|  |
| --- |
| Close-up image showing the leaf-sides of two oversized books side-by-side on a bookshelf, with additional books in soft focus background |
| Technical Documentation  [M4] Task 2 |
| |  |  |  | | --- | --- | --- | | Kasim Hussain 104016 |  |  | |

Contents

[Introduction 2](#_Toc11179158)

[User requirements 2](#_Toc11179159)

[Program specification 2](#_Toc11179160)

[Program design 2](#_Toc11179161)

[Screen components and their properties 2](#_Toc11179162)

[Class Properties and methods 4](#_Toc11179163)

[Event procedures and description 4](#_Toc11179164)

[Test plan and results 4](#_Toc11179165)

[Annotated code printouts 6](#_Toc11179166)

[Known issues 14](#_Toc11179167)

# Introduction

This document will explain program design, how it was tested and how the program works. This will help with tracking bugs and further modification of the program.

## User requirements

The program was originally intended to be able to run a game using rapid development application with little error. Additionally, another requirement was to use classes and implement objects. The application had to be designed built and tested with documentation.

## Program specification

The specification was to create an object-oriented game program and demonstrate the techniques used using the rapid application development model.

The features included in the application are:

* **Help panel** – The first thing the user sees when they start the program which shows how to play the game and pressing P to toggle pause. The help panel can be seen when the game is paused.
* **Game reset** – game can be reset by simply pressing the reset button
* **Score board** – total number of wins is shown on the left

## Program design

### Screen components and their properties

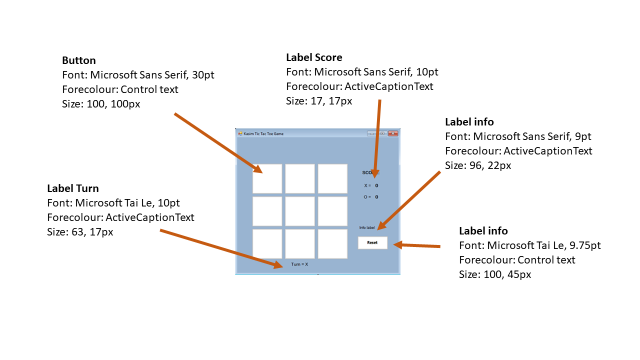
These screenshots show the screen components with their labeled properties for the game:

Figure 1 labelled properties

The sequence diagrams below will illustrate the behavior of the program. Figure 2 below shows the sequence for creating a new game and figure 3 for playing a game.

Figure 3 Playing game diagram

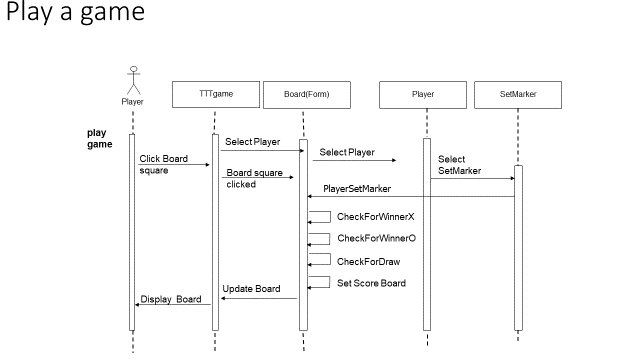
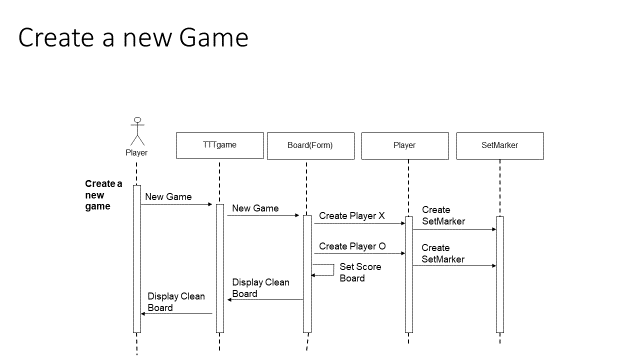
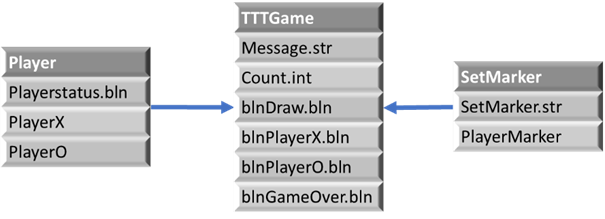


Figure 2 New/Resetting game diagram



Class diagram



These diagrams are very important

### Class Properties and methods

### Event procedures and description

The table below shows all the event procedures with their description.

Below shown is a flowchart for the processing tasks

The flowchart below on the left shows the process for the whole game from start to end and the flow chart on the right shows the process of the timer­­­

### Test plan and results

The table below shows the test for each event in the program and their results:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Date** | **Expected Result** | **Actual Result** | **Corrective Action** |
| Form loads | 5/04/18 | Form loads without any difficulty and displays board | The form loads with no problem | N/A |
| Welcome message displayed and click ok | 02/05/19 | Welcome message is displayed and disappears after clicking ‘ok’ | Welcome message is displayed and form is shown after clicking ‘ok’ | N/A |
| Click on button TopLeft | 02/05/19 | X is displayed on button TopLeft | X is displayed on button | N/A |
| Click on button TopCentre | 02/05/19 | O is displayed button TopCentre | O is displayed button | N/A |
| Click on button TopRight | 02/05/19 | X is displayed on button TopRight | X is displayed on button | N/A |
| Click on button CentreLeft | 02/05/19 | O is displayed on button CentreLeft | O is displayed on button | N/A |
| Click on button CentreCentre | 02/05/19 | X is displayed on button CentreCentre | X is displayed on button | N/A |
| Click on button CentreRight | 02/05/19 | O is displayed on button CentreRight | O is displayed on button | N/A |
| Click on button BottomLeft | 02/05/19 | X is displayed on button BottomLeft | X is displayed on button | N/A |
| Click on button BottomCentre | 02/05/19 | Nothing displayed BottomCentre | Nothing displayed | N/A |
| Click on button BottomRight | 02/05/19 | Nothing displayed BottomRight | Nothing displayed | N/A |
| Click reset button | 02/05/19 | The grid clears and “Game reset” displayed | The grid clears and “Game reset” displayed | N/A |
| Pressing Esc | 25/02/18 | Exits the form | Application closes | N/A |

## Annotated code printouts

The code below will show all the code from the application fully commented:

// Author: Kasim Hussain 104016

// Description: Tic Tac Toe is implemented in the form class

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kasim104016TicTacToe

{

public partial class TTTgame : Form

{

//Properties

private Boolean blnDraw;

private Boolean blnPlayerX;

private Boolean blnPlayerO;

private Boolean blnGameOver;

//Player class declarations

Player PlayerX;

Player PlayerO;

public TTTgame()

{

InitializeComponent();

ResetGame();

}

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

{

string message = "Welcome to Kasim's Tic Tac Toe game";

MessageBox.Show(message);

}

//create players

public void CreatePlayers()

{

if (PlayerX == null)

{

PlayerX = new Player(true, "X");

}

if (PlayerO == null)

{

PlayerO = new Player(false, "O");

}

}

//resets board and game state

private void ResetGame()

{

//creat players

CreatePlayers();

//Clear the board

TopLeft.Text = "";

TopLeft.BackColor = Color.White;

TopCenter.Text = "";

TopCenter.BackColor = Color.White;

TopRight.Text = "";

TopRight.BackColor = Color.White;

CenterLeft.Text = "";

CenterLeft.BackColor = Color.White;

CenterCenter.Text = "";

CenterCenter.BackColor = Color.White;

CenterRight.Text = "";

CenterRight.BackColor = Color.White;

BottomLeft.Text = "";

BottomLeft.BackColor = Color.White;

BottomCenter.Text = "";

BottomCenter.BackColor = Color.White;

BottomRight.Text = "";

BottomRight.BackColor = Color.White;

lblMessage.Text = "";

blnGameOver = false;

blnDraw = false;

blnPlayerX = false;

blnPlayerO = false;

labelTurn.Text = "Turn = X";

PlayerX.PlayerStatus = true;

PlayerO.PlayerStatus = false;

}

// Events for clicking buttons

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

{

Game\_Click(TopLeft);

}

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

{

Game\_Click(TopCenter);

}

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

{

Game\_Click(TopRight);

}

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

{

Game\_Click(CenterLeft);

}

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

{

Game\_Click(CenterCenter);

}

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

{

Game\_Click(CenterRight);

}

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

{

Game\_Click(BottomLeft);

}

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

{

Game\_Click(BottomCenter);

}

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

{

Game\_Click(BottomRight);

}

//Event handler for click event (all squares)

void Game\_Click(Button BoardSquare)

{

if (blnGameOver == false)

{

//Player X turn to pick a position

if (PlayerX.PlayerStatus == true)

{

if (BoardSquare.Text == "")

{

BoardSquare.Text = PlayerX.PlayerMarker;

}

else

{

lblMessage.Text = "The choice made wasn't empty";

}

}

//Player O turn to pick a position

if (PlayerO.PlayerStatus == true)

{

if (BoardSquare.Text == "")

{

BoardSquare.Text = PlayerO.PlayerMarker;

}

else

{

lblMessage.Text = "The choice made wasn't empty";

}

}

//Check if X has Won

CheckXWinningLine();

if (blnPlayerX == true)

{

lblMessage.Text = "X Wins";

ScoreBoard();

blnGameOver = true;

}

//Check if O has Won

CheckOWinningLine();

if (blnPlayerO == true)

{

lblMessage.Text = "O Wins";

ScoreBoard();

blnGameOver = true;

}

//Check for a draw

CheckDraw();

if (blnDraw == true)

{

lblMessage.Text = "Game Drawn";

blnGameOver = true;

}

//Change turn of player

if (PlayerO.PlayerStatus == true)

{

labelTurn.Text = "Turn = X";

PlayerO.PlayerStatus = false;

PlayerX.PlayerStatus = true;

}

else

{

labelTurn.Text = "Turn = O";

PlayerO.PlayerStatus = true;

PlayerX.PlayerStatus = false;

}

}

}

//Function to check for a draw

private void CheckDraw()

{

int count = 1;

if (TopLeft.Text != "")

{

count++;

}

if (TopCenter.Text != "")

{

count++;

}

if (TopRight.Text != "")

{

count++;

}

if (CenterLeft.Text != "")

{

count++;

}

if (CenterCenter.Text != "")

{

count++;

}

if (CenterRight.Text != "")

{

count++;

}

if (BottomLeft.Text != "")

{

count++;

}

if (BottomCenter.Text != "")

{

count++;

}

if (BottomRight.Text != "")

{

count++;

}

if (count == 9)

{

blnDraw = true;

}

}

// Check for X winning line

private void CheckXWinningLine()

{

//Check HORIZONTAL lines for winner

if (TopLeft.Text == "X" & TopCenter.Text == "X" & TopRight.Text == "X")

{

blnPlayerX = true;

}

if (CenterLeft.Text == "X" & CenterCenter.Text == "X" & CenterRight.Text == "X")

{

blnPlayerX = true;

}

if (BottomLeft.Text == "X" & BottomCenter.Text == "X" & BottomRight.Text == "X")

{

blnPlayerX = true;

}

// Check VERTICAL lines for winner

if (TopLeft.Text == "X" & CenterLeft.Text == "X" & BottomLeft.Text == "X")

{

blnPlayerX = true;

}

if (TopCenter.Text == "X" & CenterCenter.Text == "X" & BottomCenter.Text == "X")

{

blnPlayerX = true;

}

if (TopRight.Text == "X" & CenterRight.Text == "X" & BottomRight.Text == "X")

{

blnPlayerX = true;

}

//Check DIAGONAL lines for winner

if (TopLeft.Text == "X" & CenterCenter.Text == "X" & BottomRight.Text == "X")

{

blnPlayerX = true;

}

if (TopRight.Text == "X" & CenterCenter.Text == "X" & BottomLeft.Text == "X")

{

blnPlayerX = true;

}

}

// Check for O winning line

private void CheckOWinningLine()

{

//Check HORIZONTAL lines for winner

if (TopLeft.Text == "O" & TopCenter.Text == "O" & TopRight.Text == "O")

{

blnPlayerO = true;

}

if (CenterLeft.Text == "O" & CenterCenter.Text == "O" & CenterRight.Text == "O")

{

blnPlayerO = true;

}

if (BottomLeft.Text == "O" & BottomCenter.Text == "O" & BottomRight.Text == "O")

{

blnPlayerO = true;

}

// Check VERTICAL lines for winner

if (TopLeft.Text == "O" & CenterLeft.Text == "O" & BottomLeft.Text == "O")

{

blnPlayerO = true;

}

if (TopCenter.Text == "O" & CenterCenter.Text == "O" & BottomCenter.Text == "O")

{

blnPlayerO = true;

}

if (TopRight.Text == "O" & CenterRight.Text == "O" & BottomRight.Text == "O")

{

blnPlayerO = true;

}

//Check DIAGONAL lines for winner

if (TopLeft.Text == "O" & CenterCenter.Text == "O" & BottomRight.Text == "O")

{

blnPlayerO = true;

}

if (TopRight.Text == "O" & CenterCenter.Text == "O" & BottomLeft.Text == "O")

{

blnPlayerO = true;

}

}

//Score board update function

private void ScoreBoard()

{

int Converter;

if (PlayerX.PlayerStatus == true)

{

Converter = Convert.ToInt32(labelXScore.Text);

Converter = Converter + 1;

labelXScore.Text = Converter.ToString();

}

if (PlayerO.PlayerStatus == true)

{

Converter = Convert.ToInt32(labelOScore.Text);

Converter = Converter + 1;

labelOScore.Text = Converter.ToString();

}

}

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

{

ResetGame();

lblMessage.Text = "Game Reset";

}

}

}

Player class

// Date: 10/05/2019

// Author: Kasim Hussain 104016

// Description: Tic Tac Toe is implimented in the form class

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Kasim104016TicTacToe

{

class Player

{

// private variables within the scope of the class

private bool blnPlayerStatus;

private SetMarker mkPlayermarker;

public Player(bool pPlayStatus, string pMark)

{

blnPlayerStatus = pPlayStatus;

mkPlayermarker = new SetMarker(pMark);

}

//PlayerStatus property

public bool PlayerStatus

{

get { return blnPlayerStatus; }

set { blnPlayerStatus = value; }

}

//PlayerMarker property

public string PlayerMarker

{

get { return mkPlayermarker.PlayerMarker; }

}

}

}

Setmarker Class

// Author: Kasim Hussain 104016

// Description: Tic Tac Toe is implimented in the form class

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Kasim104016TicTacToe

{

class SetMarker

{

private string strMarker;

//Constructor

public SetMarker(string pMarker)

{

strMarker = pMarker;

}

//PlayerMarker property

public string PlayerMarker

{

get { return strMarker; }

set { strMarker = value; }

}

}

}

## Known issues

From the test there are very little errors, this is because the rapid application development (RAD) approach was taken which included iterating between the design and coding. However, there still could be room for improvement in the program such as adding more features such as being able to increase the board size. If there were errors in the program a debug of the code would take place as well as making sure that all the code that is required for the program to work is there. As well as this, making sure that the events are tied to the buttons if the buttons did not display anything.