Автор: Сиромятников Марк КІТ-119а

Дата: 11.12.2021

**Лабораторна робота 2**

**Тема**. Колекції об'єктів С#. Форматований вивід

Задачі:

1. Створення власного класа контейнера для реалізації колекції об'єктів.
2. Для розроблених класів контейнерів забезпечити можливість використання їх об'єктів у циклах foreach як джерела даних.
3. Накопичення даних списку студентів ВНЗ у вигляді колекції об'єктів.
4. Пошук та відображення даних особистої справи обраного студента.

Опис класів

Container – власний клас контейнера для реалізації колекції об'єктів;

ContainerEnumerator – клас, який реалізує інтерфейс IEnumerator;

Текст програми

Container.cs

using System;

using System.Collections;

using syromiatnikov01;

namespace syromiatnikov02

{

/// <summary>

/// Class Container

/// class that implements class container

/// for collection of students

/// </summary>

public class Container : IEnumerable

{

/// <summary>

/// Private field students

/// </summary>

protected Student[] \_students;

public Student[] Students => \_students;

/// <summary>

/// Constructor with one parameter

/// </summary>

/// <param name="students"></param>

public Container(Student[] students)

{

\_students = new Student[students.Length];

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

{

\_students[i] = students[i];

}

}

/// <summary>

/// Method that adds student to collection

/// </summary>

/// <param name="student"></param>

public void Add(Student student)

{

if (student == null)

{

Console.WriteLine("Student is null");

return;

}

var position = \_students.Length;

Array.Resize(ref \_students, \_students.Length + 1);

\_students[position] = student;

}

/// <summary>

/// Method that removes student from collection

/// </summary>

/// <param name="student"></param>

/// <returns>True if student was removed otherwise false</returns>

public bool Remove(Student student)

{

if (student == null)

{

return false;

}

var pos = -1;

for (var i = 0; i < \_students.Length; i++)

{

if (\_students[i].Equals(student))

{

pos = i;

break;

}

}

if (pos == -1)

{

return false;

}

var students = new Student[\_students.Length - 1];

for (var i = 0; i < pos; i++)

{

students[i] = \_students[i];

}

for (var i = pos + 1; i < \_students.Length; i++)

{

students[i - 1] = \_students[i];

}

\_students = students;

return true;

}

/// <summary>

/// Method that finds student in collection

/// </summary>

/// <param name="student"></param>

/// <returns>If such student exists returns it otherwise null</returns>

public Student Find(Student student)

{

for (var i = 0; i < \_students.Length; i++)

{

if (\_students[i].Equals(student))

{

return \_students[i];

}

}

return null;

}

/// <summary>

/// Implemented GetEnumerator method

/// </summary>

/// <returns>ContainerEnum</returns>

public IEnumerator GetEnumerator()

{

return new ContainerEnumerator(\_students);

}

}

}

ContainerEnumerator.cs

using syromiatnikov01;

using System;

using System.Collections;

namespace syromiatnikov02

{

/// <summary>

/// Class ContainerEnum

/// class that implements IEnumerator for student class

/// </summary>

public sealed class ContainerEnumerator : IEnumerator

{

/// <summary>

/// Private fields of a class

/// </summary>

private Student[] \_students;

private int \_position = -1;

/// <summary>

/// Constructor with one parameter

/// </summary>

/// <param name="students"></param>

public ContainerEnumerator(Student[] students)

{

\_students = students;

}

/// <summary>

/// Implemented Current property

/// </summary>

public object Current

{

get

{

try

{

return \_students[\_position];

}

catch (IndexOutOfRangeException)

{

throw new InvalidOperationException();

}

}

}

/// <summary>

/// Implemented MoveNext method

/// </summary>

/// <returns></returns>

public bool MoveNext()

{

\_position++;

return \_position < \_students.Length;

}

/// <summary>

/// Implemented Reset method

/// </summary>

public void Reset()

{

\_position = -1;

}

}

}

Program.cs

using System;

using syromiatnikov01;

namespace syromiatnikov02

{

class Program

{

static void Main(string[] args)

{

var customStudent = new Student("Momot", "Roman", "Evegenievich", DateTime.Parse("10-8-2001"), DateTime.Parse("16-05-2019"), "119a", "CIT", "Computer engineering", 80);

var students = new Student[] { new Student("Bily", "Vadim", "Ivanovich", DateTime.Parse("12-6-2001"), DateTime.Parse("16-05-2019"), "119a", "CIT", "Computer engineering", 100),

new Student("Menshakov", "Dmytro", "Olegovich", DateTime.Parse("16-11-2000"), DateTime.Parse("23-8-2019"), "119a", "CIT", "Computer engineering", 90)};

var list = new Container(students);

list.Add(customStudent);

foreach (var item in list)

{

Console.WriteLine(item.ToString());

}

list.Remove(new Student("Menshakov", "Dmytro", "Olegovich", DateTime.Parse("16-11-2000"), DateTime.Parse("23-8-2019"), "119a", "CIT", "Computer engineering", 90));

foreach (var item in list)

{

Console.WriteLine(item.ToString());

}

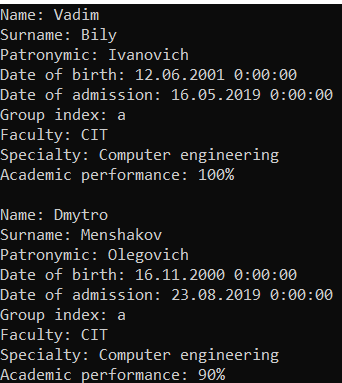
var stud = list.Find(customStudent);

Console.ReadLine();

}

}

}



Результати роботи програми

**Висновок:** у результаті виконання лабораторної роботи було створено клас Container для реалізації колекції об'єктів, для нього була забезпечена можливість використання його об'єктів у циклі foreach як джерела даних.