Автор: Абдуллін Олексій

Група: КІТ-119а

Дата: 17.12.2021

Лабораторна робота№ 4

Тема: Властивості класу. Обробка рядків. StringBuilder

Задачі:

1. Вивід для обраного студента назви групи (абревіатура назви факультету, номер спеціальності, рік надходження, індекс).
2. Вивід для обраного студента номера курсу та семестру на поточний момент.
3. Розрахунок та вивід для обраного студента віку на поточний момент (до дня).
4. Продемонструвати ефективне використання StringBuilder для обробки рядків.

Опис класів

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

MyCollectionEnum – клас, який реалізує інтерфейс IEnumerator;

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 Abdullin04
{
    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 Abdullin04
{
    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 Abdullin04
{
    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 Abdullin04
{
    [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;
using System.Xml;

namespace Abdullin04
{
    public class Menu
    {
        public void MenuStudents()
        {
            var MyCollection = new MyCollection();
            MyCollection.Add(new Student());

            int option;
            bool inMenu = true;
            string path = "lab03.json";
            var serializer = new DataContractJsonSerializer(typeof(List<Student>));
            while (inMenu)
            {
                Console.WriteLine("Menu options:");
                Console.WriteLine("1. Add");
                Console.WriteLine("2. Remove");
                Console.WriteLine("3. Show all students");
                Console.WriteLine("4. Serialization");
                Console.WriteLine("5. Deserialization");
                Console.WriteLine("0. Exit");
                Console.Write("Enter your option: ");

                if (!int.TryParse(Console.ReadLine(), out option))
                {
                    Console.WriteLine("\nError! Invalid datatype.\n");
                    option = -1;
                }

                switch (option)
                {
                    case 1:
                        Regex regex_string = new Regex(@"^[a-z]+$",
RegexOptions.IgnoreCase);

                        string firstname;
                        string surname;
                        string groupIndex;
                        string faculty;
                        int specialization;
                        int academicPerformance;
                        DateTime dateOfBirth;
                        DateTime dateOfEnter;

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

                        Console.Write("Enter surname of student: ");
```

```
                        surname = Console.ReadLine();
                        if (!regex_string.IsMatch(surname))
                        {
                            Console.WriteLine("\nError! Invalid datatype.\n");
                            break;
                        }

                        Console.Write("Enter index of group: ");
                        groupIndex = Console.ReadLine();
                        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.Write("\nError! Invalid datatype. \n");
                            break;
                        }

                        bool result = MyCollection.Remove(id);

                        if (result)
                        {
                            Console.Write("\nStudent was deleted succsessfully.\n");
                        }
                        else
                        {
                            Console.Write("\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;
                }
            }
        }
    }
}
```

Результат виконання програми

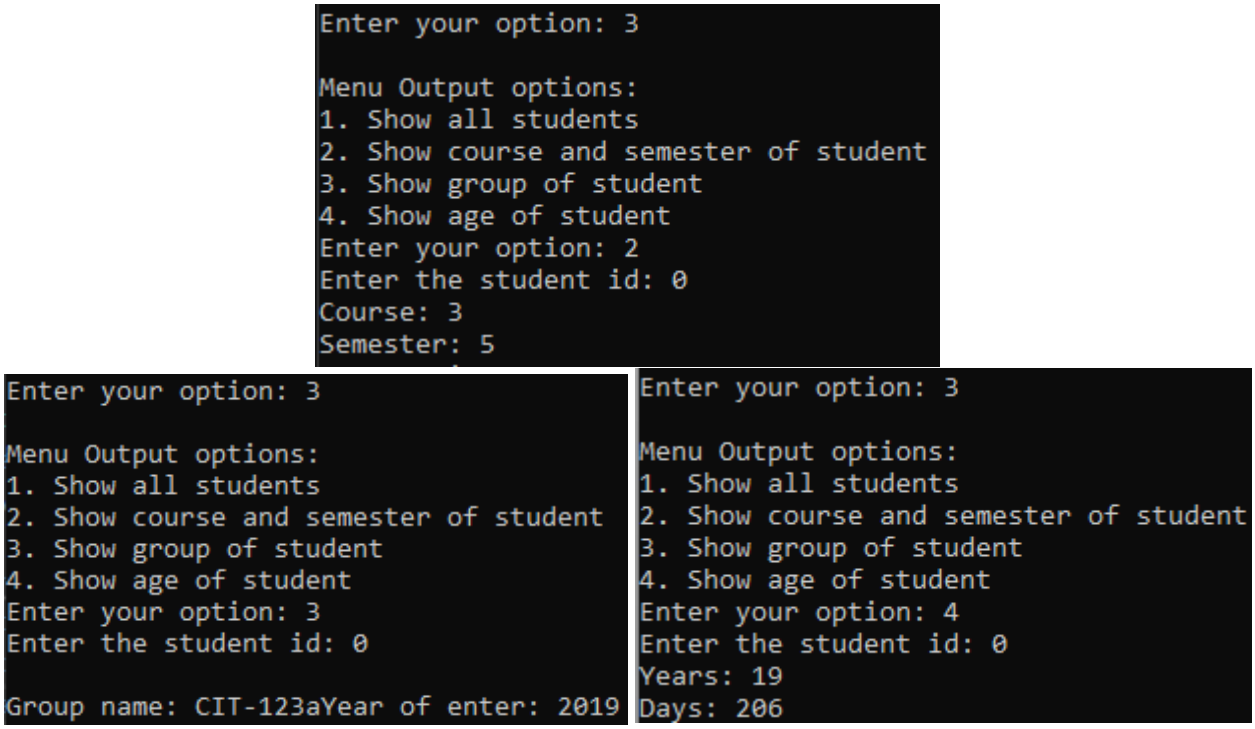

Рисунок 1 – Результати роботи програми

Висновок: У результаті виконання лабораторної роботи було додано можливість вивести для обраного студента назву групи (абревіатура назви факультету, номер спеціальності, рік надходження, індекс), номер курсу та семестру на поточний момент, віку на поточний момент (до дня).