**Peer Evaluation for Lab 3 – Memory**

|  |  |
| --- | --- |
| Your name: (Your lab is the one being evaluated) | Amanda Akins |
| Name(s) of peer evaluator(s) | Chris Henry |
| Date: | 5/2/16 |

Instructions  
You should have already completed Lab 3. After you and a peer have evaluated your work, you will submit this evaluation along with screen shots and source code indicated in moodle. You may make corrections to your work as a result of the evaluation.

|  |  |
| --- | --- |
| ***Concentration Problem*** | |
| Does the application use any form level variables? What are they? Is it reasonable to use an instance variable for each of these? | Yes |
| Application includes definitions for appropriate set of methods? What are they? Does each pass parameters and return values appropriately? **EXTRA CREDIT: Do the methods that manipulate the array of strings pass the array in as a parameter rather than using the instance variable?** | Yes |
| Application includes definitions for appropriate set of event handlers? What are they? Does the application function appropriately? Can you interact with the application in a way that causes it to “degrade” and not gracefully? | Yes |

|  |  |
| --- | --- |
| ***Programming style for all programs*** | |
| Is proper indentation used? Is each event handler indented properly? | Yes |
| Are comments used appropriately? | Yes |
| Do variable names use camel case? (camelCase for example) | Yes |
| Do method names use Title Case (or Pascal Case?) | Yes |

General comments and notes:

Screen Shots and Source Code

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace Concentration

{

public partial class boardForm : Form

{

public boardForm()

{

InitializeComponent();

}

#region Instance Variables

//\* You'll need an array of strings named cards, 2 indexes and the number of matches

//\* My array is set up to hold 21 strings and I ignore the 0 element because I named the picture boxes

//\* starting at 1 rather than 0

string[] cards = new string[21];

int matches = 0;

int firstPick = NOT\_PICKED\_YET, secondPick = NOT\_PICKED\_YET; // stores first and second indexes

const int NOT\_PICKED\_YET = -1;

// this is for the timer that displays the elapsed time. I included it for illustration purposed only.

private int seconds = 0;

#endregion

#region Methods

//\* Write some methods

// you may need more methods but here's the list that I used

// this should fill the cards (array of strings) with card file names

private void FillCards()

{

string[] values = { "a", "5", "j", "q", "k" };

string[] suits = { "c", "d", "h", "s" };

int index = 1;

foreach (string value in values)

{

foreach (string suit in suits)

{

// \*Add a string for the filename of the card

// to the array. The first card, for example, should be

// cards[1] = "cardac.jpg";

cards[index] = "card" + value + suit + ".jpg";

index++;

}

}

}

//\* determines if the 2 cards are a match. The "value" of the card is in the filename at the 5th char.

private bool IsMatch(int index1, int index2)

{

if (cards[index1].Substring(4, 1) == cards[index2].Substring(4, 1))

return true;

else

return false;

}

//\* Shuffles the strings in the cards array. I'll give you an algorithm in class for this.

private void ShuffleCards()

{

Random randomNumberGenerator = new Random();

for (int index = 1; index <= 20; index++)

{

int rndIndex = randomNumberGenerator.Next(1, 21);

string temp = cards[index];

cards[index] = cards[rndIndex];

cards[rndIndex] = temp;

}

}

// loads the specified card picture box with the filename from the array

private void LoadCard(int i)

{

PictureBox card = (PictureBox)this.Controls["card" + i];

card.Image = Image.FromFile(System.Environment.CurrentDirectory + "\\Cards\\" + cards[i]);

}

//\* loads an image for the back of a card in the specified card

private void LoadCardBack(int i)

{

PictureBox card = (PictureBox)this.Controls["card" + i];

card.Image = Image.FromFile(System.Environment.CurrentDirectory + "\\Cards\\" + "black\_back.jpg");

}

//\* disables and hides the specified card - set visible property to false

private void HideCard(int i)

{

PictureBox card = (PictureBox)this.Controls["card" + i];

card.Visible = false;

}

//\* disables and hides all cards. Call HideCard in a loop.

private void HideAllCards()

{

for (int i = 1; i <= 20; i++)

{

HideCard(i);

}

}

//\* disables the specified card

private void DisableCard(int i)

{

PictureBox card = (PictureBox)this.Controls["card" + i];

card.Enabled = false;

}

//\* disables all cards. Call DisableCard in a loop.

private void DisableAllCards()

{

for (int i = 1; i <= 20; i++)

{

DisableCard(i);

}

}

private void ShowCard(int i)

{

PictureBox card = (PictureBox)this.Controls["card" + i];

card.Visible = true;

}

//\* enables and shows all of the cards, make picture box

private void ShowAllCards()

{

for (int i = 1; i <= 20; i++)

{

ShowCard(i);

}

}

//\* enables the cards that are still visible on the board

private void EnableAllVisibleCards()

{

for (int i = 1; i <= 20; i++)

{

PictureBox card = (PictureBox)this.Controls["card" + i];

if (card.Visible == true)

{

card.Enabled = true;

}

else

{

card.Enabled = false;

}

}

}

#endregion

//\* finish this

private void frmBoard\_Load(object sender, EventArgs e)

{

gameTimer.Enabled = true;

lblElapsedTime.Text = seconds.ToString();

// fill the cards array - you have a method

FillCards();

// shuffle the cards.

ShuffleCards();

// show the back of all cards - you have a method

for (int i = 1; i <= 20; i++)

{

LoadCardBack(i);

}

}

// This is only for illustration purposes

private void gameTimer\_Tick(object sender, EventArgs e)

{

seconds++;

lblElapsedTime.Text = seconds.ToString();

}

//\* Finish this. See pseudocode below.

private void card\_Click(object sender, EventArgs e)

{

PictureBox card = (PictureBox)sender;

int cardIndex = int.Parse(card.Name.Substring(4));

/\* if the first card is not picked yet

\* save the index

\* load the card image

\* disable the card so the user can't click the same card twice

\* else (the first card has been picked so this is the second card)

\* save the index of the second card

\* load the card image

\* disable all the cards

\* start the flip timer - flipTimer.Enabled = true;

\* end if

\*/

if (firstPick == NOT\_PICKED\_YET)

{

firstPick = cardIndex;

LoadCard(firstPick);

DisableCard(firstPick);

}

else

{

secondPick = cardIndex;

LoadCard(secondPick);

DisableAllCards();

flipTimer.Enabled = true;

}

}

//\* ToDo: Finish this. See pseudocode below.

private void flipTimer\_Tick(object sender, EventArgs e)

{

/\* stop the timer

\* if there's a match

\* hide the first card clicked

\* hide the second card clicked

\* reset both indexes

\* increment the number of matches

\* if the game is over

\* if the user wants to play again

\* shuffle the cards - ShuffleCards();

\* show the card backs and enable all of the cards - LoadCardBack();

\* else

\* exit the application?

\* end if

\* else

\* enable all of the cards left on the board - EnableAllCards();

\* end if

\* else

\* show the back of the first card clicked

\* show the back of the second card clicked

\* reset both indexes

\* enable all of the cards left on the board

\* end if

\*/

flipTimer.Enabled = false;

if (IsMatch(firstPick, secondPick))

{

HideCard(firstPick);

HideCard(secondPick);

firstPick = NOT\_PICKED\_YET;

secondPick = NOT\_PICKED\_YET;

matches++;

if (matches == 10)

{

MessageBox.Show("WOOHOO!");

}

}

else

{

LoadCardBack(firstPick);

LoadCardBack(secondPick);

firstPick = NOT\_PICKED\_YET;

secondPick = NOT\_PICKED\_YET;

}

EnableAllVisibleCards();

}

private void newGameButton\_Click(object sender, EventArgs e)

{

EnableAllVisibleCards();

}

}

}



