Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Курсова робота з дисципліни

**«**Об’єктно-орієнтоване програмування**»**

Тема: «Довідник туриста»

Виконавець:

ст. гр.

Керівник:

проф.

Комісія:

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Робота захищена з оцінкою

Харків 2016

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**

Кафедра: ***Програмної інженерії***

Дисципліна: ***Об’єктно-орієнтоване програмування***

Спеціальність: ***Програмна інженерія***

Курс 1 .Група *ПІ-15-1* . Семестр *2* .

***ЗАВДАННЯ***

***на курсовий проект студента***

*\_\_\_ Євченко Богдан Олегович \_\_\_\_\_\_\_*

(Прізвище, Ім'я, По батькові)

1. Тема проекту: *Справочник туриста.*
2. Термін здачі студентом закінченого проекту:
3. Вихідні дані до проекту: *Турагенства и предлагаемые услуги: страна, город (или маршрут круиза), условия проживания и проезда, экскурсионное обслуживание, сервис принимающей стороны, стоимость путевки.*

Зміст пояснювальної записки (перелік питань, що їх належить розробити) вступ, опис об’єкту розробки: *Специфікація програми, проектна специфікація, інструкція користувача, висновки.*

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми | 1-03-2016 р. |
| 2 | Формулювання вимог до програми | 2-03-2016 – 12-03-2016 р. |
| 3 | Розробка моделі програми. | 12-03-2016 – 20-03-2016 р. |
| 4 | Розробка підсистеми зберігання та пошуку. | 21-03-2016 – 29-03-2016 р. |
| 5 | Розробка інтерфейсу програми. | 30-03-2016 – 12-04-2016 р. |
| 6 | Розробка функції розширеного пошуку, редагування даних, вдосконалення інтерфейсу. | 13-04-2016 – 17-04-2016 р. |
| 7 | Тестування і доопрацювання розробленої програмної системи. | 18-04-2016 – 29-04-2016 р. |
| 8 | Оформлення пояснювальної записки, додатків, графічного матеріалу. | 25-04-2016 – 31-05-2016 р. |
| 9 | Захист |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Євченко Богдан Олегович

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бондарєв Володимир Михайлович

« 1 »\_березня\_ 2016 р.

# РЕФЕРАТ

Пояснювальна записка до курсової роботи: 28с., 22 рис., 1 прилож., 3 ист.

Мета роботи - створення програмної системи під назвою «Справочник туриста» для роботи з базою даних користувачів.

Метод розробки - Microsoft Visual Studio 2015 , WINDOWS FORMS, .NET Framework 4.5

В результаті розробки отримана програмна система під назвою «Справочник туриста» для роботи з базою даних турагентств. Система дає можливість зберігати, продивлятися, редагувати та шукати необхідну інформацію про турагентства та їх послуги: країна, місто (чи маршрут), умови проживання та проїзду, вартість путівки.

КУРСОВИЙ ПРОЕКТ, ООП, ПРОГРАМНА СИСТЕМА, VISUAL STUDIО 2015, ДОВІДНИК, ТУРИЗМ, ТУРАГЕНТСТВО, ПУТІВКА, КРУЇЗ.

Github: https://github.com/olderor/coursa4

СОДЕРЖАНИЕ

[РЕФЕРАТ 4](#_Toc452411739)

[ВВЕДЕНИЕ 6](#_Toc452411740)

[1 СПЕЦИФИКАЦИЯ ПРОГРАММЫ 8](#_Toc452411741)

[1.1 Основные требования и функции программы 8](#_Toc452411742)

[1.2 Функции программы 8](#_Toc452411743)

[1.3 Пользовательский интерфейс 9](#_Toc452411744)

[*2 ПРОЕКТНАЯ СПЕЦИФИКАЦИЯ* 13](#_Toc452411745)

[ВЫВОДЫ 24](#_Toc452411746)

[ПЕРЕЧЕНЬ ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 25](#_Toc452411747)

# ВВЕДЕНИЕ

В мире существует достаточно много различных турагентств, каждое из которых предоставляет огромное количество услуг. Среди такого огромного количества различных услуг довольно сложно найти интересующую пользователя поездку. Отталкиваясь от этой проблематики ставятся требования к рассматриваемой программе. Она позволяет сделать хранение данных компактным и удобным, а поиск интересующего тура или круиза простым и одновременно гибким.

В данной курсовой работе требуется разработать программу «Справочник туриста». Целью данной курсовой работы является создания удобного и многофункционального справочника на основе объектно-ориентированного подхода, а также получение навыков в реализации этого подхода, проектировании и реализации схемы данных.

Разработанная программа будет иметь огромное практическое применение. Она может быть полезна широкому ряду пользователей: например, заядлым путешественникам, которые ищут какой-то определенный тур, любителям и новичкам, чтобы найти экономный и не менее интересный вариант путешествия, а также работникам турагентств для составления базы собственных услуг и изучения возможностей конкурентов.

Задачи выполнения работы:

* Исследование предметной области с целью выявления особенностей тематики, основных сущностей и их атрибутов;
* Проектирование иерархии классов на основе обнаруженных особенностях выделенных сущностей;
* Применение принципов объектно-ориентированного программирования;
* Использования текстовых документов в качестве хранилища информации.

Средой разработки является Visual Studio 2015 Community. Языком разработки является объектно-ориентированный язык программирования C#. Программа будет разрабатываться с использованием интерфейса программирования приложений Windows Forms.

В настоящее время oбъектно-ориентированное программирование является абсолютным лидером в области прикладного программирования.

# СПЕЦИФИКАЦИЯ ПРОГРАММЫ

## 1.1 Основные требования и функции программы

Целью курсового проекта является написание программы «Справочник туриста», а также написание документации, сопровождающей его.

Программа должна иметь функции, предоставляющие пользователю возможности для просмотра, редактирования, сохранения и загрузки данных.

Устойчивость программы должна быть максимальной: программа не должна терять работоспособность ни при каких, даже некорректных действиях пользователя. Всякие действия, грозящие потерей информации, должны быть подтверждены пользователем.

Интерфейс пользователя должен быть понятным и приятным внешне, а также предоставлять удобные возможности для работы с данными.

На любом этапе нажатие любой клавиши должно игнорироваться или вызывать предусмотренные действия, описанные в средствах помощи.

## 1.2 Функции программы

А. Предоставление данных для просмотра, а именно: для турагентств – название и адрес турагентства, количество предоставляемых путешествий; для путешествий – название, тип, место или путь путешествия, его краткое описание, длительность и цена, а также его вид, услуги, которые в него включены и транспорт, который будет использоваться во время путешествия.

Б. Добавление в программу новых и изменение существующих турагентств и путешествий. Добавление или изменение происходит в случае, если все поля заполнены логически правильно, в противном случае пользователь получит сообщение с указанием на то, в чем заключается ошибка, но при этом данные занесенные в поля диалоговой формы сохранятся. При выборе функции сохранения или возврата пользователю предоставляется возможность подтвердить свое решение или отменить его соответственно.

В. Удаление существующих турагентств или путешествий. Следует отметить, что удаление, как и изменение выполняется относительно выделенного в вышеупомянутых списках элемента. При выборе функции удаления пользователю предоставляется возможность отменить свое решение или подтвердить его.

Г. Загрузка списка турагентств и путешествий с диска: подразумевает загрузку последнего сохраненного списка соответствующей туристической коллекции.

Д. Функция выхода: обрабатывается как через меню, так и через кнопку “крестик”.

Е. Функция поиска: позволяет фильтровать список турагентств или путешествий в зависимости от введенных пользователем данных.

Ж. Функция помощи – дает пользователю полноценную справку о текущем окне программы и его возможностях.

1.3 Пользовательский интерфейс

Данная программа имеет 4 окна, каждое из которых предоставляет пользователю возможность выполнения какой-либо из функций программы.

При запуске программы открывается основное окно, в котором пользователь имеет возможность просматривать список турагентств, добавление новых турагентств. В основном окне расположены элементы для просмотра данных о турагентстве, таких как название, адрес и количество предоставляемых путешествий.

Так же в основном окне реализована возможность поиска, что позволяет найти нужное турагентство среди этого списка, основываясь на имеющихся данных.

Рассмотрим меню в основном окне форме.

Подменю “File” (“Файл”) содержит следующие пункты:

Пункт “Add agency” (“Добавить агентство”) позволяет пользователю вносить в список новые турагентства. При нажатии на выше указанный пункт откроется окно в которое необходимо ввести название турагентства и его адрес. Необходимо помнить, что все поля обязательны для заполнения, но если Вы забудете это правило, то вместо ошибки получите сообщение с указанием на то, что следует исправить, при этом ранее введенные Вами данные останутся нетронутыми.

Пункт “Exit” (“Выход”) вызывает диалоговое окно, где пользователь может отменить свое решение. При нажатии на кнопку “Да” приложение будет закрыто.

Подменю “Help” (“Помощь”) содержит следующие пункты:

Пункт “View Help” (“Просмотреть Помощь”) позволяет пользователю просмотреть информацию о функциональных возможностях программы в данном окне и о том, как их правильно применять. При нажатии на выше указанный пункт откроется информационное окно с указанной выше информацией.

“About” (“Про программу”) позволяет пользователю просмотреть информацию о программе (название и краткое описание программы, копирайт и информация о создателе программы).

В основном окне пользователю предоставляется возможность выбора нужного ему турагентства, а также возможность просмотреть все имеющиеся путешествия.

В первом случае, будет открыто новое окно, в котором пользователь может просмотреть путешествия, предлагаемые этим турагентством, а также просмотреть, редактировать информацию об этом турагентстве с возможностью его удалить.

В окне турагентства расположены элементы для просмотра данных о предлагаемых путешествиях, таких как название, тип, место или путь путешествия, а также его краткое описание, длительность и цена. Пользователь может выбрать интересующее его путешествие и просмотреть или отредактировать информацию об этом путешествии.

Так же в этом окне реализована возможность поиска, что позволяет найти нужное путешествие среди этого списка, основываясь на имеющихся данных и данных, введенных пользователем.

Рассмотрим меню в окне форме.

Подменю “File” (“Файл”) содержит следующие пункты:

Пункт “Agency” (“Агентство”) содержит следующие пункты:

Пункт “Add travel” (“Добавить путешествие”) позволяет пользователю добавить новое путешествие в список предоставляемых этим турагентством. Для этого откроется новое окно, описание которого находится ниже.

Пункт “Edit” (“Редактировать”) вызывает окно, где пользователь может редактировать информацию о турагентстве (название и адрес) с возможностью сохранить изменения или их отменить.

Пункт “Delete” (“Удалить”) позволяет пользователю удалить всю информацию о текущем турагентстве. Для этого откроется диалоговое окно, где пользователь может отменить свое решение. При нажатии на кнопку “Да” данные будут удалены.

Пункт “Exit” (“Выход”), функционал которого идентичен и описан ранее.

Подменю “Help” (“Помощь”) функционал которого идентичен и описан ранее.

В случае выбора путешествия или добавления нового путешествия будет открыто новое окно. В окне турагентства расположены элементы для просмотра и редактирования данных о путешествии, такие как название, тип, вид, активность, место или путь путешествия, а также его краткое описание, длительность и цена. Пользователь может редактировать указанную информацию, сохранить или отменить изменения, а также удалить путешествие из базы (в случае, если было выбрано определенное путешествие).

# 2 ПРОЕКТНАЯ СПЕЦИФИКАЦИЯ

2.1 Объектная модель программы

В данном проекте был реализован удобный пользовательский интерфейс, который похож на большинство Windows – приложений.

В данном курсовом проекте было реализовано систему классов (Рис 2.1), которые тесно взаимодействую между собой.

Program

static Класс

Object

FileManager

Class

MyList

static Класс

TravelAgencyCollection

Singleton class

TravelAgency

Class

Travel

Class

Location

Class

Рисунок 2.1 – «Объектная спецификация программы»

Класс Location (Рис 2.2) содержит два открытых свойства – название места (Place) и страна, в которой находится (Country), а также поле «только для чтения» IsCorrect, показывающее корректные ли данные поля. Так же в класс входит метод преобразования объекта в строку (ToString).

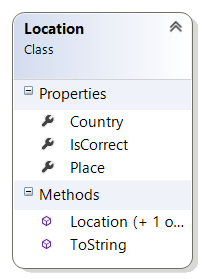


Рисунок 2.2 – «Класс Location»

Класс Travel (Рис. 2.3) – путешествие. Содержит ряд открытых полей: вид (Activeness), цена (Cost), длительность в днях (DayLength), описание (Description), комбинация (Inclusion) того, что включено в цену путешествия, список мест (Route), из которых состоит путешествие, название (Title), используемый транспорт (Transport), тип (Type), владелец (Owner) (какое туристическое агентство предоставляет). Также имеет свойства только для чтения: путь (Route) и свойство (IsCorrect), показывающее корректное ли путешествие. Имеет методы: добавить (AddLocation) и удалить (RemoveLocation) место, скопировать данные из другого объекта типа Travel (Copy).

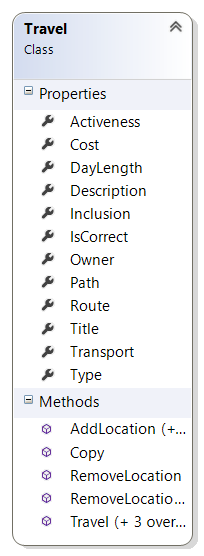


Рисунок 2.3 – «Класс Travel»

Класс TravelAgency (Рис 2.4) – туристическое агентство. Содержит ряд открытых полей: адрес (Address), название (Name), список путешествий (Travels). Также имеет свойства только для чтения: количество путешествий (TravelsCount) и свойство (IsCorrect), показывающее корректное ли туристическое агентство. Имеет методы: добавить (Add) и удалить (Remove) путешествие, скопировать данные из другого объекта типа TravelAgency (Copy).

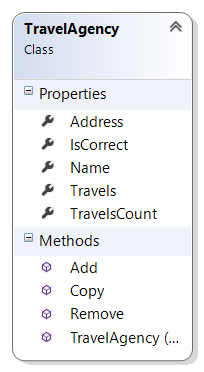


Рисунок 2.4 – «Класс TravelAgency»

Синглтон TravelAgencyCollection (Рис 2.5) – класс, для работы со списком туристических агентств. Содержит скрытое статическое поле – экземпляр класса (defaultInstance), статический метод получения этого экземпляра (GetDefaultInstance). Открытое поле – список туристических агентств (TravelAgencies). Имеет методы: добавить (Add) и удалить туристическое агентство (Remove), считывание (Read) и сохранение (Save) данных.

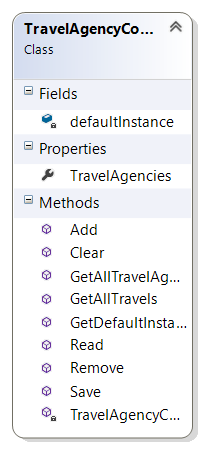


Рисунок 2.5 – «Класс TravelAgencyCollection»

В проекте необходим лишь один глобальный экземпляр TravelAgencyCollection, который будет содержать в себе базу данных и работать с ней. Суть заключается в существовании лишь одного экземпляра класса и получение его через метод GetDefaultInstance, который автоматически считает информацию из файла при первой инициализации.

Статический класс MyList (Рис 2.6) – класс, для работы со списком туристических агентств. Имеет статические методы: сравнения двух списков (Compare) и фильрации (Filter), упорядочивания элементов по полю (SortBy ).

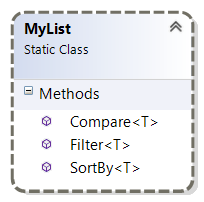


Рисунок 2.6 – «Класс MyList»

Метод Filter производит поиск и выбор данных из списка, в полях которого содержится каждый из заданных фильтров. Метод SortBy производит упорядочивание элементов из списка по заданному полю в заданном направлении (по умолчанию в порядке возрастания). В реализации методов используется запрос LINQ, а также механизм Reflection (Отражение) для более гибкой и универсальной работы функций.

Класс FileManager (Рис 2.7) – класс для работы с файлами. Содержит поле – путь к файлу, методы чтения (ReadCollection) и записи (SaveCollection) коллекции в файл, а также скрытые методы обобщенного чтения (Read) и записи (Save). Следует упомянуть, что класс использует методы сериализации\десериализации объектов.

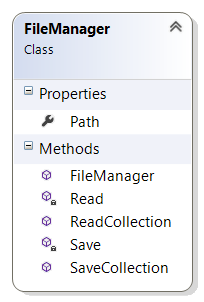


Рисунок 2.7 – «Содержимое класса FileManager»

3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для установки программы достаточно распаковать архив в желаемую папку. Далее следует запустить приложение из полученной папки.

1. Для работы с туристическими агентствами нужно использовать основное окно программы (Рис. 3.1). В окне находится меню. Первое подменю “File” (Рис. 3.2), позволяет управлять программой (добавить новое туристическое агентство “Add”, “Add agency”, выход из приложения “Exit”). Второе подменю “Help” даст помощь по использованию окна “View help” и информацию о программе и ее создателе “About”. Для получения помощи в использовании окна, выберите пункт “Help”. После чего будет открыто стандартное диалоговое окно с инструкциями.

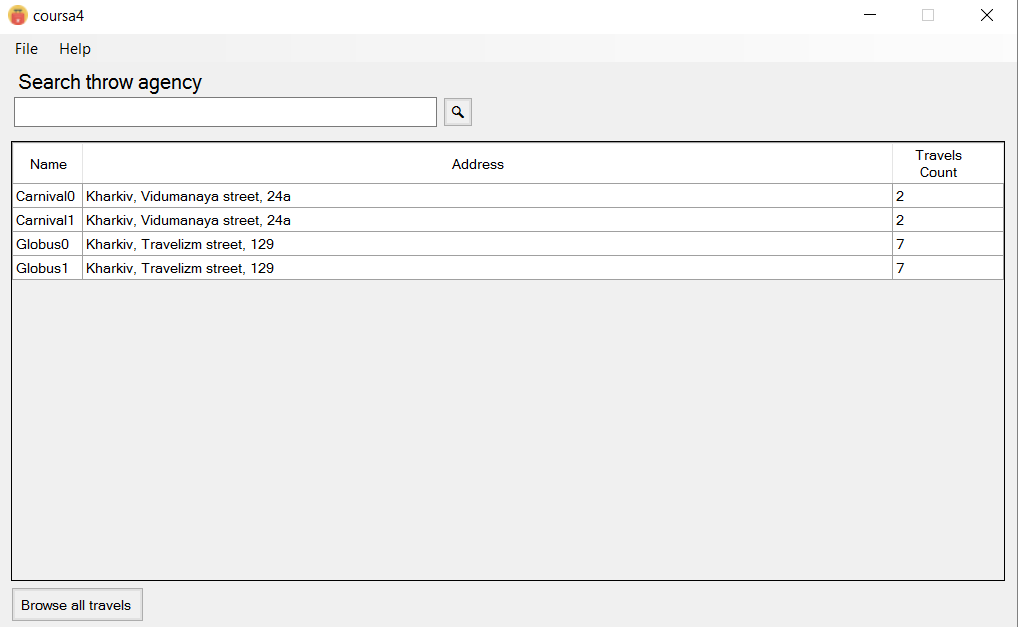


Рисунок 3.1 – «Основное окно»

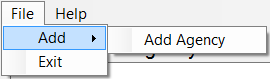


Рисунок 3.2 – «Подменю «Файл»»

* 1. Для добавления нового туристического агентства необходимо выбрать пункт “Add agency”, ввести данные и подтвердить свое решение. (Рис. 3.3)

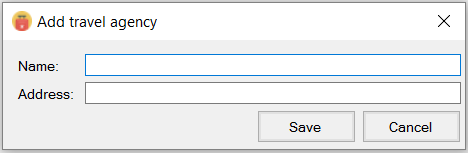


Рисунок 3.3 – «Окно добавления нового туристического агентства»

В случае ввода некорректных данных, программа предупредит об этом соответствующим сообщением (Рис. 3.4)

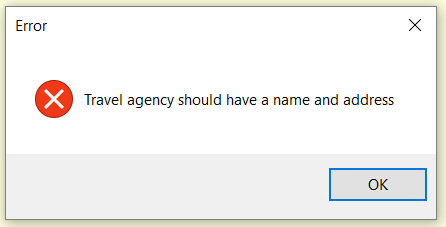


Рисунок 3.4 – «Диалоговое окно об ошибке»

Если решение будет отменено, программа спросит, необходимо ли сохранить введенные данные. (Рис. 3.5)

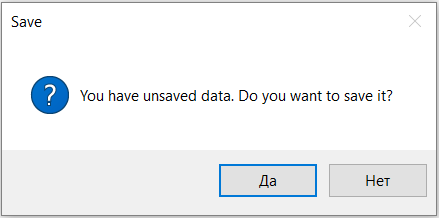


Рисунок 3.5 – «Диалоговое окно о сохранении»

1.2 Для поиска необходимых данных можно воспользоваться панелью поиска. (Рис. 3.6). Введите ключевые слова через пробел в поле поиска, после чего нажмите соответствующую кнопку рядом. Для сброса поиска очистите поле поиска и нажмите на ту же кнопку.



Рисунок 3.6 – «Панель поиска»

1.3 Для упорядочивания по столбцам, нажмите левой кнопкой мыши на заголовок нужного Вам столбца. (Рис. 3.7). Нажмите еще раз для упорядочивания в обратном порядке.

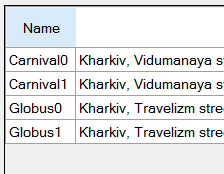


Рисунок 3.7 – «Упорядочивание по столбцам»

1.4 Для просмотра всех имеющихся путешествий, нажмите на кнопку “Browse all travels”.

1.5 Для выбора конкретного агентства два раза нажмите левой кнопкой мыши на интересующее агентство.

2. Для работы с туристическими агентствами нужно использовать необходимо выбрать туристическое агентство или выбрать режим просмотра всех путешествий (см. п. 1) (Рис. 3.8). В окне вы обнаружите меню. Первое подменю “File”, позволяет управлять программой (выход из приложения “Exit”). Второе подменю “Help” даст помощь по использованию окна “View help” и информацию о программе и ее создателе “About”. Для получения помощи в использовании окна, выберите пункт “Help”. После чего будет открыто стандартное диалоговое окно с инструкциями.

Воспользоваться поиском и упорядочиванием путешествий можно аналогично с основным окном (см. п. 1).

Для выбора конкретного путешествия два раза нажмите левой кнопкой мыши на интересующее Вас путешествие.

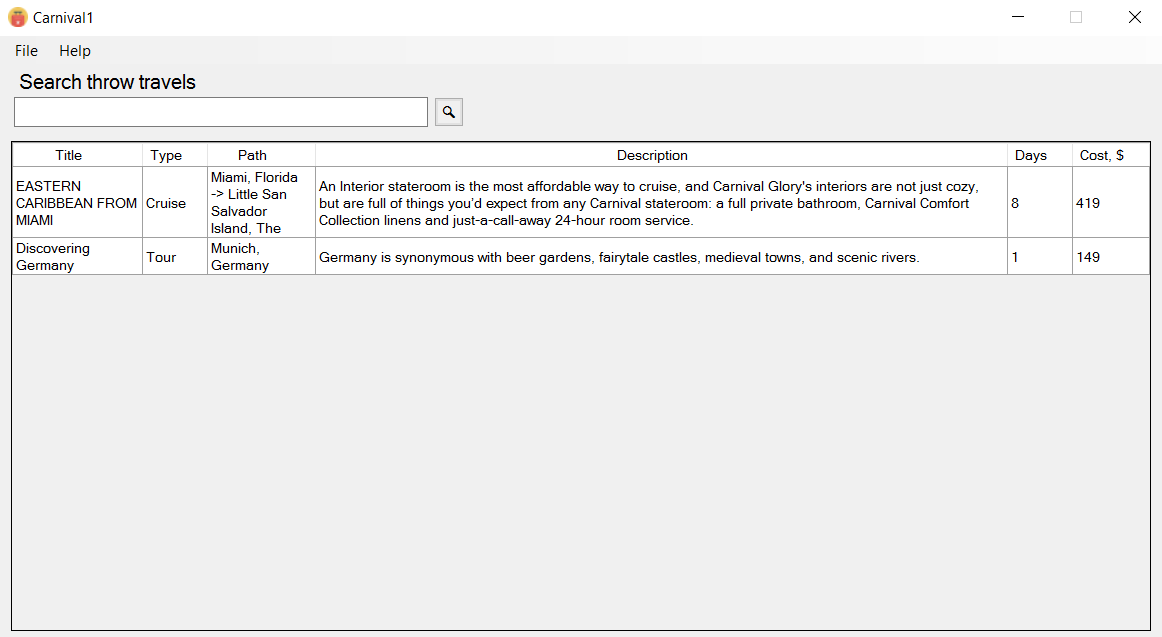


Рисунок 3.8 – «Окно всех путешествий»

2.1 Управление агентством осуществляется с помощью подменю “Agency”, которое находится в меню “File” (Рис. 3.9). Тут можно добавить новое путешествие в список предлагаемых, редактировать данные об агентстве и удалить информацию о нем.

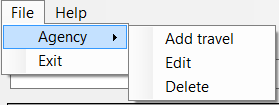


Рисунок 3.9 – «Подменю «Agency»»

2.2 Для добавления нового путешествия, просмотра информации и редактировании путешествия используется окно путешествия (Рис 3.10)

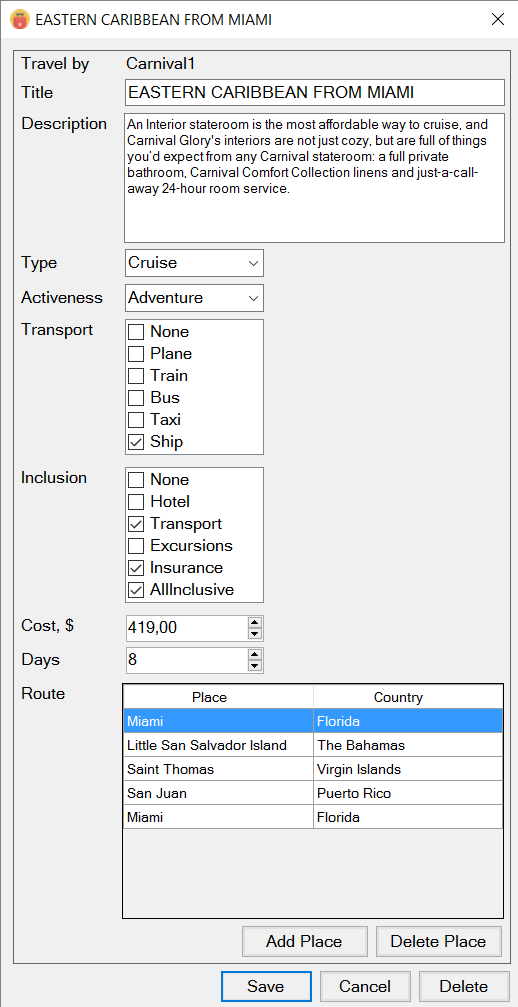


Рисунок 3.10 – «Окно путешествия»

Информация о названии путешествия и места, по которым осуществляется путешествие, необходимы к заполнению. Остальная информация указывается по желанию. Чтобы удалить путешествие, нажмите кнопку “Delete”.

# ВЫВОДЫ

В результате выполнения поставленной задачи был разработан программный продукт, отвечающий всем требованиям и целям, поставленным при его планировании. Был разработанный удобный пользовательский интерфейс. Реализована функция поиска и упорядочивания туристических агентств и путешествий. Были перехвачены и обработаны все исключительные ситуации, что позволяет программе работать без перебоя. Реализована функция сохранения файла и записи в файл. Были проработаны все возможные способы потери информации, например, если пользователь случайно закроет окно - то появится предупреждение, о том, что при закрытии приложения, несохраненные данные будут утеряны.

Во время разработки программы большое внимание уделялось следованию парадигме объектно-ориентированного программирования. Значительное внимание было уделено оптимизации и тестированию программы, а также разработке интерфейса.

Объектно-ориентированный подход дает возможность на ранних этапах разработки учесть все особенности создающихся программ, необходимый набор функций, структуру баз данных, что в дальнейшем максимально исключает необходимость переработки уже написанных компонентов программы.

Программа реализована полноценно с удобным и информативным интерфейсом. Благодаря понятной архитектуре и несложному управлению работать с базой данных может и неподготовленный пользователь.

Данный программный продукт существенно облегчает жизнь, как и профессионалам в сфере туризма, так и простым любителям путешествий, а также работникам туристических агентств. Программа будет иметь огромное практическое применение. Справочник позволит с легкостью вести собственную базу агентств и путешествий, а встроенные функции, предотвращающие потерю данных, позволит пользователю не беспокоится о том, что его труд может бесследно исчезнуть.

# ПЕРЕЧЕНЬ ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бондарев В.М., Объектно-ориентированное программирование на С#. Учебное пособие. – Харьков: СМИТ, 2009г. – 224 с.
2. Эндрю Троельсен. C# и платформа .NET. Библиотека программиста. – СПб.: Питер, 2005. – 796 с.
3. Microsoft Developer Network. Библиотека MSDN. Разработка на .NET [Электронный ресурc] / MSDN – сеть разработчиков Microsoft. – Режим доступа : www/URL:https://msdn.microsoft.com/ru-ru/ – 25.05.2016 г. – Загл. с экрана.
4. Елементи інформатики [Текст]: довідник / В. С. Височанський, А. І. Кардаш, В. С. Костєв, В. В. Черняхівський.- К. :Наук. думка, 2003.- 192с.
5. Герберт Шилдт - C# 4.0. Полное руководство. – Харьков: СМИТ, 2011. – 1023 с.

Додаток А

/// <summary>

/// Класс, отвечающий за чтение и сохранение данных.

/// </summary>

class FileManager

{

/// <summary>

/// </summary>

/// <param name="path"> Путь к файлу. </param>

public FileManager(string path)

{

Path = path;

}

public string Path { get; set; }

/// <summary>

/// Чтение данных из файла.

/// </summary>

/// <returns> Возвращает коллекцию, которая была сохранена по заданному пути.

/// В случае, если прочесть невозможно, возвращает null. </returns>

public TravelAgencyCollection ReadCollection()

{

return Read() as TravelAgencyCollection;

}

/// <summary>

/// Сохранение коллекции.

/// </summary>

/// <param name="collection"> коллекция, которую необходимо сохранить. </param>

public void SaveCollection(TravelAgencyCollection collection)

{

Save(collection);

}

/// <summary>

/// Чтение данных из файла.

/// </summary>

/// <returns> Возвращает объект, который был сохранен по заданному пути.

/// В случае, если прочесть невозможно, возвращает null. </returns>

private object Read()

{

// Проверка на существование файла

if (!File.Exists(Path))

{

return null;

}

Stream stream = new FileStream(Path,

FileMode.Open,

FileAccess.Read,

FileShare.Read);

try

{

IFormatter formatter = new BinaryFormatter();

object obj = formatter.Deserialize(stream);

stream.Close();

return obj;

}

catch { }

stream.Close();

return null;

}

/// <summary>

/// Сохранение объекта.

/// </summary>

/// <param name="obj"> Объект, который необходимо сохранить. </param>

private void Save(object obj)

{

IFormatter formatter = new BinaryFormatter();

Stream stream = new FileStream(Path,

FileMode.Create,

FileAccess.Write, FileShare.None);

formatter.Serialize(stream, obj);

stream.Close();

}

}

[Serializable]

public class Location

{

/// <summary>

/// </summary>

/// <param name="place"> Название места. </param>

/// <param name="country"> Страна, в которой находится. </param>

public Location(string place, string country)

{

Country = country;

Place = place;

}

public Location()

{

Country = "";

Place = "";

}

public string Country { get; set; }

public string Place { get; set; }

public bool IsCorrect

{

get

{

if (Country == "")

return false;

if (Place == "")

return false;

return true;

}

}

public override string ToString()

{

return Place + ", " + Country;

}

}

static class MyList

{

/// <summary>

/// Фильтрация коллекции.

/// </summary>

/// <typeparam name="T"> Тип объекта, список которого нужно отфильтровать. </typeparam>

/// <param name="list"> Список объектов, который нужно отфильтровать. </param>

/// <param name="filters"> Массив фильтров. </param>

/// <returns> Возвращает отфильтрованный список. </returns>

public static List<T> Filter<T>(List<T> list, string[] filters)

{

// Запрос выбора объектов, у которых для каждого из фильтров есть хотя бы одно поле, в котором содержится фильтр.

List<T> result = list.Where(ta => filters.All(f => ta.GetType().GetProperties().Any(

p => !p.PropertyType.IsGenericType && p.GetValue(ta).ToString().ToLower().Contains(f.ToLower())

))).ToList();

return result;

}

/// <summary>

/// Сортировка коллекции по параметру.

/// </summary>

/// <typeparam name="T"> Тип объекта, список которого нужно упорядочить. </typeparam>

/// <param name="list"> Список объектов, который нужно упорядочить. </param>

/// <param name="property"> Поле, по которому нужно упорядочить. </param>

/// <param name="direction"> Направление упорядочивания. </param>

/// <returns> Возвращает упорядоченный список. </returns>

public static List<T> SortBy<T>(List<T> list, string property, ListSortDirection direction = ListSortDirection.Ascending)

{

List<T> result;

// Сортировка списка по значению из заданного поля.

if (direction == ListSortDirection.Ascending)

result = list.OrderBy(el =>

el.GetType().GetProperty(property).GetValue(el, null)).ToList();

else

result = list.OrderByDescending(el => el.GetType().GetProperty(property).GetValue(el, null)).ToList();

return result;

}

/// <summary>

/// Сравнение двух списков.

/// </summary>

/// <typeparam name="T"> Тип объектов, которые находятся в списке. </typeparam>

/// <param name="list1"> Первый список. </param>

/// <param name="list2"> Второй список. </param>

/// <returns> Возвращает true, если соответсвующие элементы списков равны, иначе false. </returns>

public static bool Compare<T>(List<T> list1, List<T> list2)

{

if (list1.Count != list2.Count)

return false;

for (int i = 0; i < list1.Count; i++)

if (!list1[i].Equals(list2[i]))

return false;

return true;

}

}

public enum TravelType

{

Tour,

Cruise

}

public enum Activeness

{

Calm,

Pleasure,

Active,

Sport,

Educational,

Adventure

}

[Flags]

public enum Transport

{

None = 0x000,

Plane = 0x001,

Train = 0x002,

Bus = 0x004,

Taxi = 0x008,

Ship = 0x010

}

[Flags]

public enum Inclusion

{

None = 0x000,

Hotel = 0x001,

Transport = 0x002,

Excursions = 0x004,

Insurance = 0x008,

AllInclusive = 0x010 // Food and Drinks

}

[Serializable]

public class Travel

{

public Travel()

{

Route = new BindingList<Location>();

}

public Travel(Travel travel)

{

Copy(travel);

}

public Travel(

TravelType type,

string title,

Activeness activeness,

Transport transport,

Inclusion inclusion,

double cost,

int dayLength,

string description,

params Location[] locations) :

this(type, title, activeness, transport, inclusion, cost, dayLength, description, locations.ToList())

{ }

public Travel(TravelType type,

string title,

Activeness activeness,

Transport transport,

Inclusion inclusion,

double cost,

int dayLength,

string description,

List<Location> locations)

{

Type = type;

Title = title;

Activeness = activeness;

Transport = transport;

Inclusion = inclusion;

DayLength = dayLength;

Cost = cost;

Description = description;

Route = new BindingList<Location>(locations);

}

public TravelAgency Owner { get; set; }

public TravelType Type { get; set; }

public string Title { get; set; }

public Activeness Activeness { get; set; }

public Transport Transport { get; set; }

public Inclusion Inclusion { get; set; }

public double Cost { get; set; }

public int DayLength { get; set; }

public string Description { get; set; }

public BindingList<Location> Route { get; set; }

public string Path

{

get

{

if (Route.Count == 0)

return "";

string result = Route[0].ToString();

for (int i = 1; i < Route.Count; i++)

result += " -> " + Route[i].ToString();

return result;

}

}

/// <summary>

/// Проверка на корректность данных.

/// </summary>

public bool IsCorrect

{

get

{

if (Route.Count == 0)

return false;

if (Title == "")

return false;

foreach (Location l in Route)

if (!l.IsCorrect)

return false;

return true;

}

}

public void Copy(Travel travel)

{

Owner = travel.Owner;

Type = travel.Type;

Title = travel.Title;

Activeness = travel.Activeness;

Transport = travel.Transport;

Inclusion = travel.Inclusion;

DayLength = travel.DayLength;

Cost = travel.Cost;

Description = travel.Description;

Route = new BindingList<Location>(travel.Route.ToList());

}

public void RemoveLocation(Location location)

{

Route.Remove(location);

}

public void RemoveLocationAt(int index)

{

Route.RemoveAt(index);

}

public void AddLocation(string place, string country)

{

Route.Add(new Location(place, country));

}

public void AddLocation()

{

Route.Add(new Location());

}

}

public class TravelAgency

{

public TravelAgency()

{

Name = "";

Address = "";

Travels = new List<Travel>();

}

public TravelAgency(TravelAgency travelAgency)

{

Copy(travelAgency);

}

public TravelAgency(string name, string address, params Travel[] travels)

{

Name = name;

Address = address;

Travels = new List<Travel>(travels);

foreach (Travel t in travels)

t.Owner = this;

}

public TravelAgency(string name, string address, IEnumerable<Travel> travels)

{

Name = name;

Address = address;

Travels = new List<Travel>(travels.ToList());

foreach (Travel t in travels)

t.Owner = this;

}

public List<Travel> Travels { get; set; }

public string Name { get; set; }

public string Address { get; set; }

public int TravelsCount

{

get

{

return Travels.Count;

}

}

/// <summary>

/// Копирует данные из туристического агентства в текущий экземпляр.

/// </summary>

/// <param name="travelAgency"> Другое туристическое агентстсво. </param>

public void Copy(TravelAgency travelAgency)

{

Name = travelAgency.Name;

Address = travelAgency.Address;

Travels = new List<Travel>(travelAgency.Travels);

}

/// <summary>

/// Удаляет путешествие из списка путешествий.

/// </summary>

/// <param name="travel"> Путешествие, которое нужно удалить. </param>

public void Remove(Travel travel)

{

Travels.Remove(travel);

}

/// <summary>

/// Добавляет путешествие в список путешествий.

/// </summary>

/// <param name="travel"> Путешествие, которое нужно добавить. </param>

public void Add(Travel travel)

{

travel.Owner = this;

Travels.Add(travel);

}

/// <summary>

/// Проверка на корректность данных.

/// </summary>

public bool IsCorrect

{

get

{

if (Name == "")

return false;

if (Address == "")

return false;

return true;

}

}

}

/// <summary>

/// Синглтон с коллекцией туристических агентств.

/// </summary>

[Serializable]

class TravelAgencyCollection

{

private TravelAgencyCollection()

{

}

public BindingList<TravelAgency> TravelAgencies { get; private set; }

private static TravelAgencyCollection defaultInstance;

public static TravelAgencyCollection GetDefaultInstance()

{

if (defaultInstance == null)

{

defaultInstance = new TravelAgencyCollection();

defaultInstance.Read();

}

return defaultInstance;

}

/// <summary>

/// </summary>

/// <returns> Возвращает список всех туристических агентств. </returns>

public List<TravelAgency> GetAllTravelAgencies()

{

return TravelAgencies.ToList();

}

/// <summary>

/// </summary>

/// <returns> Возвращает список всех путешествий. </returns>

public List<Travel> GetAllTravels()

{

List<Travel> travels = new List<Travel>();

foreach (TravelAgency ta in TravelAgencies)

travels.AddRange(ta.Travels);

return travels;

}

/// <summary>

/// Сохраняет текущее состояние коллекции.

/// </summary>

public void Save()

{

FileManager fm = new FileManager("data.txt");

fm.SaveCollection(this);

}

/// <summary>

/// Загрузка данных из файла в коллекцию.

/// </summary>

public void Read()

{

FileManager fm = new FileManager("data.txt");

TravelAgencyCollection tac = fm.ReadCollection();

if (tac != null && tac.TravelAgencies != null)

{

TravelAgencies = tac.TravelAgencies;

}

else

{

TravelAgencies = new BindingList<TravelAgency>();

}

}

/// <summary>

/// Удаляет все туристические агентства из коллекции.

/// </summary>

public void Clear()

{

TravelAgencies.Clear();

}

/// <summary>

/// Добавляет туристическое агентство в коллекцию.

/// </summary>

/// <param name="travelAgency"> Туристическое агентство, которое необходимо добавить. </param>

public void Add(TravelAgency travelAgency)

{

TravelAgencies.Add(travelAgency);

}

/// <summary>

/// Удаляет туристическое агентство из коллекции.

/// </summary>

/// <param name="travelAgency"> Туристическое агентство, которое необходимо удалить. </param>

public void Remove(TravelAgency travelAgency)

{

TravelAgencies.Remove(travelAgency);

}

}

public partial class AboutTravelAgencyForm : Form

{

public AboutTravelAgencyForm() : this(new TravelAgency())

{

isNew = true;

Text = "Add travel agency";

}

public AboutTravelAgencyForm(TravelAgency travelAgency)

{

InitializeComponent();

Text = "Edit";

originalTravelAgency = travelAgency;

this.travelAgency = new TravelAgency(originalTravelAgency);

nameTextBox.DataBindings.Add(new Binding("Text", this.travelAgency, "Name", true, DataSourceUpdateMode.OnPropertyChanged));

addressTextBox.DataBindings.Add(new Binding("Text", this.travelAgency, "Address", true, DataSourceUpdateMode.OnPropertyChanged));

}

private TravelAgency originalTravelAgency;

private TravelAgency travelAgency;

private bool saved = false;

private bool isNew = false;

private void saveData()

{

// Проверка на пустые поля.

if (!travelAgency.IsCorrect)

{

MessageBox.Show("Travel agency should have a name and address", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

TravelAgencyCollection tac = TravelAgencyCollection.GetDefaultInstance();

if (isNew)

{

tac.Add(travelAgency);

}

else

{

originalTravelAgency.Copy(travelAgency);

}

tac.Save();

saved = true;

}

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

{

saveData();

if (saved)

this.Close();

}

private void AddTravelAgencyForm\_FormClosing(object sender, FormClosingEventArgs e)

{

if (!saved)

{

// Запрос подтверждения выхода из формы.

DialogResult dr = MessageBox.Show("You have unsaved data. Do you want to save it?", "Save", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (dr == DialogResult.Yes)

{

saveData();

if (!saved)

e.Cancel = true;

}

}

}

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

{

this.Close();

}

}

namespace coursa4

{

public partial class AboutTravelForm : Form

{

public AboutTravelForm(TravelAgency travelAgency) : this(travelAgency, new Travel())

{

deleteButton.Visible = false;

saveButton.Location = cancelButton.Location;

cancelButton.Location = deleteButton.Location;

isNew = true;

}

public AboutTravelForm(TravelAgency travelAgency, Travel travel)

{

InitializeComponent();

originalTravel = travel;

this.travelAgency = travelAgency;

this.travel = new Travel(travel);

if (travelAgency != null)

infoLabel.Text = travelAgency.Name;

else

infoLabel.Text = travel.Owner.Name;

// Установка связей и источников данных для перечислений.

travelTypeComboBox.DataSource = Enum.GetValues(typeof(TravelType));

travelTypeComboBox.SelectedItem = this.travel.Type;

activenessComboBox.DataSource = Enum.GetValues(typeof(Activeness));

activenessComboBox.SelectedItem = this.travel.Activeness;

((ListBox)transportCheckedListBox).DataSource = Enum.GetValues(typeof(Transport));

((ListBox)transportCheckedListBox).DisplayMember = "Name";

setEnumToCheckedListBox(transportCheckedListBox, Enum.GetValues(typeof(Transport)), (int)travel.Transport);

((ListBox)inclusionCheckedListBox).DataSource = Enum.GetValues(typeof(Inclusion));

((ListBox)inclusionCheckedListBox).DisplayMember = "Name";

setEnumToCheckedListBox(inclusionCheckedListBox, Enum.GetValues(typeof(Inclusion)), (int)travel.Inclusion);

titleTextBox.DataBindings.Add(new Binding("Text", this.travel, "Title", true, DataSourceUpdateMode.OnPropertyChanged));

costNumericUpDown.DataBindings.Add(new Binding("Value", this.travel, "Cost", true, DataSourceUpdateMode.OnPropertyChanged));

daysNumericUpDown.DataBindings.Add(new Binding("Value", this.travel, "DayLength", true, DataSourceUpdateMode.OnPropertyChanged));

descriptionTextBox.DataBindings.Add(new Binding("Text", this.travel, "Description", true, DataSourceUpdateMode.OnPropertyChanged));

routeGridView.DataBindings.Add(new Binding("DataSource", this.travel, "Route", true, DataSourceUpdateMode.OnPropertyChanged));

DataBindings.Add(new Binding("Text", titleTextBox, "Text"));

if (this.travel.Route.Count != 0)

deletePlaceButton.Enabled = true;

}

private Travel travel;

private Travel originalTravel;

private TravelAgency travelAgency;

private bool saved = false;

private bool isNew = false;

private void saveData()

{

// Проверка на корректность входных данных.

if (!travel.IsCorrect)

{

MessageBox.Show("Travel should have a title and contain at least 1 place, all of them should have place name and country.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

TravelAgencyCollection tac = TravelAgencyCollection.GetDefaultInstance();

if (isNew)

{

travelAgency.Add(travel);

}

else

{

originalTravel.Copy(travel);

}

tac.Save();

saved = true;

}

/// <summary>

/// Устанавливает значения, которые указаны в объекте, в чекбокс.

/// </summary>

/// <param name="listBox"> Чекбокс, в который нужно установить значения. </param>

/// <param name="enumValues"> Все возможные значения перечисления. </param>

/// <param name="value"> Значение объекта. </param>

private void setEnumToCheckedListBox(CheckedListBox listBox,

Array enumValues, int value)

{

for (int i = 0; i < enumValues.Length; i++)

{

int enumValue = (int)enumValues.GetValue(i);

if ((value & enumValue) == enumValue)

{

listBox.SetItemCheckState(i, CheckState.Checked);

}

}

}

private void transportCheckedListBox\_ItemCheck(object sender, ItemCheckEventArgs e)

{

transportCheckedListBox.ClearSelected();

// Если был выбран None, нужно снять со всех остальных значений отметки.

// С None нельзя снять отметку, разве что только если было выбрано другое значение.

if (e.Index == 0)

{

if (travel.Transport == Transport.None)

{

e.NewValue = CheckState.Checked;

return;

}

if (e.NewValue == CheckState.Checked)

{

foreach (int i in transportCheckedListBox.CheckedIndices)

transportCheckedListBox.SetItemCheckState(i, CheckState.Unchecked);

}

return;

}

if (e.NewValue == CheckState.Checked)

{

travel.Transport |= (Transport)Enum.GetValues(typeof(Transport)).GetValue(e.Index);

transportCheckedListBox.SetItemCheckState(0, CheckState.Unchecked);

}

else

{

travel.Transport &= ~(Transport)Enum.GetValues(typeof(Transport)).GetValue(e.Index);

if (travel.Transport == Transport.None)

{

transportCheckedListBox.SetItemCheckState(0, CheckState.Checked);

}

}

}

private void inclusionCheckedListBox\_ItemCheck(object sender, ItemCheckEventArgs e)

{

inclusionCheckedListBox.ClearSelected();

// Если был выбран None, нужно снять со всех остальных значений отметки.

// С None нельзя снять отметку, разве что только если было выбрано другое значение.

if (e.Index == 0)

{

if (travel.Inclusion == Inclusion.None)

{

e.NewValue = CheckState.Checked;

return;

}

if (e.NewValue == CheckState.Checked)

{

foreach (int i in inclusionCheckedListBox.CheckedIndices)

inclusionCheckedListBox.SetItemCheckState(i, CheckState.Unchecked);

}

return;

}

if (e.NewValue == CheckState.Checked)

{

travel.Inclusion |= (Inclusion)Enum.GetValues(typeof(Inclusion)).GetValue(e.Index);

inclusionCheckedListBox.SetItemCheckState(0, CheckState.Unchecked);

}

else

{

travel.Inclusion &= ~(Inclusion)Enum.GetValues(typeof(Inclusion)).GetValue(e.Index);

if (travel.Inclusion == Inclusion.None)

{

inclusionCheckedListBox.SetItemCheckState(0, CheckState.Checked);

}

}

}

private void travelTypeComboBox\_SelectionChangeCommitted(object sender, EventArgs e)

{

travel.Type = (TravelType)travelTypeComboBox.SelectedValue;

}

private void activenessComboBox\_SelectionChangeCommitted(object sender, EventArgs e)

{

travel.Activeness = (Activeness)activenessComboBox.SelectedValue;

}

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

{

saveData();

if (saved)

this.Close();

}

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

{

this.Close();

}

private void AboutTravelForm\_FormClosing(object sender, FormClosingEventArgs e)

{

if (!saved)

{

// Запрос подтверждения выхода из формы.

DialogResult dr = MessageBox.Show("You have unsaved data. Do you want to save it?", "Save", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (dr == DialogResult.Yes)

{

saveData();

if (!saved)

e.Cancel = true;

}

}

}

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

{

DialogResult dr = MessageBox.Show("Are you sure you want to delete data?", "Warning", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (dr == DialogResult.Yes)

{

travelAgency.Remove(originalTravel);

TravelAgencyCollection.GetDefaultInstance().Save();

saved = true;

this.Close();

}

}

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

{

// Выбираем все отмеченные ячейки и удаляем их. После устанавливаем выделение на следующий элемент.

int index = 0;

DataGridViewSelectedRowCollection rows = routeGridView.SelectedRows;

for (int i = 0; i < rows.Count; i++)

{

index = rows[i].Index;

travel.RemoveLocationAt(index);

if (travel.Route.Count == 0)

deletePlaceButton.Enabled = false;

}

routeGridView.ClearSelection();

int count = routeGridView.Rows.Count;

if (count != 0)

{

if (count <= index && count != 0)

routeGridView.Rows[count - 1].Selected = true;

else

routeGridView.Rows[index].Selected = true;

}

}

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

{

travel.AddLocation();

deletePlaceButton.Enabled = true;

}

}

/// <summary>

/// Форма, отображающая список туристических агентств.

/// </summary>

public partial class TravelAgenciesForm : Form

{

/// <summary>

/// Инициализация формы с глобальным списком агентств.

/// </summary>

public TravelAgenciesForm()

{

InitializeComponent();

TravelAgencyCollection tac = TravelAgencyCollection.GetDefaultInstance();

source = new BindingSource();

source.DataSource = tac.TravelAgencies;

travelAgenciesGridView.DataSource = source;

}

private BindingSource source;

/// <summary>

/// Обновление данных.

/// </summary>

private void updateGridView(object sender, FormClosedEventArgs e)

{

filter();

}

/// <summary>

/// Фильтрация данных.

/// </summary>

private void filter()

{

TravelAgencyCollection tac = TravelAgencyCollection.GetDefaultInstance();

string[] filters = filterTextBox.Text.Split(' ');

source.DataSource = new BindingList<TravelAgency>(MyList.Filter<TravelAgency>(tac.GetAllTravelAgencies(), filters));

}

/// <summary>

/// Обработка выбора туристического агентсва.

/// </summary>

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

{

// Проверка на выбор заголовка столбца.

if (e.RowIndex == -1)

return;

int index = travelAgenciesGridView.CurrentRow.Index;

TravelAgency ta = (source.DataSource as BindingList<TravelAgency>)[index];

TravelAgencyForm taf = new TravelAgencyForm(ta);

taf.Show();

taf.FormClosed += updateGridView;

}

/// <summary>

/// Запрет выделения фона ячейки.

/// </summary>

private void travelAgenciesGridView\_SelectionChanged(object sender, EventArgs e)

{

travelAgenciesGridView.ClearSelection();

}

/// <summary>

/// Открытие формы с просмотром всех путешествий.

/// </summary>

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

{

TravelAgencyForm form = new TravelAgencyForm(TravelAgencyCollection.GetDefaultInstance().GetAllTravels());

form.Show();

form.FormClosed += updateGridView;

}

/// <summary>

/// Фильтрация туристических агентств по нажатию кнопки.

/// </summary>

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

{

filter();

}

/// <summary>

/// Фильтрация туристических агентств по нажатию Enter.

/// </summary>

private void filterTextBox\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

filter();

}

}

/// <summary>

/// Выход из программы.

/// </summary>

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

{

Application.Exit();

}

/// <summary>

/// Добавление агентства.

/// </summary>

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

{

AboutTravelAgencyForm form = new AboutTravelAgencyForm();

form.Show();

form.FormClosed += updateGridView;

}

/// <summary>

/// Выбор агентства.

/// </summary>

private void travelAgenciesGridView\_ColumnHeaderMouseClick(object sender, DataGridViewCellMouseEventArgs e)

{

List<TravelAgency> data = (source.DataSource as BindingList<TravelAgency>).ToList();

string property = travelAgenciesGridView.Columns[e.ColumnIndex].DataPropertyName;

List<TravelAgency> sortedData = MyList.SortBy<TravelAgency>(data, property);

// Если списки одинаковы, меняем порядок

if (MyList.Compare<TravelAgency>(sortedData, data))

sortedData.Reverse();

source.DataSource = new BindingList<TravelAgency>(sortedData);

}

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

{

MessageBox.Show("coursa4 - the best travel guide!\nCreated by Bohdan Yevchenko.\nCopyright © 2016, All Rights Reserved.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

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

{

MessageBox.Show("Double click on travel agency to browse and edit travel agency.\nClick on column header to order column.\nEnter search filters separated by space and click search button to filter agencies.\nClick on 'browse all' button to see all travels we have.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Question);

}

}

/// <summary>

/// Форма, отображающая список путешествий.

/// </summary>

public partial class TravelAgencyForm : Form

{

/// <summary>

/// </summary>

/// <param name="travelAgency"> Туристическое агентство, путешествия которого необходимо показать. </param>

public TravelAgencyForm(TravelAgency travelAgency) : this(travelAgency.Travels)

{

this.travelAgency = travelAgency;

DataBindings.Add(new Binding("Text", travelAgency, "Name"));

ToolStripItemCollection fileMenu = (menuStrip1.Items[0] as ToolStripMenuItem).DropDownItems;

fileMenu[0].Visible = true;

}

/// <summary>

///

/// </summary>

/// <param name="travels"> Список путешествий. </param>

public TravelAgencyForm(BindingList<Travel> travels) : this(travels.ToList()) { }

/// <summary>

///

/// </summary>

/// <param name="travels"> Список путешествий. </param>

public TravelAgencyForm(List<Travel> travels)

{

InitializeComponent();

source = new BindingSource();

ToolStripItemCollection fileMenu = (menuStrip1.Items[0] as ToolStripMenuItem).DropDownItems;

fileMenu[0].Visible = false;

Text = "List of travels";

this.travels = new BindingList<Travel>(travels);

source.DataSource = this.travels;

travelAgencyGridView.DataSource = source;

}

private TravelAgency travelAgency;

private BindingList<Travel> travels;

private BindingSource source;

/// <summary>

/// Фильтрация данных.

/// </summary>

private void filter()

{

string[] filters = filterTextBox.Text.Split(' ');

source.DataSource = new BindingList<Travel>(MyList.Filter<Travel>(travels.ToList(), filters));

}

/// <summary>

/// Запрет выделения фона ячейки.

/// </summary>

private void travelAgencyGridView\_SelectionChanged(object sender, EventArgs e)

{

travelAgencyGridView.ClearSelection();

}

/// <summary>

/// Фильтрация путешествий по нажатию кнопки.

/// </summary>

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

{

filter();

}

/// <summary>

/// Фильтрация путешествий по нажатию Enter.

/// </summary>

private void filterTextBox\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

filter();

}

}

/// <summary>

/// Обновление данных.

/// </summary>

private void updateGridView(object sender, FormClosedEventArgs e)

{

filter();

if (travelAgency != null)

Text = travelAgency.Name;

}

/// <summary>

/// Добавление нового путешествия в список путешествий текущего агентства.

/// </summary>

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

{

AboutTravelForm form = new AboutTravelForm(travelAgency);

form.Show();

form.FormClosed += updateGridView;

}

/// <summary>

/// Редактирование текущего агентства.

/// </summary>

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

{

AboutTravelAgencyForm form = new AboutTravelAgencyForm(travelAgency);

form.Show();

form.FormClosed += updateGridView;

}

/// <summary>

/// Удаление текущего агентства.

/// </summary>

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

{

DialogResult dr = MessageBox.Show("Are you sure you want to proceed? This data will be lost.", "Warning", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (dr == DialogResult.Yes)

{

TravelAgencyCollection tac = TravelAgencyCollection.GetDefaultInstance();

tac.Remove(travelAgency);

tac.Save();

this.Close();

}

}

/// <summary>

/// Выход из программы.

/// </summary>

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

{

Application.Exit();

}

/// <summary>

/// Выбор путешествия из списка.

/// </summary>

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

{

// Проверка на выбор заголовка столбца.

if (e.RowIndex == -1)

return;

int index = travelAgencyGridView.CurrentRow.Index;

Travel travel = (source.DataSource as BindingList<Travel>)[index];

AboutTravelForm form = new AboutTravelForm(travelAgency, travel);

form.Show();

form.FormClosed += updateGridView;

}

/// <summary>

/// Возвращение к предыдущей форме.

/// </summary>

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

{

this.Close();

}

/// <summary>

/// Сортировка путешествий по полю.

/// </summary>

private void travelAgencyGridView\_ColumnHeaderMouseClick(object sender, DataGridViewCellMouseEventArgs e)

{

List<Travel> data = (source.DataSource as BindingList<Travel>).ToList();

string property = travelAgencyGridView.Columns[e.ColumnIndex].DataPropertyName;

List<Travel> sortedData = MyList.SortBy<Travel>(data, property);

// Если списки одинаковы, меняем порядок

if (MyList.Compare<Travel>(sortedData, data))

sortedData.Reverse();

source.DataSource = new BindingList<Travel>(sortedData);

}

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

{

MessageBox.Show("Double click on travel to browse and edit travel information.\nClick on column header to order column.\nEnter search filters separated by space and click search button to filter travels.\nClick on 'Agency' in menu to make actions on the agency.\nClick 'Back' button to return to list of travel agencies.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Question);

}

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

{

MessageBox.Show("coursa4 - the best travel guide!\nCreated by Bohdan Yevchenko.\nCopyright © 2016, All Rights Reserved.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}