn);
                    final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/info-blue.png"));

                    JOptionPane.showMessageDialog(null, "Game Loaded From " + fileName + "\nEnjoy and don't ever give up on anything.", "Game Load Successful", JOptionPane.INFORMATION_MESSAGE, icon);
                }

                catch(FileNotFoundException fnfex)
                {

                    final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/error-red.png"));
                    JOptionPane.showMessageDialog(null, fnfex.getMessage() + "\nTo load a game state from .csv file, have a input.csv file in the game root directory.", "File Not Found", JOptionPane.ERROR_MESSAGE, icon);

                }
                catch(FileSystemException fsex)

                {
                    final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/error-red.png"));

                    JOptionPane.showMessageDialog(null, fsex.getMessage(), "Incorrect .csv format", JOptionPane.ERROR_MESSAGE, icon);
                }

                catch(Exception ex)
                {

                    final ImageIcon icon = new ImageIcon(this.getClass().getResource("icons/error-red.png"));
                    JOptionPane.showMessageDialog(null, ex.toString() + "\nPlease make sure the .csv file is in the correct format.\nThe file should contain two lines of comma separated values,\none line for the starting game state and a second for the winning state.", "Error: Game Load Failed", JOptionPane.ERROR_MESSAGE, icon);

                }
            }

        }).start();
    }

    /**

     * Function only used by setRandomState function
     * This function should never be used in conjunction with anything else

     * Only the original author determines the usage of this function
     * Include and consult the original author before using this function for optimal software reliability, safety and functionality

     * @param a consult the original author
     * @param b consult the original author

     * @return consult the original author
     */

    private boolean contains(int[][] a, int b)
    {

        for(int i = 0; i < a.length; i++)
            for(int ii : a[i])

                if(ii == b)
                    return true;

        return false;

    }

    /**
     * Act as a destructor for cleaning up possible malfunctions and cleaning up memory after game has terminated

     * @throws Throwable
     */

    @Override
    protected void finalize() throws Throwable

    {
        super.finalize();

        gameTracker.close();
    }
}/**
 * @author Zander Labuschagne 23585137

 * Number Node class to be used as nodes in the puzzle game(Game Engine's Pistons)
 */

public class NumberNode

{
    //Instance Variables

    /**
     * integer containing the value of this node

     */
    private int info;

    /**
     * integer specifying the x coordinate of this node(latitude)

     */
    private int x;

    /**
     * integer specifying the y coordinate of this node(longitude)

     */
    private int y;

    /**
     * Reference to the node left adjacent of this node

     */
    private NumberNode left;

    /**
     * Reference to the node right adjacent to this node

     * can be null
     */

    private NumberNode right;
    /**

     * Reference to the node upwards adjacent to this node
     * can be null

     */
    private NumberNode up;

    /**
     * Reference to the node downwards adjacent to this node

     * can be null
     */

    private NumberNode down;

    /**
     * Default COnstructor

     */
    public NumberNode()

    {
        this(0, 0, 0);

    }

    /**
     * Overloaded Constructor #1

     * @param info value of this node
     * @param x x coordinate of this node(latitude)

     * @param y y coordinate of this node(longitude)
     */

    public NumberNode(int info, int x, int y)
    {

        this(info, x, y, null, null, null, null);
    }

    /**

     * Overloaded Constructor #2
     * @param info value of this node

     * @param x x coordinate of this node(latitude)
     * @param y y coordinate of this node(longitude)

     * @param left Reference to the node left adjacent to this node
     * @param right Reference to the node right adjacent of this node

     * @param up Reference to the node upwards adjacent to this node
     * @param down Reference to the node downwards adjacent to this node

     */
    public NumberNode(int info, int x, int y, NumberNode left, NumberNode right, NumberNode up, NumberNode down)

    {
        setInfo(info);

        setX(x);
        setY(y);

        setLeft(left);
        setRight(right);

        setUp(up);
        setDown(down);

    }

    //Mutators
    /**

     * sets the value of this node
     * @param info value of this node

     */
    public void setInfo(int info)

    {
        this.info = info;

    }

    /**
     * sets the x coordinate of this node

     * latitude
     * @param x x coordinate of this node

     */
    public void setX(int x)

    {
        this.x = x;

    }

    /**
     * sets the y coordinate of this node

     * longitude
     * @param y y coordinate of this node

     */
    public void setY(int y)

    {
        this.y = y;

    }

    /**
     * sets the reference to the node left adjacent to this node

     * @param left reference to the left adjacent node
     */

    public void setLeft(NumberNode left)
    {

        this.left = left;
    }

    /**

     * sets the reference to the node right adjacent to this node
     * @param right reference to the right adjacent node

     */
    public void setRight(NumberNode right)

    {
        this.right = right;

    }

    /**
     * sets the reference to the node upwards adjacent to this node

     * @param up reference to the upwards adjacent node
     */

    public void setUp(NumberNode up)
    {

        this.up = up;
    }

    /**

     * sets the reference to the node downwards adjacent to this node
     * @param down reference to the downwards adjacent node

     */
    public void setDown(NumberNode down)

    {
        this.down = down;

    }

    //Accessors
    /**

     * get the value of this node
     * @return integer value of this node

     */
    public int getInfo()

    {
        return info;

    }

    /**
     * get the x coordinate of this node

     * latitude
     * @return x coordinate of this node

     */
    public int getX()

    {
        return x;

    }

    /**
     * get the y coordinate of this node

     * longitude
     * @return y coordinate of this node

     */
    public int getY()

    {
        return y;

    }

    /**
     * get the reference to the node left adjacent to this node

     * @return reference to the node left adjacent to this node
     */

    public NumberNode getLeft()
    {

        return left;
    }

    /**

     * get the reference to the node right adjacent to this node
     * @return reference to the node right adjacent to this node

     */
    public NumberNode getRight()

    {
        return right;

    }

    /**
     * get the reference to the node upwards adjacent to this node

     * @return reference to the node upwards adjacent to this node
     */

    public NumberNode getUp()
    {

        return up;
    }

    /**

     * get the reference to the node downwards adjacent to this node
     * @return reference to the node downwards adjacent to this node

     */
    public NumberNode getDown()

    {
        return down;

    }

    /**
     * Determine if this node is the origin node

     * @return true if this node is the origin node
     */

    public boolean isOrigin()
    {

        return (x == 0 && y == 0);
    }

    /**

     * Determine if this node is the null node
     * null node is the node occupying the empty block

     * @return true if this node is the null node
     */

    public boolean isNullNode()
    {

        return getInfo() == 0;
    }

}
```