Boss Puzzle (Nine puzzle)

Report (ITRW 317)

By:

Anneke Lourens

24087122

Due:

18 March 2016

index

1 Introduction

This report is about the Nine-Puzzle. The document will give a little bit of background information and also where did the nine puzzle start. This report will also include how to play the game. A step by step guide is included that tell you how to set up the puzzle and how to run the Nine-puzzle game that is included in the folder.

2 Literature Study

2.1 Background and History

The objective of the 9-puzzle is to move the tiles in the given 3X3 board to make a picture complete or to give the solution (Reinefeld).

3 User Guide

3.1 How to make a .csv file

Step 1:
Open the file named "waarde_puzzle.csv" in Notepad++ or Notepad. Remember that the file must have "," in between the numbers and there must only be 8 numbers and a lowercase b where the blank space in the puzzle must be (Figure 1). Figure 2 is an example of what the .csv file looks in notepad++ or Notepad. The first row is the puzzle that is presented to the player and the second row is the solved puzzle. If the player has saved its puzzle to continue later there will be a third row in the .csv file that shows the moves that the player has made already.

3.2 How to begin the Nine-puzzle

Step 1:
If you have your own .csv file - right click on the execute.bat file and open it with notepad. Change the filename of the .csv file. Save the file and exit it. Double-click on the execute.bat file to run the program. Figure 1
Step 2:
If the program has run you will see Figure 3
Step 3:
Enter a "y" for you are a new player or "n" for you are not a new player Figure 4
Step 4:
After step 3 is completed the begin puzzle will be given. Figure 5
Step 5:
Enter the number that you want to move like in Figure 6. The game will move that number to the blank space(b) and give the updated puzzle again.

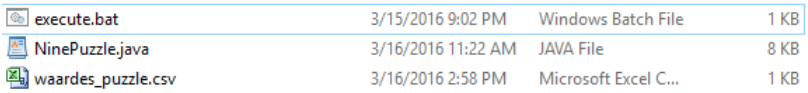


Figure 1: This is the files that one get which will run the nine puzzle

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

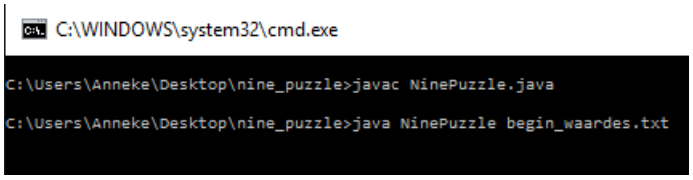
Figure 2: How the data looks in the .csv file. This data will be used in the nine puzzle.

3.3 Functionality

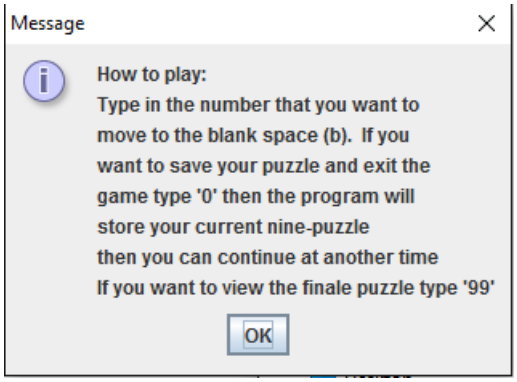
How to stop, save and exit the program:
If you want to stop playing and play again later then you can type ”0” then the puzzle will be saved and the game will exit Figure 7.

If you have won the game, the program will save your work and show a message to congratulate you and show how many moves you have made to solve the puzzle Figure 8.

If you want to view the finale puzzle you type in 99 Figure 9.



(a) If you have run the execute.bat file this lines will show in the CMD



(b) The program will show this message directly after you have run the execute.bat file

Figure 3

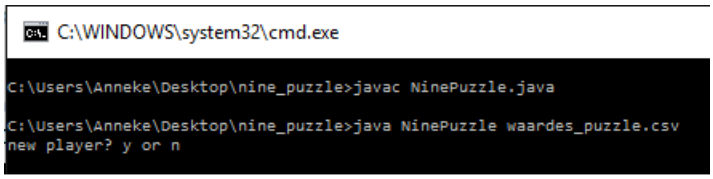


Figure 4: You will be asked if you are a new player or not

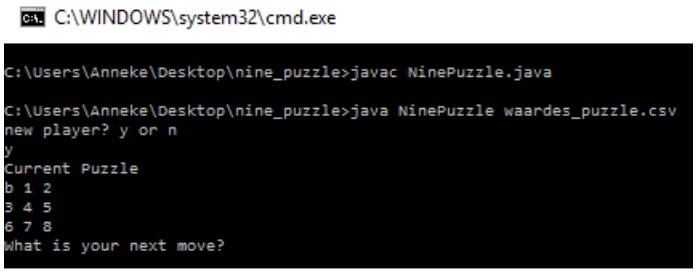


Figure 5: After specifying that you are a new player or not the program will give the begin puzzle

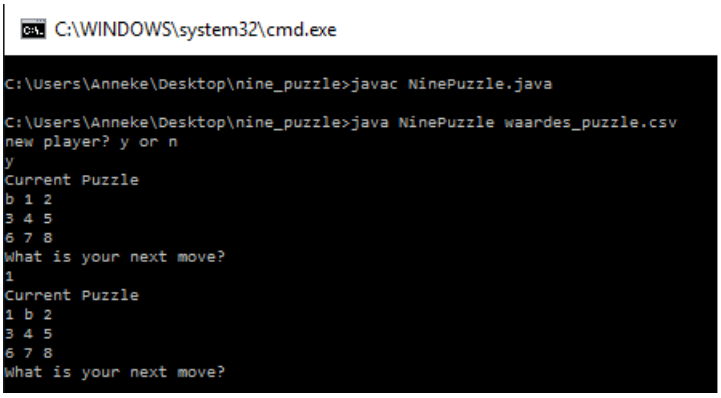
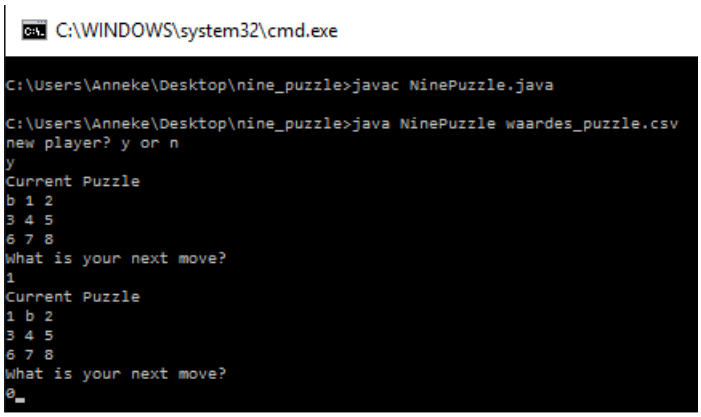
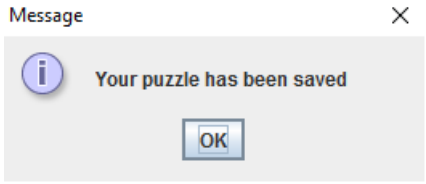


Figure 6: After a move is made



(a) If you type in 0



(b) The message will show that your puzzle has been saved

Figure 7

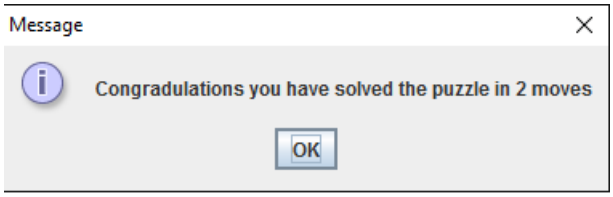


Figure 8: The congratulation message

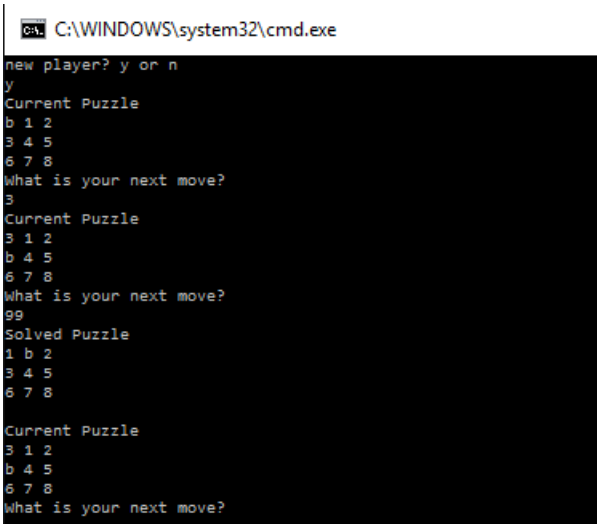


Figure 9: Solved puzzle is shown in the program

4 Code

```
import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import javax.swing.JOptionPane;

public class NinePuzzle {
  public static void main(String[] args) throws FileNotFoundException {
    // scanner to read out of txt file
    Scanner input = new Scanner(args[0]);
    // gebruiker se skuiwe
    Scanner gebruiker = new Scanner(System.in);
    boolean solved = false;
    boolean save = false;
    int count = 0;
    // create arrays
    int[] current = new int[9];
    int[] finaal = new int[9];
    // array for the string read in csv file
    String[] userInputs = new String[1];
    String filename = input.nextLine();
    // call method that reads the csv file
    setUpArray(current, finaal, userInputs, filename);
    // display how the game is played
    String message = "How to play: \n" +
                         "Type in the number that you want to \n" +
                     "move to the blank space (b).  If you \n"+
                                    "want to save your puzzle and exit the \n"+
                                    "game type '0' then the program will \n"+
                                    "store your current nine-puzzle \n" +
                     "then you can continue at another time \n"+
                                    "If you want to view the finale puzzle type '99'";
    JOptionPane.showMessageDialog(null, message);
        System.out.println("new player? y or n");
        String newUser = gebruiker.nextLine();
        String yn = newUser.substring(0, 1);
        if (yn.equals("y") ||  yn.equals("Y")) {
          userInputs[0] = "";
      writeArrayToFile(current, finaal, userInputs[0], filename);
        }
        count = userInputs.length;
    while ( !solved && !save) {
      System.out.println("Current Puzzle");
      puzzle(current); // print current puzzle
      int lees = 0;
      // ask user what his next move is going to be
      System.out.println("What is your next move?");
      lees = gebruiker.nextInt();
      // test if the input from user is 0 or any other number between 1 - 8
      if ( lees != 0 ) {
        if(lees == 99) {
          System.out.println("Solved Puzzle");
          puzzle(finaal);
          System.out.println("");
          } else {
          // move the index of the tiles
            if (move_index(lees, current)) {
              count++;
              solved = compare_solution(finaal, current);
              userInputs[0] += Integer.toString(lees) + ",";
                      writeArrayToFile(current,finaal,userInputs[0],filename);
                    }
          }
      } else { //if the user enters 0
        //write current puzzle to csv file
        writeArrayToFile(current, finaal, userInputs[0], filename);
        //stoor na txt file as begin waardes
        save = true;
      }//end if
      //show the final puzzle to the user
    }// end while
    // show message that say he has solved  the puzzle
    if ( solved ) {
                count = count-1;
      String message1 ="Congradulations you have solved the puzzle in" +" "+count +" "+ "moves";
      JOptionPane.showMessageDialog(null, message1);
    }
        if(save){
          String message2 = "Your puzzle has been saved";
      JOptionPane.showMessageDialog(null, message2);
        }
}//end main

  public static void writeArrayToFile(int [] current,
                                      int [] finaal,
                                     String userinputs,
                                     String filename)
                  throws FileNotFoundException {
```

```
  try {
    PrintWriter outputStream = new PrintWriter(filename);
    for ( int k = 0; k < 9; k++ ) {
      if ( k != 8 ) {
     // write the current puzzle to file
        if(current[k] == 0) {
          outputStream.print("b,");
        } else {
          outputStream.print(current[k] + ",");
        }
      } else {
          outputStream.print(current[k] + "\r\n");
      }
    }
  for(int h = 0; h < 9; h++ ) {
    if (h != 8) {
      if (finaal[h] == 0) {
        outputStream.print("b,");
      } else {
          outputStream.print(finaal[h] + ",");
      }
    } else {
      outputStream.print(finaal[h] + "\r\n");
    }//end else
  }
  if (userinputs.length() > 0) {
    String user_inputs = userinputs.substring(0, userinputs.length() - 1) + "\r\n";
    outputStream.println(user_inputs);
  }
  outputStream.close();
  } catch (FileNotFoundException e) {
      e.printStackTrace();
  }//end catch
}
//read csv file into an array
public static void setUpArray(int [] current, int [] finaal,
                              String [] userinputs, String filename)
                                                                  throws FileNotFoundException {
  File File = new File(filename);
  try {
    Scanner inputStream = new Scanner(File);
    int lines_read = 0;
    while (inputStream.hasNext() ) {
      String data = inputStream.next();//gets a whole line
      String [] values = data.split(",");
      if ( lines_read == 0 ) {
        for ( int v = 0; v < 9; v++ ) {
          if (values[v].equals("b")) {
            current[v] = 0;
          } else {
            current[v] = Integer.parseInt(values[v]);
          }
        }
      } else if ( lines_read == 1 ) {
                for ( int w = 0; w < 9; w++ ) {
                  if (values[w].equals("b")) {
                    finaal[w] = 0;
                  } else {
                      finaal[w] = Integer.parseInt(values[w]);
                  }
                }
      } else if ( lines_read == 2 ) {
                      if (!data.equals(null) || !data.equals("")) {
                  userinputs[0] = data + ",";
                      } else {
                          userinputs[0] = "";
                      }
      }
    lines_read++;
  }
    inputStream.close();
  }
  catch(FileNotFoundException e) {
    e.printStackTrace();
  }
}//end setuparray
//print puzzle
public static int puzzle(int [] current) {
  for(int k = 0; k < 9; k++) {
    if (current[k] == 0) {
      System.out.print("b ");
    } else {
        System.out.print(current[k] +" ");
    }
    if ( ((k + 1) % 3) == 0 ) {
      System.out.print("\n");
    }
  }//end for
  return 0;
}//end puzzle

//swap the index of the tile that has been moved
public static boolean move_index(int lees, int [] current) {
```

```
    int[][] lookup = {{1, 3, -1, -1},
                      {0, 2, 4, -1},
                      {1, 5, -1, -1},
                      {0, 4, 6, -1},
                      {1, 3, 5, 7},
                      {2, 4, 8, -1},
                      {3, 7, -1, -1},
                      {4, 6, 8, -1},
                      {5, 7, -1, -1}};
    int zero_position = get_index(0, current);
    int value_position = get_index(lees, current);
    for ( int i = 0; i < 4; i++) {
      if ( lookup[zero_position][i] == value_position ) {
        current[value_position] = 0;
        current[zero_position] = lees;
    return true;
      }
    }
  return false;
  }//end move_index
  //get the index of the blank space and the number that the user want to move
  public static int get_index(int lees, int [] current) {
    for ( int i = 0; i < 9; i++ ) {
      if ( lees == current[i] ) {
        return i;
      }
    }
  return 0;
  }
  //compare wether the current puzzle has the same values of the final puzzle
  public static boolean compare_solution(int [] finaal, int [] current) {
    for ( int i = 0; i < 9; i++) {
      if ( finaal[i] != current[i] ) {
        return false;
      }
    }
    return true;
  }//end compare
}//end class
```

5 Bibliography

Reinefeld, A.Complete Solution of the Eight-Puzzle and the Bene t of Node Ordering in IDA*. `http://www.laborion.com/Portals/0/Ai/Heuristicssearch/93ijcai.pdf` Date of access: 8 Maart 2016.