НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА ПРОГРАМУВАННЯ»

*Звiт з лабораторної роботи №1*

*Тема: «Розробка програм на С#.Консольні програми.»*

Виконав:   
ст. гр. КІТ-119б  
Шарма О.Р.

Перевірив:  
Бартош М.В.

Харків – 2021

***Індивідуальне завдання:***

1. Введення і збереження анкетних даних особової справи студента вузу: П.І.Б., дата

народження, дата надходження, індекс академ. групи (а, б ...), фт, спеціальність,

успішність (в%). Забезпечити валідацію даних, що вводяться.

2. Доступ до особистих полях реалізувати за допомогою властивостей класу, забезпечивши

необхідні обчислення і перевірку коректності встановлюваних значень. для

спрощеного доступу використовувати автоматично реалізовані властивості.

3. Відображення введених даних.

4. Демонстрація роботи з масивами об'єктів.

5. У звіті детально описати структуру розроблених класів і їх взаємозв'язку.

***Опис програми***

Дана програма містить 6 класів з методами.

**Клас MainMenu**

Головний клас програми, містить:

1. Метод Start

Функції що виконуються :

* 1. Створення контейнеру, який міститиме масив об’єктів типу Student.
  2. Запит у користувача чи бажає він додати студента.
  3. Якщо так, то запит користувача заповнити всі дані.
  4. Створення на основі даних об’єкта типу Student.
  5. Додання об’єкту до контейнеру.
  6. Виведення на екран вміст контейнера за допомогою foreach.

**Клас Student**

Містить поля – інформація про студенів. Містить сеттери та геттери, а також метод toString та два конструктори.

**Клас StudentArray**

Наслідується від IEnumerable та IEnumerator. Містить поля – масив студентів, розмір масиву. Також містить методи GetEnumerator, MoveNext,

**Клас Input**

Відповідає за вхідні дані

**Клас Validator**

Відповідає за валідацію вхідних даних

ТЕКСТ ПРОГРАМИ

**Файл MainMenu**

using System;

using System.Collections.Generic;

using System.Text;

namespace DotNetLab1

{

class Mainmenu

{

public static void Start()

{

var studentArray = new StudentArray();

while (true)

{

PrintMainMenu();

int choice;

try

{

choice = Input.EnterInt("choice");

switch (choice)

{

case 1:

{

studentArray.AddStudent(CreateStudent());

break;

}

case 2:

{

studentArray.printStudents();

studentArray.DeleteStudentByIndex(Input.EnterInt("index"));

break;

}

case 3:

{

studentArray.printStudents();

break;

}

case 0:

{

return;

}

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

public static void PrintMainMenu()

{

Console.WriteLine("Enter your choice:");

Console.WriteLine("1. Add a student");

Console.WriteLine("2. Remove student");

Console.WriteLine("3. Print list of students");

Console.WriteLine("0. Exit");

}

public static Student CreateStudent()

{

string surname = Input.EnterName("student's surname");

string name = Input.EnterName("student's name");

string patronymic = Input.EnterName("student's patronymic");

DateTime dob = Input.EnterDate("day of birth");

DateTime enterDate = Input.EnterDate("acquiring date");

string groupIndex = Input.EnterString("group index");

string faculty = Input.EnterUniInfo("faculty");

string specialty = Input.EnterUniInfo("speciality");

int academicPerformance = Input.EnterPercents("academic performance, in percents");

return new Student(surname, name, patronymic, dob, enterDate, groupIndex, faculty, specialty, academicPerformance);

}

}

}

**Файл Input**

using Microsoft.VisualBasic.CompilerServices;

using System;

using System.Collections.Generic;

using System.Text;

namespace DotNetLab1

{

class Input

{

private static string \_errorMessage = "Invalid input. Check and try again";

public static string EnterName(string fieldName)

{

Console.WriteLine("Enter " + fieldName + ":");

string name = Console.ReadLine();

while (!Validator.ValidateName(name))

{

Console.WriteLine(\_errorMessage);

name = Console.ReadLine();

}

return name;

}

public static string EnterUniInfo(string fieldName)

{

Console.WriteLine("Enter " + fieldName + ":");

string sentence = Console.ReadLine();

while (!Validator.ValidateSentence(sentence))

{

Console.WriteLine(\_errorMessage);

sentence = Console.ReadLine();

}

return sentence;

}

public static string EnterString(string fieldName)

{

Console.WriteLine("Enter " + fieldName + ":");

string str = Console.ReadLine();

return str;

}

public static int EnterPercents(string fieldName)

{

Console.WriteLine("Enter " + fieldName + ":");

while (true)

{

try

{

int value = Convert.ToInt32(Console.ReadLine());

if (Validator.ValidateIntByRange(0, 100, value))

{

return value;

}

continue;

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

public static int EnterInt(string fieldName)

{

Console.WriteLine("Enter " + fieldName + ":");

while (true)

{

try

{

int value = Convert.ToInt32(Console.ReadLine());

return value;

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

public static DateTime EnterDate(String fieldName)

{

Console.WriteLine("Enter " + fieldName + ":");

while (true)

{

try

{

Console.WriteLine("Enter year:");

int year = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter month:");

int month = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter day:");

int day = Convert.ToInt32(Console.ReadLine());

return new DateTime(year, month, day);

}

catch (Exception e)

{

Console.WriteLine("An exception occured while date input:" + e.Message);

}

}

}

}

}

**Файл Student**

using System;

using System.Collections.Generic;

using System.Text;

namespace DotNetLab1

{

class Student

{

public string Surname { get; set; }

public string name { get; set; }

public string patronymic { get; set; }

public DateTime dateOfBirth { get; set; }

public DateTime enterDate { get; set; }

public string groupIndex { get; set; }

public string faculty { get; set; }

public string specialty { get; set; }

public int academicPerformance { get; set; }

public Student() { }

public Student(string surname, string name, string patronymic, DateTime dob, DateTime enterDate, string groupIndex, string faculty, string specialty, int academicPerformance)

{

this.Surname = surname;

this.name = name;

this.patronymic = patronymic;

dateOfBirth = dob;

this.enterDate = enterDate;

this.groupIndex = groupIndex;

this.faculty = faculty;

this.specialty = specialty;

this.academicPerformance = academicPerformance;

}

public override string ToString()

{

return $"Surname: {Surname}\nName: {name}\nPatronymic: {patronymic}\nDate of birth: {dateOfBirth.ToString()}\nEnter date: {enterDate}\nGroup index: {groupIndex}\n" +

$"Faculty: {faculty}\nSpecialty: {specialty}\nAcademic performance: {academicPerformance}\n";

}

}

}

**Файл StudentArray**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Text;

namespace DotNetLab1

{

class StudentArray : System.Collections.IEnumerable

{

public Student[] students { get; set; }

public bool empty { get; set; } = false;

public void AddStudent(Student student)

{

if (students == null)

{

students = new Student[0];

}

var newArr = new Student[students.Length + 1];

for (int i = 0, length = students.Length; i < length; i++)

{

newArr[i] = students[i];

}

newArr[students.Length] = student;

students = newArr;

if (!empty && students.Length > 0)

{

empty = true;

}

}

public void DeleteStudentByIndex(int index)

{

if (students != null)

{

if (checkIndex(index))

{

Student[] newArr = new Student[students.Length - 1];

for (int i = 0; i < index; i++)

{

newArr[i] = students[i];

}

for (int i = index + 1, length = students.Length; i < length; i++)

{

newArr[i] = students[i];

}

students = newArr;

if (empty && students.Length == 0) empty = false;

}

else

{

Console.WriteLine("Index out of range");

}

}

else

{

Console.WriteLine("Array is empty");

}

}

private bool checkIndex(int index)

{

return index >= 0 && index <= students.Length - 1;

}

public void printStudents()

{

if (students != null)

{

int i = 0;

foreach (Student student in students)

{

Console.WriteLine(student);

}

}

else

{

Console.WriteLine("Array is empty");

}

}

//public void findStudent();

IEnumerator IEnumerable.GetEnumerator() => GetEnumerator();

public StudentsEnum GetEnumerator()

{

return new StudentsEnum(students);

}

}

internal class StudentsEnum : IEnumerator

{

private Student[] students;

int position = -1;

public StudentsEnum(Student[] students)

{

this.students = students;

}

public bool MoveNext()

{

position++;

return (position < students.Length);

}

public void Reset()

{

position = -1;

}

object IEnumerator.Current

{

get

{

return Current;

}

}

public Student Current

{

get

{

try

{

return students[position];

}

catch (IndexOutOfRangeException)

{

throw new InvalidOperationException();

}

}

}

}

}

**Файл Validator**

using System;

using System.Collections.Generic;

using System.Text;

using System.Text.RegularExpressions;

namespace DotNetLab1

{

class Validator

{

private static string \_namePattern = @"^(?<firstchar>(?=[A-Za-z]))((?<alphachars>[A-Za-z])|(?<specialchars>[A-Za-z]['-](?=[A-Za-z]))|(?<spaces> (?=[A-Za-z])))\*$";

private static string \_sentencePattern = @"\b[^\d\W]+\b";

public static bool ValidateName(string name)

{

return Regex.IsMatch(name.Trim(), \_namePattern, RegexOptions.IgnoreCase);

}

public static bool ValidateSentence(string sentence)

{

return Regex.IsMatch(sentence.Trim(), \_sentencePattern, RegexOptions.IgnoreCase);

}

public static bool ValidateIntByRange(int min, int max, int value)

{

return value >= min && value <= max;

}

}

}

РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ

Результати роботи програми приведено нижче на скріншотах:

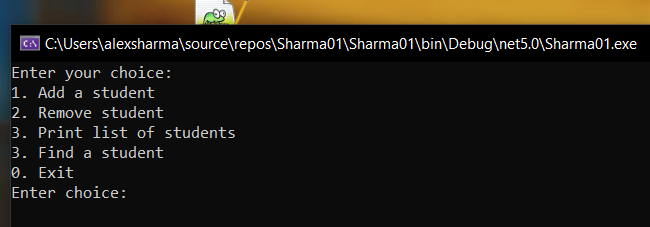


Рисунок 1 – головне меню

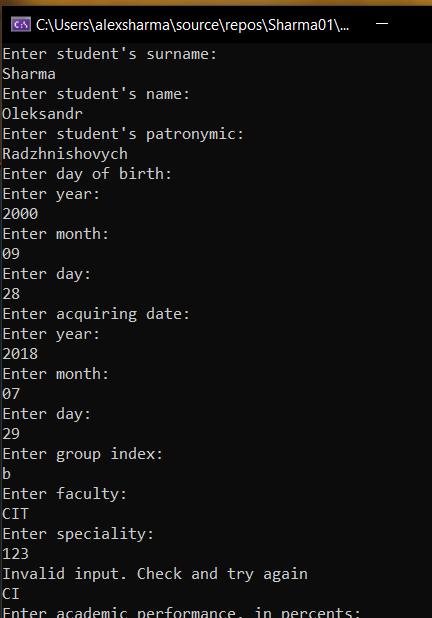


Рисунок 2 – додаємо студента

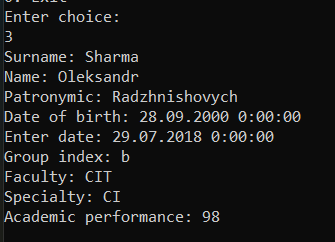


Рисунок 3 – додавання успішне

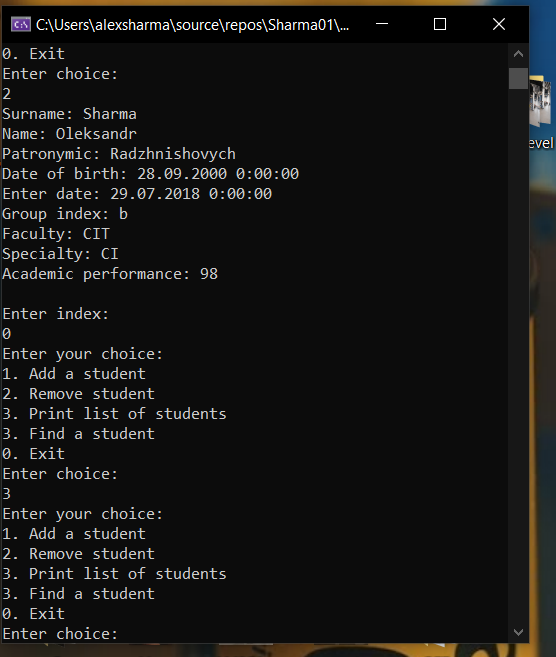


Рисунок 4 – видалення студента

Висновки:

В ході данної лабораторної роботи було розроблено та відлагоджено програму на мові C#, реалізовані усі задачі лабораторної роботи