Programming for Computing

Assignment 2 (60% of the Year)

"My Juke Box"

The objective of this assignment is for you to create an application using Visual Studio 2017 and C#, which is to emulate a "PC Juke Box". Due to the complexity of this application an executable example called, "My Juke Box", is available for download and execution from the course Blackboard site. Please compare your application's functionality, with the version on the blackboard for completeness. For this task you will need to use most of the procedures and functions you have been shown and that have been demonstrated throughout the year.

This assignment consists of four sections, each of which are briefly described here and then presented in more detail in the remaining sections of this document. The first section focuses on creating the main Graphical User Interface (GUI) of the "My Juke Box". The second section is for adding the functionality of selecting 'Genre' and 'Tracks' from the main GUI. The third is the Version Control showing your understanding of a generic concept and the fourth and final section, is for those of you who like a challenge and are more familiar with Visual Studio. This section is called, "Advanced Programming" where you will be asked to create a secondary window called "Setup", which will enable the user (**NOT** you!) to add and remove 'Genre' and 'Sound Tracks'.

You should submit your completed assignment in e-format via "Blackboard", as stated in the "Hand In" section of this document.

Please Note.

This is an individual assignment and **NOT** a group task. Should you require additional information, please contact your me using the following:

E : P.O'Neill@shu.ac.uk

T : 0114 225 6990

Alternatively, ask me during one of your scheduled tutorial sessions, or in one of the drop in sessions that will run in place of the scheduled Lectures.

Section 1. The Interface – (10 marks)

Create the Software Application's Main Interface

Write a program in C# to emulate a Juke Box, the application should be called "My Juke Box", like the sample application I have created and is available to you on Blackboard.

No.	Task	Available Marks
1,	Create the Graphical User Interface (GUI), which contains all of the elements as shown on page 2 of this document, Figure 1.	10

(You may use your own choice of background image.)

The GUI.

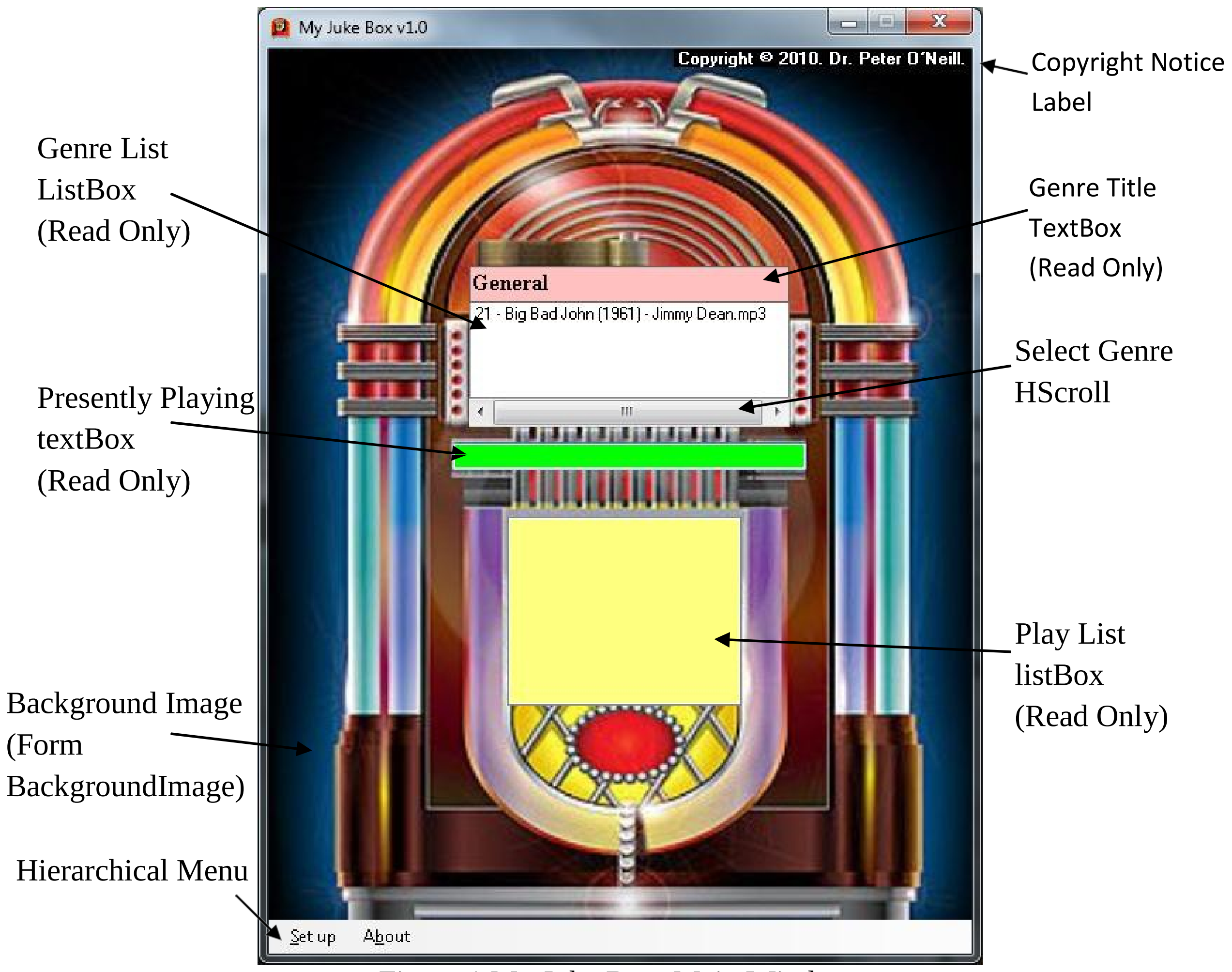

My Juke Box v1.0
Copyright © 2010. Dr. Peter O'Neill.
Copyright Notice Label
Genre List ListBox (Read Only)
Genre Title TextBox (Read Only)
General
21 - Big Bad John (1961) - Jimmy Dean.mp3
Select Genre HScroll
Presently Playing textBox (Read Only)
Play List listBox (Read Only)
Background Image (Form BackgroundImage)
Hierarchical Menu
Set up
About

Figure 1 My Juke Box, Main Window.

<u>Section 2. The Programs Non-Functional & Functional Procedures – (60 marks)</u>

All of the functionality outlined in **<u>Section 2</u>**, are based on the "My Juke Box" interface shown in Figure 1 above, unless stated otherwise.

<u>Please Note.</u> Many additional attributes and elements of source code will be required to make these tasks work correctly. Therefore, you should use your own initiative by writing your own code to ensure your application's functionality.

Non-Functional Procedures

No.	Task	Available Marks
2,	Create an array of listBox/es or a 2 dimensional String Array to contain the information required by the "My Juke Box" application, about the Genre? (This was explained in the Lecture, about this Assignment!)	5

No.	Task	Available Marks
3,	Write a procedure, which reads from a file, the ‘Genre Title’ and each of the ‘Tracks’ into one of the array’s suggested in Task 2.	20

Functional Procedures

No. Task **Available Marks**

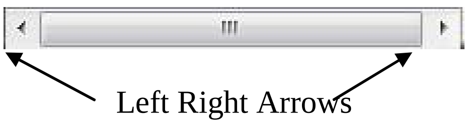
4, On double-clicking a ‘Track’ on the “Genre List”, have it that the ‘Track’ is copied to the “Play List”. (Add the selected item to the “Play List”.) — 5

5, When the “Play List” contains a ‘Track’ and :

- No ‘Track’ is playing, have the first item move from the “Play List” into the “Presently Playing” TextBox and start to play the track.
- a ‘Track’ is playing, have the Juke Box wait until the present track stops playing and then have the first item from the “Play List” move into the “Presently Playing” textBox and play that track.

15

6, On selecting the left or right arrows on the “Select Genre” horizontal scroll bar as shown here :

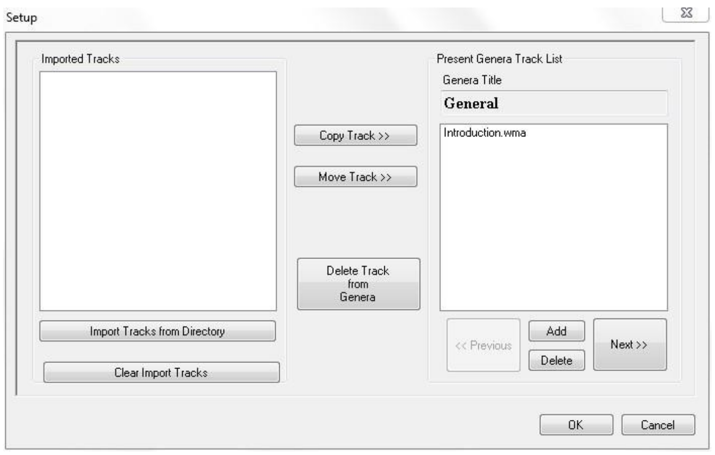


have the “Genre Title” and “Genre List” updated with the next or previous Genre.

5

7, Implement the “Menu” and all of its functional activities.

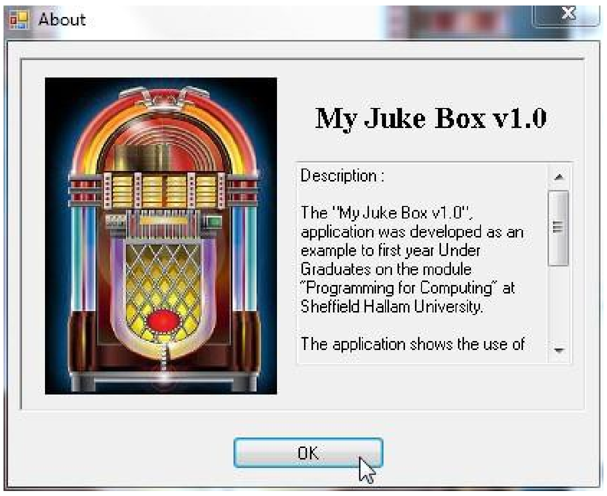
a) Clicking on ‘Setup’, have the following interface displayed:



5

Figure 2, Setup Window.
(An enlarged version of this is available on page 5.)

b) Clicking on ‘About’ on the menu, have the following interface display:



Description :

The "My Juke Box v1.0", application was developed as an example to first year Under Graduates on the module "Programming for Computing" at Sheffield Hallam University.

The application shows the use of

Figure 3, About Window.

5

on the clicking of the ‘OK’ button have it close the ‘About Window’, returning the focus to the Main Window depicted in Figure 1, of this document.

Section 3. Version Control – (20 marks)

GitHub is widely used by software houses, particularly were a considerable number of software developers are collaborating on the same project, (in most cases these days), but it can also be useful when one person is developing as you can revert to previous versions.

No.	**Task**	**Mark**
8.	In this section we would like you to keep SubVersions of your source code while you are writing it. A minimum of 15 "Commits" to the SVN is required. You will need to invite both Jan and me, using the email addresses in this document.	20

(Please see Black Board for instructions on setting this up.)

.

Section 4. Advanced Programming – (10 marks)

All of the functionality outlined in **Section 4**, are based on the "Setup Window", shown and described here:

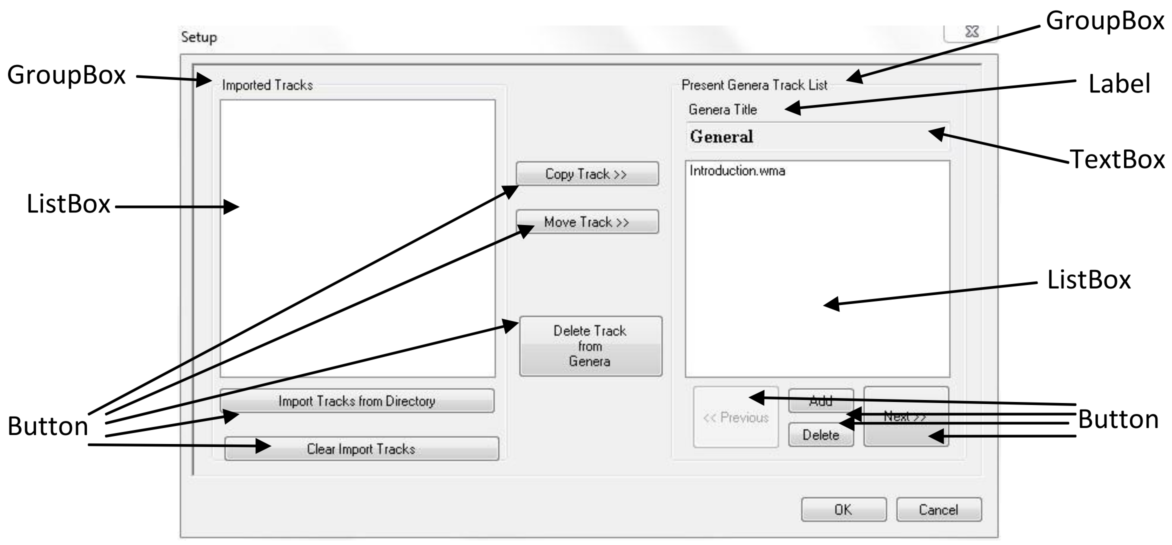


Figure 4, Setup Window with Description.

No.	Task	Available Marks
8,	Populate the selected 'Genre' from the configuration file. (my example is in the sub-directory 'Media')	5
9,	On selecting the buttons contained within the "Present Genre Track List", have them undertake the functions as suggested by their title and shown in the example application.	2
10,	Create the functionality to save the new configuration file (see 'Hints and Tips' at the end of the document for the code for the "Import Tracks from Director"), when the user selects the "OK" button.	3

(See my example for ideas on what is required.)

Handing In via Blackboard

You are to Hand in a:

- Word Document.
- Zip file.

The Word Document will contain:

1. a screenshots of your running program.
2. the code from your .cs file.
3. Information to direct us to the "GitHub", e.g., Username.

Your Word Document should be titled <Your Full Name Student Number Assignment Designation.docx>, e.g., in my case, "**O'Neill P 209244 Ass2.docx**".

Inside the Word Document, you should have <Your Full Name and Student Number Assignment Designation> in the Header of each page and in the 'Footer' should have 'Programming for Computing' and the page number.

In Windows, you can copy a shot of the active window by pressing <Shift> plus <Prt-sc> keys

Your Zip files will contain:

- Your Visual Studio 2017 Project directory, which should be a zip file that will include all the source code in Visual Basic and the executable version of your application.

This is an all-electronic hand-in. Work will **NOT** be marked if handed-in on paper.

Cheating. This is an individual assignment - I will report any cheating to the examinations board.

Hand In Date: By the date and time stated on Black Board

You will hand in the required files on, or before the date and time stated on Black Board. Please see the video under the "Assignment" button on Blackboard, for a complete description on the method to submit your work. Alternatively, if you are still having difficulties in knowing what or how to submit, please come and see me.

In order to pass this module you are required to score a minimum of 40% in each assignment. Failing to accomplish this mark can result in you being required to, 'Resit' a single assignment, or undertake the complete year and in some cases, being asked to leave the University. (Please see the University's Procedures for more information.)

Hints and Tips

Ask if you get stuck - I won't tell you the answer, but I will guide you !

E : P.O'Neill@shu.ac.uk

T : 0114 225 6990

General Guidance

Break the application into it smaller parts, the 'Marking' scheme above should give you some ideas.

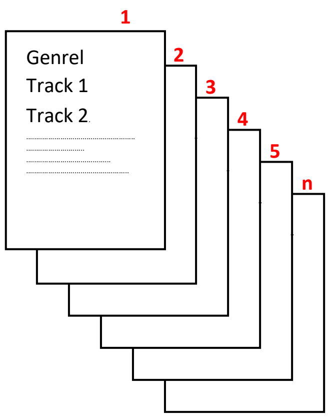
Develop a single part, but think how this part you are developing, influences other parts of your application.

Use "Pseudo Code", or some other design notation to work out the design of your application. If you do not complete your application, you can submit the design and I can give you some credit for your understanding of the problem.

Conceptual Hints

In Memory

In the memory we can create an array of ListBox and conceptually this is how we can represent them:

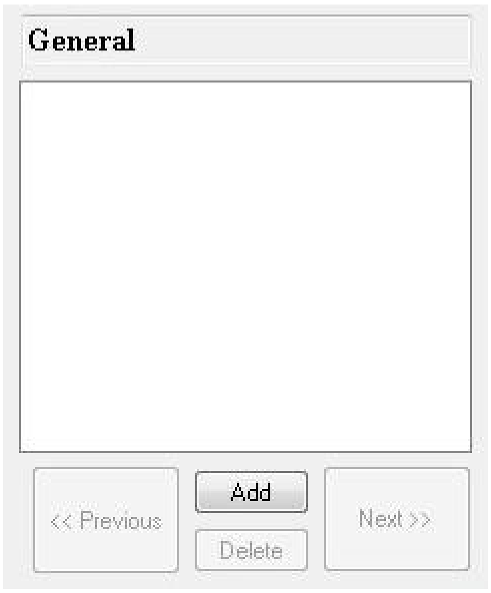


Int_Number_of_Genre

Below, I have created a function, which I call when the user selects the "Next" or "Previous" buttons on the interface shown on the following page. I increment or decrement the number "Int_Show_Genre_Number" if it is less than the last list "Int_Number_of_Genre", or greater than -1, e.g., is the first list and then I send it to the function as shown here:

```
Populate_Genre_List(Int_Show_Genre_Number)
```

On Interface



Useful Source Code

I am here providing you with the global variables, that I used within my code and the function calls, I make at the start of the application:

```
// Include the Class to enable Read and Write to Disk functions
Imports System.IO


        // Gets the string of the path to the media files for the application
        public string StrApplicationMediaPath = Directory.GetCurrentDirectory();
        // Holds the number of Genre, which becomes the number of columns in the array
        public int Int_Number_of_Genre;

        // Create memory allocation for an array of the ListBox/es
        ListBox[] Media_Library;
        // Tells the JukeBox it is, or not, Playing
        bool IsPlaying = false;

        private void JukeBox_Shown(object sender, EventArgs e)
        {
            if (Load_Media_Lists() == false)
            {
                MessageBox.Show("Unable to load the 'Media Content'.");
                Load_Default();
            }

            if (Initailise() == false)
            {
                MessageBox.Show("Unable to display the 'Media Content'.");
                Close();
            }
        }
```

Initially, I declare an array of type 'ListBox', which consist of one (`ListBox[] Media_Library;`). However, in the `Load_Media_Lists()` method I initiate this array type, so we can have additional 'Genre', this also means that the declared type remains a global:

```
Media_Library = new ListBox[Int_Number_of_Genre];
```

and create each one as required with the line :

```
' Create an array of ListBox/es
Media_Library[genre) = new ListBox();
```

Below again is useful bit of code for the "Advanced Programming" section, but remember the components are what I have called them, e.g., my 'Import Tracks from Directory' is called, "`Btn_Import_Tracks`" as shown in the code below:

Button Import Tracks in Setup Window

```
private void btn_Import_Tracks_Click(object sender, EventArgs e)
{
    // Tells the application that something has changed
    bool_Requires_Saving = true;
    // Let the user select the directory the music is comming from
    if( folderBrowserDialog1.ShowDialog() == DialogResult.OK)
    {
       // Populates the list array with all the files with the stated extension
       foreach (string file in _
       Directory.EnumerateFiles(folderBrowserDialog1.SelectedPath,"*.*",_
       SearchOption.AllDirectories).Where(s => s.EndsWith(".mp3") || _
       s.EndsWith(".wma") || s.EndsWith(".wav") || s.EndsWith(".MP3") || _
       s.EndsWith(".WMA") || s.EndsWith(".WAV")))
              {
                    lst_Imported.Items.Add(file);
                }

                if(lst_Imported.Items.Count > -1)
                {
                    btn_Import_Tracks.Enabled = false;
                }
                else
                {
                    btn_Import_Tracks.Enabled = true;
                }
            }
        }
```