Автор: Меньшаков Дмитро КІТ-119а

Дата: 11.12.2021

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

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 menshakov01;
using System.Runtime.Serialization.Json;
using System.IO;
using System.Text;

namespace menshakov04
{
```

```
    /// <summary>
    /// Class Container
    /// class that implements class container
    /// for collection of students
    /// </summary>
    public sealed class Container
    {
        /// <summary>
        /// Private field students
        /// </summary>
        private Student[] _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)
            {
                throw new ArgumentNullException(nameof(student), "Student is null");
            }

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

            for (var i = 0; i < _students.Length; i++)
            {
                newArr[i] = _students[i];
            }

            newArr[newArr.Length - 1] = student;
            _students = newArr;
        }

        /// <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 newArr = new Student[_students.Length - 1];

            for (var i = 0; i < pos; i++)
            {
                newArr[i] = _students[i];
            }
            for (var i = pos + 1; i < _students.Length; i++)
            {
                newArr[i - 1] = _students[i];
            }

            _students = newArr;
            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>
        /// Method that writes students' data to JSON file
        /// </summary>
        public void WriteToFile()
        {
            var jsonFormatter = new DataContractJsonSerializer(typeof(Student[]));

            try
            {
                using (var file = new FileStream("students.json", FileMode.Create))
                {
                    try
                    {
                        jsonFormatter.WriteObject(file, _students);
```

```
                        Console.WriteLine("Data were successfully written to file\n")
;
                    }
                    catch (System.Runtime.Serialization.SerializationException ex)
                    {
                        Console.WriteLine(ex.Message);
                    }
                }
            }
            catch (UnauthorizedAccessException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Method that reads students' data from JSON file
        /// </summary>
        public void ReadFromFile()
        {
            if (_students != null)
            {
                var jsonFormatter = new DataContractJsonSerializer(typeof(Student[]))
;

                try
                {
                    using (var file = new FileStream("students.json", FileMode.Open))
                    {
                        try
                        {
                            _students = jsonFormatter.ReadObject(file) as Student[];
                            Console.WriteLine("Data were 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");
            }
        }

        /// <summary>
        /// Method that allows to edit data of chosen student
        /// </summary>
        /// <param name="student"></param>
        public void EditData(Student student)
        {
            var pos = -1;
```

```
            for (var i = 0; i < _students.Length; i++)
            {
                if (_students[i].Equals(student))
                {
                    pos = i;
                    break;
                }
            }

            if (pos != -1)
            {
                Console.WriteLine("Enter what field you want to edit:\n1) Name\n2) Su
rname\n3) Patronymic\n4) Date of birth\n5) Date of admission\n" +
                    "6) Group index\n7) Faculty\n8) Specialty\n9) Academic performanc
e\n");
                var option = Console.ReadLine();
                try
                {
                    switch (option)
                    {
                        case "Name":
                            _students[pos].Name = Console.ReadLine();
                            break;
                        case "Surname":
                            _students[pos].Surname = Console.ReadLine();
                            break;
                        case "Patronymic":
                            _students[pos].Patronymic = Console.ReadLine();
                            break;
                        case "Date of birth":
                            _students[pos].DateOfBirth = DateTime.Parse(Console.ReadL
ine());
                            break;
                        case "Date of admission":
                            _students[pos].DateOfAdmission = DateTime.Parse(Console.R
eadLine());
                            break;
                        case "Group index":
                            _students[pos].GroupIndex = char.Parse(Console.ReadLine()
);
                            break;
                        case "Faculty":
                            _students[pos].Faculty = Console.ReadLine();
                            break;
                        case "Specialty":
                            _students[pos].Specialty = Console.ReadLine();
                            break;
                        case "Academic performance":
                            _students[pos].AcademicPerformance = int.Parse(Console.Re
adLine());
                            break;
                        default:
                            Console.WriteLine("Invalid option\n");
                            break;
                    }
                }
                catch (FormatException ex)
                {
                    Console.WriteLine(ex.Message);
                }
            }
```

```
            else
            {
                Console.WriteLine("There is no such student in collection\n");
            }
        }

        /// <summary>
        /// Method that prints chosen data about student
        /// </summary>
        /// <param name="student"></param>
        public void ShowData(Student student)
        {
            var pos = -1;

            for (var i = 0; i < _students.Length; i++)
            {
                if (_students[i].Equals(student))
                {
                    pos = i;
                    break;
                }
            }

            if (pos != -1)
            {
                var dataForPrint = new StringBuilder();
                Console.WriteLine("Enter what data you want to get:\n1) group index\n
2) course\n3) age\n");
                var option = Console.ReadLine();
                switch (option)
                {
                    case "group index":
                        dataForPrint.AppendFormat("\nFaculty: {0}\nSpecialty: {1}\nDa
te of admission: {2}\nGroup index: {3}", student.Faculty,
                            student.Specialty, student.DateOfAdmission.Year, student.
GroupIndex);
                        Console.WriteLine(dataForPrint.ToString());
                        dataForPrint.Clear();
                        break;
                    case "course":
                        dataForPrint.AppendFormat("\nCourse: {0}\nSemester: {1}\n", (
DateTime.Now.Year - student.DateOfAdmission.Year) + 1,
                            Math.Ceiling((double)((12 * (DateTime.Now.Year -
 student.DateOfAdmission.Year) + DateTime.Now.Month - student.DateOfAdmission.Month)
                            - 2 * (DateTime.Now.Year -
 student.DateOfAdmission.Year))) / 5);
                        Console.WriteLine(dataForPrint.ToString());
                        dataForPrint.Clear();
                        break;
                    case "age":
                        dataForPrint.AppendFormat("\nYears: {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(dataForPrint.ToString());
                        dataForPrint.Clear();
                        break;
                    default:
                        Console.WriteLine("Invalid option\n");
                        break;
                }
```

```
            }
            else
            {
                Console.WriteLine("There is no such student in collection\n");
            }
        }

        /// <summary>
        /// Implemented GetEnumerator method
        /// </summary>
        /// <returns>ContainerEnum</returns>
        public IEnumerator GetEnumerator()
        {
            return new ContainerEnumerator(_students);
        }
    }
}
```

ContainerEnumerator.cs

```
using menshakov01;
using System;
using System.Collections;

namespace menshakov02
{
    /// <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 menshakov01;

namespace menshakov04
{
    class Program
    {
        static void Main(string[] args)
        {
            var customStudent = new Student("Momot", "Roman", "Evegenievich", DateTim
e.Parse("10-8-2001"), DateTime.Parse("16-05-
2019"), 'a', "CIT", "Computer engineering", 80);
            var students = new Student[] { new Student("Bily", "Vadim", "Ivanovich",
DateTime.Parse("12-6-2001"), DateTime.Parse("16-05-
2019"), 'a', "CIT", "Computer engineering", 100),
                new Student("Menshakov", "Dmytro", "Olegovich", DateTime.Parse("16-
11-2000"), DateTime.Parse("23-8-2019"), 'a', "CIT", "Computer engineering", 90)};
            var list = new Container(students);
            list.Add(customStudent);
            /*list.WriteToFile();
            list.ReadFromFile();*/
            list.ShowData(customStudent);
            list.EditData(customStudent);
            foreach (var item in list)
            {
                Console.WriteLine(item.ToString());
            }

            list.Remove(new Student("Menshakov", "Dmytro", "Olegovich", DateTime.Pars
e("16-11-2000"), DateTime.Parse("23-8-
2019"), 'a', "CIT", "Computer engineering", 90));
            foreach (var item in list)
            {
```

```
                Console.WriteLine(item.ToString());
            }

            var stud = list.Find(customStudent);

            Console.ReadLine();
        }
    }
}
```

```
Enter what data you want to get:
1) group index
2) course
3) age

age

Years: 20
Month: 3
Days: 11

Name: Vadim
Surname: Bily
Patronymic: Ivanovich
Date of birth: 12.06.2001 0:00:00
Date of admission: 16.05.2019 0:00:00
Group index: a
Faculty: CIT
Specialty: Computer engineering
Academic performance: 100%

Name: Dmytro
Surname: Menshakov
Patronymic: Olegovich
Date of birth: 16.11.2000 0:00:00
Date of admission: 23.08.2019 0:00:00
Group index: a
Faculty: CIT
Specialty: Computer engineering
Academic performance: 90%

Name: Roman
Surname: Momot
Patronymic: Evegenievich
Date of birth: 10.08.2001 0:00:00
Date of admission: 16.05.2019 0:00:00
Group index: a
Faculty: CIT
Specialty: Computer engineering
Academic performance: 80%
```

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).