**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

ФАИС

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

по дисциплине **«Языки программирования высокого уровня»**

на тему: **«Делегаты и события»**

Выполнил: студент гр. ИП-22

Коваленко А. И.

Принял: ст. преподаватель Романькова Т. Л.

Дата сдачи отчета: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: ­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2022

**Делегаты и события**

**Цель**: *получить навыки использования делегатов и событий*

**Вариант 7.**

1. Подготовить текстовый файл, содержащий информацию о подписчиках на журналы: фамилию, адрес, названия журналов (разделителем в файле служит точка с запятой).
2. Подготовить текстовый файл, содержащий информацию о журналах: название, стоимость годовой подписки.
3. Класс **Подписчик** должен содержать следующие элементы: поля для хранения фамилии, адреса, поле-массив для хранения стоимости подписки на каждый журнал, поле-массив с названиями журналов, индексатор для доступа к элементам поля-массива с названиями, свойство для определения общей стоимости подписки, метод для вывода информации, методы для реализации шаблона «наблюдатель» (см. ниже).
4. Класс **Журнал** должен содержать следующие элементы: поля для хранения названия журнала, стоимости годовой подписки, методы для реализации шаблона «наблюдатель» (см. ниже).
5. Разработать программу, которая выполняет следующие действия:

* Считывает информацию из первого файла в массив объектов класса **Подписчик**.
* Считывает информацию из второго файла в массив объектов класса **Журнал**.
* Выводит информацию о подписчиках в виде.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№* | *Фамилия* | *Адрес* | *Издания* | *Стоимость* |
| *1* | *Иванов* | *Ул.Советская, д.25, кв.67* | *Крокодил, Садовод* | *98000* |
| *2* | *Петров* | *Ул.Победы, д.12, кв.5* | *Мурзилка* | *12000* |

* Информация в таблице должна быть отсортирована по убыванию стоимости.

1. Реализовать паттерн Наблюдатель
   1. Класс Подписчик реализует объект наблюдателя
   2. Класс Журнал реализует объект наблюдения

* На основе делегата при изменении названия журнала автоматически изменяется название соответствующего журнала в списке изданий для каждого подписчика. Для этого каждый подписчик должен быть «зарегистрирован» в соответствующем объекте класса **Журнал**. Журнал, для которого меняется название, выбирается пользователем.
* На основе стантартной обработки событий, при изменении стоимости годовой подписки на журнал ставятся в известность все подписчики на данный журнал кроме тех кто проживает на ул.Советская.

Листинг программы:

Program.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text.RegularExpressions;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

Subscriber[] allSubscribers = ReadingDataFromFileOfSubscribers();

Journal[] allJournals = ReadingDataFromFileOfJournals();

Array.Sort(allSubscribers);

RegistrationSubscribers(allSubscribers, allJournals);

OutputTableInformation(allSubscribers);

SelectingJournalToChangeName(allJournals, allSubscribers);

OutputTableInformation(allSubscribers);

SelectingJournalToChangePriceSubscription(allJournals, allSubscribers);

OutputTableInformation(allSubscribers);

Console.ReadKey();

}

public static string InputName()

{

Console.Write($"Введите новое название журнала: ");

string newNameJournal = Console.ReadLine();

return newNameJournal;

}

private static void OutputTableInformation(Subscriber[] allSubscribers)

{

Table table = new Table();

table.HeaderTable();

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

{

allSubscribers[i].InformationOutput(i + 1);

}

table.BottomTable();

}

private static Subscriber[] ReadingDataFromFileOfSubscribers()

{

Subscriber[] allSubscribers = new Subscriber[File.ReadLines("subscribers.txt").Count()];

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

{

allSubscribers[i] = new Subscriber();

}

try

{

if (allSubscribers.Length == 0)

{

throw new Exception("Файл с данными о подписчиках пуст.");

}

StreamReader readerFirstFile = new StreamReader("subscribers.txt");

string line;

for (int i = 0; !readerFirstFile.EndOfStream; i++)

{

line = readerFirstFile.ReadLine();

string[] data = line.Split(new char[] { ';' }, StringSplitOptions.RemoveEmptyEntries);

if (data.Length == 3)

{

allSubscribers[i].LastName = data[0];

allSubscribers[i].Address = data[1];

string[] namesOfJournalsOfLine = data[2].Split(new char[] { ',' }, StringSplitOptions.RemoveEmptyEntries);

allSubscribers[i].AllCostOfSubscriptions = new Dictionary<string, double>(namesOfJournalsOfLine.Length);/////

for (int j = 0; j < namesOfJournalsOfLine.Length; j++)

{

StreamReader readerSecondFile = new StreamReader("journals.txt");

for (int k = 0; !readerSecondFile.EndOfStream; k++)

{

string[] dataFromSecondFile = readerSecondFile.ReadLine().Split(new char[] { ';' }, StringSplitOptions.RemoveEmptyEntries);

if (dataFromSecondFile.Length != 2)

{

throw new Exception($"Данные о журналах не достоверные. {i + 1} строка.");

}

if (dataFromSecondFile[0] == namesOfJournalsOfLine[j].Trim())

{

allSubscribers[i].AllCostOfSubscriptions.Add(namesOfJournalsOfLine[j].Trim(), Convert.ToDouble(dataFromSecondFile[1]));

break;

}

}

readerSecondFile.Close();

}

}

else

{

throw new Exception($"Данные о подписчиках в файле не достоверные. {i + 1} строка.");

}

}

readerFirstFile.Close();

}

catch(Exception e)

{

Console.WriteLine($"Ошибка: {e.Message}");

return null;

}

return allSubscribers;

}

private static Journal[] ReadingDataFromFileOfJournals()

{

Journal[] allJournals = new Journal[File.ReadLines("journals.txt").Count()];

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

{

allJournals[i] = new Journal();

}

try

{

if (allJournals.Length == 0)

{

throw new Exception("Файл с данными о журналах пуст.");

}

StreamReader readerSecondFile = new StreamReader("journals.txt");

string line;

for (int i = 0; !readerSecondFile.EndOfStream; i++)

{

line = readerSecondFile.ReadLine();

string[] data = line.Split(new char[] { ';' }, StringSplitOptions.RemoveEmptyEntries);

if (data.Length == 2)

{

allJournals[i].NameOfJournal = data[0];

allJournals[i].SubscriptionPrice = Convert.ToDouble(data[1]);

}

else

{

throw new Exception($"Данные о журналах в файле не достоверные. {i + 1} строка.");

}

}

readerSecondFile.Close();

}

catch(Exception e)

{

Console.WriteLine($"Ошибка: {e.Message}");

return null;

}

return allJournals;

}

private static void RegistrationSubscribers(Subscriber[] allSubscribers, Journal[] allJournals)

{

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

{

for (int j = 0; j < allSubscribers.Length; j++)

{

foreach(KeyValuePair<string, double> costOfSubscriptions in allSubscribers[j].AllCostOfSubscriptions)

{

if (costOfSubscriptions.Key == allJournals[i].NameOfJournal)

{

allJournals[i].ObserverRegistration(new ChangeInName(allSubscribers[j].ChangeOfNameJournal));

break;

}

}

}

}

}

private static void SelectingJournalToChangeName(Journal[] allJournals, Subscriber[] allSubscribers)

{

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

{

Console.WriteLine($"{i+1}-{allJournals[i].NameOfJournal}");

}

int number;

Console.Write("Введите номер журнала, у которого необходимо изменить название: ");

string input = Console.ReadLine();

while (!Int32.TryParse(input, out number) || number > allJournals.Length || number < 1)

{

Console.Write("Произошла ошибка. Введите номер журнала, у которого необходимо изменить название: ");

input = Console.ReadLine();

}

allJournals[number - 1].NameOfJournal = InputName();

}

public static double InputPrice()

{

double price;

Console.Write($"Введите новую годовую стоимость подписки журнала: ");

string input = Console.ReadLine();

while (!Double.TryParse(input, out price) || price > 1000 || price < 1)

{

Console.Write($"Произошла ошибка. Введите новую годовую стоимость подписки журнала: ");

input = Console.ReadLine();

}

return price;

}

private static void SelectingJournalToChangePriceSubscription(Journal[] allJournals, Subscriber[] allSubscribers)

{

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

{

Console.WriteLine($"{i + 1}-{allJournals[i].NameOfJournal}-{allJournals[i].SubscriptionPrice}");

}

int number;

Console.Write("Введите номер журнала, у которого необходимо изменить стоимость годовой подписки: ");

string input = Console.ReadLine();

while (!Int32.TryParse(input, out number) || number > allJournals.Length || number < 1)

{

Console.Write("Произошла ошибка. Введите номер журнала, у которого необходимо изменить стоимость годовой подписки: ");

input = Console.ReadLine();

}

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

{

string pattern = @"(.\*)ул.Советская(.\*)";

if (!Regex.IsMatch(allSubscribers[i].Address, pattern, RegexOptions.IgnoreCase))

{

foreach (KeyValuePair<string, double> costOfSubscriptions in allSubscribers[i].AllCostOfSubscriptions)

{

if (costOfSubscriptions.Key == allJournals[number - 1].NameOfJournal)

{

allJournals[number - 1].PriceChange += allSubscribers[i].ChangeOfPriceJournal;

break;

}

}

}

}

allJournals[number - 1].SubscriptionPrice = InputPrice();

}

}

}

Subscriber.cs

using System;

using System.Collections.Generic;

namespace ConsoleApp1

{

class Subscriber : IComparable

{

private string lastName;

private string address;

public Dictionary<string, double> AllCostOfSubscriptions { get; set; }

public double TotalCostOfSubscription

{

get

{

double sum = 0;

foreach (KeyValuePair<string, double> costOfSubscriptions in AllCostOfSubscriptions)

{

sum += costOfSubscriptions.Value;

}

return sum;

}

}

public string LastName

{

get => lastName;

set => lastName = value;

}

public string Address

{

get => address;

set => address = value;

}

public void InformationOutput(int number)

{

Console.WriteLine("├────────┼───────────────────┼────────────────────────────────────────┼──────────────────────────────┼────────────────────┤");

Console.Write($"│ {number,-6} │ {LastName,-17} │ {Address,-38} │ ");

bool flag = false;

foreach (KeyValuePair<string, double> costOfSubscriptions in AllCostOfSubscriptions)

{

if (flag == false)

{

Console.Write($" {costOfSubscriptions.Key, -28}│");

Console.WriteLine($" {TotalCostOfSubscription, -18} │");

flag = true;

}

else

{

Console.WriteLine($"│ │ │ │ {costOfSubscriptions.Key,-28}│ │");

}

}

}

public int CompareTo(object obj)

{

Subscriber subscriber = obj as Subscriber;

if(subscriber != null)

{

return subscriber.TotalCostOfSubscription.CompareTo(this.TotalCostOfSubscription);

}

else

{

throw new Exception("Невозможно сравнить два объекта");

}

}

public void ChangeOfNameJournal(string newNameJournal, string old)

{

foreach(KeyValuePair<string, double> costOfSubscriptions in AllCostOfSubscriptions)

{

if (costOfSubscriptions.Key == old)

{

double tmp = costOfSubscriptions.Value;

AllCostOfSubscriptions.Remove(old);

AllCostOfSubscriptions.Add(newNameJournal, tmp);

break;

}

}

}

public void ChangeOfPriceJournal(double newPriceJournal, string key)

{

//foreach (KeyValuePair<string, double> costOfSubscriptions in AllCostOfSubscriptions)

//{

// if (costOfSubscriptions.Key == key)

// {

// //AllCostOfSubscriptions.Remove(key);

// //AllCostOfSubscriptions.Add(key, newPriceJournal);

// break;

// }

//}

AllCostOfSubscriptions[key] = newPriceJournal;

}

}

}

Journal.cs

using System;

namespace ConsoleApp1

{

public delegate void ChangeInName(string newNameJournal, string key);

public delegate void SubscriptionPriceEventHandler(double newPrice, string key);

class Journal

{

private string nameOfJournal;

private double subscriptionPrice;

public string NameOfJournal

{

get => nameOfJournal;

set

{

//if (observer != null)

//{

// observer(value, nameOfJournal);

//}

observer?.Invoke(value, nameOfJournal);

nameOfJournal = value;

}

}

public double SubscriptionPrice

{

get => subscriptionPrice;

set

{

PriceChange?.Invoke(value, nameOfJournal);

subscriptionPrice = value;

}

}

public event SubscriptionPriceEventHandler PriceChange;

ChangeInName observer;

public void ObserverRegistration(ChangeInName ob)

{

observer += ob;

}

}

}

Table.cs

using System;

namespace ConsoleApp1

{

class Table

{

public void HeaderTable()

{

Console.WriteLine("┌────────┬───────────────────┬────────────────────────────────────────┬──────────────────────────────┬────────────────────┐");

Console.WriteLine("│ № │ Фамилия │ Адрес │ Издания │ Стоимость │");

Console.WriteLine("│ │ │ │ │ │");

}

public void BottomTable()

{

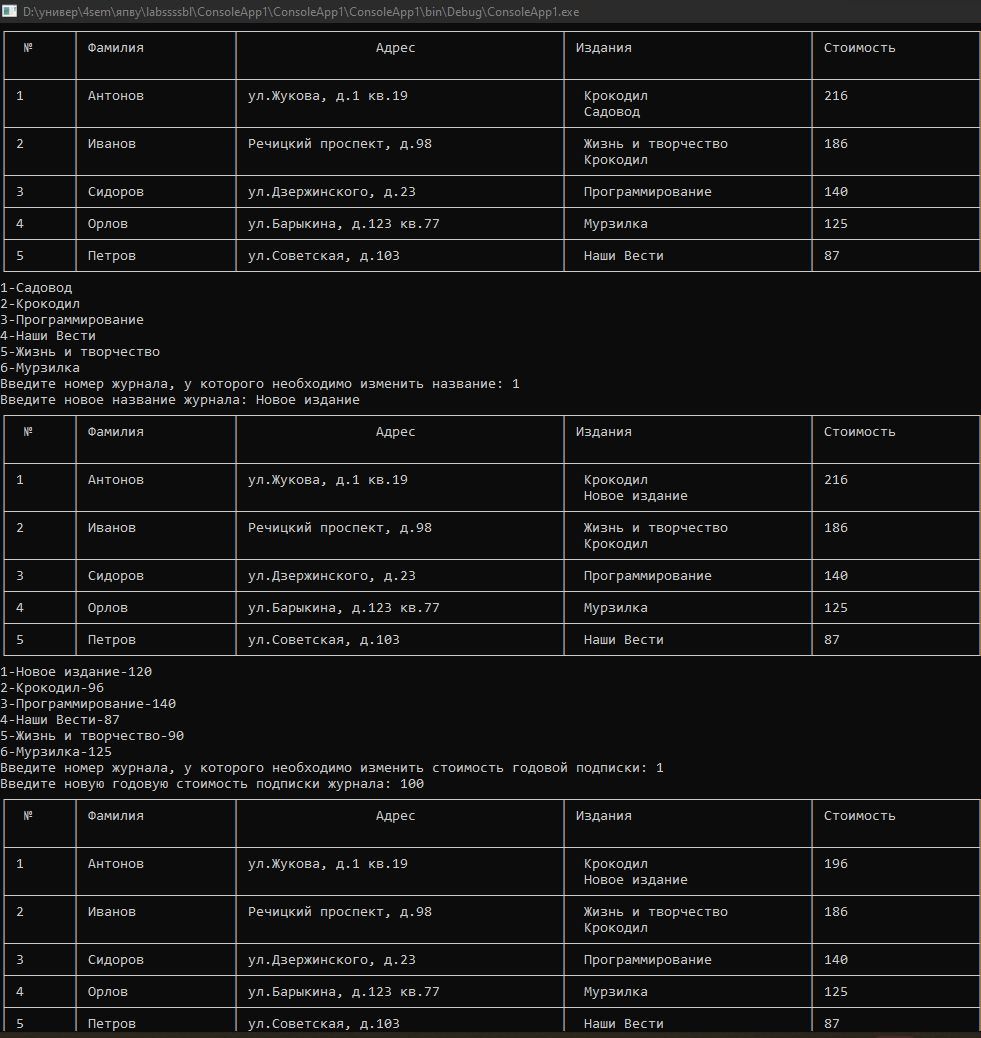
Console.WriteLine("└────────┴───────────────────┴────────────────────────────────────────┴──────────────────────────────┴────────────────────┘");

}

}

}

**Результат выполнения программы:**

****

**Вывод:**  
В ходе данной лабораторной работы были получены навыки в теме делегаты и события.