

Автор: Абдуллін Олексій

Група: КІТ-119а

Дата: 17.12.2021

Лабораторна робота № 3

Тема: Обробка колекцій об'єктів. Робота з файлами

Задачі:

1. Збереження введених даних у файлі для подальшого редагування.
2. Відновлення раніше збережених даних із файлу.
3. Пошук та редагування даних студента.
4. Видалення даних особистої справи студента.

Опис класів

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

MyCollectionEnum – клас, який реалізує інтерфейс IEnumerable;

Student – клас, який відображує студента;

IPrinter – інтерфейс для виводу у консоль;

Menu – клас для роботи меню;

Текст програми

MyCollectionEnum

```
using System;
using System.Collections;
using System.Collections.Generic;

namespace Abdullin03
{
    public class MyCollectionEnum : IEnumerator
    {
        public List<Student> _stud;

        int position = -1;

        public MyCollectionEnum(List<Student> stud)
        {
            _stud = stud;
        }

        public bool MoveNext()
        {
            position++;
            return (position < _stud.Count);
        }

        public void Reset()
        {
            position = -1;
        }

        object IEnumerator.Current
        {
            get
            {
                return Current;
            }
        }

        public Student Current
        {
            get
            {
                try
                {
                    return _stud[position];
                }
                catch (IndexOutOfRangeException)
                {
                    throw new InvalidOperationException();
                }
            }
        }
    }
}
```

MyCollection

```
using System.Collections;
using System.Collections.Generic;

namespace Abdullin03
{
    public class MyCollection : IEnumerable
    {
        private List<Student> _studentsArray = new List<Student>();

        public void Add(Student student)
        {
            if (student is null)
            {
                student = new Student();
            }
            _studentsArray.Add(student);
        }

        public bool Remove(int id)
        {
            if (id >= _studentsArray.Count || 0 > id)
            {
                return false;
            }
            _studentsArray.RemoveAt(id);
            return true;
        }

        public void Clear()
        {
            _studentsArray.Clear();
        }

        public Student GetStudentById(int id)
        {
            int i = 0;
            if (id >= _studentsArray.Count || 0 > id)
            {
                return null;
            }
            foreach (var stud in _studentsArray)
            {
                if (id == i)
                {
                    return stud;
                }
            }
            return null;
        }

        public Student GetStudent(Student student)
        {
            foreach (var stud in _studentsArray)
            {
                if (stud.Equals(student))
                {
                    return student;
                }
            }
            return null;
        }

        public List<Student> GetStudents()
        {
            return _studentsArray;
        }
    }
}
```

```

    }

    IEnumerator IEnumerable.GetEnumerator()
    {
        return (IEnumerator)GetEnumerator();
    }

    public MyCollectionEnum GetEnumerator()
    {
        return new MyCollectionEnum(_studentsArray);
    }
}

```

IPrinter

```

using System;

namespace Abdullin03
{
    public interface IPrinter
    {
        void Print(string str);
    }
    public class ConsolePrinter : IPrinter
    {
        public void Print(string str)
        {
            Console.WriteLine(str);
        }
    }
}

```

Student

```

using System;
using System.Runtime.Serialization;

namespace Abdullin03
{
    [DataContract]
    public class Student
    {
        [DataMember]
        public string FirstName { get; set; }
        [DataMember]
        public string SurName { get; set; }
        [DataMember]
        public string GroupIndex { get; set; }
        [DataMember]
        public string Faculty { get; set; }
        [DataMember]
        public int Specialization { get; set; }
        [DataMember]
        public int AcademicPerformance { get; set; }
        [DataMember]
        public DateTime DateOfBirth { get; set; }
        [DataMember]
        public DateTime DateOfEnter { get; set; }
        [IgnoreDataMember]
        public IPrinter Printer { get; set; }
        public Student() : this("Oleksii", "Abdullin", "a", "CIT", 123, 86, new
DateTime(2002, 5, 31), new DateTime(2019, 8, 12))
        {

```

```

    }
    public Student(string FirstName, string SurName, string GroupIndex, string
Faculty,
        int Specialization, int AcademicPerformance, DateTime DateOfBirth, DateTime
DateOfEnter)
    {
        this.FirstName = FirstName;
        this.SurName = SurName;
        this.GroupIndex = GroupIndex;
        this.Faculty = Faculty;
        this.Specialization = Specialization;
        this.AcademicPerformance = AcademicPerformance;
        this.DateOfBirth = DateOfBirth;
        this.DateOfEnter = DateOfEnter;
        this.Printer = new ConsolePrinter();
    }
    public void Print()
    {
        Printer.Print(ToString());
    }
    public override string ToString()
    {
        return "Fristname: " + FirstName + "Surname: " + SurName +
            "\nDate of birth: " + DateOfBirth.Day + "." + DateOfBirth.Month + "." +
DateOfBirth.Year +
            "\nDate of enter: " + DateOfEnter.Day + "." + DateOfEnter.Month + "." +
DateOfEnter.Year +
            "\nIndex of group: " + GroupIndex + "\nFaculty: " + Faculty +
            "\nSpecialization: " + Specialization + "\nAcademic Performance: " +
AcademicPerformance + "\n";
    }

    public override bool Equals(object obj)
    {
        if (obj == null)
        {
            return false;
        }
        Student s = obj as Student;
        if (s == null)
        {
            return false;
        }
        return s.FirstName == this.FirstName &&
            s.SurName == this.SurName &&
            s.GroupIndex == this.GroupIndex &&
            s.Faculty == this.Faculty &&
            s.Specialization == this.Specialization &&
            s.AcademicPerformance == this.AcademicPerformance &&
            s.DateOfBirth == this.DateOfBirth &&
            s.DateOfEnter == this.DateOfEnter;
    }
}
}
}

```

Menu

```

using System;
using System.Collections.Generic;
using System.Text.RegularExpressions;
using System.Runtime.Serialization.Json;
using System.IO;
using System.Text;

```



```

        if (!regex_string.IsMatch(groupIndex))
        {
            Console.WriteLine("\nError! Invalid datatype.\n");
            break;
        }

        Console.Write("Enter faculty of student: ");
        faculty = Console.ReadLine();
        if (!regex_string.IsMatch(faculty))
        {
            Console.WriteLine("\nError! Invalid datatype.\n");
            break;
        }

        Console.Write("Enter specialization of student: ");
        if (!int.TryParse(Console.ReadLine(), out specialization))
        {
            Console.WriteLine("\nError! Invalid datatype.\n");
            break;
        }

        Console.Write("Enter academic performance of student: ");
        if (!int.TryParse(Console.ReadLine(), out academicPerformance))
        {
            Console.WriteLine("\nError! Invalid datatype.\n");
            break;
        }

        if (academicPerformance > 100 || academicPerformance < 0)
        {
            Console.WriteLine("\nError! Invalid value\n");
            break;
        }

        Console.Write("Enter date of birth of student (e.g. 01/01/2001 or
1.1.2001): ");

        if (!DateTime.TryParse(Console.ReadLine(), out dateOfBirth))
        {
            Console.WriteLine("\nError! Invalid datatype.\n");
            break;
        }

        Console.Write("Enter date of enter to university (e.g. 01/01/2001
or 1.1.2001): ");

        if (!DateTime.TryParse(Console.ReadLine(), out dateOfEnter))
        {
            Console.WriteLine("\nError! Invalid datatype.\n");
            break;
        }

        Student s = new Student(firstname, surname, groupIndex, faculty,
specialization,
            academicPerformance, dateOfBirth, dateOfEnter);
        MyCollection.Add(s);
        break;
    case 2:
        int id;
        Console.Write("\nEnter student id: ");
        if (!int.TryParse(Console.ReadLine(), out id))
        {
            Console.WriteLine("\nError! Invalid datatype. \n");
            break;
        }

        bool result = MyCollection.Remove(id);

```

```

        if (result)
        {
            Console.WriteLine("\nStudent was deleted successfully.\n");
        }
        else
        {
            Console.WriteLine("\nError! Invalid student id. \n");
        }
        break;
    case 3:
        int i = 0;
        foreach (var stud in MyCollection)
        {
            Console.WriteLine("\nStudent ID: " + i);
            stud.Print();
            i++;
        }
        break;
    case 4:
        using (var file = new FileStream(path, FileMode.Create))
        {
            using (var jsonw =
JsonReaderWriterFactory.CreateJsonWriter(file, Encoding.GetEncoding("utf-8")))
            {
                serializer.WriteObject(jsonw,
MyCollection.GetStudents());
                jsonw.Flush();
            }
        }
        break;
    case 5:
        List<Student> obj = Activator.CreateInstance<List<Student>>();
        using (FileStream file = new FileStream(path, FileMode.Open))
        {
            using (XmlDictionaryReader jsonr =
JsonReaderWriterFactory.CreateJsonReader(file,
Encoding.GetEncoding("utf-8"),
XmlDictionaryReaderQuotas.Max, null))
            {
                obj = serializer.ReadObject(jsonr) as List<Student>;
            }
        }
        MyCollection.Clear();
        foreach (var stud in obj)
        {
            stud.Printer = new ConsolePrinter();
            MyCollection.Add(stud);
        }
        break;
    case 0:
        inMenu = false;
        break;
    default:
        if (option == -1)
        {
            break;
        }
        Console.WriteLine("\nIncorrect option. Try again.\n");
        break;
    }
}
}
}
}
}

```


Результати роботи програми

```
Menu options:
1. Add
2. Remove
3. Show all students
4. Serialization
5. Deserialization
0. Exit
Enter your option: 5
Menu options:
1. Add
2. Remove
3. Show all students
4. Serialization
5. Deserialization
0. Exit
Enter your option: 3

Student ID: 0
Fristname: OleksiiSurname: Abdullin
Date of birth: 31.5.2002
Date of enter: 12.8.2019
Index of group: a
Faculty: CIT
Specialization: 123
Academic Performance: 86

Student ID: 1
Fristname: BogdanSurname: Jinx
Date of birth: 7.7.2002
Date of enter: 4.8.2020
Index of group: c
Faculty: CS
Specialization: 121
Academic Performance: 67
```

Рисунок 1 – Результати роботи програми

Висновок: У результаті виконання лабораторної роботи було модифіковано клас Container для збереження поточних даних у файл у форматі Json та їх відновлення, було додано методи пошуку та видалення особистих даних студента.