Knight's Tour Application

PROG 37721 ASSIGNMENT 1 SUMMER 2019

2019 June 15

Seunghyun Ban 991 479 657

Knight's Tour Application

Table Of Contents

1. Cell.cs

```
Form1.cs    NonIntellegentMode.cs    HeuristicMode.cs    Program.cs    SeunghyunBanNon
Assignment1                                             Assignment1.Cell

1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace Assignment1
8   {
        65 references
9       class Cell
10      {
            12 references
11          public int X { get; set; }
            12 references
12          public int Y { get; set; }
            9 references
13          public int value { get; set; }
14
            0 references
15          public Cell()
16          {
17          }
            18 references
18          public Cell(int x, int y)
19          {
20              this.X = x;
21              this.Y = y;
22          }
            25 references
23          public Cell(int x, int y, int value)
24          {
25              this.X = x;
26              this.Y = y;
27              this.value = value;
28          }
29
30      }
31  }
32
```

This class includes Property of int X, Y and value, also 3 constructors.

X indicates a specific row number in chess board.

Y indicates a specific column number in chess board.

value is a value of a specific cell, this value will help to decide where knight should go.

2. PlayMode.cs

```
Form1.cs    NonIntellegentMode.cs    HeuristicMode.cs    Program.cs
Assignment1                                             Assignmer
```

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Assignment1
{
    2 references
    interface PlayMode
    {
        4 references
        Cell[,] Initialize();
        4 references
        ArrayList PossibleRoutes(int x, int y);
        18 references
        bool checkCellValid(int x, int y);
    }
}
```

This class is interface which parent class of **NonIntelligentMode** and **HeuristicMode.**

NonintelligentMode class and HeuristicMode class both sharing

1. Initialize() function that returns Cell[,] multidimension array that contain value of whole chess board.
2. PossibeRoutes(x, y) function that check the possible route of value of row and column and return ArrayList that contain possible route of x and y position
3. checkCellValid(x, y) function check the next cell is valid or invalid for knight's next move. It will return true and false.

3. NonIntelligentMethod.cs

Form1.cs | NonIntellegentMode.cs* | HeuristicMode.cs | Cell.cs

Assignment1 | Assignment1.NonIntellegentM | PossibleRoutes(int x, int y)

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Assignment1
{
    2 references
    class NonIntellegentMode:PlayMode
    {
        private Cell[,] board_nonIN = new Cell[8, 8];

        //Initiallize board
        4 references
        public Cell[,] Initialize()
        {
            for (int x = 0; x < 8; x++)
            {
                for (int y = 0; y < 8; y++)
                {
                    board_nonIN[x, y] = new Cell(x, y, 0);
                }
            }
            return board_nonIN;
        }
        //check possible route and return as arraylist
        4 references
        public ArrayList PossibleRoutes(int x, int y)
        {
            ArrayList list = new ArrayList();

            int X = x + 2;
            int Y = y + 1;
            if (checkCellValid(X, Y) == true)
                list.Add(new Cell(X, Y));

            X = x + 2;
            Y = y - 1;
            if (checkCellValid(X, Y) == true)
                list.Add(new Cell(X, Y));

            X = x - 2;
            Y = y + 1;
            if (checkCellValid(X, Y) == true)
                list.Add(new Cell(X, Y));

            X = x - 2;
            Y = y - 1;
            if (checkCellValid(X, Y) == true)
                list.Add(new Cell(X, Y));
```

100 %

```
            X = x + 1;
            Y = y + 2;
            if (checkCellValid(X, Y) == true)
                list.Add(new Cell(X, Y));

            X = x + 1;
            Y = y - 2;
            if (checkCellValid(X, Y) == true)
                list.Add(new Cell(X, Y));

            X = x - 1;
            Y = y + 2;
            if (checkCellValid(X, Y) == true)
                list.Add(new Cell(X, Y));


            X = x - 1;
            Y = y - 2;
            if (checkCellValid(X, Y) == true)
                list.Add(new Cell(X, Y));


            return list;

        }

        18 references
        public bool checkCellValid(int x, int y)
        {
            if (x >= 0 && x <= 7 && y >= 0 && y <= 7 && board_nonIN[x, y].value == 0)
            {
                return true;
            }
            else
                return false;
        }
    }
}
```

NonintelligentMode class is child class of **PlayMode**. It will inherit Initialize, PossibleRoute and checkCellValid function.

4. HeuristicMode.cs

Form1.cs | NonIntellegentMode.cs* | HeuristicMode.cs* | Cell.cs | Program.cs | SeunghyunBa

Assignment1 | Assignment1.Heuristic

```
using System.Text;
using System.Threading.Tasks;

namespace Assignment1
{
    2 references
    class HeuristicMode:PlayMode
    {

        private Cell[,] board_IN = new Cell[8, 8];

        //initialize borad with Heuristic value + 100
        //add 100, It will help to divide passed cell and unpassed cell
        4 references
        public Cell[,] Initialize()
        {
            for (int x = 0; x < 8; x++)
            {
                if (x == 0 || x == 7)
                {
                    board_IN[x, 0] = new Cell(x, 0, 102);
                    board_IN[x, 1] = new Cell(x, 1, 103);
                    board_IN[x, 2] = new Cell(x, 2, 104);
                    board_IN[x, 3] = new Cell(x, 3, 104);
                    board_IN[x, 4] = new Cell(x, 4, 104);
                    board_IN[x, 5] = new Cell(x, 5, 104);
                    board_IN[x, 6] = new Cell(x, 6, 103);
                    board_IN[x, 7] = new Cell(x, 7, 102);

                }
                else if (x == 1 || x == 6)
                {
                    board_IN[x, 0] = new Cell(x, 0, 103);
                    board_IN[x, 1] = new Cell(x, 1, 104);
                    board_IN[x, 2] = new Cell(x, 2, 106);
                    board_IN[x, 3] = new Cell(x, 3, 106);
                    board_IN[x, 4] = new Cell(x, 4, 106);
                    board_IN[x, 5] = new Cell(x, 5, 106);
                    board_IN[x, 6] = new Cell(x, 6, 104);
                    board_IN[x, 7] = new Cell(x, 7, 103);

                }
                else
                {
                    board_IN[x, 0] = new Cell(x, 0, 104);
                    board_IN[x, 1] = new Cell(x, 1, 106);
                    board_IN[x, 2] = new Cell(x, 2, 108);
                    board_IN[x, 3] = new Cell(x, 3, 108);
                    board_IN[x, 4] = new Cell(x, 4, 108);
                    board_IN[x, 5] = new Cell(x, 5, 108);
                    board_IN[x, 6] = new Cell(x, 6, 106);
                    board_IN[x, 7] = new Cell(x, 7, 104);
```

100 %

Form1.cs | NonIntellegentMode.cs* | HeuristicMode.cs* | Cell.cs | Program.cs | Seunghyu

Assignment1 | Assignment1.Heur

```
                }

            }
            return board_IN;
        }
        4 references
        public ArrayList PossibleRoutes(int x, int y)
        {
            //list for possible route
            ArrayList list = new ArrayList();

            //filtered list for intelligent way
            ArrayList filteredList = new ArrayList();

            int X = x + 2;
            int Y = y + 1;
            if (checkCellValid(X, Y) == true)
                list.Add(new Cell(X, Y));

            X = x + 2;
            Y = y - 1;
            if (checkCellValid(X, Y) == true)
                list.Add(new Cell(X, Y));

            X = x - 2;
            Y = y + 1;
            if (checkCellValid(X, Y) == true)
                list.Add(new Cell(X, Y));

            X = x - 2;
            Y = y - 1;
            if (checkCellValid(X, Y) == true)
                list.Add(new Cell(X, Y));

            X = x + 1;
            Y = y + 2;
            if (checkCellValid(X, Y) == true)
                list.Add(new Cell(X, Y));

            X = x + 1;
            Y = y - 2;
            if (checkCellValid(X, Y) == true)
                list.Add(new Cell(X, Y));

            X = x - 1;
            Y = y + 2;
            if (checkCellValid(X, Y) == true)
                list.Add(new Cell(X, Y));

            X = x - 1;
            Y = y - 2;
```

100 %

```
                X = x - 1;
                Y = y - 2;
                if (checkCellValid(X, Y) == true)
                    list.Add(new Cell(X, Y));

                //find the lowest value of cell. It will be next cell
                int low = 1000;
                for (int i = 0; i < list.Count; i++)
                {
                    Cell c = (Cell)list[i];

                    int candidate = board_IN[c.X, c.Y].value;

                    if (candidate < low)
                        low = candidate;
                }

                //match the lowest value with cell position and return filtered list
                for (int i = 0; i < list.Count; i++)
                {
                    Cell c1 = (Cell)list[i];


                    if (low == board_IN[c1.X, c1.Y].value)
                        filteredList.Add(new Cell(c1.X, c1.Y));
                }
                return filteredList;
            }

            //check next cell is valid or not
            18 references
            public bool checkCellValid(int x, int y)
            {
                if (x >= 0 && x <= 7 && y >= 0 && y <= 7 && board_IN[x, y].value > 100)
                {
                    return true;
                }
                else
                    return false;
            }
        }
    }

```

HeuristicMode class is child class of **PlayMode**. It will inherit Initialize, PossibleRoute and checkCellValid function.

5.Form1.cs

Form1.cs* | NonIntellegentMode.cs* | HeuristicMode.cs* | Cell.cs | Program.cs | SeunghyunBanNonIntel

Assignment1 | Assignment1.Form1

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

/*
 * File:   Form1.cs
 * Author: Seunghyun Ban
 * Created on May 13, 2019, 12:31 PM
 */

namespace Assignment1
{
    3 references
    public partial class Form1 : Form
    {
        static string path;
        //initial cell does not cahnge untill we click another position
        Cell initialCell;
        Cell newCell;

        1 reference
        public Form1()
        {
            InitializeComponent();
            button1.Enabled = false;
            button2.Enabled = false;
            button3.Enabled = false;
        }
        1 reference
```

Form1 class inherits Form class. And design window application form.

```
1 reference
private void tableLayoutPanel1_MouseClick(object sender, MouseEventArgs e)
{
    tableLayoutPanel1.Controls.Clear();
    button1.Enabled = true;
    button2.Enabled = true;

    //get position of cell that user click on the table
    int x = 0;
    int verticalOffset = 0;
    foreach (int h in tableLayoutPanel1.GetRowHeights())
    {
        int y = 0;
        int horizontalOffset = 0;
        foreach (int w in tableLayoutPanel1.GetColumnWidths())
        {
            Rectangle rectangle = new Rectangle(horizontalOffset, verticalOffset, w, h);
            if (rectangle.Contains(e.Location))
            {
                MessageBox.Show ("You select" + " ("+ x +", "+ y +") " + "!! Let's start game!" );

                initialCell = new Cell(x, y);
            }
            horizontalOffset += w;
            y++;
        }
        verticalOffset += h;
        x++;
    }
}
//button for nonInterlligent method
```

When we mouse click on the tablelayoutpanel (chess board), It will give a clicked position of board cell.

And the position data will be saved in **initialCell** object that never change until user click another cell.

Button1 for NonIntelligent Mode

Form1.cs* | NonIntellegentMode.cs* | HeuristicMode.cs* | Cell.cs | Program.cs

Assignment1 | Assignment1.Form1 | tableLayoutPanel1_MouseClick(obje

```
        }
        //button for nonInterlligent method
        1 reference
        private void button1_Click(object sender, EventArgs e)
        {
            path = @"..\..\SeunghyunBanNonIntelligentMethod.txt";
            File.WriteAllText(path, "");

            int time = Int32.Parse(textBox1.Text);

            int valueOnTable = 0;
            while (time > 0) {
                int count = 0;

                Cell[,] board = new Cell[8, 8];
                NonIntellegentMode nonIN = new NonIntellegentMode();

                board = nonIN.Initialize();
                if (count == 0)
                    newCell = initialCell;

                board[newCell.X, newCell.Y].value = ++count;
                tableLayoutPanel1.Controls.Clear();

                displayValue(count);
                ArrayList list;

                do
                {
                    list = nonIN.PossibleRoutes(newCell.X, newCell.Y);
                    Random r = new Random();
                    int way = r.Next(list.Count);
                    if (list.Count > 0)
                    {
                        //Console.WriteLine("Random: " + a);
                        newCell = (Cell)list[way];
                        board[newCell.X, newCell.Y].value = ++count;
                        if (time == 1)
                        {
                            displayValue(count);
                        }
                    }
                    else
                        break;
                } while (list.Count > 0);
                valueOnTable++;
                writeTotxt(path, valueOnTable, count);
                time--;
            }
            button3.Enabled = true;
        }
```

Logic of NonIntelligent Mode

1. Get value of "time" from textbox
2. Get value of position x, y by clicking chess board
3. Save position data in Cell class
4. Initialize chess board and store in "board" multidimension array
5. Create new object of **nonIntelligentMode** class
6. **PossibleRoutes** method will return arraylist of available route "list"
   - **CheckCellValid** will validate available position
   - All of the cell value is zero, but when knight move to cell, that cell value will be changed. So, the function will check if the cell value is 0 or not, and position is outside of board or not
7. Display data by using label on the chess board
8. Write result in txt file.

Button2 for Heuristic Mode

```
115     private void button2_Click(object sender, EventArgs e)
116     {
117         path = @"..\..\SeunghyunBanHeuristicsMethod.txt";
118         File.WriteAllText(path, "");
119         int time = Int32.Parse(textBox1.Text);
120         int valueOnTable = 0;
121         while (time > 0)
122         {
123             int count = 0;
124
125             Cell[,] board = new Cell[8, 8];
126             HeuristicMode IN = new HeuristicMode();
127
128             board = IN.Initialize();
129             if (count == 0)
130                 newCell = initialCell;
131
132             board[newCell.X, newCell.Y].value = ++count;
133
134             tableLayoutPanel1.Controls.Clear();
135             displayValue(count);
136
137             ArrayList list;
138             do
139             {
140                 list = IN.PossibleRoutes(newCell.X, newCell.Y);
141                 Random r = new Random();
142                 int way = r.Next(list.Count);
143
144                 if (list.Count > 0)
145                 {
146                     //Console.WriteLine("Random: " + a);
147                     newCell = (Cell)list[way];
148                     board[newCell.X, newCell.Y].value = ++count;
149                     if (time == 1)
150                     {
151                         displayValue(count);
152                     }
153                 }
154                 else
155                     break;
156
157             } while (list.Count > 0);
158             valueOnTable++;
159             writeTotxt(path, valueOnTable, count);
160             time--;
161         }
162         //string textFile = ""
163         button3.Enabled = true;
164     }
165
        4 references
```

Logic of Heuristic Mode

1. Get value of "time" from textbox
2. Get value of position x, y by clicking chess board
3. Save position data in Cell class
4. Initialize chess board and store in "board" multidimension array
5. Create new object of **HeuristicMode** class
6. **PossibleRoutes** method will return arraylist of available route "list"
   - Cell value follow the Heuristic way, initialized manually.
   - **CheckCellValid** will validate available position. I added 100 to every cell so I can check the valid cell by checking cell is greater than 100 or not and position is out of table or not.
   - After make possible route arraylist "list", It will be filtered the value by using heuristic way -Find a lowest value of cell
   - It will make arraylist of "filteredList" after filtering list of available routes from "list".
7. Display data by using label on the chess board
8. Write result in txt file.

```
4 references
public void displayValue(int count)
{
    Label tour_number = new Label();

    if (count.GetType().Equals(typeof(int)))
        tour_number.Text = count.ToString();
    else
        MessageBox.Show("Please enter Integer");

    tour_number.AutoSize = true;
    tour_number.BackColor = Color.Transparent;
    tour_number.Anchor = AnchorStyles.None;
    tableLayoutPanel1.Controls.Add(tour_number, newCell.Y, newCell.X);
}

2 references
public void writeTotxt(string path, int count, int cou)
{
    if (File.Exists(path))
    {
        using (StreamWriter sw = File.AppendText(path))
        {
            sw.WriteLine("Trial {0}: The knight was able to successfully touch {1} squares.", count, cou);
        }
    }
    else
        MessageBox.Show("File does not exit", "File IO error", MessageBoxButtons.OK, MessageBoxIcon.Error);
}

1 reference
private void button3_Click(object sender, EventArgs e)
{
    Process.Start("notepad.exe", path);
}

1 reference
private void tableLayoutPanel1_CellPaint_1(object sender, TableLayoutCellPaintEventArgs e)
{
    if ((e.Column + e.Row) % 2 == 1)
        e.Graphics.FillRectangle(Brushes.Beige, e.CellBounds);
    else
        e.Graphics.FillRectangle(Brushes.White, e.CellBounds);
}
}
}
```

displayValue function will display value of cell on the chess board**.**

wirteTotxt function write a result in the specific txt file.

Button3_click is button open the result txt file for user