Кам’янець-Подільський національний університет

 імені Івана Огієнка

КАФЕДРА КОМП’ЮТЕРНИХ НАУК

Навчальна дисципліна «Об’єктно-орієнтоване програмування»

ЛАБОРАТОРНА РОБОТА #0106

Тема: Класи колекцій і протоколи ітерації

Варіант №3

Виконав:

студент 1-го курсу

групи KNms1-B21

Волошин В.Ю.

Прийняла:

старший викладач,

Мястковська М.О.

Кам’янець-Подільський – 2023

1. **Хід роботи.**

Класи колекцій та протоколи ітерації є ключовими елементами в мові програмування C#, які дозволяють зручно та ефективно опрацьовувати набори даних. У C# існують різноманітні класи колекцій, такі як List<T>, Dictionary<TKey, TValue>, Queue<T>, Stack<T>, та ArrayList, кожен із яких має свої особливості та призначення для роботи з даними.

При використанні колекцій важливо розуміти протоколи ітерації. Інтерфейси IEnumerable, IEnumerator, IEnumerable<T>, та IEnumerator<T> визначають стандарти для ітерації через елементи колекцій. Це дозволяє зручно використовувати цикл foreach для перегляду даних в колекціях.

Класи колекцій надають різноманітні методи для роботи з елементами, такі як Add та Remove для додавання та видалення елементів. Властивості Count та Length надають інформацію про кількість елементів у колекції.

Використання цих класів і протоколів ітерації робить код більш гнучким, дозволяючи зручно працювати з колекціями будь-якого типу та розміру. Ці концепції допомагають полегшити управління даними в програмах, забезпечуючи ефективні та оптимізовані методи опрацювання ітерацій та маніпулювання елементами колекцій.

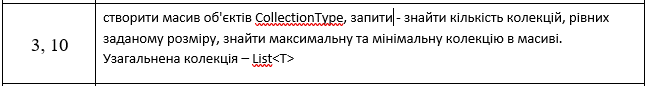
1. **Завдання до виконання.**

Створіть узагальнений клас CollectionType<T>. Визначити в класі конструктори, методи додавання та видалення елементів, інші необхідні методи та, якщо потрібно, перевантажені операції. Визначити індексатори та властивості. CollectionType можна реалізувати на основі стандартних колекцій (List, Stack, Array тощо).

Візьміть, створений тип (клас) з лабораторної №1, та реалізувати в ньому інтерфейс IComparable<T>. Використовуйте цей клас як параметр вашого узагальненого класу. Створіть кілька колекцій. Виконайте сортування, LINQ-запити відповідно до варіанта.

Виконайте кілька складних LINQToObject запитів (мінімум 5) до колекції об'єктів, використовуючи одночасно більше трьох операцій (приклад: where + select + orderBy, first + any + min).

Створіть узагальнену стандартну колекцію з простору імен System.Collections вказану у варіанті з рядками та виконайте введення-виведення, пошук рядків, що містять певне значення, підрахунок кількості рядків довжини n, сортування у зростаючому та спадному порядку.



1. **Лістинг програми.**

class CollectionType<T>

{

    private List<T> items;

    public CollectionType()

    {

        items = new List<T>();

    }

    public CollectionType(IEnumerable<T> initialItems)

    {

        items = new List<T>(initialItems);

    }

    // Додавання елемента

    public void AddItem(T item)

    {

        items.Add(item);

    }

    // Видалення елемента

    public void RemoveItem(T item)

    {

        items.Remove(item);

    }

    // Отримання елемента за індексом

    public T this[int index]

    {

        get

        {

            if (index < 0 || index >= items.Count)

            {

                throw new IndexOutOfRangeException("Index is out of range.");

            }

            return items[index];

        }

        set

        {

            if (index < 0 || index >= items.Count)

            {

                throw new IndexOutOfRangeException("Index is out of range.");

            }

            items[index] = value;

        }

    }

    // Кількість елементів у колекції

    public int Count

    {

        get { return items.Count; }

    }

    // Індексатор для доступу за значенням елемента

    public int IndexOf(T item)

    {

        return items.IndexOf(item);

    }

    // Вивід всіх елементів на екран

    public void DisplayItems()

    {

        foreach (var item in items)

        {

            Console.WriteLine(item);

        }

    }

}

public class Employee : IComparable<Employee>

{

    public string LastName { get; set; }

    public string Initials { get; set; }

    public int BirthYear { get; set; }

    public double Salary { get; set; }

    public static int ObjectCount { get; private set; }

    public Employee(string lastName, string initials, int birthYear, double salary)

    {

        LastName = lastName;

        Initials = initials;

        BirthYear = birthYear;

        Salary = salary;

        ObjectCount++;

    }

    // Реалізація методу CompareTo для IComparable<T>

    public int CompareTo(Employee other)

    {

        // Порівнюємо спочатку за прізвищем, потім за роком народження

        int result = LastName.CompareTo(other.LastName);

        if (result == 0)

        {

            result = BirthYear.CompareTo(other.BirthYear);

        }

        return result;

    }

}

class CollectionHandler<T>

{

    private List<T> collection;

    public CollectionHandler(List<T> initialCollection)

    {

        collection = initialCollection;

    }

    public void PrintCollection()

    {

        foreach (var item in collection)

        {

            Console.WriteLine(item);

        }

    }

    public void SearchByValue(T searchValue)

    {

        collection = collection.Where(item => item.Equals(searchValue)).ToList();

    }

    public int CountStringsByLength(int length)

    {

        return collection.Count(item => item.ToString().Length == length);

    }

    public void SortCollectionAscending()

    {

        collection = collection.OrderBy(item => item.ToString()).ToList();

    }

    public void SortCollectionDescending()

    {

        collection = collection.OrderByDescending(item => item.ToString()).ToList();

    }

}

class Program

{

    static void Main()

    {

        //task 1

        // Приклад використання узагальненого класу CollectionType<T>

        CollectionType<int> intCollection = new CollectionType<int>();

        intCollection.AddItem(1);

        intCollection.AddItem(2);

        intCollection.AddItem(3);

        Console.WriteLine("Collection of integers:");

        intCollection.DisplayItems();

        // Приклад використання індексатора

        Console.WriteLine("Element at index 1: " + intCollection[1]);

        // Приклад використання методу видалення

        intCollection.RemoveItem(2);

        Console.WriteLine("Collection after removing element 2:");

        intCollection.DisplayItems();

        Console.WriteLine();

        //task 2

        // Створення колекції з об'єктами Employee

        List<Employee> employees = new List<Employee>

        {

            new Employee("Doe", "J.", 1980, 50000),

            new Employee("Smith", "A.", 1990, 60000),

            new Employee("Johnson", "M.", 1985, 55000),

            new Employee("Williams", "S.", 1982, 52000)

        };

        // Вивід несортованого списку

        Console.WriteLine("Unsorted employees:");

        foreach (var employee in employees)

        {

            Console.WriteLine($"{employee.LastName}, {employee.Initials}, {employee.BirthYear}, {employee.Salary}");

        }

        // Сортування за прізвищем та роком народження

        employees.Sort();

        // Вивід відсортованого списку

        Console.WriteLine("\nSorted employees:");

        foreach (var employee in employees)

        {

            Console.WriteLine($"{employee.LastName}, {employee.Initials}, {employee.BirthYear}, {employee.Salary}");

        }

        // LINQ-запит для виводу працівників, які мають зарплату більше 55000

        var highSalaryEmployees = from emp in employees

                                   where emp.Salary > 55000

                                   select emp;

        Console.WriteLine("\nEmployees with salary > 55000:");

        foreach (var employee in highSalaryEmployees)

        {

            Console.WriteLine($"{employee.LastName}, {employee.Initials}, {employee.BirthYear}, {employee.Salary}");

        }

        Console.WriteLine();

        //task 3

        // 1. Вивести інформацію про працівників, які мають зарплату більше 50000 та впорядкувати за прізвищем.

        var highSalaryOrderedByLastName = employees

            .Where(e => e.Salary > 50000)

            .OrderBy(e => e.LastName)

            .Select(e => $"{e.LastName}, {e.Initials}, {e.BirthYear}, {e.Salary}");

        Console.WriteLine("1. Employees with salary > 50000, ordered by last name:");

        foreach (var info in highSalaryOrderedByLastName)

        {

            Console.WriteLine(info);

        }

        Console.WriteLine();

        // 2. Знайти працівника з найвищою зарплатою та вивести його інформацію.

        var highestSalaryEmployee = employees.OrderByDescending(e => e.Salary).First();

        Console.WriteLine("2. Employee with the highest salary:");

        Console.WriteLine($"{highestSalaryEmployee.LastName}, {highestSalaryEmployee.Initials}, {highestSalaryEmployee.BirthYear}, {highestSalaryEmployee.Salary}");

        Console.WriteLine();

        // 3. Знайти середню зарплату всіх працівників.

        var averageSalary = employees.Average(e => e.Salary);

        Console.WriteLine($"3. Average salary of all employees: {averageSalary}");

        Console.WriteLine();

        // 4. Вивести перші три працівники, які народилися після 1980 року, впорядковані за роком народження.

        var youngestEmployees = employees

            .Where(e => e.BirthYear > 1980)

            .OrderBy(e => e.BirthYear)

            .Take(3)

            .Select(e => $"{e.LastName}, {e.Initials}, {e.BirthYear}, {e.Salary}");

        Console.WriteLine("4. First three employees born after 1980, ordered by birth year:");

        foreach (var info in youngestEmployees)

        {

            Console.WriteLine(info);

        }

        Console.WriteLine();

        // 5. Перевірити, чи існує хоча б один працівник із зарплатою більше 60000.

        var hasHighSalaryEmployee = employees.Any(e => e.Salary > 60000);

        Console.WriteLine($"5. Does any employee have a salary greater than 60000? {hasHighSalaryEmployee}");

        Console.WriteLine();

        // task 4

        List<string> stringCollection = new List<string>

        {

            "apple", "orange", "banana", "grape", "kiwi", "pear", "mango", "strawberry"

        };

        CollectionHandler<string> stringCollectionHandler = new CollectionHandler<string>(stringCollection);

        Console.WriteLine("All lines:");

        stringCollectionHandler.PrintCollection();

        string searchValue = "kiwi";

        Console.WriteLine($"\nLines containing '{searchValue}':");

        stringCollectionHandler.SearchByValue(searchValue);

        stringCollectionHandler.PrintCollection();

        int lengthToCount = 6;

        Console.WriteLine($"\nNumber of rows of length {lengthToCount}: {stringCollectionHandler.CountStringsByLength(lengthToCount)}");

        Console.WriteLine("\nSort in ascending order:");

        stringCollectionHandler.SortCollectionAscending();

        stringCollectionHandler.PrintCollection();

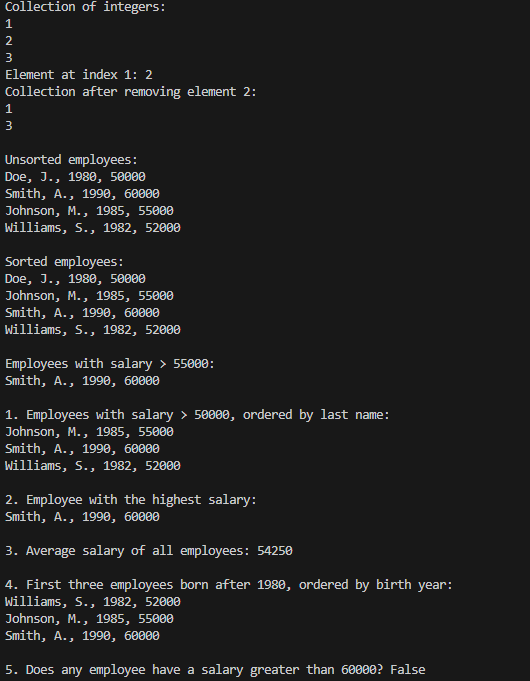
        Console.WriteLine("\nSort in descending order:");

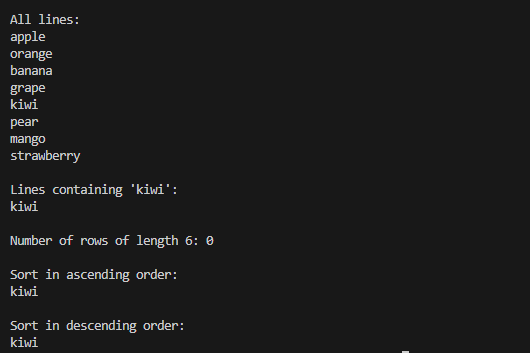
        stringCollectionHandler.SortCollectionDescending();

        stringCollectionHandler.PrintCollection();

    }

}

1. **Результат роботи.**



1. **Посилання на проект.**