Liam McManus

**PROJECT DOCUMENT - UNO**

Advanced Computing



**KEY**

ANALYSIS

Pages 3 - 5

Project Specification

Requirement Specification

Functional Requirements

End User Requirements

Scopes, Constraints and Boundaries

Inputs, Processes and Output

UML Case Diagram

GANTT Chart

DESIGN

Pages 6 - 14

Pseudocode

Pages 15 - 16

Wireframe

Main Menu

Main Playspace

Entity Relationship Diagram

SQL Design

IMPLEMENTATION

Pages 17 - 20

Screenshots

Database

Program

Pages 21 - 43

Code

Research required

Ongoing testing

TESTING

Pages 44 - 56

Requirement Specification

Functional Requirements

Test Plan

All individual tests

Input Validation

Deck Randomisation

Name Randomisation

Persona Test

EVALUATION

Page 57 - 58

Fitness for purpose

Matching Design

Testing

Maintainability and Robustness

**ANALYSIS**

REQUIREMENT SPECIFICATION

This program will be a digital version of the card game UNO or switch this will link to a database and create a leaderboard of the winners and losers which will then be sorted and accessed through the program using bubble sort.

This leaderboard will be imported into Visual Basic using a read file from the access database. On import the record will be ordered by wins descending showing the winner at the top this will be shown in a box on the starting page. This starting screen will also show a start button, a description of the rules and as said before a window to access the leaderboard, on the press of the start button the program brings up another menu which will have the user input their name and a 4-letter tag once they input these the game will start.

The game is based on a record system with each player being a different object of a record with procedures adding on cards. The game will proceed as normal with each player having a card removed and added; this will all be randomly chosen out of the NPC’s decks; these cards will have individual values associated with them dictating which card they can be placed on and what they can do. When it comes to the playable characters' turn they are given an option of what card to choose and the method associated with that card is carried out. This continues until either the main player is out or everyone else is out. This is then output to the database the name of the winner.

This project meets the requirements as

* Players will input their name and name tag
* The details from access will be bubble sorted after they are read and input into an array
* The program will be modular with procedures and functions
* The program will update the access database with the newly sorted data.

REQUIREMENT SPECIFICATION

Functional Requirements

An external database containing player details will have to be created

Procedure to read previous players’ details from access to an array

Procedure to get player inputs and assign them to an array

Procedure to bubble sort this array

Procedure to randomise NPCs’ names and move the cards from the central class to each array

Procedure to allow the user to place a card and run a method associated with a button press

Procedure to export the player’s details and names from the array to update access document

Procedure to display from an array

End-User Requirements

Users should be able to view other players’ score in the main menu

Users should be able to input their name and tag

Users should be able to start the game

Users should be assigned cards

Users should be able to select which card to place

Users should be able to see the leaderboard post-game

**SCOPE, CONSTRAINTS AND BOUNDARIES**

SCOPE

This involves creating a modular program. The deliverables include;

A detailed design of the program structure

A test plan with a completed test plan data table

A working program

The results of testing

An evaluation report

CONSTRAINTS

Time

The work must be completed by March 23rd

The code must be completed within class time

Technical

The program must be created using Visual Basic

Must run through windows OS

BOUNDARIES

The program will read the leaderboard from an Access database

The players tag must be 4 chCODE AND PSEUDOCODE

**INPUTS, PROCESSES AND OUTPUTS**

INPUTS

* Data from external database containing previous player details which includes the NoOfWins, playerName and playerTag
* The program will ask the user to input their details including playerName and playerTag this tag will be limited to 4 characters
* The players decision throughout the game in which option to play

PROCESSES

* Read in data from external access file and storing these details in an array of a record
* Input validation on playerTag < 4 Characters
* Randomise other 3 NPC names
* Randomise the centreDeck record containing the details of 52 cards
* Move 28 of these instances to 4 classes evenly with 7 in each
* Use validation to check if the option the user picks can be ran
* Procedure to randomise the NPCs choice
* Run method to update players deck based on players choice
* Method to move 2 instances from record to players class (+2 function)
* Method to move 4 instances from record to players class (+4 function)
* Method to change which records are being accessed (Reverse function)
* If statement to determine whether game won or how bad loss is
* Update the wins based on last game in the playerDetails array
* Bubble sort to sort the playerDetails array

OUTPUTS

* Write the playerDetails array to the access file it was initially read from
* Display this array as a leaderboard
* Display the simplified leaderboard only including the playerTag and wins

**RESOURCES REQUIRED**

To create this programme I require a computer capable of running software such as coding packages and desktop publishing software, these programmes include Visual Basic, Serif PagePlus, Serif DrawPlus, Access and Word. For the actual programme I will require a deck of cards I’ve designed and a background designed.

HARDWARE

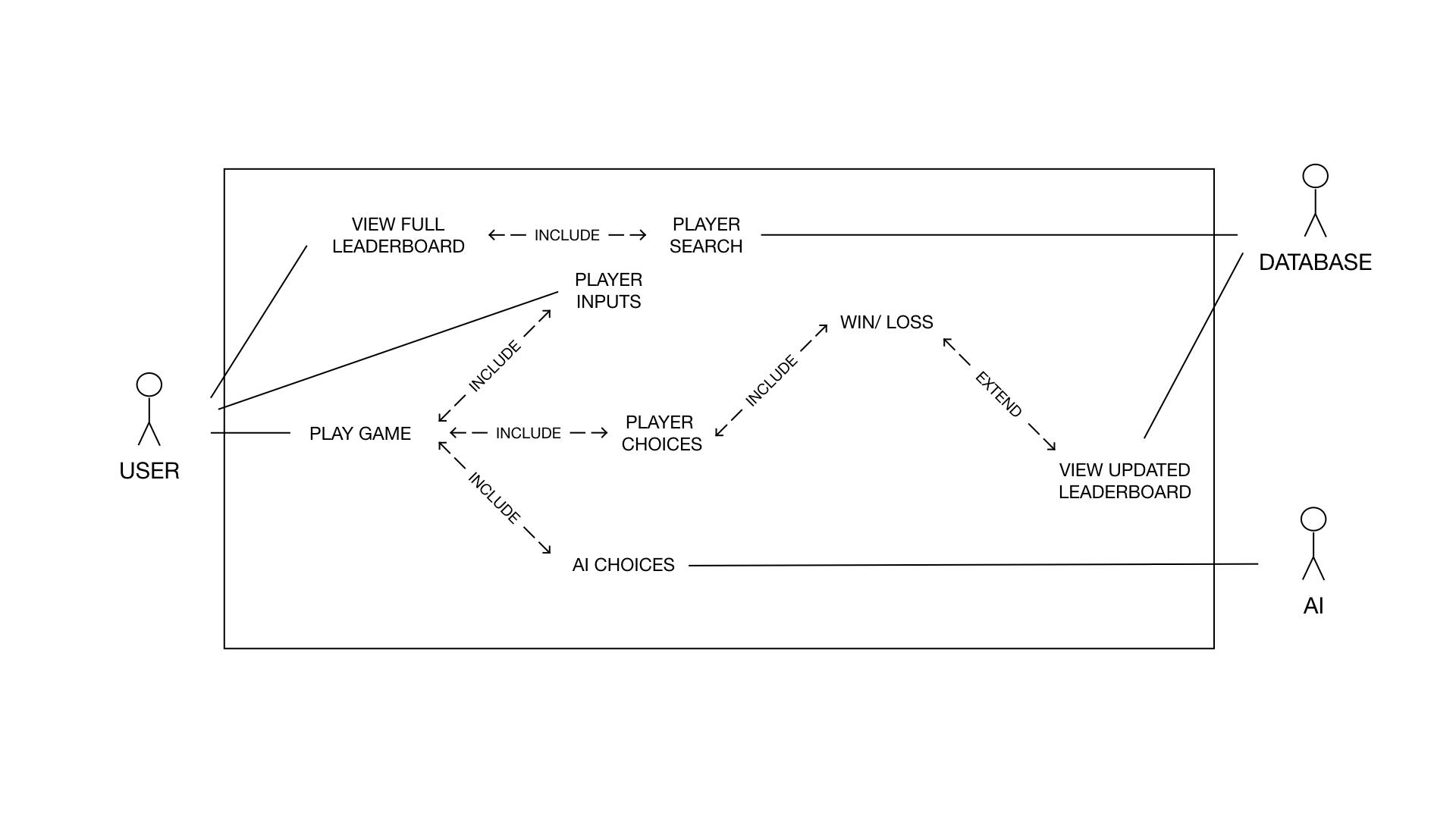
* Computer capable of running required software
* Thumb Drive able to store project for portability

SOFTWARE

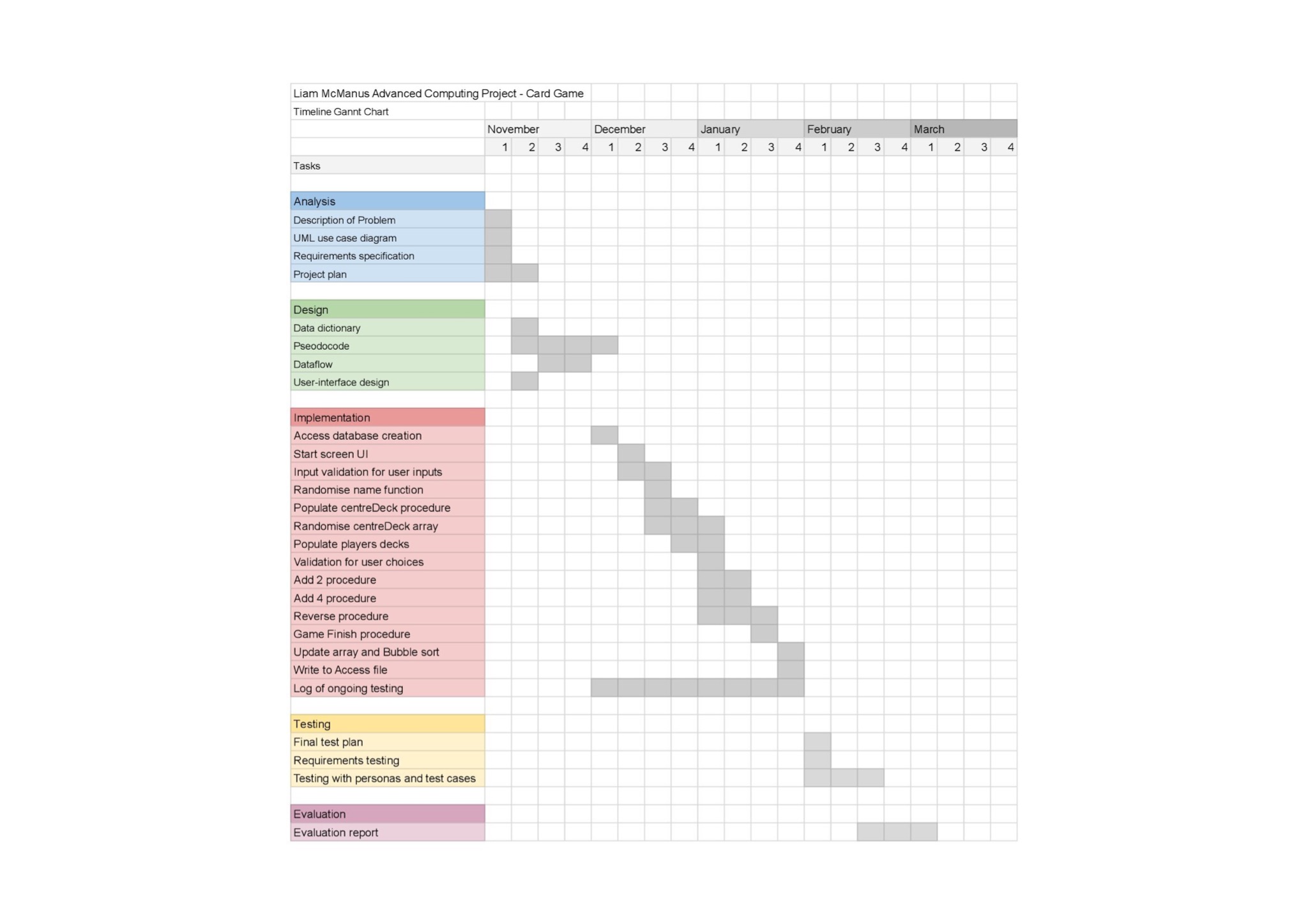
* Serif PagePlus
* Visual Basic
* Access Database

IMAGES

* Page with every card
* Button Design
* Background Design



**UML USE CASE DIAGRAM**

**GANTT CHART**

**DESIGN**

**RANDOMISING NAME**

* This code generates 4 random random characters
* And creates a tag with all 4

‘INITIALISING VARIABLES

Set rName to an empty string

Declare counter as an integer and initialise it to 0

Declare alphabet as an array of 26 strings

Declare letter as an array of 4 strings

Declare rand as an integer and initialise it to 0

‘INITIALISE THE ALPHABET IN AN ARRAY

For counter from 0 to 25 do

Set alphabet(counter) to the character corresponding to ascii of 65 plus the value of counter

‘FOR EACH LETTER IN 4 LETTER TAG RANDOMISE

For each counter from 0 to 3, do the following:

Generate a random integer between 0 and 25 and assign it to rand

Set letter(counter) to the element of alphabet with index

‘FOR EACH LETTER IN 4 LETTER TAG RANDOMISE

Set rName to all 4 characters

Convert rName to uppercase

Set rand to 0

**VALIDATION OF TAG**

* The code validates the tag input by the user
* If length is not 4 player re-enters
* Loop until length = 4
* Convert to uppercase

‘DO LOOP UNTIL LENGTH VALID

Do

If the length of player.playerTag is not equal to 4 Then

Prompt the user to enter a four-letter tag using InputBox

Assign the entered tag to player.playerTag

End If

Loop until the length of player.playerTag is equal to 4

‘CONVERT TO UPPERCASE

Convert player.playerTag to uppercase

Return player

**DECLARATION**

‘DEFINING CARD DETAILS

Structure cardDetails

String Type

String Colour

String Addition

String Number

End Structure

‘DEFINING PLAYER DETAILS

Structure playerDetails

String playerName

String playerTag

Integer Wins

End Structure

**DISPLAY PLAYER CARD**

* The code sets the path for the image
* Then changes the path depending on which card is in the array
* This is filename then returned to the main program

‘INITIALISE THE FILENAME

Set filename to the path where the card images are stored

‘IF TO DETERMINE WHICH CARD SHOULD BE DISPLAYED

If the type of the card counter in playerDeck is "colour" and its number is not "Null" Then

Else if the type of the card is "reverse" Then

Else if the type of the card is "plus" Then

Else if the type of the card is an empty string Then

End If

Return filename

**DISPLAY CENTRE DECK**

* The code sets the path for the image
* Then changes the path depending on which card is in the array
* This is filename then returned to the main program

‘INITIALISE THE FILENAME

Set filename to the path where the card images are stored

‘IF TO DETERMINE WHICH CARD SHOULD BE DISPLAYED

If the type of the played card is "colour" and its number is not "Null" Then

Set pllayed card's colour and number to filename to get the image filename

Else if the type of the played card is "reverse" Then

Set played card's colour and "reverse" to filename to get the image filename

Else if the type of the played card is "plus" Then

Set played card's colour, "+", and its addition to filename to get the image filename

End If

Return filename

**READ IN AND RANDOMISE**

‘OPEN FILE

OpenFile(1)

‘READ IN

Do

Read from file into array

counter = counter + 1

Loop Until EOF(1)

‘FILE CLOSE

FileClose(1)

‘RANDOMISE CENTRE DECK

For index to EOA

Dim rnd As New Random

t = rnd.Next(0, centreDeck.Count - 1)

Temp = centreDeck(t)

centreDeck(t) = centreDeck(index)

centreDeck(index) = Temp

Next

**DISPLAY CENTRE DECK**

* The code sets the path for the image
* Then changes the path depending on which card is in the array
* This is filename then returned to the main program

For counter = 0 To 7

playerDeck(counter) = centreDeck(counter2)

counter2 = counter2 + 1

NPC1Deck(counter) = centreDeck(counter2)

counter2 = counter2 + 1

NPC2Deck(counter) = centreDeck(counter2)

counter2 = counter2 + 1

NPC3Deck(counter) = centreDeck(counter2)

counter2 = counter2 + 1

Next counter

playedCard = centreDeck(32)

deckcounter = counter2

**CODE TO DO ON LOAD OF MAIN PLAYSPACE**

* The code initialises the counts for three NPC decks (NPC1Deck, NPC2Deck, NPC3Deck)
* The code prompts the player to enter their name and player tag using an InputBox
* The player's validated tag is displayed on a label (player\_label).
* Three NPC names are randomly generated using the randomiseName function and assigned to NPC1, NPC2, and NPC3 variables.
* The NPC names and counts are displayed on corresponding labels

On Form2 Load:

‘INITIALISING VARIABLES

Set NPCCount to the number of cards in NPCDeck

‘GET INPUTS FOR PLAYER NAME AND TAG

Prompt player to enter their name

Prompt player to enter their player tag

‘VALIDATE THE TAG

Call validateTag function

‘DISPLAY TAG

Display player tag on player\_label

‘RAND TAG FOR NPCs

Generate a random NPC name

Display NPC1 name on npc1\_label

Generate a random NPC name

Display NPC2 name on npc2\_label

Generate a random NPC name

Display NPC3 name on npc3\_label

‘SET UP GAME BY POPULATING DECKS AND CARD ARRAYS

Populate the centre deck using populateDeck function

Deal cards from centre deck to playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard and deckCounter using dealCards function

‘DISPLAY THESE UPDATED DECKS

Display playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard using displayDecks function

‘DISPLAY DECK COUNTS

Set NPCdeckCount1 label text to NPC1Count

Set NPCdeckCount2 label text to NPC2Count

Set NPCdeckCount3 label text to NPC3Count

**PLAY GAME FUNCTION**

* The code continues the game once player has played
* Then it hides the players cards while others play

‘AS PLAYER HAS JUST PLAYED TAKE ONE OF THEIR CARD COUNTER

playerDeckCount - 1

‘WHILE NOBODY HAS WON AND PLAYER HAS PLAYED

While visible is False and NPCdeckCount not equal 0 DO

‘LOOP FOR 3 NPCs

Do

Call the AIPlay function

If the played varable is True, decrease NPC1Count by 1 and call the displayDecks function passing in playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, and playedCard variables as arguments, and call the losegame function

Loop for 3

‘LOOP FOR 3 NPCs

Set the cards to visible to allow player to play

End the loop

**AI TAKE CARD**

* The code runs through the NPCDeck and finds a playable card
* Then it hides the players cards while others play

Do loop

‘IF CARD IS NULL REPLACE WITH TAKEN CARD

if addition, type, colour and number of the card at the NPC1Deck are Null then

‘REPLACE CARD

takenCard = current index of NPC1Deck

set taken = True

set visible = False

call the visibility function

Counter = counter + 1

‘LOOP UNTIL TAKEN

Loop until the counter is equal to 7 or taken is True.

**CODE TO VALIDATE PLAYED CARD**

* This code is for the player to validate played card.
* The if statement is determine whether the proposed card is the same colour as the previously played card or the same number.
* The elseif after is to determine whether its an addition card

‘DISPLAY DECK COUNTS

‘IF TO DETERMINE WHETHER IT MATCHES COLOURS

if playedCard.Colour = Colour and

playedCard.Colour is not "null" or

Number = playedCard.Number and

Addition = "null" then

‘SET CARDS TO NULL

Set card to null

‘SET PLAYER CARD FROM DECK TO NULL

playerDeck(counter).Addition = ""

playerDeck(counter).Type = ""

playerDeck(counter).Colour = ""

playerDeck(counter).Number = ""

‘VARIABLE TO START GAME

played = true

‘IF TO DETERMINE WHETHER ITS A PLUS CARD

elseif playedCard.Colour = Card.Colour and

(Card.Addition = "2" or Card.Addition = "4") then

‘SET CARDS TO NULL

playedCard = Card

Card.Addition = ""

Card.Type = ""

Card.Colour = ""

Card.Number = ""

‘VARIABLE TO START GAME

played = true

adding = adding + playedCard.Addition

‘SET PLAYER CARD FROM DECK TO NULL

playerDeck(counter).Addition = ""

playerDeck(counter).Type = ""

playerDeck(counter).Colour = ""

playerDeck(counter).Number = ""

else

Display message box saying "Please play a valid card"

endif

**CODE TO FOR AI TO PLAY CARD**

* This code is for the NPCs deck to validate play card
* This is similar to the last procedure though instead
* The elseif after is to determine whether its an addition card

‘DISPLAY DECK COUNTS

‘IF TO DETERMINE WHETHER IT MATCHES COLOURS

if playedCard.Colour = Colour and

playedCard.Colour is not "null" or

Number = playedCard.Number and

Addition = "null" then

‘DISPLAY DECK COUNTS

playedCard = Card

‘SET CARDS TO NULL

Set Card.Addition = ""

Set Card.Type = ""

Set Card.Colour = ""

Set Card.Number = ""

‘SET PLAYER CARD FROM DECK TO NULL

Set NPCDeck(counter) NULL

‘VARIABLE TO START GAME

played = true

‘IF TO DETERMINE WHETHER ITS A PLUS CARD

elseif playedCard.Colour = Card.Colour and

(Card.Addition = "2" or Card.Addition = "4") then

‘SET CARDS TO NULL

Set playedCard = Card

Set Card.Addition = ""

Set Card.Type = ""

Set Card.Colour = ""

Set Card.Number = ""

‘VARIABLE TO START GAME

played = true

adding = adding + playedCard.Addition

‘SET PLAYER CARD FROM DECK TO NULL

Set NPCDeck(counter) Null

‘IF NOT VALID RERUN TAKE CARD

If played = False Then

AItakeCard(placeholderAIDeck)

End If

**CODE IN BUTTON FOR PLAYER TO PLAY CARD**

* This code is for the NPCs deck to validate play card
* This is similar to the last procedure though instead
* The elseif after is to determine whether its an addition card

‘SET THE CARD PLAYED TO THE BUTTON PRESSED THEN VALIDATE

playedCard = playerDeck(button pressed)

Call playCard function

‘IF CARD VALID DO

If played = True Then

‘REMOVE CARD FROM DECK

Set playerCard thats been played to Null.

‘CONTINUE GAME

Set the cards to not visible to stop player from playing

Call gamePlay

End If

**BUBBLE SORT FOR PLAYERS**

‘LOOP FOR ARRAY

Do

‘IF COUNTER NULL THEN END OF ARRAY

If playersTag(counter) NULL Then

endarray = True

End If

‘RUNNING COUNTER UNTIL EOF

counter = counter + 1

Loop Until counter = 50 Or endarray = True

n = counter

swapped = True

While swapped And n >= 0

swapped = False

For counter = 0 To n - 2

‘BUBBLE SORT SWAP

If players(counter).wins < players(counter + 1).wins Then

player = players(counter)

players(counter) = players(counter + 1)

players(counter + 1) = player

swapped = True

End If

n = n - 1

End While.

**WIREFRAMES**

START MENU

BUTTON START

Displays two input boxes one Player Name second Player Tag

Player Tag is validated to length 4 and all caps

Takes the player to the main playspace

BUTTON BUBBLE SORT

Bubble sorts the array

And is then displayed with the display button

DATA DICTIONARY

| **Primary**  **Foreign** | **Field**  **Name** | **Required** | **Data Type** | **Length** | **Validation** |
| --- | --- | --- | --- | --- | --- |
|  | PlayerName | N | String | 20 - 30 |  |
| PK | PlayerTag | Y | String | 4 | Length = 4 and  All caps |
|  | Wins | Y | Integer |  |  |

MAIN PLAYSPACE



BUTTON PLAY CARD

Plays the card displayed in the picture

And removes from picture box

PICTURE BOXES

Displays the card at that point in the array

NPC LABEL

Displays the number of cards in npc deck

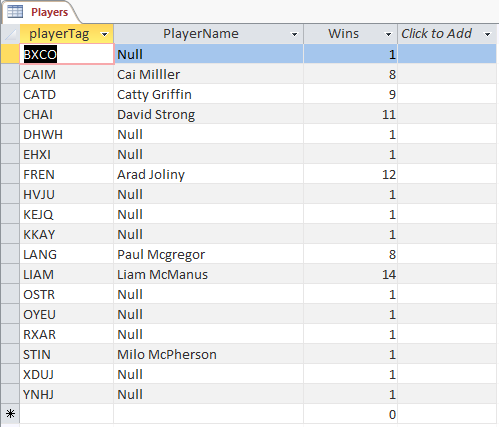
ENTITY RELATIONSHIP



SQL DESIGN

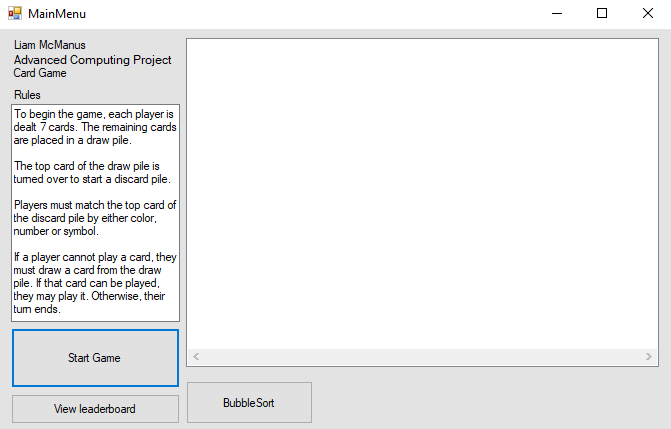
Select \* FROM [Players]

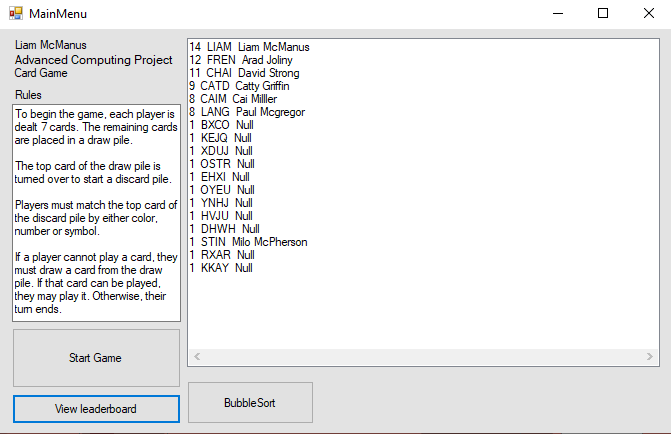
INSERT INTO [Players] VALUES ( 'Null' , '" & NPC2 & "', '1')

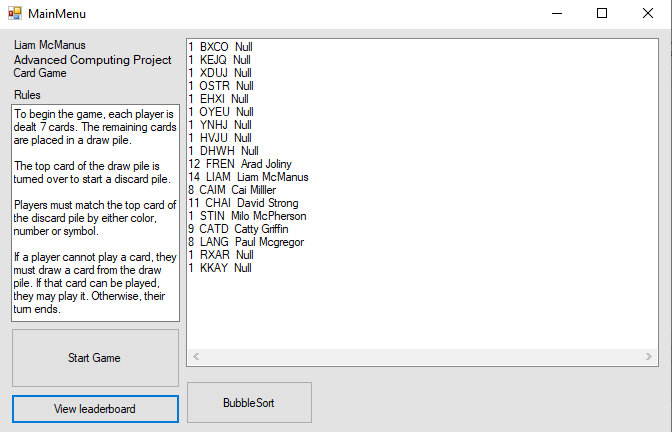
**IMPLEMENTATION**

SCREENSHOTS

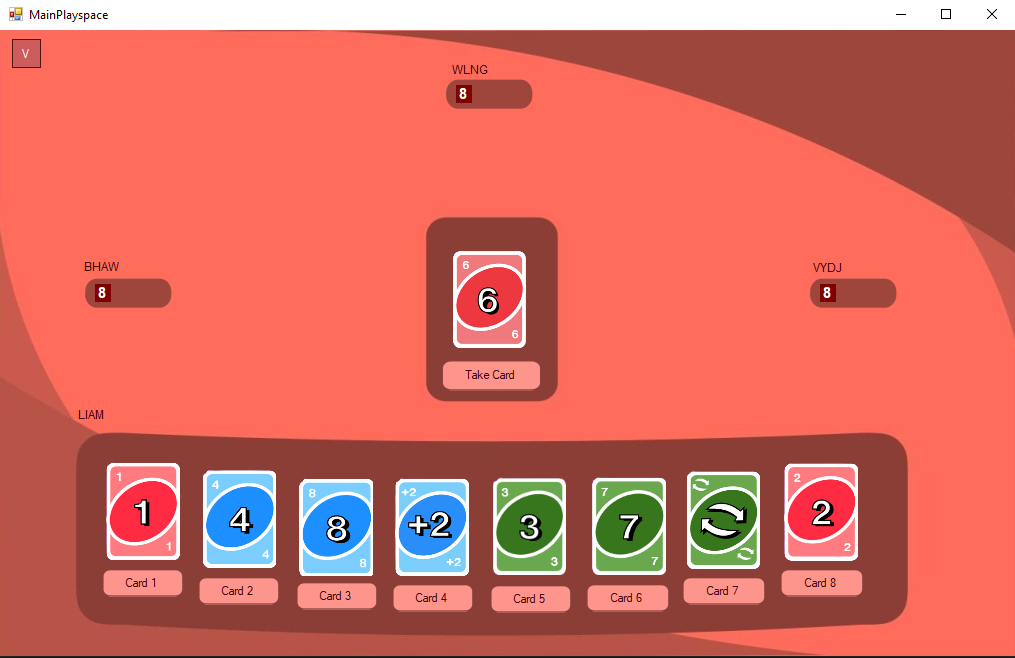
DATABASE

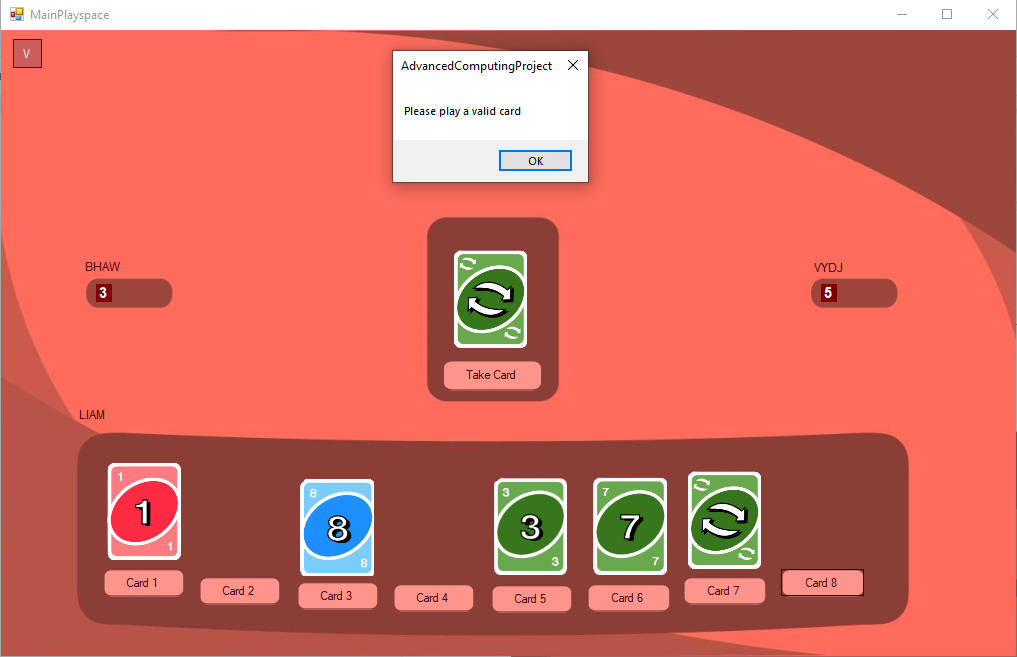
MAIN MENU

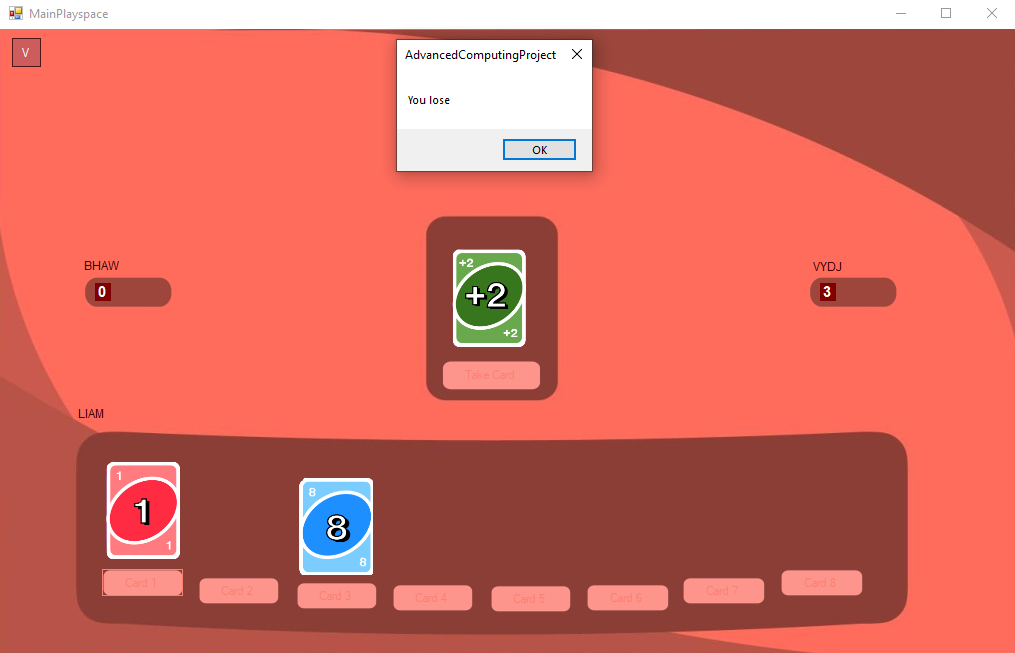


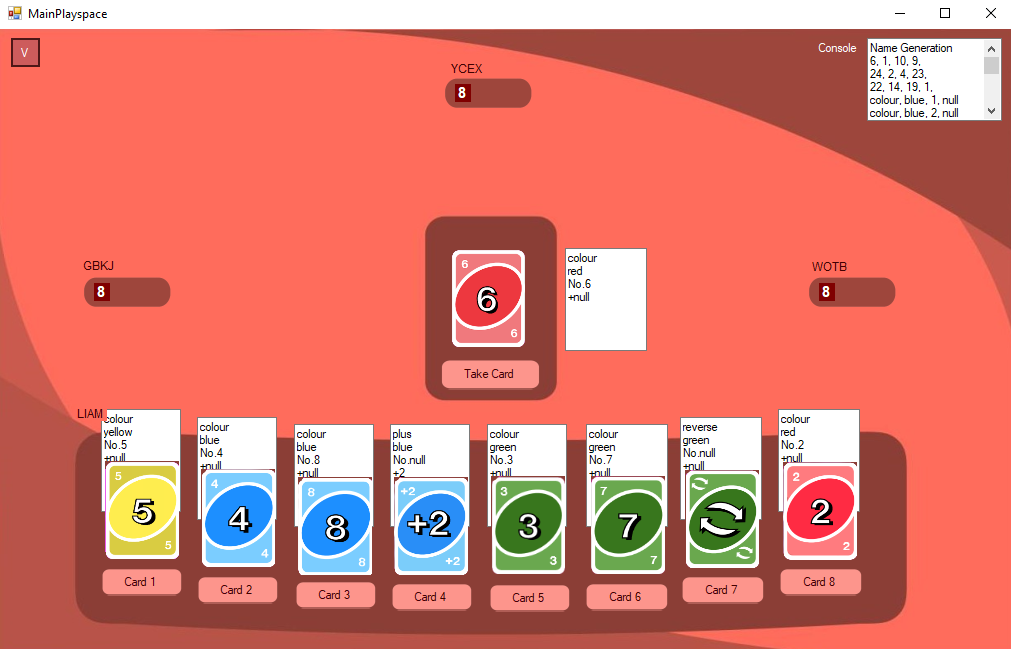


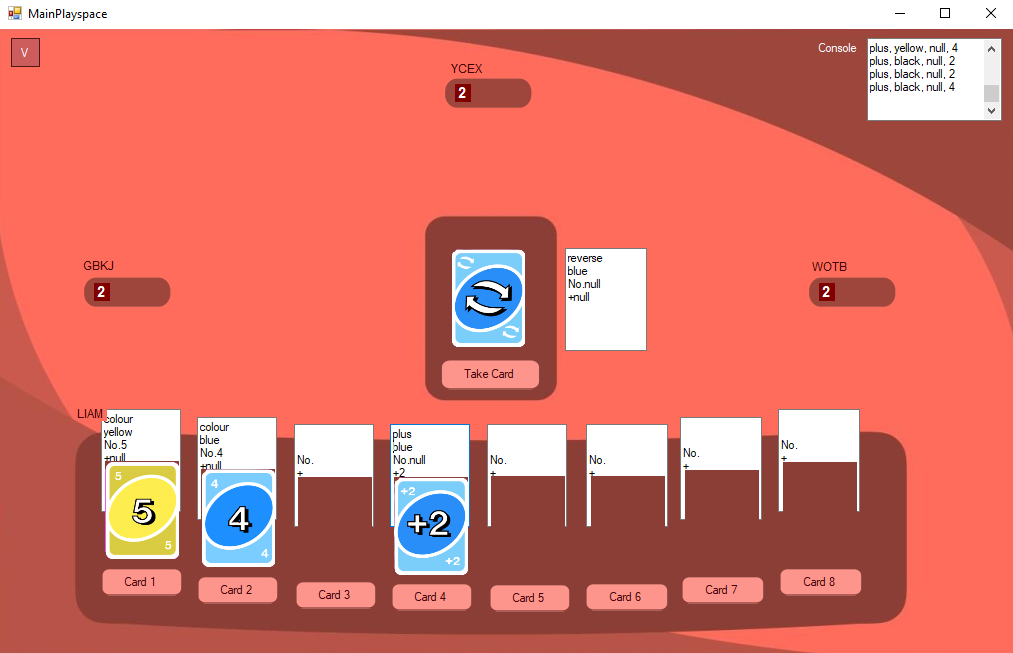
MAIN PLAYSPACE











CODE

FORM1

Imports System.Media

Imports System.Reflection

Imports System.Reflection.Emit

Imports System.Runtime.InteropServices

Imports System.Data.OleDb

Public Class mainPlayspace

Public Structure cardDetails

Dim Type As String

Dim Colour As String

Dim Addition As String

Dim Number As String

End Structure

Public Structure playerDetails

Dim playerName As String

Dim playerTag As String

Dim Wins As Integer

End Structure

'------------------------------------------------------------------------------------------

'Variables

'Declaring variables

Dim player As playerDetails

Dim gameFin As Boolean

Dim NPC1 As String

Dim NPC2 As String

Dim NPC3 As String

Dim NPC1Count As String

Dim NPC2Count As String

Dim NPC3Count As String

Dim deck(7) As cardDetails

Dim centreDeck(55) As cardDetails

Dim playedCard As cardDetails

Dim playerDeck(7) As cardDetails

Dim NPC1Deck(7) As cardDetails

Dim NPC2Deck(7) As cardDetails

Dim NPC3Deck(7) As cardDetails

Dim filename As String = "N:\Advanced Computing\cards finale\"

Dim played As Boolean = False

Dim visible As Boolean

Dim playCardCard As cardDetails

Dim adding As Integer

Dim deckCounter As Integer

Dim takenCard As cardDetails

Dim playerDeckCount As Integer

Dim players(100) As playerDetails

Dim placeholderAIDeck() As cardDetails

'------------------------------------------------------------------------------------------

'Procedure to execute on load

Private Sub Form2\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

NPC1Count = NPC1Deck.Count

NPC2Count = NPC2Deck.Count

NPC3Count = NPC3Deck.Count

Dim rName As String = ""

player.playerName = InputBox("Please enter your name")

player.playerTag = InputBox("Please enter your player tag")

validateTag(player)

player\_label.Text = player.playerTag

txtConsole.Text += "Name Generation" & vbNewLine

'Setting NPC names

randomiseName(rName)

NPC1 = rName

npc1\_label.Text = "" & NPC1

randomiseName(rName)

NPC2 = rName

npc2\_label.Text = "" & NPC2

randomiseName(rName)

NPC3 = rName

npc3\_label.Text = "" & NPC3

'Setting up UI

Call populateDeck(centreDeck)

Call dealCards(centreDeck, playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard, deckCounter)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

NPCdeckCount1.Text = NPC1Count

NPCdeckCount2.Text = NPC2Count

NPCdeckCount3.Text = NPC3Count

End Sub

'------------------------------------------------------------------------------------------

'Function to validate inputs for playerTag

Function validateTag(ByRef player As playerDetails)

Do

If Len(player.playerTag) <> 4 Then

player.playerTag = InputBox("Please enter a four letter tag")

End If

Loop Until Len(player.playerTag) = 4

player.playerTag = UCase(player.playerTag)

Return player

End Function

'Function to randomise the names of other players

Function randomiseName(ByRef rName As String)

rName = ""

Randomize()

Dim counter As Integer

Dim alphabet(26) As String

Dim letter(4) As String

Dim rand As Integer

For counter = 0 To 25

alphabet(counter) = Chr(65 + counter)

Next counter

For counter = 0 To 3

rand = Int(25 \* Rnd())

letter(counter) = alphabet(rand)

txtConsole.Text += "" & rand & ", "

Next counter

rName = letter(0) & letter(1) & letter(2) & letter(3)

rName = UCase(rName)

txtConsole.Text += vbNewLine

rand = 0

Return rName

End Function

'Function to display player deck

Function displayCard(ByRef filename As String, ByRef counter As Integer)

filename = "N:\Advanced Computing\cards finale\"

If playerDeck(counter).Type = "colour" And playerDeck(counter).Number <> "Null" Then

filename = filename & playerDeck(counter).Colour & playerDeck(counter).Number & ".png"

ElseIf playerDeck(counter).Type = "reverse" Then

filename = filename & playerDeck(counter).Colour & "reverse.png"

ElseIf playerDeck(counter).Type = "plus" Then

filename = filename & playerDeck(counter).Colour & "+" & playerDeck(counter).Addition & ".png"

ElseIf playerDeck(counter).Type = "" Then

filename = ""

End If

Return filename

End Function

'Function to display centre deck

Function displayCentreDeck(ByRef filename As String)

filename = "N:\Advanced Computing\cards finale\"

If playedCard.Type = "colour" And playedCard.Number <> "Null" Then

filename = filename & playedCard.Colour & playedCard.Number & ".png"

ElseIf playedCard.Type = "reverse" Then

filename = filename & playedCard.Colour & "reverse.png"

ElseIf playedCard.Type = "plus" Then

filename = filename & playedCard.Colour & "+" & playedCard.Addition & ".png"

End If

Return filename

End Function

'------------------------------------------------------------------------------------------

'Game functions

Function playCard(ByRef playCardCard As cardDetails, ByRef playedCard As cardDetails, ByRef counter As Integer, ByRef played As Boolean)

played = False

If (playedCard.Colour = playCardCard.Colour And playedCard.Colour <> "null" And playCardCard.Addition = "null") Or (playCardCard.Number = playedCard.Number And playCardCard.Addition = "null") Then

playedCard = playCardCard

playCardCard.Addition = ""

playCardCard.Type = ""

playCardCard.Colour = ""

playCardCard.Number = ""

playerDeck(counter).Addition = ""

playerDeck(counter).Type = ""

playerDeck(counter).Colour = ""

playerDeck(counter).Number = ""

played = True

ElseIf playedCard.Colour = playCardCard.Colour And playCardCard.Addition = "2" Or playCardCard.Addition = "4" Then

playedCard = playCardCard

playCardCard.Addition = ""

playCardCard.Type = ""

playCardCard.Colour = ""

playCardCard.Number = ""

played = True

adding = adding + playedCard.Addition

playerDeck(counter).Addition = ""

playerDeck(counter).Type = ""

playerDeck(counter).Colour = ""

playerDeck(counter).Number = ""

If adding <> 0 Then

If adding = 2 Then

Call AItakeCard(placeholderAIDeck)

Call AItakeCard(placeholderAIDeck)

NPC3Count = NPC3Count + 2

adding = 0

ElseIf adding = 4 Then

Call AItakeCard(placeholderAIDeck)

Call AItakeCard(placeholderAIDeck)

Call AItakeCard(placeholderAIDeck)

Call AItakeCard(placeholderAIDeck)

NPC3Count = NPC3Count + 4

adding = 0

End If

End If

Else

MsgBox("Please play a valid card")

End If

Call gamePlay(gameFin)

Return played

End Function

'AI PLAY CARD

Function AIplayCard(ByRef playCardCard As cardDetails, ByRef playedCard As cardDetails, ByRef counter As Integer, ByRef played As Boolean)

counter = 0

played = False

Do

playCardCard = deck(counter)

If playedCard.Colour = playCardCard.Colour And playedCard.Colour <> "null" Or playCardCard.Number = playedCard.Number And playCardCard.Addition = "null" Then

playedCard = playCardCard

playCardCard.Addition = ""

playCardCard.Type = ""

playCardCard.Colour = ""

playCardCard.Number = ""

NPC1Deck(counter).Addition = ""

NPC1Deck(counter).Type = ""

NPC1Deck(counter).Colour = ""

NPC1Deck(counter).Number = ""

played = True

ElseIf playedCard.Colour = playCardCard.Colour And (playCardCard.Addition = "2" Or playCardCard.Addition = "4") Then

playedCard = playCardCard

playCardCard.Addition = ""

playCardCard.Type = ""

playCardCard.Colour = ""

playCardCard.Number = ""

played = True

adding = adding + playedCard.Addition

NPC1Deck(counter).Addition = ""

NPC1Deck(counter).Type = ""

NPC1Deck(counter).Colour = ""

NPC1Deck(counter).Number = ""

End If

counter = counter + 1

Loop Until played = True Or counter = 8

If played = False Then

AItakeCard(placeholderAIDeck)

End If

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

Return played

End Function

Function AIplayCard2(ByRef playCardCard As cardDetails, ByRef playedCard As cardDetails, ByRef counter As Integer, ByRef played As Boolean)

counter = 0

played = False

Do

playCardCard = deck(counter)

If playedCard.Colour = playCardCard.Colour And playedCard.Colour <> "null" Or playCardCard.Number = playedCard.Number And playCardCard.Addition = "null" Then

playedCard = playCardCard

playCardCard.Addition = ""

playCardCard.Type = ""

playCardCard.Colour = ""

playCardCard.Number = ""

NPC2Deck(counter).Addition = ""

NPC2Deck(counter).Type = ""

NPC2Deck(counter).Colour = ""

NPC2Deck(counter).Number = ""

played = True

ElseIf playedCard.Colour = playCardCard.Colour And (playCardCard.Addition = "2" Or playCardCard.Addition = "4") Then

playedCard = playCardCard

playCardCard.Addition = ""

playCardCard.Type = ""

playCardCard.Colour = ""

playCardCard.Number = ""

played = True

adding = adding + playedCard.Addition

NPC2Deck(counter).Addition = ""

NPC2Deck(counter).Type = ""

NPC2Deck(counter).Colour = ""

NPC2Deck(counter).Number = ""

End If

counter = counter + 1

Loop Until played = True Or counter = 8

If played = False Then

AItakeCard2(placeholderAIDeck)

End If

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

Return played

End Function

Function AIplayCard3(ByRef playCardCard As cardDetails, ByRef playedCard As cardDetails, ByRef counter As Integer, ByRef played As Boolean)

counter = 0

played = False

Do

playCardCard = deck(counter)

If playedCard.Colour = playCardCard.Colour And playedCard.Colour <> "null" Or playCardCard.Number = playedCard.Number And playCardCard.Addition = "null" Then

playedCard = playCardCard

playCardCard.Addition = ""

playCardCard.Type = ""

playCardCard.Colour = ""

playCardCard.Number = ""

NPC3Deck(counter).Addition = ""

NPC3Deck(counter).Type = ""

NPC3Deck(counter).Colour = ""

NPC3Deck(counter).Number = ""

played = True

ElseIf playedCard.Colour = playCardCard.Colour And (playCardCard.Addition = "2" Or playCardCard.Addition = "4") Then

playedCard = playCardCard

playCardCard.Addition = ""

playCardCard.Type = ""

playCardCard.Colour = ""

playCardCard.Number = ""

played = True

adding = adding + playedCard.Addition

NPC3Deck(counter).Addition = ""

NPC3Deck(counter).Type = ""

NPC3Deck(counter).Colour = ""

NPC3Deck(counter).Number = ""

End If

counter = counter + 1

Loop Until played = True Or counter = 8

If played = False Then

AItakeCard3(placeholderAIDeck)

End If

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

Return played

End Function

'AI TAKE

Private Sub takeCard\_Click(sender As Object, e As EventArgs) Handles takeCard.Click

Dim taken As Boolean

Dim counter As Integer

takenCard = centreDeck(deckCounter)

Do

If playerDeck(counter).Addition = "" And playerDeck(counter).Type = "" And playerDeck(counter).Colour = "" And playerDeck(counter).Number = "" Then

playerDeck(counter) = takenCard

taken = True

visible = False

visibility(visible)

deckCounter = deckCounter - 1

End If

counter = counter + 1

Loop Until deckCounter = 7 Or taken = True

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

Call gamePlay(gameFin)

End Sub

Sub AItakeCard(ByRef placeholderAIDeck() As cardDetails)

Dim taken As Boolean

Dim counter As Integer

counter = 0

takenCard = centreDeck(deckCounter)

Do

If NPC1Deck(counter).Addition = "" And NPC1Deck(counter).Type = "" And NPC1Deck(counter).Colour = "" And NPC1Deck(counter).Number = "" Then

NPC1Deck(counter) = takenCard

taken = True

visible = False

visibility(visible)

End If

counter = counter + 1

Loop Until counter = 7 Or taken = True

End Sub

Sub AItakeCard3(ByRef placeholderAIDeck() As cardDetails)

Dim taken As Boolean

Dim counter As Integer

counter = 0

takenCard = centreDeck(deckCounter)

Do

If NPC3Deck(counter).Addition = "" And NPC3Deck(counter).Type = "" And NPC3Deck(counter).Colour = "" And NPC3Deck(counter).Number = "" Then

NPC3Deck(counter) = takenCard

taken = True

visible = False

visibility(visible)

End If

counter = counter + 1

Loop Until counter = 7 Or taken = True

End Sub

Sub AItakeCard2(ByRef placeholderAIDeck() As cardDetails)

Dim taken As Boolean

Dim counter As Integer

counter = 0

takenCard = centreDeck(deckCounter)

Do

If NPC2Deck(counter).Addition = "" And NPC2Deck(counter).Type = "" And NPC2Deck(counter).Colour = "" And NPC2Deck(counter).Number = "" Then

NPC2Deck(counter) = takenCard

taken = True

visible = False

visibility(visible)

End If

counter = counter + 1

Loop Until counter = 7 Or taken = True

End Sub

'------------------------------------------------------------------------------------------

'Procedures to execute in load

Sub populateDeck(ByRef centreDeck() As cardDetails)

Dim txtfileName As String

Dim counter As Integer = 0

Dim t As Integer

Dim Temp As cardDetails

txtfileName = "N:\Advanced Computing\uno.txt"

Randomize()

'Reading from deck file

FileOpen(1, txtfileName, OpenMode.Input)

Do

Input(1, centreDeck(counter).Type)

Input(1, centreDeck(counter).Colour)

Input(1, centreDeck(counter).Number)

Input(1, centreDeck(counter).Addition)

txtConsole.Text += centreDeck(counter).Type & ", "

txtConsole.Text += centreDeck(counter).Colour & ", "

txtConsole.Text += centreDeck(counter).Number & ", "

txtConsole.Text += centreDeck(counter).Addition & vbNewLine

counter = counter + 1

Loop Until EOF(1)

FileClose(1)

For index As Integer = 0 To centreDeck.Count - 1

Dim rnd As New Random

t = rnd.Next(0, centreDeck.Count - 1)

Temp = centreDeck(t)

centreDeck(t) = centreDeck(index)

centreDeck(index) = Temp

Next

txtConsole.Text += vbNewLine & "Randomised Array" & vbNewLine

For counter = 0 To 55

txtConsole.Text += centreDeck(counter).Type & ", "

txtConsole.Text += centreDeck(counter).Colour & ", "

txtConsole.Text += centreDeck(counter).Number & ", "

txtConsole.Text += centreDeck(counter).Addition & vbNewLine

Next counter

End Sub

'Procedures to deal out cards

Sub dealCards(ByRef centreDeck() As cardDetails,

ByRef playerDeck() As cardDetails,

ByRef NPC1Deck() As cardDetails,

ByRef NPC2Deck() As cardDetails,

ByRef NPC3Deck() As cardDetails,

ByRef playedCard As cardDetails,

ByRef deckcounter As Integer)

Dim counter As Integer

Dim counter2 As Integer = 0

For counter = 0 To 7

playerDeck(counter) = centreDeck(counter2)

counter2 = counter2 + 1

NPC1Deck(counter) = centreDeck(counter2)

counter2 = counter2 + 1

NPC2Deck(counter) = centreDeck(counter2)

counter2 = counter2 + 1

NPC3Deck(counter) = centreDeck(counter2)

counter2 = counter2 + 1

Next counter

playedCard = centreDeck(32)

deckcounter = counter2

End Sub

'Procedures to display decks

Sub displayDecks(ByRef playerDeck() As cardDetails,

ByRef NPC1Deck() As cardDetails,

ByRef NPC2Deck() As cardDetails,

ByRef NPC3Deck() As cardDetails,

ByRef playedCard As cardDetails)

Dim counter As Integer = 0

PlayerDeck1.Text = playerDeck(0).Type & vbNewLine & playerDeck(0).Colour & vbNewLine & "No." & playerDeck(0).Number & vbNewLine & "+" & playerDeck(0).Addition

PlayerDeck2.Text = playerDeck(1).Type & vbNewLine & playerDeck(1).Colour & vbNewLine & "No." & playerDeck(1).Number & vbNewLine & "+" & playerDeck(1).Addition

PlayerDeck3.Text = playerDeck(2).Type & vbNewLine & playerDeck(2).Colour & vbNewLine & "No." & playerDeck(2).Number & vbNewLine & "+" & playerDeck(2).Addition

PlayerDeck4.Text = playerDeck(3).Type & vbNewLine & playerDeck(3).Colour & vbNewLine & "No." & playerDeck(3).Number & vbNewLine & "+" & playerDeck(3).Addition

PlayerDeck5.Text = playerDeck(4).Type & vbNewLine & playerDeck(4).Colour & vbNewLine & "No." & playerDeck(4).Number & vbNewLine & "+" & playerDeck(4).Addition

PlayerDeck6.Text = playerDeck(5).Type & vbNewLine & playerDeck(5).Colour & vbNewLine & "No." & playerDeck(5).Number & vbNewLine & "+" & playerDeck(5).Addition

PlayerDeck7.Text = playerDeck(6).Type & vbNewLine & playerDeck(6).Colour & vbNewLine & "No." & playerDeck(6).Number & vbNewLine & "+" & playerDeck(6).Addition

PlayerDeck8.Text = playerDeck(7).Type & vbNewLine & playerDeck(7).Colour & vbNewLine & "No." & playerDeck(7).Number & vbNewLine & "+" & playerDeck(7).Addition

playedCardBox.Text = playedCard.Type & vbNewLine & playedCard.Colour & vbNewLine & "No." & playedCard.Number & vbNewLine & "+" & playedCard.Addition

displayCard(filename, counter)

CardDisplay1.ImageLocation = filename

counter = counter + 1

displayCard(filename, counter)

CardDisplay2.ImageLocation = filename

counter = counter + 1

displayCard(filename, counter)

CardDisplay3.ImageLocation = filename

counter = counter + 1

displayCard(filename, counter)

CardDisplay4.ImageLocation = filename

counter = counter + 1

displayCard(filename, counter)

CardDisplay5.ImageLocation = filename

counter = counter + 1

displayCard(filename, counter)

CardDisplay6.ImageLocation = filename

counter = counter + 1

displayCard(filename, counter)

CardDisplay7.ImageLocation = filename

counter = counter + 1

displayCard(filename, counter)

CardDisplay8.ImageLocation = filename

counter = counter + 1

displayCentreDeck(filename)

DisplayCentreCard.ImageLocation = filename

NPCdeckCount1.Text = NPC1Count

NPCdeckCount2.Text = NPC2Count

NPCdeckCount3.Text = NPC3Count

End Sub

'Toggle visibilty of buttons

Sub visibility(ByRef visible As Integer)

PlayerCard1.Visible = visible

PlayerCard2.Visible = visible

PlayerCard3.Visible = visible

PlayerCard4.Visible = visible

PlayerCard5.Visible = visible

PlayerCard6.Visible = visible

PlayerCard7.Visible = visible

PlayerCard8.Visible = visible

takeCard.Visible = visible

End Sub

'Toggle visibilty of buttons

'------------------------------------------------------------------------------------------

Sub gamePlay(ByRef gameFin As Boolean)

playerDeckCount = playerDeckCount - 1

While visible = False And NPC1Count <> 0 And NPC2Count <> 0 And NPC3Count <> 0

Call AIPLay1()

If played = True Then

NPC1Count = NPC1Count - 1

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

losegame()

End If

Call AIPLay2()

If played = True Then

NPC2Count = NPC2Count - 1

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

losegame()

End If

Call AIPLay3()

If played = True Then

NPC3Count = NPC3Count - 1

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

losegame()

End If

visible = True

Call visibility(visible)

End While

End Sub

Function losegame()

If NPC1Count = 0 Or NPC2Count = 0 Or NPC3Count = 0 Then

If NPC1Count = 0 Then

MsgBox("You lose")

'Connect to database

Dim SQLReader As OleDbDataReader

Dim connection\_type As String = "Provider=Microsoft.ACE.OLEDB.12.0;"

Dim file\_location As String = "Data Source=N:\Advanced Computing\AH1\test.accdb"

Dim conn As OleDbConnection

conn = New OleDbConnection(connection\_type & file\_location)

conn.Open()

'Convert stored data to SQL query - edited from book!!!

Dim query As String = "INSERT INTO [Players] VALUES ( 'Null' , '" & NPC1 & "', '1');"

'Insert partial data

'Dim query As String = "INSERT INTO [customers] (firstname, surname) VALUES ( ' " & firstname & " ', ' " & surname & "');"

'Execute the built query

Dim command As New OleDbCommand(query, conn)

SQLReader = command.ExecuteReader()

End

ElseIf NPC2Count = 0 Then

MsgBox("You lose")

'Connect to database

Dim SQLReader As OleDbDataReader

Dim connection\_type As String = "Provider=Microsoft.ACE.OLEDB.12.0;"

Dim file\_location As String = "Data Source=N:\Advanced Computing\AH1\test.accdb"

Dim conn As OleDbConnection

conn = New OleDbConnection(connection\_type & file\_location)

conn.Open()

'Convert stored data to SQL query - edited from book!!!

Dim query As String = "INSERT INTO [Players] VALUES ( 'Null' , '" & NPC2 & "', '1');"

'Insert partial data

'Dim query As String = "INSERT INTO [customers] (firstname, surname) VALUES ( ' " & firstname & " ', ' " & surname & "');"

'Execute the built query

Dim command As New OleDbCommand(query, conn)

SQLReader = command.ExecuteReader()

End

ElseIf NPC3Count = 0 Then

MsgBox("You lose")

'Connect to database

Dim SQLReader As OleDbDataReader

Dim connection\_type As String = "Provider=Microsoft.ACE.OLEDB.12.0;"

Dim file\_location As String = "Data Source=N:\Advanced Computing\AH1\test.accdb"

Dim conn As OleDbConnection

conn = New OleDbConnection(connection\_type & file\_location)

conn.Open()

'Convert stored data to SQL query - edited from book!!!

Dim query As String = "INSERT INTO [Players] VALUES ( 'Null' , '" & NPC2 & "', '1');"

'Insert partial data

'Dim query As String = "INSERT INTO [customers] (firstname, surname) VALUES ( ' " & firstname & " ', ' " & surname & "');"

'Execute the built query

Dim command As New OleDbCommand(query, conn)

SQLReader = command.ExecuteReader()

End

ElseIf playerDeckCount = 0 Then

MsgBox("You win")

End

End If

End If

End Function

'------------------------------------------------------------------------------------------

Private Sub PlayerCard1\_Click(sender As Object, e As EventArgs) Handles PlayerCard1.Click

Dim counter As Integer = 0

playCardCard = playerDeck(counter)

playCard(playCardCard, playedCard, counter, played)

If played = True Then

playerDeck(counter).Addition = ""

playerDeck(counter).Type = ""

playerDeck(counter).Colour = ""

playerDeck(counter).Number = ""

visible = False

Call visibility(visible)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

Call gamePlay(gameFin)

End If

End Sub

Private Sub PlayerCard2\_Click(sender As Object, e As EventArgs) Handles PlayerCard2.Click

Dim counter As Integer = 1

playCardCard = playerDeck(counter)

playCard(playCardCard, playedCard, counter, played)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

If played = True Then

playerDeck(counter).Addition = ""

playerDeck(counter).Type = ""

playerDeck(counter).Colour = ""

playerDeck(counter).Number = ""

visible = False

Call visibility(visible)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

Call gamePlay(gameFin)

End If

End Sub

Private Sub PlayerCard3\_Click(sender As Object, e As EventArgs) Handles PlayerCard3.Click

Dim counter As Integer = 2

playCardCard = playerDeck(counter)

playCard(playCardCard, playedCard, counter, played)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

If played = True Then

playerDeck(counter).Addition = ""

playerDeck(counter).Type = ""

playerDeck(counter).Colour = ""

playerDeck(counter).Number = ""

visible = False

Call visibility(visible)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

Call gamePlay(gameFin)

End If

End Sub

Private Sub PlayerCard4\_Click(sender As Object, e As EventArgs) Handles PlayerCard4.Click

Dim counter As Integer = 3

playCardCard = playerDeck(counter)

playCard(playCardCard, playedCard, counter, played)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

If played = True Then

playerDeck(counter).Addition = ""

playerDeck(counter).Type = ""

playerDeck(counter).Colour = ""

playerDeck(counter).Number = ""

visible = False

Call visibility(visible)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

Call gamePlay(gameFin)

End If

End Sub

Private Sub PlayerCard5\_Click(sender As Object, e As EventArgs) Handles PlayerCard5.Click

Dim counter As Integer = 4

playCardCard = playerDeck(counter)

playCard(playCardCard, playedCard, counter, played)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

If played = True Then

playerDeck(counter).Addition = ""

playerDeck(counter).Type = ""

playerDeck(counter).Colour = ""

playerDeck(counter).Number = ""

visible = False

Call visibility(visible)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

Call gamePlay(gameFin)

End If

End Sub

Private Sub PlayerCard6\_Click(sender As Object, e As EventArgs) Handles PlayerCard6.Click

Dim counter As Integer = 5

playCardCard = playerDeck(counter)

playCard(playCardCard, playedCard, counter, played)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

If played = True Then

playerDeck(counter).Addition = ""

playerDeck(counter).Type = ""

playerDeck(counter).Colour = ""

playerDeck(counter).Number = ""

visible = False

Call visibility(visible)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

Call gamePlay(gameFin)

End If

End Sub

Private Sub PlayerCard7\_Click(sender As Object, e As EventArgs) Handles PlayerCard7.Click

Dim counter As Integer = 6

playCardCard = playerDeck(counter)

playCard(playCardCard, playedCard, counter, played)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

If played = True Then

playerDeck(counter).Addition = ""

playerDeck(counter).Type = ""

playerDeck(counter).Colour = ""

playerDeck(counter).Number = ""

visible = False

Call visibility(visible)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

Call gamePlay(gameFin)

End If

End Sub

Private Sub PlayerCard8\_Click(sender As Object, e As EventArgs) Handles PlayerCard8.Click

Dim counter As Integer = 7

playCardCard = playerDeck(counter)

playCard(playCardCard, playedCard, counter, played)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

If played = True Then

playerDeck(counter).Addition = ""

playerDeck(counter).Type = ""

playerDeck(counter).Colour = ""

playerDeck(counter).Number = ""

visible = False

Call visibility(visible)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

Call gamePlay(gameFin)

End If

End Sub

'------------------------------------------------------------------------------------------

'AI PLay

Private Sub AIPLay1()

Dim counter As Integer

deck = NPC1Deck

Call AIplayCard(playCardCard, playedCard, counter, played)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

End Sub

Private Sub AIPLay2()

Dim counter As Integer

deck = NPC2Deck

Call AIplayCard2(playCardCard, playedCard, counter, played)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

End Sub

Private Sub AIPLay3()

Dim counter As Integer

deck = NPC3Deck

Call AIplayCard3(playCardCard, playedCard, counter, played)

Call displayDecks(playerDeck, NPC1Deck, NPC2Deck, NPC3Deck, playedCard)

End Sub

'------------------------------------------------------------------------------------------

'WRITE TO DATABASE

Private Sub Button10\_Click(sender As Object, e As EventArgs) Handles Button10.Click

Dim boolea1 As Boolean

If PlayerDeck1.Visible = True Then

boolea1 = False

Else

boolea1 = True

End If

PlayerDeck1.Visible = boolea1

PlayerDeck2.Visible = boolea1

PlayerDeck3.Visible = boolea1

PlayerDeck4.Visible = boolea1

PlayerDeck5.Visible = boolea1

PlayerDeck6.Visible = boolea1

PlayerDeck7.Visible = boolea1

PlayerDeck8.Visible = boolea1

playedCardBox.Visible = boolea1

txtConsole.Visible = boolea1

Label3.Visible = boolea1

End Sub

End Class

**FORM2**

Imports System.Data.OleDb

Public Class Form1

Public Structure playerDetails

Dim wins As Integer

Dim playerTag As String

Dim playerName As String

End Structure

Dim myForm2 As New mainPlayspace

Public players(50) As playerDetails

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

myForm2.Show()

Me.Hide()

End Sub

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Dim counter As Integer

'Connect to database

Dim SQLReader As OleDbDataReader

Dim connection\_type As String = "Provider=Microsoft.ACE.OLEDB.12.0;"

Dim file\_location As String = "Data Source=N:\Advanced Computing\AH1\test.accdb"

Dim conn As OleDbConnection\

conn = New OleDbConnection(connection\_type & file\_location)

conn.Open()

'Select and display results

Dim query As String = "Select \* FROM [Players]"

Dim command As New OleDbCommand(query, conn)

SQLReader = command.ExecuteReader()

If SQLReader.HasRows Then

While SQLReader.Read

players(counter).wins = SQLReader("Wins")

players(counter).playerName = SQLReader("PlayerName")

players(counter).playerTag = SQLReader("playerTag")

counter = counter + 1

End While

Else

MsgBox("this file isnt valid")

End If

End Sub

Private Sub Button2\_Click(sender As Object, e As EventArgs) Handles Button2.Click

Dim counter As Integer

ListBox1.Items.Clear()

Do

ListBox1.Items.Add(players(counter).wins & " " & players(counter).playerTag & " " & players(counter).playerName)

counter = counter + 1

Loop Until players(counter).wins = 0 And players(counter).playerTag = ""

End Sub

Private Sub Button3\_Click(sender As Object, e As EventArgs) Handles Button3.Click

Dim n As Integer

Dim counter As Integer

Dim endarray As Boolean

Dim player As playerDetails

endarray = False

Do

If players(counter).playerTag = "" Then

endarray = True

End If

counter = counter + 1

Loop Until counter = 50 Or endarray = True

n = counter

Dim swapped As Boolean

swapped = True

While swapped And n >= 0

swapped = False

For counter = 0 To n - 2

If players(counter).wins < players(counter + 1).wins Then

player = players(counter)

players(counter) = players(counter + 1)

players(counter + 1) = player

swapped = True

End If

Next counter

n = n - 1

End While

End Sub

Private Sub Label5\_Click(sender As Object, e As EventArgs) Handles Label5.Click

End Sub

**RESEARCH**

Researching what language to implement in

Initially I had decided upon HTML, javascript and javascript, as previously I had made a hangman and card match program but due to the lack of experience with it, then I considered implementing it in python but due to visual basics easy UI tools and the expertise with it compared to visual basic I finally decided to go forward with that.

Deciding upon implementation style

The code was originally going to be a text-based version of UNO similar to a battleships game I coded in python but once I came across how to implement picture boxes with code I felt it was a simple decision which I feel has created quite a nice final product, the code is still primarily text-based but the picture boxes make it easier to understand as a user.

Coding and Technical aspects

I learned;

* How to alter picture boxes with code
* How to randomise arrays
* How to use SQL to connect a database to code
* How to change the visibility of objects with code
* How to implement multiple forms in visual basic
* How to transfer variables across forms
* How to design UNO cards using serif software

**ONGOING TESTING**

Ongoing testing is crucial in ensuring that a software program works as intended and meets the requirements and expectations of its users. It allows developers to identify and fix bugs, errors, and other issues that may arise during the development process or after the software has been released.

**TESTING**

**Functional Requirements**

An external database containing player details will have to be created

Procedure to read previous players’ details from access to an array

Procedure to get player inputs and assign them to an array

Procedure to bubble sort this array

Procedure to randomise NPCs’ names and move the cards from the central class to each array

Procedure to allow the user to place a card and run a method associated with a button press

Procedure to have other players play a card

Procedure to determine winner

Procedure to export the player’s details and names from the array to update access document

Procedure to display from an array

**Test Plan**

To test these requirements of my program I’ve set up the following tests

***Click of View leaderboard button***

This test includes screenshots demonstrating it working

Displays leaderboard demonstrating

Procedure to read previous players’ details from access to an array

***Click of Bubble sort button***

This test includes screenshots demonstrating it working

Bubble sorts leaderboard demonstrating

Procedure to bubble sort this array

***Click of Start button***

This test includes multiple test data of

A table displaying the NPC1 Deck

A table displaying the NPC2 Deck

A table displaying the NPC3 Deck

A table of NPC Names

A table displaying the Player Deck

Starts game demonstrating

Procedure to get player inputs and assign them to an array

Procedure to randomise NPCs’ names and move the cards from the central class to each array

***Persona Test***

This test includes in depth detailing of each play and result in the game

Runs through a game demonstrating

Procedure to allow the user to place a card and run a method associated with a button press

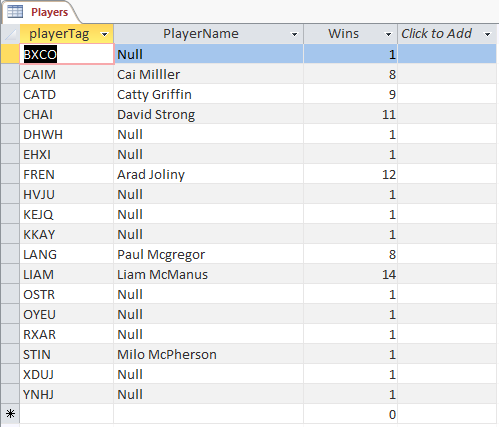
Procedure to have other players play a card

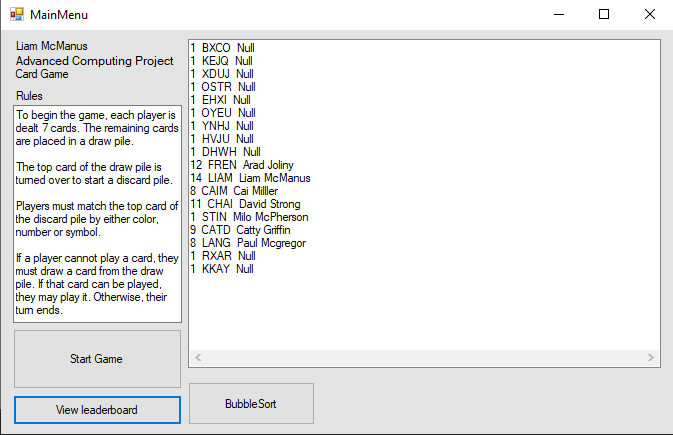
Procedure to determine winner

Procedure to export the player’s details and names from the array to update access document

Procedure to display from an array

**SCREENSHOTS**

External Database

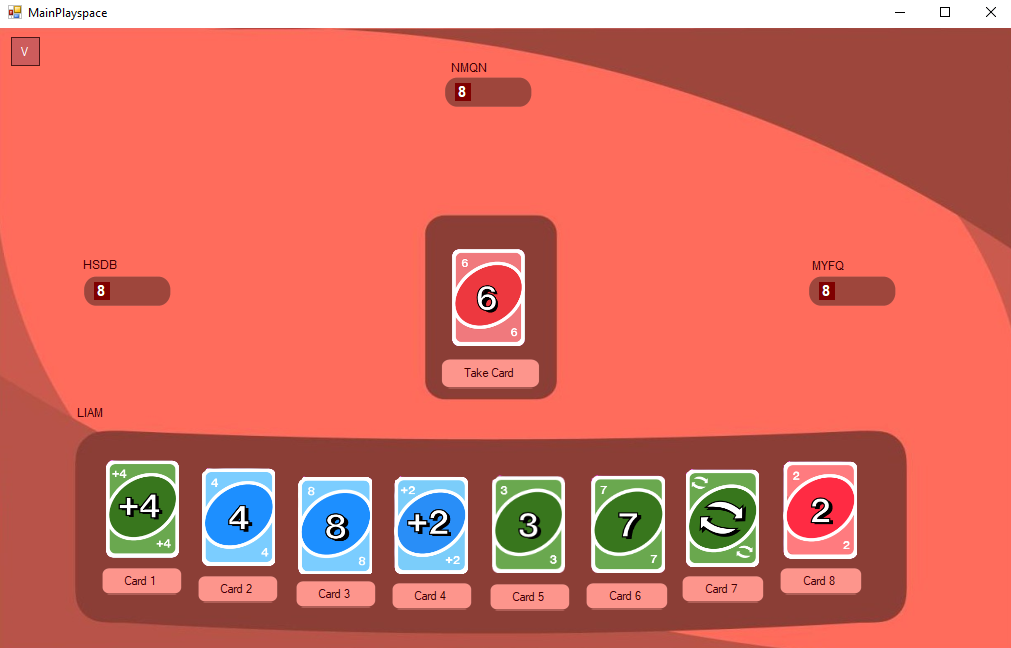
Read previous players’ details from access to an array

get player inputs and assign them to an array

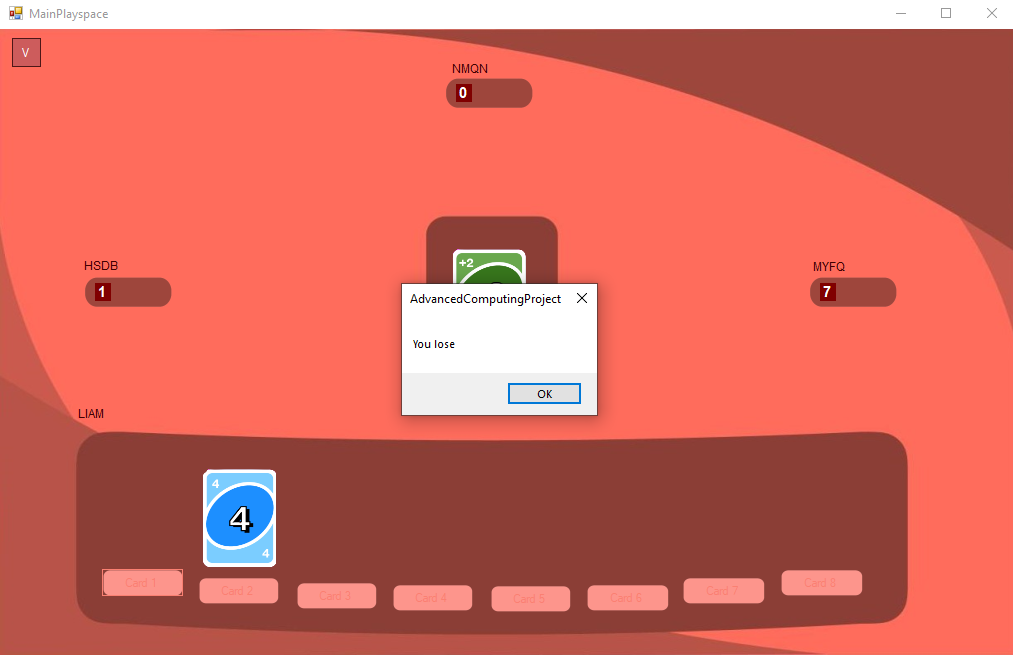
Get player inputs

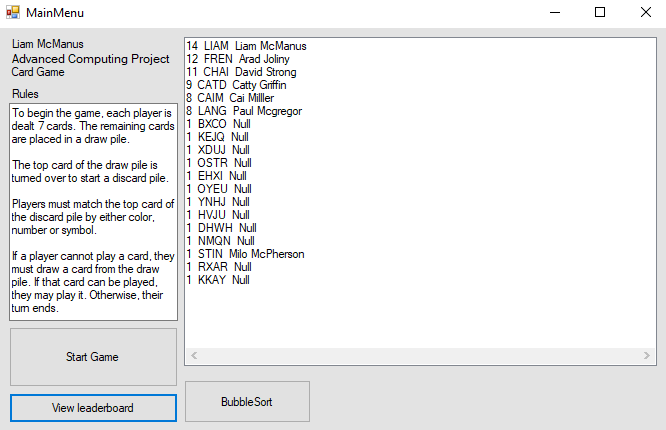
randomise NPCs’ names and move the cards from the central class to each array

Procedure to allow the user to place a card and run a method associated with a button press



Procedure to have other players play a card

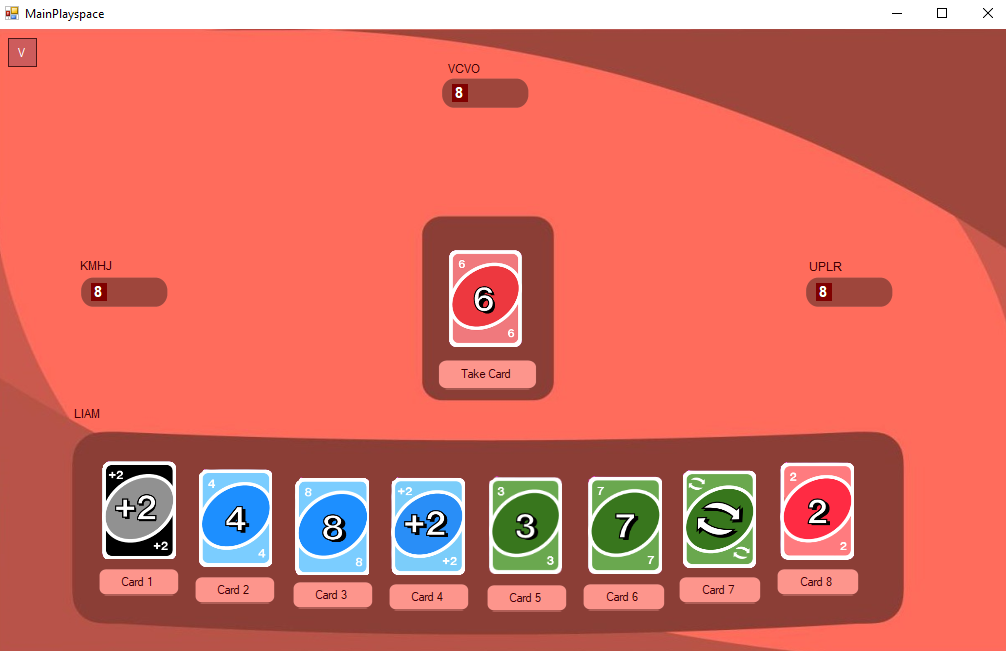
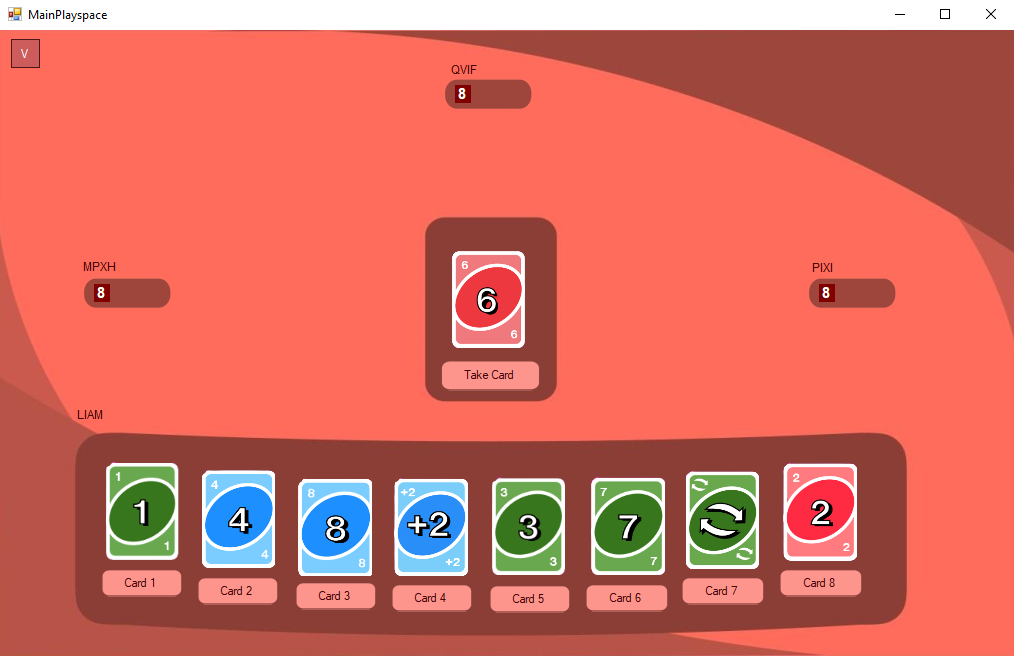
Procedure to determine winner

Procedure to bubble sort

**TESTING TABLE**

**Randomising Player Array**

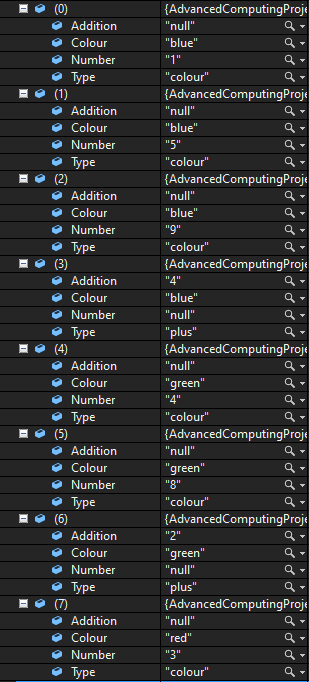
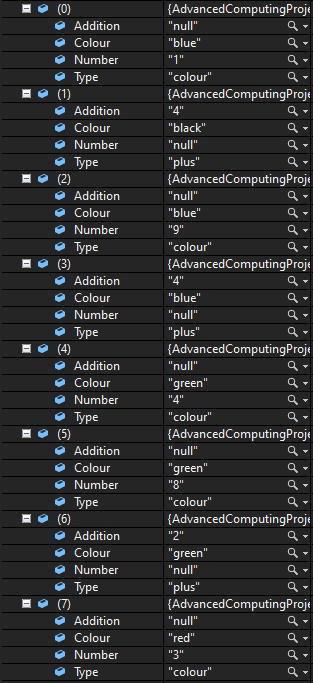
| ***CARD 1*** | ***CARD 2*** | ***CARD 3*** | ***CARD 4*** | ***CARD 5*** | ***CARD 6*** | ***CARD 7*** | ***CARD 8*** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Green reverse | Blue 4 | Blue 8 | Blue plus 2 | Green reverse | Green reverse | Green reverse | Red 2 |
| Blue 4 | Blue 3 | Blue plus 2 | Blue plus 2 | Green 3 | Green 7 | Green reverse | Blue plus 2 |
| Yellow 5 | Green 2 | Blue 8 | Blue plus 2 | Green 3 | Green 7 | Green 7 | Green reverse |
| Green 3 | Blue 4 | Green plus 2 | Green 7 | Green 2 | Green 7 | Green reverse | Red 2 |
| Black plus 2 | Blue 4 | Blue 8 | Green 2 | Green 2 | Red 2 | Red 2 | Red 2 |
| Red 8 | Blue 4 | Blue 8 | Blue plus 2 | Green 2 | Green 7 | Green reverse | Green 2 |
| Black plus 2 | Blue 4 | Blue 8 | Blue plus 2 | Green 3 | Green 7 | Green reverse | Red 2 |



This test shows the randomising deck and moving from the centre deck to each independent array. Although this test data is repetitive it works well enough for the program. I will talk about this later in more depth in the evaluation. The next three tests are to test the procedure to move these cards into the NPC decks.

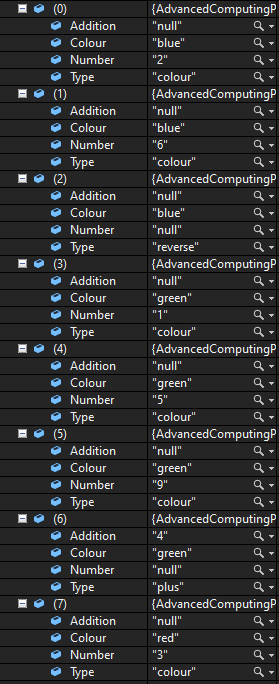
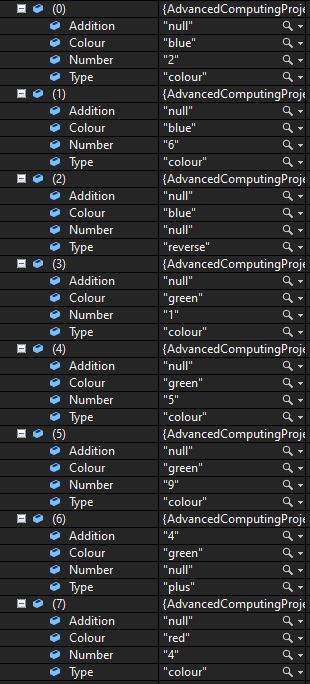
**Randomising NPC1 Card Array**

| ***CARD 1*** | ***CARD 2*** | ***CARD 3*** | ***CARD 4*** | ***CARD 5*** | ***CARD 6*** | ***CARD 7*** | ***CARD 8*** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Blue 1 | Black plus 2 | Blue 9 | Blue 9 | Green 4 | Green 8 | Green plus 2 | Red 3 |
| Blue 1 | Blue 5 | Blue 9 | Blue plus 4 | Green 4 | Green 8 | Green plus 2 | Red 3 |
| Blue 1 | Red 4 | Blue 9 | Blue plus 4 | Green 4 | Blue plus 4 | Green plus 4 | Red 3 |
| Blue 4 | Blue 1 | Blue 9 | Black plus 2 | Green 4 | Black plus 2 | Green 1 | Red plus 2 |



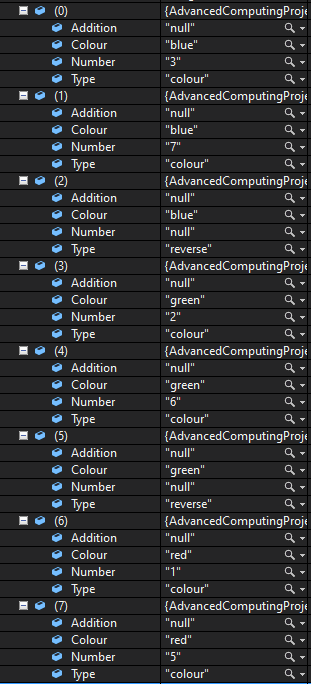
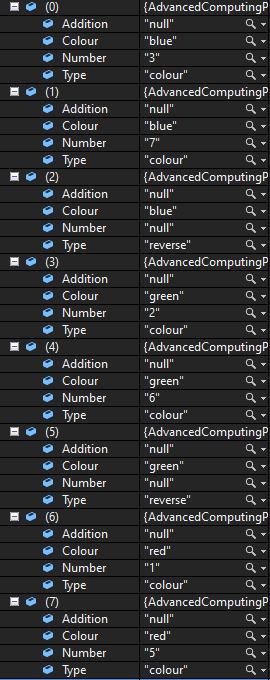
**Randomising NPC2 Card Array**

| ***CARD 1*** | ***CARD 2*** | ***CARD 3*** | ***CARD 4*** | ***CARD 5*** | ***CARD 6*** | ***CARD 7*** | ***CARD 8*** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Blue 2 | Blue 5 | Blue 8 | Blue reverse | Green 5 | Green 9 | Red 2 | Green 9 |
| Blue 2 | Blue 6 | Blue reverse | Green 1 | Green 5 | Green 9 | Green plus 4 | Red 4 |
| Blue 2 | Blue 6 | Blue reverse | Green 1 | Green 5 | Green 9 | Green plus 4 | Red 3 |
| Blue 2 | Blue 5 | Blue 8 | Blue reverse | Green 5 | Green 9 | Red 2 | Green 9 |

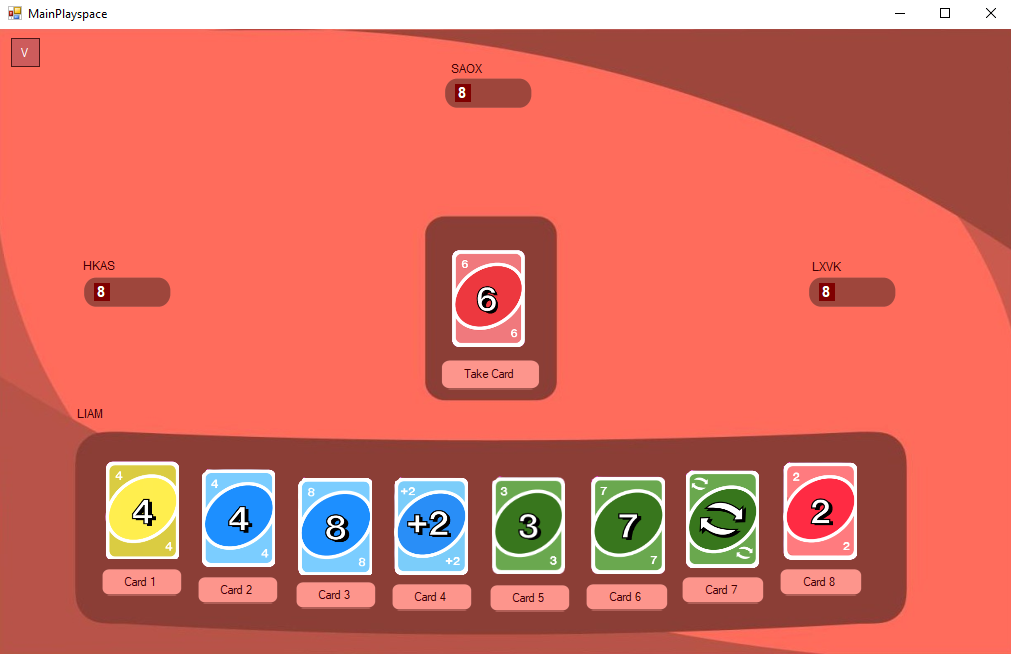


**Randomising NPC3 Card Array**

| ***CARD 1*** | ***CARD 2*** | ***CARD 3*** | ***CARD 4*** | ***CARD 5*** | ***CARD 6*** | ***CARD 7*** | ***CARD 8*** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Blue 3 | Blue 7 | Blue reverse | Green 2 | Green 6 | Green reverse | Red 1 | Red 5 |
| Blue 3 | Blue 7 | Blue reverse | Green 2 | Green 6 | Green reverse | Red 1 | Red 5 |
| Blue 3 | Blue 7 | Blue reverse | Green 2 | Green 6 | Green reverse | Red 1 | Red 5 |
| Blue 3 | Blue 7 | Blue reverse | Green 2 | Green 6 | Green reverse | Red 1 | Red 5 |

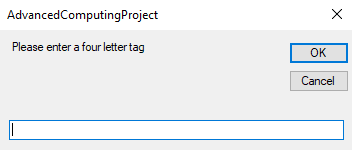


**Randomising NPC Names**

****

| ***NPC 1*** | ***NPC 2*** | ***NPC 3*** |
| --- | --- | --- |
| BMON | MDES | LPTV |
| HKAS | SOAX | LXVK |
| VQNM | HGCR | GTRU |
| FUHL | LQJP | KEYS |

**Validate Tag Input**



| ***INPUT*** | ***EXPECTED OUTPUT*** | ***ACTUAL OUTPUT*** |
| --- | --- | --- |
| LIAM | Continues code without msg box popping up. | Continues code without msg box popping up. |
| Liam | Continues code and changes tag to uppercase | Continues code and changes tag to uppercase |
| LIAM2 | Continues code without msg box popping up. | Continues code without msg box popping up. |
| Liam2 | Msg box appears telling you to input again | Msg box appears telling you to input again |
| 1111 | Continues code without msg box popping up. | Continues code without msg box popping up. |

This code runs through the input validation of the tag input as you can see above I've tested Valid, Extreme and Exceptional data.

Test Case

**Persona - Alan Fordyce**

Alan Fordyce was born in a small town, Killin, in Scotland in the mid-1970s. He grew up in a working-class family and was the youngest of three siblings. His father was a factory worker, and his mother worked part-time as a cleaner. As a child, Alan has never been good with technology and struggles to work simple programs.

| ***INPUT*** | | ***EXPECTED OUTPUT*** | | ***ACTUAL OUTPUT*** |
| --- | --- | --- | --- | --- |
| Alan loads up the program and instantly clicks start. | | The program loads to the main playspace and does the associated procedures | | The program loads to the main playspace and does the associated procedures. |
| ***THE GAME STARTS*** | | | | |
| **CARD PLAYED** | **NPC1 COUNT** | **NPC2 COUNT** | **NPC3 COUNT** | **CENTRE DECK** |
| Red 2 | 8 | 8 | 8 | Red 6 |
| Green 3 | 7 | 7 | 7 | Blue 3 |
| Green 7 | 6 | 6 | 6 | Green 2 |
| Green Reverse | 5 | 5 | 5 | Green 6 |
| Blue Reverse | 4 | 4 | 3 | Blue 7 |
| Green 4 | 3 | 3 | 5 | Green 3 |
| Blue 3 | 2 | 2 | 4 | Green Reverse |
| Take Card | 1 | 1 | 3 | Green Plus 4 |
| ***Alan has finished the game losing NPC1 called TVSA has won*** | | | | |

Test Case

**Persona - Harry Squatter**

Harry's entire life has been influenced by technology; it started with his parents, who were both powerful coders. They were members of the Order of the Programmers, a secret society of software engineers and developers who fought against the evil Voldewart. Harry is very familiar with computers and digital games.

| ***INPUT*** | | ***EXPECTED OUTPUT*** | | ***ACTUAL OUTPUT*** |
| --- | --- | --- | --- | --- |
| Harrys loads up the program and clicks the display leaderboard button. | | The program displays the leaderboard. | | The program displays the leaderboard. |
| Harry then clicks the bubble sort button. | | This bubble sorts the sorted array but doesn't yet display it. | | This bubble sorts the sorted array but doesn't yet display it. |
| Harry once again clicks the display leaderboard button. | | The leaderboard refreshes showing the updated leaderboard. | | The leaderboard refreshes showing the updated leaderboard. |
| Harry clicks start. | | The program loads to the main playspace and does the associated procedures. | | The program loads to the main playspace and does the associated procedures. |
| ***THE GAME STARTS*** | | | | |
| **CARD PLAYED** | **NPC1 COUNT** | **NPC2 COUNT** | **NPC3 COUNT** | **CENTRE DECK** |
| Red 2 | 8 | 8 | 8 | Red 2 |
| Blue 4 | 7 | 7 | 7 | Blue 7 |
| Blue 8 | 6 | 6 | 6 | Blue Reverse |
| Blue Plus 4 | 9 | 5 | 5 | Green 2 |
| Blue Reverse | 8 | 4 | 5 | Green 6 |
| Green 7 | 7 | 3 | 5 | Green 3 |
| Take Card | 6 | 2 | 4 | Blue 4 |
| Take Card | 5 | 1 | 3 | Green Plus 4 |
| ***Harry has finished the game losing NPC2 called GREG has won*** | | | | |
| ***INPUT*** | | ***EXPECTED OUTPUT*** | | ***ACTUAL OUTPUT*** |
| Harry reloads the program and views the database. | | Database display showing TVSA as 1 win | | Database display showing TVSA as 1 win |

**USER GUIDE**

BASIC INFORMATION AND RULES

To begin the game, each player is dealt 7 cards. The remaining cards are placed in a draw pile. The top card of the draw pile is turned over to start a discard pile. Players must match the top card of the discard pile by either colour, number or symbol. If a player cannot play a card, they must draw a card from the draw pile. If that card can be played, they may play it. Otherwise, their turn ends.

USER INTERFACE

The program utilises a very easy to understand user interface which allows the program to be used by even the most inexperienced computer users to play.

**HOW TO PLAY**

On load up you will be presented with the following options;

* VIEW LEADERBOARD

This allows you to view the leaderboard including previous winners.

* BUBBLE SORT

This sorts the leaderboard to win descending but you will have to refresh the leaderboard to see the board change.

* START GAME

This starts the main game.

On click of start game;

* INPUT YOUR NAME

This name will be your name shown on the leaderboard if you win.

* INPUT PLAYER TAG

This tag will be your main identifier if you win

Then you load into the main playspace;

* PLAY CARD

Wait for other players to play then

* PLAY CARD AGAIN

This repeats until end of game

**EVALUATION**

Fitness for purpose

**ORIGINAL REQUIREMENT SPECIFICATION**

This program is a digital version of the card game UNO or switch. This links to a database through SQL and creates a leaderboard of the winners which is then sorted and accessed through the program using bubble sort.

This leaderboard is imported into Visual Basic using a SQL line from the program. On import the record is then ordered by wins descending showing the winner at the top this is shown in a box on the starting page. This starting screen also shows a start button, a description of the rules and as said before a window to access the leaderboard, on the press of the start button the program brings up another menu which will have the user input their name and a 4-letter tag once they input these the game will start.

The game is based on a record system with each player being a different object of a record with procedures adding on cards. The game will proceed as normal with each player having a card removed and added; this will all be randomly chosen out of the NPC’s decks; these cards will have individual values associated with them dictating which card they can be placed on and what they can do. When it comes to the playable characters' turn they are given an option of what card to choose and the method associated with that card is carried out. This continues until either the main player is out or everyone else is out. This is then output to the database the name of the winner.

**IMPROVEMENTS**

Although my code works as intended it could be improved, due to the way I designed it, a crucial flaw has arisen in the fact you can only take 8 cards which although not detrimental to the program still changes the way the game is played from the physical edition this could be changed later on without serious effort recoding the program although I believe the program works well enough currently.

The randomising of the centre deck array could also use a change this is one of the things I learnt individually, You will notice in all my testing once this centre deck is divided into its individual hands it gets quite repetitive on rerun. Once again this doesn’t completely break the program but its something to keep in mind for further development.

Although the UI in game is quite easy to understand I might have to add in tutorials to make it easier for older generations to understand this has been shown in my Alan Fordyce test.

In the end, although it could be used, my program does meet all its functional and end-user requirements and plays through a game of UNO.

**MATCHING DESIGN**

The program matches the design almost perfectly in some points of pseudocode. I was quite vague but I would still say it matches the pseudocode. The program has some added features like in the UI i've included a button called V which toggles visibility on my annotation boxes, this helps solve errors in the code and implement new parts.

**TESTING RESULTS**

The testing varies really, the initial testing with input validation and name generating worked perfectly as intended but the randomising array function didn't work as intended it left a lot of repeated values which could probably be solved quite easily but I just couldn’t manage in the time frame set out although since its not detrimental to the program I believe this could be changed in the future. The persona testing worked perfectly as intended with the player eventually losing.

**MAINTAINABILITY AND ROBUSTNESS**

The program is completely modular using procedures and functions and it makes good use of internal commentary which makes it easier for other developers to pick up and edit my original code. A problem that appears is the repetition in my code which can make it annoying to edit although in these cases I believe it was necessary.