Чугунов В.Ю  
КІТ-119а

Лабораторна робота №1

**РОЗРОБКА ПРОГРАМ НА С#. КОНСОЛЬНІ ПРОГРАМИ**

Завдання:

1. Введення та збереження анкетних даних особистої справи студента вузу: П.І.Б., дата народження, дата надходження, індекс академ. групи (а, б...), фт, спеціальність, успішність (у%). Забезпечити валідацію даних, що вводяться.
2. Доступ до особистих полів реалізувати за допомогою властивостей класу, забезпечивши необхідні обчислення та перевірку коректності встановлюваних значень. Для спрощеного доступу використовувати властивості, що автоматично реалізуються.
3. Відображення введених даних.
4. Демонстрація роботи з масивами об'єктів.

Опис класів

Student – клас, який відображує студента;

**Текст файлу Student.cs**

using System;

using System.Runtime.Serialization;

using System.Text;

namespace Lab01

{

[DataContract]

public sealed class Student

{

private string \_fullName;

private DateTime \_dateOfBirth;

private DateTime \_dateOfReceipt;

private string \_indexOfGroup;

private string \_faculty;

private string \_specialization;

private double \_universityPerfomance;

public Student() { }

public Student(string fullName, DateTime dateOfBirth, DateTime dateOfReceipt,

string indexOfGroup, string faculty,string specialization, double universityPerfomance)

{

Name = fullName;

DateOfBirth = dateOfBirth;

DateOfReceipt = dateOfReceipt;

IndexGroup = indexOfGroup;

Faculty = faculty;

Specialization = specialization;

UniversityPerfomance = universityPerfomance;

}

[DataMember]

public string Name

{

get

{

return \_fullName;

}

set

{

if (value.Length > 40)

{

Console.WriteLine("Error! Full name length might be less than 40 symbols");

Environment.Exit(1);

}

if (string.IsNullOrEmpty(value))

{

Console.WriteLine("Error! Input of [full name] is empty!");

Environment.Exit(1);

}

\_fullName = value;

}

}

[DataMember]

public DateTime DateOfBirth

{

get

{

return \_dateOfBirth;

}

set

{

if (value < new DateTime(2000, 1, 1) || value > DateTime.Today)

{

Console.WriteLine("You've entered wrong date of birth\n");

}

\_dateOfBirth = value;

}

}

[DataMember]

public DateTime DateOfReceipt

{

get

{

return \_dateOfReceipt;

}

set

{

if (value < new DateTime(2015, 1, 1) || value > DateTime.Today)

{

Console.WriteLine("You've entered wrong date of admission\n");

}

\_dateOfReceipt = value;

}

}

[DataMember]

public string IndexGroup

{

get

{

return \_indexOfGroup;

}

set

{

int temp;

if (String.IsNullOrEmpty(value))

{

Console.WriteLine("Error! Input of [index of group] is empty!");

Environment.Exit(1);

}

if (value.Length > 3)

{

Console.WriteLine("Error! Length of [index of group] might be less than 3!");

Environment.Exit(1);

}

if (int.TryParse(value, out temp))

{

Console.WriteLine("Error! Input [group index] is not correct!");

Environment.Exit(1);

}

\_indexOfGroup = value;

}

}

[DataMember]

public string Faculty

{

get

{

return \_faculty;

}

set

{

int temp;

if (String.IsNullOrEmpty(value))

{

Console.WriteLine("Error! Input of [faculty] is empty!");

Environment.Exit(1);

}

if (value.Length > 30)

{

Console.WriteLine("Error! Length of [faculty] might be less than 30!");

Environment.Exit(1);

}

if (int.TryParse(value, out temp))

{

Console.WriteLine("Error! Input [faculty] is not correct!");

Environment.Exit(1);

}

\_faculty = value;

}

}

[DataMember]

public string Specialization

{

get

{

return \_specialization;

}

set

{

int temp;

if (String.IsNullOrEmpty(value))

{

Console.WriteLine("Error! Input of [Specialization] is empty!");

Environment.Exit(1);

}

if (value.Length > 20)

{

Console.WriteLine("Error! Length of [Specialization] might be less than 20!");

Environment.Exit(1);

}

if (int.TryParse(value, out temp))

{

Console.WriteLine("Error! Input [Specialization] is not correct!");

Environment.Exit(1);

}

\_specialization = value;

}

}

[DataMember]

public double UniversityPerfomance

{

get

{

return \_universityPerfomance;

}

set

{

if (value < 0 || value > 100)

{

Console.WriteLine("You've entered wrong university performance\n");

}

\_universityPerfomance = value;

}

}

public override string ToString()

{

var builder = new StringBuilder();

builder.Append("\nName:").Append(\_fullName).Append("\nDate of birth: ").Append(\_dateOfBirth)

.Append("\nDate of Receipt: ").Append(\_dateOfReceipt).Append("\nGroup Index: ").Append(\_indexOfGroup)

.Append("\nFaculty: ").Append(\_faculty).Append("\nSpecialization").Append(\_specialization)

.Append("\nUniversity Performance: ").Append(\_universityPerfomance).Append("%\n");

/\* return $"\nName: {\_fullName}\nDate of birth: {\_dateOfBirth}\nDate of Receipt: {\_dateOfReceipt}\nGroup Index: {\_indexOfGroup}\nFaculty: {\_faculty}\n" +

$"Specialization: {\_specialization}\nUniversity Performance: {\_universityPerfomance}%\n";\*/

return builder.ToString();

}

public override bool Equals(object obj)

{

Student another = obj as Student;

return another != null && (Name).Equals((another.Name));

}

public override int GetHashCode()

{

return (Name).GetHashCode();

}

}

}

**Текст файлу Program.cs**

using System;

namespace Lab01

{

class Program

{

static void Main(string[] args)

{

var students = new Student[] { new Student("Vadim Chugunov", DateTime.Parse("01-11-2001"),

DateTime.Parse("10-05-2019"), "a", "CIT", "Engineering", 12.4),

new Student("Oleksandr Ivanchenko", DateTime.Parse("18-11-2002"), DateTime.Parse("15-1-2019"),

"b", "CIT", "Engineering", 90.5)};

// Printing out students' data

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

{

Console.WriteLine(students[i].ToString());

}

Console.ReadLine();

}

}

}

**РЕЗУЛЬТАТ РОБОТИ**

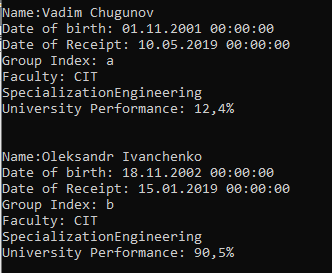
****

Рисунок 1 – Результат роботи програми

**ВИСНОВОК**

Під час виконання лабораторної роботи, набули практичні навички з реалізації класу Student, який відображує модель студента. Створили масив таких об’єктів.