Project 1 – Blackjack

EL2014-0043

Donovan van Heerden

Contents

- Source Code
- Screenshots
- User Documentation

Source Code:

```
//  Author:         Donovan van Heerden
//  Date:           07/08/2014
//  Purpose:        Emulate the Blackjack Card Game.

using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class frmBlackjack : Form
    {
        //Declare Variables and Collections.
        Random random = new Random();
        ArrayList Deck = new ArrayList();
        ArrayList Dealer = new ArrayList();
        ArrayList Player = new ArrayList();
        ArrayList Dpbx = new ArrayList();
        ArrayList Ppbx = new ArrayList();

        //Delaring the Scores and "total" value variables for use.
        int DScore, PScore,Wins,Losses,Ties;

        public frmBlackjack()
        {
            InitializeComponent();
        }

        //  Method StartGame() starts the game, playing the first 2 cards
        //  for both the dealer and player are played.
        protected void StartGame()
        {
            int RandNum;
            DScore = 0;
            PScore = 0;

            #region ArrPictureBoxAssign

            #region Dealerpbx
            Dpbx.Add(pbxD1);
            Dpbx.Add(pbxD2);
            Dpbx.Add(pbxD3);
            Dpbx.Add(pbxD4);
            Dpbx.Add(pbxD5);
            Dpbx.Add(pbxD6);
            Dpbx.Add(pbxD7);
            Dpbx.Add(pbxD8);
            Dpbx.Add(pbxD9);
            Dpbx.Add(pbxD10);
            Dpbx.Add(pbxD11);
            #endregion

            #region Playerpbx
            Ppbx.Add(pbxP1);
            Ppbx.Add(pbxP2);
            Ppbx.Add(pbxP3);
```

```
            Ppbx.Add(pbxP4);
            Ppbx.Add(pbxP5);
            Ppbx.Add(pbxP6);
            Ppbx.Add(pbxP7);
            Ppbx.Add(pbxP8);
            Ppbx.Add(pbxP9);
            Ppbx.Add(pbxP10);
            Ppbx.Add(pbxP11);
            #endregion

            #endregion

            #region AssignDeckValues
            for (int CardSuit = 0; CardSuit < 4; CardSuit++)
            {
                for (int CardNo = 0; CardNo < 14; CardNo++)
                {
                    if (CardNo == 1)
                    {
                        continue;
                    }
                    if (CardNo == 0)
                    {
                        Deck.Add((Deck.Count).ToString() + ", " + CardNo.ToString());
                        // MessageBox.Show((Deck.Count ).ToString() + ", " + CardNo.ToString());
                    }
                    else if ((CardNo < 10) & (CardNo >= 2))
                    {
                        Deck.Add((Deck.Count).ToString() + ", " + CardNo.ToString());
                        // MessageBox.Show((Deck.Count ).ToString() + ", " + CardNo.ToString());
                    }
                    else if (CardNo >= 10)
                    {
                        Deck.Add((Deck.Count).ToString() + ", 10");
                        // MessageBox.Show((Deck.Count ).ToString() + ", 10");
                    }

                }
            }
            #endregion

            #region DealInitialCards+Score
            for (int Pplay = 0; Pplay < 2; Pplay++)
            {
                RandNum = RandomCard(Deck.Count);
                Player.Add(Deck[RandNum]);
                Deck.Remove(Deck[RandNum]);
                //MessageBox.Show(Player[Pplay].ToString());
                PictureBox pic = (PictureBox)Ppbx[Pplay];
                pic.Image = Image.FromFile("Resources/Cards/" + GetCard(Player[Pplay].ToString()) +
".png");
                pic.Visible = true;
                PScore += GetCardValue(Player[Pplay].ToString(), PScore);

            }

            for (int Dplay = 0; Dplay < 2; Dplay++)
            {
                RandNum = RandomCard(Deck.Count);
                Dealer.Add(Deck[RandNum]);
                Deck.Remove(Deck[RandNum]);
                PictureBox pic = (PictureBox)Dpbx[Dplay];
                if (Dplay == 1)
                {
                    pic.Image = Image.FromFile("Resources/Cards/back.png");
                    DScore += GetCardValue(Dealer[Dplay].ToString(), DScore);
```

```
                }
                else
                {
                    pic.Image = Image.FromFile("Resources/Cards/" + GetCard(Dealer[Dplay].ToString()) +
".png");
                    DScore += GetCardValue(Dealer[Dplay].ToString(), DScore);
                    lblDScore.Text = "Score: " + DScore.ToString();
                }
                pic.Visible = true;


            }

            lblPScore.Text = "Score: " + PScore.ToString();



            #endregion
        }

        //  Method RandomCard(int Decksize), accepts an interger as an argument which it then uses
        //  to create a random number between 0 and that integer given.
        protected int RandomCard(int DeckSize)
        {
            int cPlayed;
            cPlayed = random.Next(DeckSize);
            //MessageBox.Show("RandomCard(): Decksize = "+DeckSize.ToString());
            return cPlayed;
        }

        //  Method GetCard(string Card), accepts a string as an argument which,
        //  which it then splits into an array and returns the first element back
        protected string GetCard(string Card)
        {
            string[] ArrCardInfo = Card.Split(',');
            //MessageBox.Show("GetCard() : Card = "+ Card);
            return ArrCardInfo[0];
        }

        //  Method GetCardValue(string Card, int Score), accepts an interger and a string
        //  as arguments, it calculates the score by splitting the string and converting the
        //  second element to an interger to calculate the score. if it finds the cards or value
        //  represents an Ace, then an the AceCheck(int Score) method is called within
GetCardValue(string Card, int Score).
        protected int GetCardValue(string Card, int Score)
        {
            if ((GetCard(Card) == "0") | (GetCard(Card) == "13") | (GetCard(Card) == "26") |
(GetCard(Card) == "39"))
            {
                return AceCheck(Score);
            }
            string[] ArrCardValue = Card.Split(',');
            return Int32.Parse(ArrCardValue[1]);
        }

        //  Method ClearBoard_Reset(), resets the game and calls the StartGame() a new one.
        protected void ClearBoard_Reset()
        {
            for (int CardCount = 0; CardCount < Player.Count; CardCount++)
            {
                PictureBox pic = (PictureBox)Ppbx[CardCount];
                pic.Visible = false;
            }

            for (int CardCount = 0; CardCount < Dealer.Count; CardCount++)
            {
```

```
                PictureBox pic = (PictureBox)Dpbx[CardCount];
                pic.Visible = false;
            }

            #region Clearing ArrayLists
            Deck.Clear();
            Player.Clear();
            Dealer.Clear();
            Dpbx.Clear();
            Ppbx.Clear();
            #endregion

            StartGame();
        }

        //  Method AceCheck(int Score), is called by GetCardValue(string Card, int Score), to determine
the Aces value.
        protected int AceCheck(int Score)
        {
            int Ace;

            if (Score > 10)
            {
                Ace = 1;
                return Ace;
            }
            else
            {
                Ace = 11;
                return Ace;
            }
        }

        //  Method DealerPlay(), plays the dealers turn and calls the CheckScore(), when dealer is
finished playing.
        protected void DealerPlay()
        {
            int RandNum;
            int Dplay = Dealer.Count;
            bool DStop = false;

            PictureBox pic = (PictureBox)Dpbx[1];
            pic.Image = Image.FromFile("Resources/Cards/" + GetCard(Dealer[1].ToString()) + ".png");
            lblDScore.Text = "Score: " + DScore.ToString();

            while (DStop == false)
            {
                if (DScore >= 17)
                {
                    DStop = true;

                    CheckScore();
                    break;
                }
                else if (DScore == 21)
                {
                    MessageBox.Show("Dealer got Blackjack! You lose!");
                    btnStay.Enabled = false;
                    btnHit.Enabled = false;
                    btnDeal.Enabled = true;
                    break;
                }
                //MessageBox.Show(Dplay.ToString());
                RandNum = RandomCard(Deck.Count);
                //MessageBox.Show(RandNum.ToString());
                Dealer.Add(Deck[RandNum]);
```

```
                //MessageBox.Show("Dealer Array: "+Dealer.Count.ToString());
                Deck.Remove(Deck[RandNum]);
                pic = (PictureBox)Dpbx[Dplay];
                //MessageBox.Show("Getting Card: "+GetCard(Dealer[Dplay].ToString()));
                pic.Image = Image.FromFile("Resources/Cards/" + GetCard(Dealer[Dplay].ToString()) +
".png");
                pic.Visible = true;
                //MessageBox.Show("Card Value:
"+GetCardValue(Dealer[Dplay].ToString(),DScore).ToString());
                DScore += GetCardValue(Dealer[Dplay].ToString(), DScore);
                lblDScore.Text = "Score: " + DScore.ToString();
                Dplay++;


            }


        }

        //  Method CheckScore(), checks the score to determine who won or lost.
        protected void CheckScore()
        {
            bool Dbust = false;
            bool Pbust = false;

            if ((DScore > 21) & (PScore > 21))
            {
                Pbust = true;
                Losses++;
                lblLosses.Text = "Losses: " + Losses;
                MessageBox.Show("Player Bust! Dealer wins!");
                btnStay.Enabled = false;
                btnHit.Enabled = false;
                btnDeal.Enabled = true;
            }
            else if (DScore > 21)
            {
                Dbust = true;
                Wins++;
                lblWins.Text = "Wins: "+Wins;
                MessageBox.Show("Dealer Bust! You win!");
                btnStay.Enabled = false;
                btnHit.Enabled = false;
                btnDeal.Enabled = true;

            }
            else if (PScore > 21)
            {
                Pbust = true;
                Losses++;
                lblLosses.Text = "Losses: " + Losses;
                MessageBox.Show("Player Bust! Dealer wins!");
                btnStay.Enabled = false;
                btnHit.Enabled = false;
                btnDeal.Enabled = true;
            }
            else
            {
                if ((DScore < PScore) & (Pbust == false))
                {
                    Wins++;
                    lblWins.Text = "Wins: " + Wins;
                    MessageBox.Show("You win!");
                    btnStay.Enabled = false;
                    btnHit.Enabled = false;
                    btnDeal.Enabled = true;
```

```
            }
            else if ((DScore > PScore) & (Dbust == false))
            {
                Losses++;
                lblLosses.Text = "Losses: " + Losses;
                MessageBox.Show("Dealer wins! You lose!");
                btnStay.Enabled = false;
                btnHit.Enabled = false;
                btnDeal.Enabled = true;

            }
            else if ((DScore == PScore) & (Dbust == false) & (Pbust == false))
            {
                Ties++;
                lblTies.Text = "Ties: " + Ties;
                MessageBox.Show("It's a tie!");
                btnStay.Enabled = false;
                btnHit.Enabled = false;
                btnDeal.Enabled = true;

            }
        }

    }

    private void btnQuit_Click(object sender, EventArgs e)
    {
        //  Exits the application.
        Application.Exit();
    }

    private void btnPlay_Click(object sender, EventArgs e)
    {
        //  Puts the buttons in their positions or makes the appropriate ones visible or invisible.
        btnPlay.Visible = false;
        btnDeal.Visible = true;
        btnDeal.Enabled = false;
        btnHit.Enabled = true;
        btnStay.Enabled = true;
        btnQuit.Location = new Point(727, 324);

        //  Sets the Wins, Losses and Tries counters to 0.
        Wins = 0;
        Losses = 0;
        Ties = 0;

        //  Attempts to do the following code and executes the catch if an error occurs.
        try
        {
            StartGame();
        }
        catch (Exception)
        {
            MessageBox.Show("Oops! An error Occured. Application Closing.");
            Application.Exit();
        }
    }

    private void btnDeal_Click(object sender, EventArgs e)
    {
        //  Disables or enables the appropriate buttons.
        btnDeal.Enabled = false;
        btnHit.Enabled = true;
        btnStay.Enabled = true;
```

```
            //  Attempts to do the following code and executes the catch if an error occurs.
            try
            {
                ClearBoard_Reset();
            }
            catch (Exception Error)
            {
                MessageBox.Show("An Error Occured: \n'"+Error.ToString()+"'\n Application Closing.");
                Application.Exit();
            }
        }

        private void btnHit_Click(object sender, EventArgs e)
        {
            //  Initialises the local Variables.
            int RandNum;
            bool bust = false;
            int Pplay = Player.Count;

            //Attempts to do the following code and executes the catch if an error occurs.
            try
            {
                if (bust == false)
                {

                    //MessageBox.Show(Pplay.ToString());
                    RandNum = RandomCard(Deck.Count);
                    //MessageBox.Show(RandNum.ToString());
                    Player.Add(Deck[RandNum]);
                    //MessageBox.Show(Player.Count.ToString());
                    Deck.Remove(Deck[RandNum]);
                    PictureBox pic = (PictureBox)Ppbx[Pplay];
                    //MessageBox.Show(GetCard(Player[Pplay].ToString()));
                    pic.Image = Image.FromFile("Resources/Cards/" + GetCard(Player[Pplay].ToString()) +
".png");
                    pic.Visible = true;
                    //MessageBox.Show(GetCardValue(Player[Pplay].ToString()).ToString());
                    PScore += GetCardValue(Player[Pplay].ToString(), PScore);
                    lblPScore.Text = "Score: " + PScore.ToString();
                    Pplay++;

                    if (PScore > 21)
                    {
                        bust = true;
                        DealerPlay();
                        btnStay.Enabled = false;
                        btnHit.Enabled = false;
                        btnDeal.Enabled = true;
                    }
                    else if (PScore == 21)
                    {
                        DealerPlay();
                        btnStay.Enabled = false;
                        btnHit.Enabled = false;
                        btnDeal.Enabled = true;
                    }

                }


            }
            catch (Exception Error)
            {
                MessageBox.Show("An Error Occured: \n'" + Error.ToString() + "'\n Application Closing.");
                Application.Exit();
            }
```

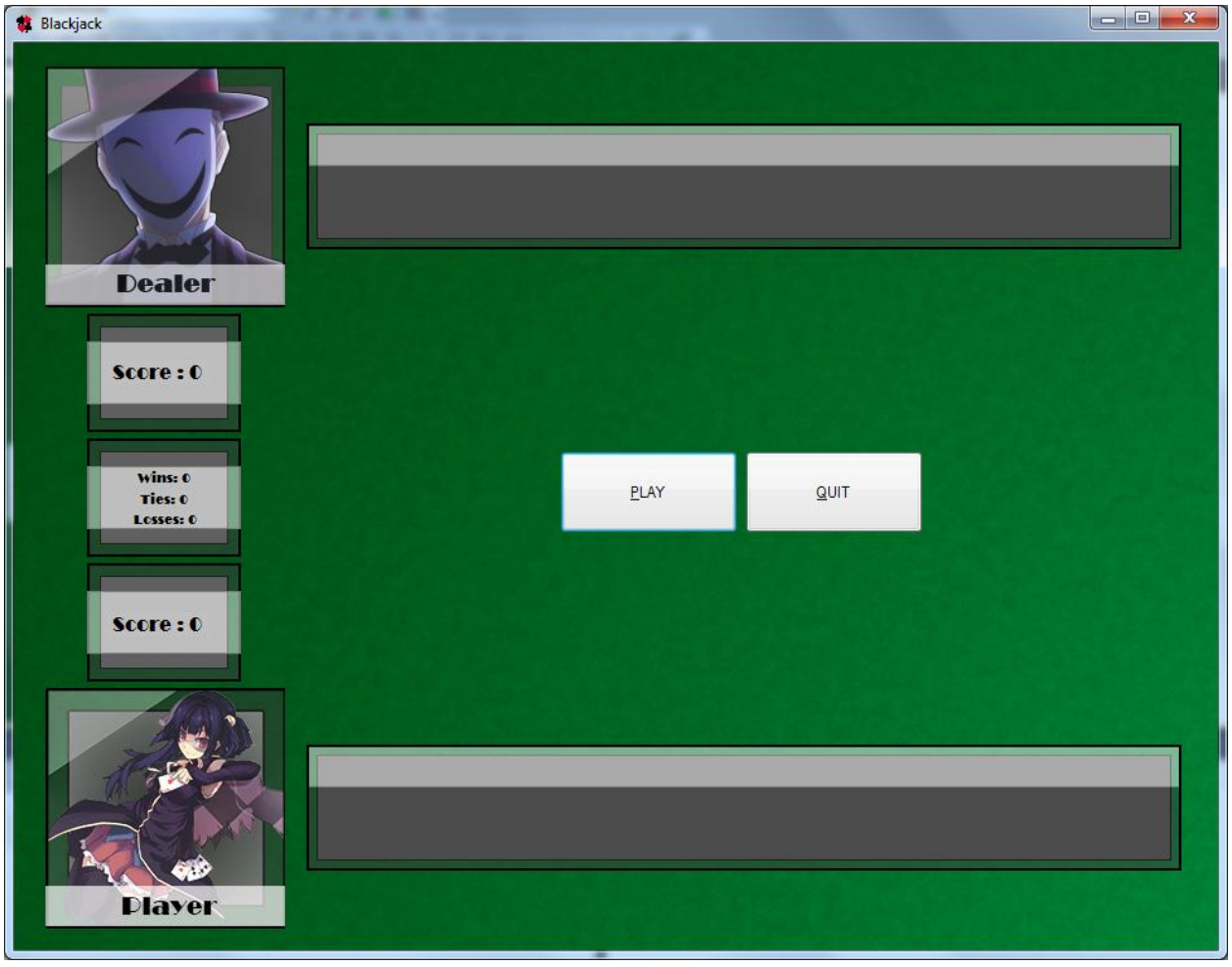
```
        }

        private void btnStay_Click(object sender, EventArgs e)
        {
            // Attempts to do the following code and executes the catch if an error occurs.
            try
            {
                if (PScore == 21)
                {
                    DealerPlay();
                    btnStay.Enabled = false;
                    btnHit.Enabled = false;
                    btnDeal.Enabled = true;
                }
                DealerPlay();
            }
            catch (Exception Error)
            {
                MessageBox.Show("An Error Occured: \n'" + Error.ToString() + "'\n Application Closing.");
                Application.Exit();
            }
        }

    }
}
```

Screenshots:

Screenshot below is when you run the program



Screenshot below is when you click the play button

Screenshot below is when the player busts and the dealer wins by default.

Screenshot below is when the dealer busts and the player wins.

Screenshot below is when the "Hit" and "Stay" buttons are disabled, and "Deal" is enabled. (End of the Game)

User Documentation

Author: Donovan van Heerden
Student No: EL2014-0043
Date: 07/08/2014
Instructor: Jason Smith
Campus: CTI East London

When the program starts, the window will have two buttons available to press, “Play” and “Quit”.

“Play”, starts the game, 2 cards are then dealt to the dealer and player. The “Play” button will then disappear and 3 new buttons appear, “Deal”, which is disabled; “Hit” and “Stay” which are enabled.

The score for the player and dealer are then calculated accordingly.

“Hit” draws another card for the player and recalculates the score; “Stay” lets the dealer play and determines who wins. If the player busts, a message is shown and if the dealer busts, a similar method is shown accordingly. A message is also shown when the appropriate player wins. After “Stay” is pressed and the message showing the winner is displayed, “Hit” and “Stay” are then disabled and “Deal” is enabled, allowing you to play again.

Total wins, losses and ties for the games played are all recorded and shown below the dealer score and above the player score.