Чугунов В.Ю  
КІТ-119а

Лабораторна робота №2

**КОЛЕКЦІЇ ОБ'ЄКТІВ С#. ФОРМАТОВАНИЙ ВИВІД**

Завдання:

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

ContainerEnum – клас, який реалізує інтерфейс IEnumerator;

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

**Текст класу Сontainer.cs**

public class Container : IEnumerable

{

private Student[] \_students;

delegate double task(Student[] student);

public Container(Student[] pArray)

{

\_students = new Student[pArray.Length];

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

{

\_students[i] = pArray[i];

}

}

/\* Add,remove,find methods\*/

public void Add(Student student)

{

if (student == null)

{

throw new ArgumentNullException(nameof(student), "Student is null");

}

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

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

{

newArr[i] = \_students[i];

}

newArr[newArr.Length - 1] = student;

\_students = newArr;

}

public bool Remove(Student student)

{

if (student == null)

{

return false;

}

int pos = -1;

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

{

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

{

pos = i;

break;

}

}

if (pos == -1)

{

return false;

}

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

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

{

newArr[i] = \_students[i];

}

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

{

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

}

\_students = newArr;

return true;

}

public Student Find(Student student)

{

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

{

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

{

return \_students[i];

}

}

return null;

}

public void EditData(Student student)

{

int index = -1;

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

{

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

{

index = i;

break;

}

}

if (index != -1)

{

Console.WriteLine("Remember, you can edit only {Fullname, DateOfReceipt, Faculty} fields!\n");

try

{

Console.WriteLine("Enter new Full Name: ");

\_students[index].Name = Console.ReadLine();

Console.WriteLine("\nEnter new Date Of Receipt: ");

\_students[index].DateOfReceipt = DateTime.Parse(Console.ReadLine());

Console.WriteLine("\nEnter new name of Faculty: ");

\_students[index].Faculty = Console.ReadLine();

Console.WriteLine("\nStudent data is modified successfuly!\n ");

}

catch (FormatException ex)

{

Console.WriteLine(ex.Message);

}

}

else

{

Console.WriteLine("There is no student in list\n");

}

}

public void ShowStudentInfo(Student student)

{

int pos = -1;

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

{

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

{

pos = i;

break;

}

}

if (pos != -1)

{

var builder = new StringBuilder();

Console.WriteLine("\n===========Data -> (Course, Group&Semester, Age)===========\n");

Console.WriteLine("Course&Semester info:");

builder.AppendFormat("Course: {0}\nSemester: {1}\n", (DateTime.Now.Year - student.DateOfReceipt.Year) + 1,

Math.Ceiling((double)((12 \* (DateTime.Now.Year - student.DateOfReceipt.Year) + DateTime.Now.Month - student.DateOfReceipt.Month)

- 2 \* (DateTime.Now.Year - student.DateOfReceipt.Year))) / 5);

Console.WriteLine(builder.ToString());

builder.Clear();

Console.WriteLine("\nGroup info:");

builder.AppendFormat("Faculty: {0}\nSpecialty: {1}\nDate of admission: {2}\nGroup index: {3}", student.Faculty,

student.Specialization, student.DateOfReceipt.Year, student.IndexGroup);

Console.WriteLine(builder.ToString());

builder.Clear();

Console.WriteLine("\nAge info:");

builder.AppendFormat("Years: {0}\nMonth: {1}\nDays: {2}\n", DateTime.Now.Year - student.DateOfBirth.Year,

(Math.Abs(DateTime.Now.Month - student.DateOfBirth.Month)) - 1, DateTime.Now.Day);

Console.WriteLine(builder.ToString());

builder.Clear();

}

else

{

Console.WriteLine("No student in List!\n");

}

}

public void WriteToFile()

{

var jsonFormatter = new DataContractJsonSerializer(typeof(Student[]));

try

{

using (var file = new FileStream("students\_list.json", FileMode.Create))

{

try

{

jsonFormatter.WriteObject(file, \_students);

Console.WriteLine("Successfully written to file\n");

}

catch (System.Runtime.Serialization.SerializationException ex)

{

Console.WriteLine(ex.Message);

}

}

}

catch (UnauthorizedAccessException ex)

{

Console.WriteLine(ex.Message);

}

}

public void ReadFromFile()

{

if (\_students != null)

{

var jsonFormatter = new DataContractJsonSerializer(typeof(Student[]));

try

{

using (var file = new FileStream("students\_list.json", FileMode.Open))

{

try

{

\_students = jsonFormatter.ReadObject(file) as Student[];

Console.WriteLine("Successfully read from file\n");

}

catch (System.Runtime.Serialization.SerializationException ex)

{

Console.WriteLine(ex.Message);

}

}

}

catch (FileNotFoundException ex)

{

Console.WriteLine(ex.Message);

}

}

else

{

Console.WriteLine("There are no students in container\n");

}

}

public void ShowTableInfo()

{

var builder = new StringBuilder();

var sep = new string('-', 76);

builder.AppendFormat("\n{0,-25}{1,-20}{2,-20}{3,-10}", "Full name", "Faculty", "Specialization", "Group Index");

Console.WriteLine(builder);

Console.WriteLine(sep);

foreach (var student in \_students)

{

builder.Clear();

builder.AppendFormat("{0,-25}{1,-20}{2,-20}{3, -10}", student.Name, student.Faculty, student.Specialization, student.IndexGroup);

Console.WriteLine(builder);

}

Console.WriteLine(sep);

}

public bool RemoveForOption()

{

var builder = new StringBuilder();

builder.Append("Enter option of the deleting:\n")

.Append("1 - Group index\n").Append("2 - Specialty\n").Append("3 - Faculty\n\n");

Console.WriteLine(builder);

var temp = Console.ReadLine();

IComparer comparator = null;

switch (temp)

{

case "1":

Console.WriteLine("Write name of group index:");

temp = Console.ReadLine();

comparator = new CompareGroups();

break;

case "2":

Console.WriteLine("Write name of specialty:");

temp = Console.ReadLine();

comparator = new CompareSpecialty();

break;

case "3":

Console.WriteLine("Write name of faculty:");

temp = Console.ReadLine();

comparator = new CompareFaculty();

break;

default:

temp = "";

Console.WriteLine("Error!\n");

break;

}

if (temp.Length != 0)

{

int sizePrev = \_students.Length;

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

{

if (comparator.Compare(\_students[i], temp) == 0)

{

Remove(\_students[i]);

i--;

}

}

if (sizePrev != \_students.Length)

{

return true;

}

}

return false;

}

public double AvgCount()

{

IComparer comparator = null;

task functionD = null;

Console.WriteLine("\n\nCount of avarage Age or University Performance:");

Console.WriteLine("1 - University Performance");

Console.WriteLine("2 - Age");

Console.WriteLine("Write 1 or 2 ");

var temp = Console.ReadLine();

if (temp == "1")

{

functionD = CountAvgUniversityPer;

}

else if (temp == "2")

{

functionD = CountAvgAge;

}

else

{

Console.WriteLine("Error!Try again!");

return -1;

}

var builder = new StringBuilder();

builder.Append("Enter option of the count:\n")

.Append("1 - Group\n").Append("2 - Specialty\n").Append("3 - Faculty\n\n");

Console.WriteLine(builder);

temp = Console.ReadLine();

switch (temp)

{

case "1":

Console.WriteLine("Write name of group:");

temp = Console.ReadLine();

comparator = new CompareGroups();

break;

case "2":

Console.WriteLine("Write name of specialty:");

temp = Console.ReadLine();

comparator = new CompareSpecialty();

break;

case "3":

Console.WriteLine("Write name of faculty:");

temp = Console.ReadLine();

comparator = new CompareFaculty();

break;

default:

temp = "";

Console.WriteLine("Error!\n");

break;

}

if (temp.Length != 0)

{

int size = 0;

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

{

if (comparator.Compare(\_students[i], temp) == 0)

{

size++;

}

}

Student[] students = new Student[size];

size = 0;

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

{

if (comparator.Compare(\_students[i], temp) == 0)

{

students[size] = \_students[i];

size++;

}

}

return functionD(students);

}

return -1;

}

public double CountAvgAge(Student[] list)

{

double temp = 0;

foreach (var listItem in list)

{

temp += DateTime.Now.Year - listItem.DateOfBirth.Year;

}

return temp / list.Length;

}

public double CountAvgUniversityPer(Student[] list)

{

double temp = 0;

foreach (var listItem in list)

{

temp += listItem.UniversityPerfomance;

}

return temp / list.Length;

}

private class CompareGroups : IComparer

{

public int Compare(object x, object y)

{

Student student = (Student)x;

return student.IndexGroup.CompareTo(y);

}

}

private class CompareSpecialty : IComparer

{

public int Compare(object x, object y)

{

Student student = (Student)x;

string data = (string)y;

return student.Specialization.CompareTo(data);

}

}

private class CompareFaculty : IComparer

{

public int Compare(object x, object y)

{

Student student = (Student)x;

string data = (string)y;

return student.Faculty.CompareTo(data);

}

}

public IEnumerator GetEnumerator()

{

return new ContainerEnum(\_students);

}

}

**Текст класу ContainerEnum.cs**

public sealed class ContainerEnum : IEnumerator

{

private Student[] \_students;

private int \_position = -1;

public ContainerEnum(Student[] students)

{

\_students = students;

}

public object Current

{

get

{

try

{

return \_students[\_position];

}

catch (IndexOutOfRangeException)

{

throw new InvalidOperationException();

}

}

}

public bool MoveNext()

{

\_position++;

return \_position < \_students.Length;

}

public void Reset()

{

\_position = -1;

}

}

**Текст файлу Program.cs**

using System;

using Lab01;

namespace Lab02

{

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", 70.4),

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

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

var firstStudent = new Student("Vadim Chugunov", DateTime.Parse("01-11-2001"), DateTime.Parse("10-05-2019"), "a", "CIT", "Engineering", 70.4);

var secondStudent = new Student("Sergey Dremen", DateTime.Parse("02-01-2002"), DateTime.Parse("10-02-2019"), "a", "CIT", "Engineering", 89.4);

var thirdStudent = new Student("Oleksandr Ivanchenko", DateTime.Parse("10-11-2002"), DateTime.Parse("15-1-2019"), "b", "CIT", "Engineering", 90.5);

var studentsArray = new Student[] { firstStudent, secondStudent };

var list = new Container(studentsArray);

Console.WriteLine("==========================List==========================");

foreach (var listItem in list)

{

Console.WriteLine(listItem.ToString());

}

list.Add(thirdStudent);

Console.WriteLine("==========================List after added element==========================");

foreach (var listItem in list)

{

Console.WriteLine(listItem.ToString());

}

}

}

}

**РЕЗУЛЬТАТ РОБОТИ**

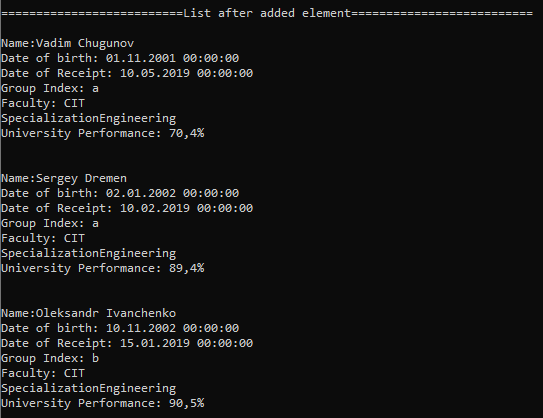


Рисунок 1 – Результат роботи програми

**ВИСНОВОК**

Під часвиконання лабораторної роботи, набули практичні навчики з створення класу Container для реалізації колекції об'єктів. У нашому випадку – список студентів.