Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни «Об’єктно-орієнтоване програмування»

ДОВІДНИК МЕЛОМАНА

Керівник, ас. каф ПІ, Ляпота В.М.

Студент гр. ПІ-15-2 Черкасов А. С.

Комісія:

проф. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дудар З.В.

проф. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бондарев В.М.

Харків 2016

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**

Кафедра: ***Програмної інженерії***

Дисципліна: Об’єктно-орієнтоване програмування

Спеціальність: ***Програмна інженерія***

Курс 1 .Група *ПІ-15-2* . Семестр 2 .

***ЗАВДАННЯ***

***на курсовий проект студента***

***Черкасова Андрія Сергійовича***

(Прізвище, Ім'я, По батькові)

1. Тема проекту:

Довідник меломана

1. Термін здачі студентом закінченого проекту: ***“28” - травня - 2016 р***.
2. Вихідні дані до проекту:

*Специфікація програми, методичні вказівки до виконання курсової роботи*

4. Зміст розрахунково-пояснювальної записки:

*Вступ, специфікація програми, проектна специфікація, інструкція , висновки.*

1. Перелік графічного матеріалу

*Екранні форми, схема об’єктної моделі, діаграми класів*  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми | 1-03-2016 р. |
| 2 | Аналіз предметної області | 1-03-2016 – 15-03-2016 р. |
| 3 | Розробка постановки задачі | 15-03-2016 – 25-03-2016 р. |
| 4 | Розробка об’єктної моделі | 25-03-2016 – 10-04-2016 р. |
| 5 | Кодування програмної системи | 10-04-2016 – 5-05-2016 р. |
| 6 | Тестування і доопрацювання розробленої програмної системи. | 5-05-2016 – 20-05-2016 р. |
| 7 | Оформлення пояснювальної записки | 20-05-2016 – 27-05-2016 р. |
| 8 | Публічний захист проекту перед комісією | 28-05-2016 р. |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ляпота В.М.

(Прізвище, Ім'я, По батькові)

« 1 »\_березня\_\_ 2016 р.

РЕФЕРАТ / ABSTRACT

Пояснювальна записка: 18 с., 20 рис., 5 джерел, 1 додаток.

Мета роботи: закріплення знань, отриманих під час вивчення дисципліни «Об’єктно-орієнтовне програмування», шляхом розробки програмної системи під назвою «Довідник меломана».

Методи розробки: Microsoft VisualStudio Enterprise 2015, Windows Forms, .NET Framework 4.6

В результаті розробки отримана програмна система під назвою «Довідник меломана» для роботи з інформаційною базою даних, чию роль виконують \*.txt файли. Розроблена програма дає швидкий та зручний доступ до пісень та виконувачів, які існують в базі, надає можливість додавати, видаляти та редагувати пісні. Також є можливості пошуку по пісням, створення та редагування власних списків програвання.

КУРСОВА РОБОТА, ДОВІДНИК МЕЛОМАНА, БАЗА, ДОВІДНИК, ПРОГРАММА.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 6

1 СПЕЦИФИКАЦИЯ ПРОГРАММЫ 8

1.1 Основные функции 8

1.2 Работа с файлами 11

1.3 Функция поиска. Меню «Help» и «About» 13

2 ПРОЕКТНАЯ СПЕЦИФИКАЦИЯ 17

2.1 Объектная модель программы 17

2.2 Реализация функций программы 18

3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 21

ВЫВОДЫ 23

ПЕРЕЧЕНЬ ССЫЛОК 24

Приложение А Частичный код программы 25

ВВЕДЕНИЕ

«Музыка – смысл моей жизни», – сказал хорватский певец Восакреч Йедрна. Таковой она является не только для него. Миллионы людей по всему земному шару повторяют эту фразу, ведь даже не обязательно быть музыкантом, для того, чтобы музыка играла большую роль в жизни отдельно взятого человека. Музыка – это древнейшее искусство. Она не ограничена ничем, кроме фантазии композитора. С ее помощью люди многие века грустят, радуются, задумываются о чем-то, отдыхают, танцуют. Разумеетя, за тысячелетия существоавния и процветания музыкального искусства появилось практически несметное количество композиций, которые не помешало бы систематизировать и структурировать. И, несомненно, у каждого человека свои вкусы, каждому нравятся разные музыкальные жанры и исполнители. Именно поэтому возникает необходимость в программном обеспечении, в котором каждый человек сможет легко и просто получить доступ к той музыке, которую он любит.

В данной курсовой работе требуется разработать программу «Справочник меломана». Целью данной курсовой работы является разработка программы-справочника, с помощью которого можно добавлять, удалять и редактировать исполнителей и их песни, осуществлять поиск по ним с разными критриями, создавать, октрывать и сохранять плей-листы. За основу взят объектно-ориентированный подход.

Задачи выполнения работы:

1. исследование предметной области с целью выявления основных принципов данной сферы;
2. проектирование иерархии классов, интерфейсов, взаимодействия компонентов на основе выделенных принципов и данных средств;
3. использование встроенные элементы среды для структуризации классов и оптимизации кода;
4. применение принципов объектно-ориентированной парадигмы к классам;
5. реализовать программное взаимодействие с базой данной формата \*.txt

Объектно-ориентированный подход требует глубокого понимания основных принципов, или, иначе, концепций, на которых он базируется. В данном подходе основными концепциями являются понятия объектов и классов.

Объектно-ориентированное программирование в настоящее время является абсолютным лидером в области прикладного программирования. Использование этого подхода предоставляет программисту широкие возможности в функциональности и в сопровождаемости проекта.

В качестве основного инструмента разработки применяется Microsoft Visual Studio 2015 Enterprise. Visual Studio представляет собой интегрированную среду разработки программ, созданную корпорацией Microsoft. Язык программирования C#.

1 СПЕЦИФИКАЦИЯ ПРОГРАММЫ

* 1. Основные функции

Работа с программой начинается с главного окна (рис. 1.1). Интерфейс программы прост в использовании и интуитивно понятен. Это является необходимым условием при разработке пользовательских интерфейсов.

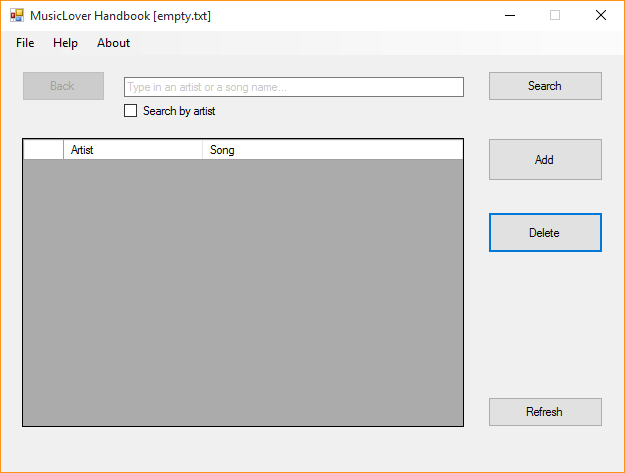


Рисунок 1.1 – Главное окно

Нажав на вкладку «Add», пользователь перейдет к окну с полями заполнения исполнителя и названия песни (рис. 1.2.1, 1.2.2).

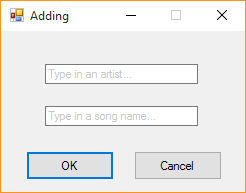


Рисунок 1.2.1 – Добавление песни

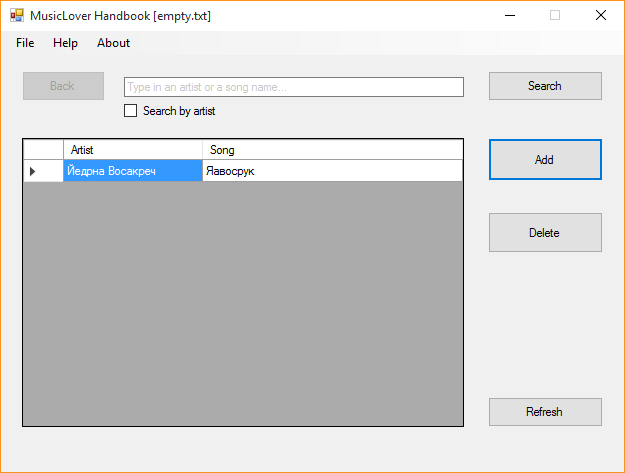


Рисунок 1.2.2 – Добавление песни

Двойной клик на добавленной песне открывает схожее окно редактирования (рис. 1.3).

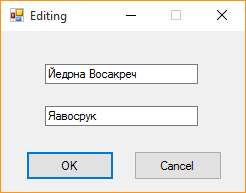


Рисунок 1.3 – Редактирование песни

Кнопка «Delete» служит для удаления выбранной песни

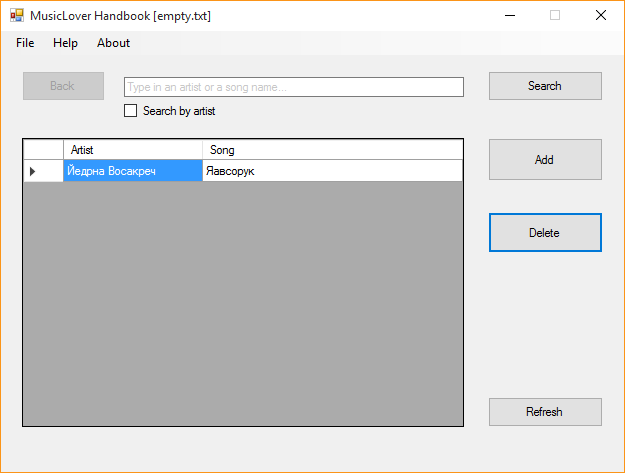


Рисунок 1.4 – Удаление песни

* 1. Работа с файлами

Вкладка «File» содержит в себе подвкладки «Open song list…», «Save» и «Save song list as…». Последняя служит для сохранения плей-листа с указанием пути и названия (рис. 1.5.1, 1.5.2).

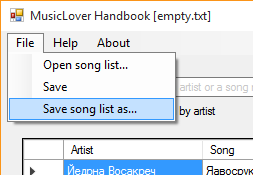


Рисунок 1.5.1 – Сохранение плей-листа

Кнопка «Save» сохраняет файл с собственным названием, не меняя его расположения. После успешной операции сохранения появляется форма с соответствующим уведомлением (рис. 1.5.3).

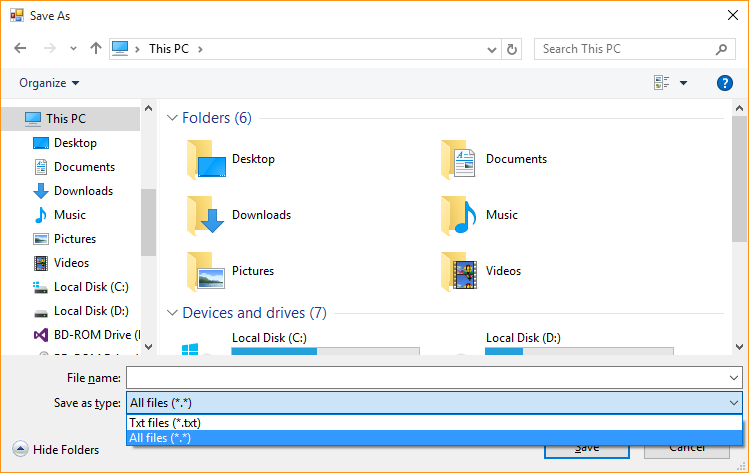


Рисунок 1.5.2 – Сохранение плей-листа

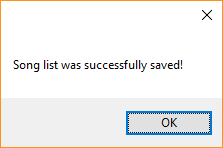


Рисунок 1.5.3 – Сохранение плей-листа

Для того, чтобы октрыть сохраненный или заранее созданный плей-лист, нужно нажать на кнопку «Open song list…» (рис 1.6.1, 1.6.2).

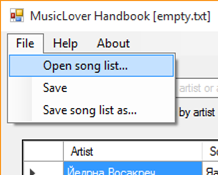


Рисунок 1.6.1 – Открытие плей-листа

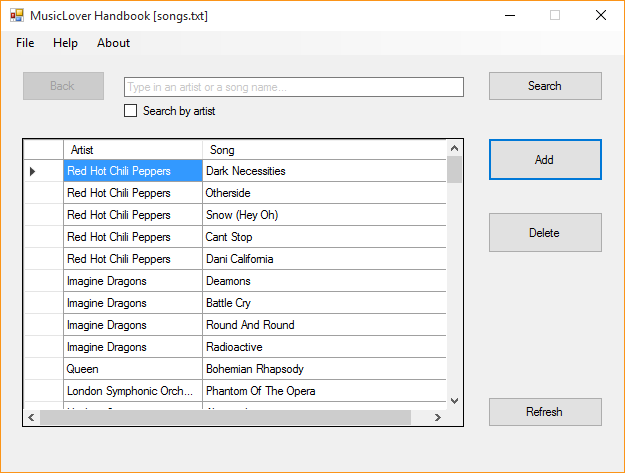


Рисунок 1.6.2 – Открытие плей-листа

* 1. Функция поиска. Меню «Help» и «About»

Для поиска по песням плей-листа предусмотрена соответствующая функция, вызываемая нажатием клависи «Search» (рис. 1.7.1). Она формирует плей-лист из песен, подходящих под критерии поиска. Можно заметить, что после её нажатия становится активна кнопка «Back». Она предусмотрена для того, чтобы после поиска пользователь по желанию мог вернуться к предыдущему плей-листу (рис. 1.7.2).

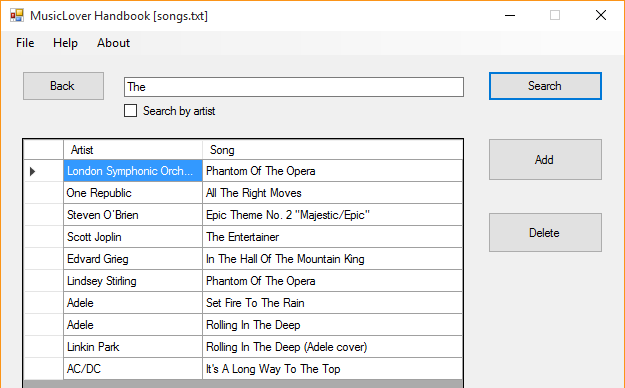


Рисунок 1.7.1 – Поиск песен

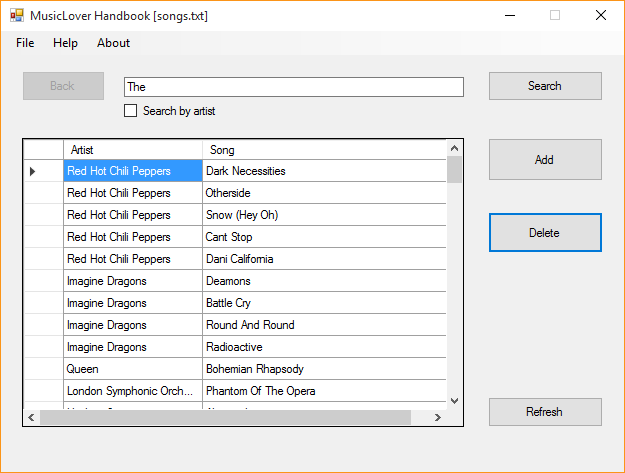


Рисунок 1.7.2 – Возвращение к предыдущему плей-листу

Если вам понравилась конкретная группа, на помощь придёт флажок «Search by artist». Если он потавлен, поиск будет вестись по исполнителю, а не по названию (рис. 1.8)

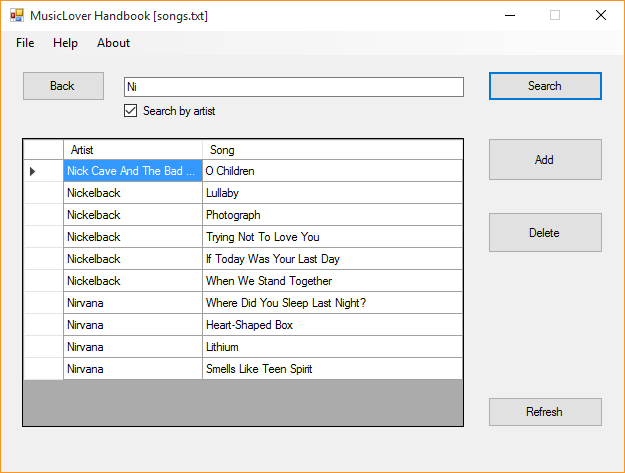


Рисунок 1.8 – Поиск песен по исполнителю

В верхнем меню также располагаются ещё 2 кнопки – «Help» и «About». Они вызывают справку и информацию о программе соответственно (рис. 1.9.1, 1.9.2). В справке содержится очень краткие и понятные инструкции для использования MusicLover Handbook.

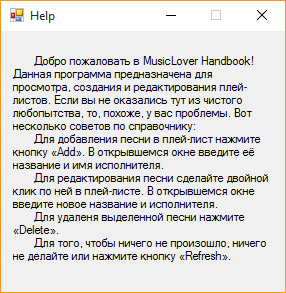


Рисунок 1.9.1 – Форма «Help»

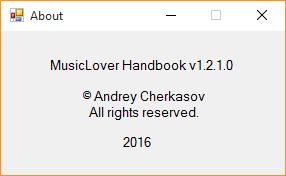


Рисунок 1.9.2 – Информация о программе

1. ПРОЕКТНАЯ СПЕЦИФИКАЦИЯ
   1. Объектная модель программы

Основным конструктивным элементом в языках ООП является модуль (module), представляющий собой логически связанную совокупность классов и объектов, а не подпрограмм, как в более ранних языках. В графическом интерфейсе данной программы содержится форма заполнения и редактирования данных, следовательно в проекте присутствуют специальные вспомогательные функции, которые считывают данные, заполняют коллекции, сохраняют данные.

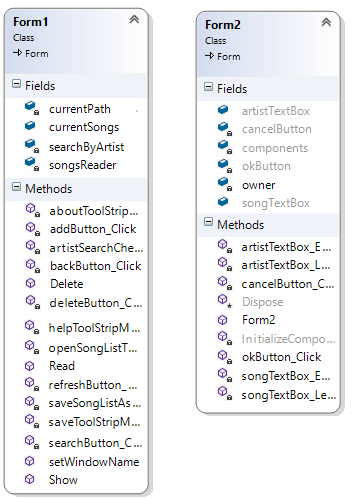


Рисунок 2.1 – Краткая схема программы

На рисунке 2.1 представлена краткая схема программы. Большая часть функционала заключена в двух формах: Form1 отвечает за такие функции программы, как удаление, поиск, открытие и сохранение, в то время как Form2 помогает с добавлением и редактированием.

* 1. Реализация функций программы

Как упоминалось ранее, данная программа состоит из нескольких компонентов, которые тесно взаимосвязаны друг с другом. В этом разделе будут рассмотрены ключевые функции. Полный набор методов можно увидеть в коде программы (приложение А).

Скелетной основой программы является словарь, типа строка – коллекция строк. Он чрезвычайно удобен для данных в программе: они представляют собой пару исполнитель – песня, поэтому любую песню легко можно найти по исполнителю, что особенно важно для поиска. Ниже представлены методы для считывания и отображения данных в элементе dataGridView1:

public void Read(TextReader reader, Dictionary<string, List<string>> dictionary)

{

dictionary.Clear();

string line;

while ((line = reader.ReadLine()) != null)

{

string artist = line.Split('↕')[0];//alt+18

string song = line.Split('↕')[1];

try

{

dictionary[artist].Add(song);

}

catch (KeyNotFoundException ex)

{

dictionary.Add(artist, new List<string> { song });

}

}

}

Рисунок 2.6 – Чтение из файла в словарь

public void Show(Dictionary<string, List<string>> dictionary)

{

dataGridView1.Rows.Clear();

foreach (string artist in dictionary.Keys)

{

foreach (string song in dictionary[artist])

{

dataGridView1.Rows.Add(artist, song);

}

}

}

Рисунок 2.7 – Добавление нового элемента

Также одним из важнейших методов программы являются методы добавления и удаления объектов. Первый открывает новую форму для введения данных

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

{

Form2 x = new Form2(this);

x.Owner = this;

x.ShowDialog();

x.Text = "Adding";

}

public partial class Form2 : Form

{

Form1 owner;

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

{

if (this.Text == "Editing")

{

owner.Delete();

}

try

{

owner.currentSongs[artistTextBox.Text].Add(songTextBox.Text);

}

catch (KeyNotFoundException ex)

{

owner.currentSongs.Add(artistTextBox.Text, new List<string> { songTextBox.Text });

}

owner.dataGridView1.Rows.Add(artistTextBox.Text, songTextBox.Text);

this.Close();

}

Рисунок 2.9 – Добавление

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

{

Delete();

}

//Used in deleteButton\_Click and in the Form2 in okButton\_click

public void Delete()

{

int i = dataGridView1.SelectedCells[0].RowIndex;

string artist = dataGridView1.Rows[i].Cells[0].Value.ToString();

string song = dataGridView1.Rows[i].Cells[1].Value.ToString();

currentSongs[artist].Remove(song);

dataGridView1.Rows.RemoveAt(i);

}

Рисунок 2.10 – Удаление

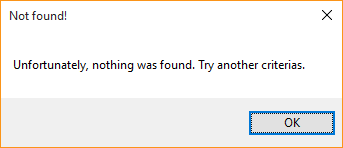
3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

При первом запуске программы открывается пустой файл empty.txt, то есть, в нашем плей-листе сейчас нет песен. Самое время добавить парочку! Для этого нужно нажать на клавишу «Add», расположенную в правой части окна. Появилось новое окно, в котором Вас попросят ввести название песни и исполнителя. Для добавления песни нажмите «ОК».

Итак, Вы добавили первую песню. Но, безусловно, бывает так, что при наборе была допущена ошибка, так и оставшаяся незамеченной до отображения песни в плей-листе. Или, быть может, вы открыли плей-лист человека, который придумал своим песням незамысловатые названия вроде «1111» и «Песня». Не беда! Двойной клик по песне открывает окно редактирования информации, набранной в предыдущем пункте, что позволяет исправить эти недоразумения.

Возможно, вам покажется долгим добавление большого количества песен. Тут на помощь приходит вышеупомянутая функция открытия готового плей-листа. Возможно, у вас есть друг, у которого схожий музыкальный вкус. Попросите его предоставить свой файл плей-листа. Для того, чтобы открыть его, нажмите кнопку в верхнем меню «File» и в выпадающем меню выберете «Open song list…». В следующем окне укажите путь к файлу. Теперь Вы можете отредактировать его так, как хотите этого сами – добавить что-то своё, удалить ненужное. Кстати, насчёт удаления. Кнопка удаления расположена прямо под кнопкой добавления. Для удаления песни из плей-листа выберите её левой кнопкой мыши и нажмите «Delete».

Понятное дело, что чем больше плейлист, тем тяжелее в нём ориентироваться. Для этого реализована функция поиска. Чтобы искать по названию песни, введите то, что хотите найти в текстовое поле и нажмите на клавишу «Search», расположенную в правом верхнем углу экрана (над кнопкой добавления). Функция поиска формирует новый плей-лист из открытого с учётом критериев поиска. Для возврата в основной плей-лист нажмите клавишу «Back». Также в MusicLover Handbook присутствует возможность поиска по исполнителю. Для этого нужно проделать всё то же самое, что и для обычного поиска, поставив флажок «Search by artist». Если программа не найдёт ваш запрос в плей-листе, будет показано соответствующее окно:



После того, как Вы отредактируете свой плей-лист, не забудьте сохранить его. Для этого в верхнем меню «File» выберете опцию «Save». Она сохранит изменения в файле плей-листа. Для того, чтобы выбрать имя записываемого файла и его директорию, выберете пункт меню «Save song list as…» и укажите нужные данные.

ВЫВОДЫ

Результатом выполнения данной курсовой работы стала программа – «Справочник меломана». Данная программа предназначена для удобного отображения песен и навигации по ним.

В проекте представлен объектно-ориентированный подход в проектировании программного обеспечения информационного плана, дающий возможность на ранних этапах разработки учесть все нюансы будущей программы, необходимый набор функций, состав и структуру баз данных, что в дальнейшем исключает необходимость переработки уже написанных компонентов программы.

В программе присутствует простой и понятный пользовательский интерфейс, так что полностью разобраться во функционале программы не составит труда даже для неопытного пользователя.

Это приложение разработано для того, чтобы каждый мог создать свой собственный плей-лист с целью его редактирования.

Планы на будущее: добавить возможность импорта плей-листа из сайта «Вконтакте», возможность отображения большего количества информации по песни, а именно длительность и текст (опционально), реализовать функции добавления самих песен для проигрывания, расширенного поиска в режиме реального времени, сделать версии для мобильных платформ, нарисовать более приятный графический интерфейс.

ПЕРЕЧЕНЬ ССЫЛОК

1. Бондарев, В.М. Объектно-ориентированное программирование на С# [Текст]: учеб. пособ. /В.М. Бондарев. – Х.: Компания СМИТ, 2009 – 224 с.
2. Буч, Г. Объектно-ориентированный анализ и проектирование [Текст]: пер. с англ – М.: 000 "И.Д. Вильяме", 2008 - 720 с.
3. Мандер, Т. Разработка пользовательского интерфейса [Текст]: пер. с англ. – М.: ДМК Пресс, 2008 – 412 с.
4. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е изд. [Текст]: пер. с англ. – М.: Вильямс, 2013 – 1312 с.
5. Шилдт, Г. C# 4.0.: Полное руководство [Текст]: пер. с англ. – М.: Вильямс, 2011 – 1056 с.

Приложение А

Частичный код программы

Класс Form1

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

setWindowName();

using (songsReader = new StreamReader(currentPath))

{

Read(songsReader, currentSongs);

}

Show(currentSongs);

}

//dictionary for storing current list of songs (artist - song)

public Dictionary<string, List<string>> currentSongs = new Dictionary<string, List<string>>();

//var for reading a file

TextReader songsReader;

//specifies whether search goes on artists or songs

bool searchByArtist = false;

//current file path

string currentPath = "empty.txt";

public void setWindowName()

{

try

{

this.Text = "MusicLover Handbook [" + currentPath.Substring(currentPath.LastIndexOf("\\") + 1) + "]";

}

catch (ArgumentOutOfRangeException ex)

{

this.Text = "MusicLover Handbook [" + currentPath + "]";

}

}

#region Read

/// <summary>

/// Puts data from the file to the dictionary

/// </summary>

/// <param name="reader">A file that we are reading from</param>

/// <param name="dictionary">Dictionary that we are writing to</param>

public void Read(TextReader reader, Dictionary<string, List<string>> dictionary)

{

dictionary.Clear();

string line;

while ((line = reader.ReadLine()) != null)

{

string artist = line.Split('↕')[0];//alt+18

string song = line.Split('↕')[1];

try

{

dictionary[artist].Add(song);

}

catch (KeyNotFoundException ex)

{

dictionary.Add(artist, new List<string> { song });

}

}

}

#endregion

#region Show

/// <summary>

///Represents dictionary in the dataGridView

/// </summary>

/// <param name="dictionary">Dictionary that we need to show</param>

public void Show(Dictionary<string, List<string>> dictionary)

{

dataGridView1.Rows.Clear();

foreach (string artist in dictionary.Keys)

{

foreach (string song in dictionary[artist])

{

dataGridView1.Rows.Add(artist, song);

}

}

}

#endregion

#region deleteButton\_Click

/// <summary>

/// Deletes selected element from current song list

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

Delete();

}

//Used in deleteButton\_Click and in the Form2 in okButton\_click

public void Delete()

{

int i = dataGridView1.SelectedCells[0].RowIndex;

string artist = dataGridView1.Rows[i].Cells[0].Value.ToString();

string song = dataGridView1.Rows[i].Cells[1].Value.ToString();

currentSongs[artist].Remove(song);

dataGridView1.Rows.RemoveAt(i);

}

#endregion

#region refreshButton\_Click

/// <summary>

/// Refill dataGridView with the current song list (originally designed to check whether deletion was right or not)

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

dataGridView1.Rows.Clear();

Show(currentSongs);

}

#endregion

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

{

Form2 x = new Form2(this);

x.Owner = this;

x.ShowDialog();

x.Text = "Adding";

}

#region openSongListToolStripMenuItem\_Click

/// <summary>

/// Displays a dialog to open a song list file

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

Stream stream = null;

OpenFileDialog dialog = new OpenFileDialog();

dialog.InitialDirectory = "c:\\";

dialog.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

dialog.FilterIndex = 2;

dialog.RestoreDirectory = true;

if (dialog.ShowDialog() == DialogResult.OK)

{

try

{

if ((stream = dialog.OpenFile()) != null)

{

using (stream)

{

using (songsReader = new StreamReader(dialog.FileName))

{

Read(songsReader, currentSongs);

Show(currentSongs);

currentPath = dialog.FileName;

setWindowName();

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Error: Could not read file from disk. Original error: " + ex.Message);

}

}

}

#endregion

//edites a song by double-clicking the row cell

private void dataGridView1\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

Form2 x = new Form2(this);

x.Visible = true;

x.Text = "Editing";

int i = dataGridView1.SelectedCells[0].RowIndex;

string artist = dataGridView1.Rows[i].Cells[0].Value.ToString();

string song = dataGridView1.Rows[i].Cells[1].Value.ToString();

x.artistTextBox.Text = artist;

x.songTextBox.Text = song;

x.artistTextBox.ForeColor = Color.Black;

x.songTextBox.ForeColor = Color.Black;

}

//changes the value of searchByArtist

private void artistSearchCheckBox\_CheckedChanged(object sender, EventArgs e)

{

searchByArtist = !searchByArtist;

}

#region searchButton\_Click

/// <summary>

/// Searches song list by artist or song name

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

Dictionary<string, List<string>> searchResult = new Dictionary<string, List<string>>();

foreach (string artist in currentSongs.Keys)

{

if (searchByArtist)

{

if (artist.Contains(searchBox.Text))

{

searchResult.Add(artist, currentSongs[artist]);

}

}

else

{

foreach (string song in currentSongs[artist])

{

if (song.Contains(searchBox.Text))

{

try

{

searchResult[artist].Add(song);

}

catch (KeyNotFoundException ex)

{

searchResult.Add(artist, new List<string> { song });

}

}

}

}

}

if (searchResult.Count == 0)

{

MessageBox.Show("Unfortunately, nothing was found. Try another criterias.", "Not found!");

searchBox.Focus();

}

else

{

Show(searchResult);

backButton.Enabled = true;

}

}

#endregion

#region backButton\_Click

/// <summary>

/// Restores current song list after search results generating

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

backButton.Enabled = false;

Show(currentSongs);

}

#endregion

#region saveToolStripMenuItem\_Click

/// <summary>

/// Saving file to the current path

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

File.Delete(currentPath);

using (StreamWriter document = new StreamWriter(currentPath))

{

foreach (string artist in currentSongs.Keys)

{

foreach (string song in currentSongs[artist])

{

document.WriteLine(artist + '↕' + song);

}

}

}

MessageBox.Show("Song list was successfully saved!");

}

#endregion

#region saveSongListAsToolStripMenuItem\_Click

/// <summary>

/// Saving file to the path chosen by user

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

SaveFileDialog dialog = new SaveFileDialog();

dialog.Filter = "Txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

dialog.FilterIndex = 2;

dialog.RestoreDirectory = true;

if (dialog.ShowDialog() == DialogResult.OK)

{

using (StreamWriter document = new StreamWriter(dialog.OpenFile()))

{

foreach (string artist in currentSongs.Keys)

{

foreach (string song in currentSongs[artist])

{

document.WriteLine(artist + '↕' + song);

}

}

}

}

MessageBox.Show("Song list was successfully saved!");

}

#endregion

#region searchBoxTextStyle

private void searchBox\_Enter(object sender, EventArgs e)

{

textBoxPlaceholderEnter(searchBox, "Type in an artist or a song name...");

}

private void searchBox\_Leave(object sender, EventArgs e)

{

textBoxPlaceholderLeave(searchBox, "Type in an artist or a song name...");

}

#endregion

#region placeholder

public void textBoxPlaceholderEnter(TextBox textBox, string defaultText)

{

if (textBox.Text == defaultText)

{

textBox.ForeColor = Color.Black;

textBox.Text = "";

}

}

public void textBoxPlaceholderLeave(TextBox textBox, string defaultText)

{

if (textBox.Text == "")

{

textBox.ForeColor = Color.Gray;

textBox.Text = defaultText;

}

}

#endregion

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

{

Form3 f = new Form3();

f.Show();

}

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

{

Form4 f4 = new Form4();

f4.Show();

}

}

Класс Form2

public partial class Form2 : Form

{

Form1 owner;

public Form2(Form1 f1)

{

this.owner = f1;

InitializeComponent();

}

#region okButton\_Click

/// <summary>

/// Confirms additing or changing

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

if (this.Text == "Editing")

{

owner.Delete();

}

try

{

owner.currentSongs[artistTextBox.Text].Add(songTextBox.Text);

}

catch (KeyNotFoundException ex)

{

owner.currentSongs.Add(artistTextBox.Text, new List<string> { songTextBox.Text });

}

owner.dataGridView1.Rows.Add(artistTextBox.Text, songTextBox.Text);

this.Close();

}

#endregion

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

{

this.Close();

}

#region artistTextBoxStyle

private void artistTextBox\_Enter(object sender, EventArgs e)

{

owner.textBoxPlaceholderEnter(artistTextBox, "Type in an artist...");

}

private void artistTextBox\_Leave(object sender, EventArgs e)

{

owner.textBoxPlaceholderLeave(artistTextBox, "Type in an artist...");

}

#endregion

#region songTextBoxStyle

private void songTextBox\_Enter(object sender, EventArgs e)

{

owner.textBoxPlaceholderEnter(songTextBox, "Type in a song name...");

}

private void songTextBox\_Leave(object sender, EventArgs e)

{

owner.textBoxPlaceholderLeave(songTextBox, "Type in a song name...");

}

#endregion

}