Лабораторна робота №7

Тема: Вбудовані інтерфейси в C#.

Мета: Вивчити особливості роботи з вбудованими інтерфейсами у C#.

Завдання: Здобути навички створення, відлагодження та тестування проектів обробки даних з використанням вбудованих інтерфейсів мовою ***C#***.

(max: 5 балів)

Теоретичні відомості:

У попередній лекції ми говорили, що інтерфейс це повністю абстрактний клас, всі методи якого абстрактні (не містять реалізації).

Це означає, що клас, який наслідує інтерфейс, зобов'язаний повністю реалізувати *всі методи інтерфейсу*. У цьому відмінність від класу, що успадковує абстрактний клас, де похідний клас може реалізувати лише *деякі методи* абстрактного класу.

Головне призначення інтерфейсів – забезпечення множинного наслідування класів.

В бібліотеці FCL платформи .Net Framework є багато вбудованих інтерфейсів, які використовуються самими класами бібліотеки FCL, а також можуть використовуватися прикладними програмами. Розглянемо деякі з них.

1. Впорядкованість об'єктів і інтерфейс IComparable

Часто, коли створюється клас, бажано задати відношення порядку на його об'єктах для того, щоб можна було якимсь чином порівнювати об'єкти. Такий клас слід оголосити спадкоємцем інтерфейсу `IComparable`. Цей інтерфейс має всього один метод `CompareTo (object obj)`, що повертає ціле значення, позитивне, негативне або рівне нулю, залежно від виконання відношення "більше", "менше" або "рівно".

Таким чином, метод `CompareTo (object obj)` повинен повертати:

- 0, якщо поточний об'єкт і параметр рівні;
- негативне число, якщо поточний об'єкт менше параметра;
- позитивне число, якщо поточний об'єкт більше параметра.

При використанні цього інтерфейсу в класі спочатку визначають метод `CompareTo`, а після цього вводять перевизначені операції для порівняння об'єктів звичним чином з використанням знаків операцій відношення (<,>,==).

Приклад 1 . Розглянемо клас Person

```
class Person
    {
        public string Name;        //ім'я
        public int Age;              // вік
        public string Role;         // роль
        public string GetName() { return Name; }
        public int GetAge() { return Age; }
          }
```

Введемо відношення порядку на класі Person, зробивши цей клас спадкоємцем інтерфейсу `IComparable`. Реалізуємо в цьому класі метод інтерфейсу `CompareTo` для порівняння об'єктів:

```
public class Person:IComparable
   {
      public int CompareTo( object pers)
      {
         const string s = "Порівнюваний об'єкт не належить класу Person";
         Person p = pers as Person;
         if (!p.Equals(null))
            return (Name.CompareTo(p.Name));
         throw new ArgumentException (s);
      }
      // інші компоненты класу
   }
```

Оскільки аргумент в методі інтерфейсу належить типу `object`, перед виконанням порівняння його потрібно привести до типу `Person`. Для приведення використовується операція **as**, що дозволяє перевірити коректність виконання приведення. Якщо приведення неможливе, то неможливо виконати і порівняння об'єктів. В цьому випадку генерується виключення, яке може обробити розумним чином лише клієнт класу, що намагався виконати порівняння. У самому класі `Person` можна лише пояснити ситуацію, передавши інформацію об'єкту виключення.

При перевірці на значення null використовується відношення `Equals`, а не звичайна рівність, яка буде перевизначена.

Операція **is** використовується в логічних виразах. Логічний вираз

```
obj is T
```

вірний (true), якщо об'єкт obj належить типу T, і невірний (false) в іншому випадку.

Оператор призначення:

```
obj = P as T;
```

призначає об'єкту `obj` об'єкт P, приведений до типу T, якщо таке приведення можливе, інакше об'єкту призначається значення null.

Визначивши метод `CompareTo` для класу `Person`, ми тим самим ввели відношення порядку для об'єктів цього класу. Звичайно, порівняння персон може виконуватися по різних критеріях: зросту, віку, зарплаті. В нашому випадку відношення порядку на об'єктах класу `Person` задається як відношення порядку на прізвищах персон. Оскільки рядки успадковують інтерфейс `IComparable`, для прізвищ персон викликається метод `CompareTo`, його результат і повертається як результат методу `CompareTo` для персон.

Введемо тепер в класі Person перевантаження операцій відношення:

```
public static bool operator <(Person p1, Person p2)
{
   return (p1.CompareTo(p2) < 0);
}
public static bool operator >(Person p1, Person p2)
{
   return (p1.CompareTo(p2) > 0);
}
public static bool operator <=(Person p1, Person p2)
{
   return (p1.CompareTo(p2) <= 0);
}
public static bool operator >=(Person p1, Person p2)
{
   return (p1.CompareTo(p2) >=0);
}
}
```

Повний код програми для цього прикладу

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Console1_Lab11
{
    public class Person :IComparable
    {
        public string Name;       //ім'я
        public int Age;           // вік
        public string Role;       // роль
        public string GetName() { return Name; }
        public int GetAge() { return Age; }
        public Person(string N)
        {
            this.Name = N;
        }
        public int CompareTo(object pers)
```

```
    {
        const string s = "Порівнюваний об'єкт не належить до класу Person";
        Person p = pers as Person;
        if (!p.Equals(null))
            return (Name.CompareTo(p.Name));
        throw new ArgumentException(s);
    }
    public static bool operator <(Person p1, Person p2)
    {
        return (p1.CompareTo(p2) < 0);
    }
    public static bool operator >(Person p1, Person p2)
    {
        return (p1.CompareTo(p2) > 0);
    }
    public static bool operator <=(Person p1, Person p2)
    {
        return (p1.CompareTo(p2) <= 0);
    }
    public static bool operator >=(Person p1, Person p2)
    {
        return (p1.CompareTo(p2) >= 0);
    }
}

class Program
{
    static void Main(string[] args)
    {
        Person prep1 = new Person("Тітков");
        Person prep2 = new Person("Коротун");
        Person prep3 = new Person("Шилін");
        Person prep4 = new Person("Крячок");
        Person prep5 = new Person("Пожидаєва");
        Person prep6 = new Person("Проскудіна");
        Console.WriteLine("{0} > {1} = {2}", prep1.Name,
           prep2.Name, (prep1 > prep2));
        Console.WriteLine("{0} >= {1} = {2}", prep3.Name,
           prep4.Name, (prep3 >= prep4));
        Console.WriteLine("{0} != {1} = {2}", prep5.Name,
           prep6.Name, (prep5 != prep6));
        Console.ReadLine();
    }
}
  }
```

В цьому прикладі зроблене досить штучне порівняння об'єктів, яке просто демонструє можливості порівняння об'єктів класу. Зробимо порівняння об'єктів за віком. Змінемо конструктор класу з ініціалізацією віку:

```
 public Person(string N,int Age)
  {
      this.Name = N;
      this.Age = Age;
          }
```

Задамо в методі CompareTo операції порівняння за віком

```
public int CompareTo(object pers)
  {
      const string s = "Порівнюваний об'єкт не належить до класу Person";
      Person p = pers as Person;
      if (!p.Equals(null))
          return (Age.CompareTo(p.Age));
     throw new ArgumentException(s);
          }
```

Можлива і наступна реалізація інтерфейсу:

```
public int CompareTo(object pers)
  {
      Person p = (Person) pers;
      if (this.Age > p.Age) return 1;
      if (this.Age < p.Age) return -1;
      return 0;
          }
```

Для спрощення виводу на консоль створимо віртуальний метод

```
virtual public void Passport()
  {
      Console.WriteLine("Name = {0} Age = {1}",Name, Age);
          }
```

Приклад 2. Внесемо зміни в метод `Main`. Створимо масив персон, елементами якого є створені об'єкти класу `prep1,...prep6`. Так як це масив – ми можемо його відсортувати викликом методу `Sort()`.

Повний код програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Console2_Lab7
{
    public class Person : IComparable
    {
        public string Name;       //имя
        public int Age;           // возраст
        public string Role;       // роль
        public string GetName() { return Name; }
        public int GetAge() { return Age; }
        public Person(string N,int Age)
        {
            this.Name = N;
            this.Age = Age;
        }
        public int CompareTo(object pers)
        {
            Person p = (Person) pers;
            if (this.Age > p.Age) return 1;
            if (this.Age < p.Age) return -1;
            return 0;
        }

         public void Passport()
        {
            Console.WriteLine("Name = {0} Age = {1}",Name, Age);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Person prep1 = new Person("Іванов",48);
            Person prep2 = new Person("Петров",36);
            Person prep3 = new Person("Соколов",30);
```

```
            Person prep4 = new Person("Сидоров",50);
            Person prep5 = new Person("Орлов",50);
            Person prep6 = new Person("Федоров",60);
            Person [] group = new Person[6];
            group[0] = prep1;
            group[1] = prep2;
            group[2] = prep3;
            group[3] = prep4;
            group[4] = prep5;
            group[5] = prep6;
            Array.Sort(group);
            foreach (Person elem in group) elem.Passport();
            Console.ReadLine();
        }
    }
}
```

Інтерфейс `IComparable` дозволяє встановити порівняння об'єктів тільки за одним критерієм. Але у багатьох алгоритмах потрібно виконувати порівняння об'єктів за різними критеріями. У C# для цього використовується інтерфейс **IComparer.**

2. Впорядкування об'єктів за кількома критеріями. Інтерфейс IComparer

Інтерфейс `IComparer` визначений в просторі імен `System.Collections`. Він містить один метод Compare, що повертає результат порівняння двох об'єктів, переданих йому як параметри:

```
interface IComparer
{
    int Compare( object ob1, object ob2 )
}
```

Принцип використання цього інтерфейсу полягає в тому, що для кожного критерію сортування об'єктів описується невеликий допоміжний клас, що реалізує цей інтерфейс. Об'єкт цього класу передається в стандартний метод сортування масиву другим аргументом.

Розглянемо приклад сортування об'єктів класу `Person` за віком і зарплатою. Модифікуємо клас `Person`, додамо в клас поле `Zarplata`, модифікуємо конструктор класу ініціалізацією цього поля.

Створимо два допоміжних класи: `SortByAge`, `SortByZarplata`, які є нащадками інтерфейсу **`IComparer`**. В них реалізовнані методи порівняння за потрібними критеріями (віком і зарплатою).

```
public class SortByAge : IComparer
  {
      //Сортування за віком
      int IComparer.Compare(object ob1, object ob2)
      {
          Person p1 = (Person)ob1;
          Person p2 = (Person)ob2;
          if (p1.Age > p2.Age) return 1;
          if (p1.Age < p2.Age) return -1;
          return 0;
      }
  }
  public class SortByZarplata : IComparer                                    //
  {
      // сортування за зарплатою
      int IComparer.Compare(object ob1, object ob2)
      {
          Person p1 = (Person)ob1;
          Person p2 = (Person)ob2;
          if (p1.Zarplata > p2.Zarplata) return 1;
          if (p1.Zarplata < p2.Zarplata) return -1;
          return 0;
       }
  }
```

Приклад 3. Повний код програми.

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

```
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Console3_Lab7
{
    // Використання інтерфейсу IComparer
    public class Person
    {
        public string Name;        //ім'я
        public int Age;            // вік
        public string Role;        // роль
        public int Zarplata;       // зарплата
        public string GetName() { return Name; }
        public int GetAge() { return Age; }

        public Person(string N,int Age, int Z)
        {
            this.Name = N;
            this.Age = Age;
            this.Zarplata = Z;
        }

         virtual public void Passport()
        {
            Console.WriteLine("Name = {0} Age = {1}  Zarplata = {2}", Name,
Age,Zarplata);
        }

         public class SortByAge : IComparer
         {
             //Сортування за віком
             int IComparer.Compare(object ob1, object ob2)
             {
                 Person p1 = (Person)ob1;
                 Person p2 = (Person)ob2;
                 if (p1.Age > p2.Age) return 1;
                 if (p1.Age < p2.Age) return -1;
                 return 0;
             }
         }

 public class SortByZarplata : IComparer                                    //
         {
             // сортування за зарплатою
             int IComparer.Compare(object ob1, object ob2)
             {
                 Person p1 = (Person)ob1;
                 Person p2 = (Person)ob2;
                 if (p1.Zarplata > p2.Zarplata) return 1;
                 if (p1.Zarplata < p2.Zarplata) return -1;
                 return 0;
              }
         }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Person prep1 = new Person("Іванов",48,1500);
            Person prep2 = new Person("Петренко",36,3500);
            Person prep3 = new Person("Соколовський",30,1000);
            Person prep4 = new Person("Сидор",50,4500);
            Person prep5 = new Person("Орловський",50,6000);
            Person prep6 = new Person("Федорович",60,1300);
            Person [] group = new Person[6];
```

```
            group[0] = prep1;
            group[1] = prep2;
            group[2] = prep3;
            group[3] = prep4;
            group[4] = prep5;
            group[5] = prep6;
            Console.WriteLine("Сортування за віком:");
            Array.Sort(group,new Person.SortByAge());
            foreach (Person elem in group) elem.Passport();
            Console.WriteLine("Сортування за зарплатою:");
            Array.Sort(group,new Person.SortByZarplata());
            foreach (Person elem in group) elem.Passport();
            Console.ReadLine();
        }
    }
}
```

3. Перелічуваність об'єктів і інтерфейси. Інтерфейс IEnumerable

Якщо клас можна розглядати як деякий контейнер об'єктів перечислення, то для того, щоб клас міг повертати ці об'єкти в циклі `foreach`, клас має бути спадкоємцем інтерфейсу **`IEnumerable`** або мати в своєму складі ітератори - методи, що повертають результат типу `IEnumerable`.

Наприклад, робота з масивами за допомогою циклу `foreach` можлива саме тому, що тип `Array` реалізує інтерфейси `IEnumerable` і `IEnumerator`. Можна створювати і власні класи, що підтримують стандартні інтерфейси, що дозволить використовувати об'єкти цих класів стандартними способами.

В інтерфейсу **`IEnumerable`** всього один метод - **`GetEnumerator()`**. В методі немає жодних вхідних аргументів, так що, здається, організувати перелічування об'єктів класу досить просто - написати реалізацію одного методу. Це дійсно просто, але вимагає розуміння процесу перелічуваності. Перша складність полягає в тому, що метод `GetEnumerator` синтаксично визначений таким чином:

```
IEnumerator    GetEnumerator()
```

Це означає, що в результаті виклику методу повинен повертатися інтерфейсний об'єкт, що належить інтерфейсу `IEnumerator` - ще одному інтерфейсу, пов'язаному з перелічуванням. Метод `GetEnumerator` вимагає створення об'єкту нумератора - об'єкту, що реалізує методи інтерфейсу `IEnumerator`.

В інтерфейсі `IEnumerator` є декілька методів, і в сукупності саме вони і дозволяють організувати процес перелічування.

Розглянемо приклад використання інтерфейсу `IEnumerable`.

Додамо в наш проект новий клас Persons, одним з полів якого буде масив об'єктів Person. Організуємо перелічування в цьому класі, що зводиться до перелічування персон масиву.

Приклад 4.

```
/// <summary>
/// Клас - спадкоємець інтерфейсу IEnumerable
/// що допускає перелічування об'єктів.
/// Перелічування  зводиться до перелічування  персон
/// заданих полем  container
/// </summary>
class Persons:IEnumerable
{
protected int size;
protected Person[] container;
Random rnd = new Random();
/// <summary>
/// Конструктор за замовчанням.
/// Створює масив з 10 персон
/// </summary>
public Persons()
{
size = 10;
container = new Person[size];
FillContainer();
}
```

```
        /// <summary>
        /// Конструктор. Створює масив заданої розмірності
        /// Створює його елементи, використовуючи рандомізацію.
        /// </summary>
        /// <param name="size">розмірність масиву</param>
        public Persons(int size)
        {
        this.size = size;
        container = new Person[size];
        FillContainer();
        }
        /// <summary>
        /// Конструктор, якому передається масив персон
        /// </summary>
        /// <param name="container"> масив Person</param>
        public Persons(Person[] container)
        {
        this.container = container;
        size = container.Length;
        }
        /// <summary>
        /// Заповнення масиву person
        /// </summary>
        void FillContainer()
        {
        for(int i = 0; i <size; i++)
        {
            int num = rnd.Next(3*size);
            int age = rnd.Next(27, 46);
            container[i]= new Person("агент_" + num, age);
        }
        }
        }
```

Створений клас `Persons` влаштований просто. У нього є поле container, яке є масивом з елементами класу `Person`. Набір конструкторів класу дозволяє передати класу масив персон або доручити самому класу його формування, використовуючи метод `FillContainer`. Оскільки клас оголошений спадкоємцем інтерфейсу `IEnumerable`, необхідно реалізувати метод `GetEnumerator`, що забезпечить перелічуваність об'єктів класу і дасть можливість використовувати цикл `for each` при роботі з об'єктами класу. В даному випадку реалізувати метод інтерфейсу нескладно, і ось як виглядає його реалізація:

```
    /// <summary>
        /// Реалізація методу інтерфейсу IEnumerable
        /// Зводиться до виклику відповідного методу
        /// для поля container - масиву
        /// для якого цей метод реалізований в бібліотеці FCL
        /// </summary>
        /// <returns>перечислитель - інтерфейсний об'єкт</returns>
        public IEnumerator GetEnumerator()
        {
        return container.GetEnumerator();
        }
```

Повний код програми до прикладу 4.

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.Linq;
using System.Text;

namespace Console4_Lab11
{
    public class Person
    {
        public string Name;       //ім'я
```

```
    public int Age;           // вік
    public string Role;       // роль
    public string GetName() { return Name; }
    public int GetAge() { return Age; }
    public Person(string N, int A)
    {
        this.Name = N;
        this.Age = A;
    }
}

class Persons:IEnumerable
{
    protected int size;
    protected Person[] container;
    Random rnd = new Random();
    /// <summary>
    /// Конструктор за замовчанням.
    /// Створює масив з 10 персон
    /// </summary>
    public Persons()
    {
        size = 10;
        container = new Person[size];
        FillContainer();
    }

    /// <summary>
    /// Конструктор. Створює масив заданої розмірності
    /// Створює його елементи, використовуючи рандомізацію.
    /// </summary>
    /// <param name="size">розмірність масиву</param>
    public Persons(int size)
    {
        this.size = size;
        container = new Person[size];
        FillContainer();
    }
    /// <summary>
    /// Конструктор, якому передається масив персон
    /// </summary>
    /// <param name="container"> масив Person</param>
    public Persons(Person[] container)
    {
        this.container = container;
        size = container.Length;
    }

    /// <summary>
    /// Заповнення масиву person
    /// </summary>
    ///

   void FillContainer()
    {
        for (int i = 0; i < size; i++)
        {
            int num = rnd.Next(3 * size);
            int age = rnd.Next(27, 46);
            container[i] = new Person("агент_" + num, age);
        }
    }
    public IEnumerator GetEnumerator()
    {
        return container.GetEnumerator();
    }
```

```
    }
```

У тесті створюється об'єкт класу `Persons` і в циклі `foreach` об'єкти перебираються, повертаючи кожного разу черговий об'єкт класу `Person`. Метод `ToString`, визначений в класі `Person`, дозволяє виводити інформацію про об'єкти.

```
class Program
{
   static void Main(string[] args)
   {
      int size = 5;
      Persons agents = new Persons(size);
      foreach (Person agent in agents)
      {
         Console.WriteLine("Прізвище:  " + agent.Name.ToString() + "    Вік:  " + agent.Age.ToString());
      }
      Console.ReadLine();
   }
}
```

4. Ітератори і інтерфейс IEnumerable

Оператор `foreach` є зручним засобом перебору елементів об'єкту. Масиви і всі стандартні колекції бібліотеки .NET дозволяють виконувати такий перебір завдяки тому, що в них реалізовані інтерфейси `IEnumerable` і `IEnumerator`. Для застосування оператора `foreach` до типу даних користувача потрібно реалізувати в ньому ці інтерфейси.

Інтерфейс `IEnumerable` визначає всього один метод — `GetEnumerator`, що повертає об'єкт типу `IEnumerator` (нумератор), який можна використовувати для перебору елементів об'єкту.

Інтерфейс `IEnumerator` задає три елементи:

- властивість `Current`, що повертає поточний елемент об'єкту;
- метод `MoveNext`, що пересуває нумератор на наступний елемент об'єкту;
- метод `Reset`, що встановлює нумератор в початок перегляду.

Цикл `foreach` використовує ці методи для перебору елементів, з яких складається об'єкт. Таким чином, якщо потрібно, щоб для перебору елементів класу міг застосовуватися цикл `foreach`, необхідно реалізувати чотири методи: `GetEnumerator`, `Current`, `MoveNext` і `Reset`. Це не цікава робота, а виконувати її доводиться часто, тому починаючи з версії 2.0 були введені засоби, що полегшують виконання перебору в об'єкті, — ітератори.

Ітератор – це блок коду, в якому задається послідовність перебору елементів об'єкту. На кожному проході циклу foreach виконується один крок ітератора, що закінчується видачею чергового значення. Видача значення виконується за допомогою оператора **yield**, який дозволяє заповнювати контейнер елементами. Його синтаксис:

```
yield return <вираз>;
```

Кожне виконання оператора **yield** додає новий елемент в контейнер. Розглянемо простий приклад використання ітератора, який створює колекцію кольорів.

```
public System.Collections.IEnumerable Rainbow()
{
    yield return "red";
    yield return "orange";
    yield return "yellow";
    yield return "green";
    yield return "blue";
    yield return "violet";
}
```

Клієнти цього класу можуть працювати з цією колекцією, наприклад так:

```
string colors = "";
        foreach(string s in tst.Rainbow())
            colors += s + "-";
```

В цьому прикладі `tst` – об'єкт класу `Testing`, а змінна s в циклі `foreach` набуде значень всіх кольорів, розміщених в контейнері за допомогою оператора `yield`. Слід зазначити, що реально жодні контейнери не створюються, а цикл `foreach` на кожному кроці викликає ітератор і створює новий елемент. Саме тому цикл `foreach` працює лише на читання елементів і не працює на запис.

Розглянемо ще один приклад створення ітератора для класу `Person`. Хай потрібно створити об'єкт, що містить масив екземплярів класу `Person`. Для підтримки перебору достатньо вказати, що клас `Persons` реалізує інтерфейс `IEnumerable` (оператор 1), і описати ітератор (оператор 2). Доступ до нього може бути здійснений через методи `MoveNext` і `Current` інтерфейсу `IEnumerator`.

Приклад 5.

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.Linq;
using System.Text;

namespace Console5_Lab7
{
    public class Person
    {
        public string Name;        //ім'я
        public int Age;            // вік
        public string Role;        // роль
        public string GetName() { return Name; }
        public int GetAge() { return Age; }
        public int Zarplata;       // зарплата
        public Person(string N, int A, int Z)
        {
            this.Name = N;
            this.Age = A;
            this.Zarplata = Z;
        }

        virtual public void Passport()
        {
            Console.WriteLine("Name = {0} Age = {1}  Zarplata = {2}", Name, Age,
Zarplata);
        }
    }

    class Persons:IEnumerable       //1
    {
        private Person[] mas;
        private int n;

        public Persons()
        {
           mas = new Person[10];
           n = 0;
        }
        public IEnumerator GetEnumerator()
        {
            for ( int i = 0; i < n; ++i ) yield return mas[i];      //2
        }
        public void Add( Person m )
        {
            if ( n >= 10 ) return;
            mas[n] = m;
            ++n;
        }
    }
    class Program
    {
        static void Main()
        {
            Persons pers = new Persons();
            pers.Add(new Person("Іванчак", 48, 1500));
            pers.Add(new Person("Петровський",36,3500));
```

```
            pers.Add(new Person("Соколовський", 30,1000));
            foreach (Person x in pers) x.Passport();
            Console.ReadKey();
        }
    }
}
```

Блок ітератора синтаксично є звичайним блоком і може зустрічатися в тілі методу, операції або частині `get` властивості, якщо відповідне значення, яке повертається має тип `IEnumerable` або `IEnumerator`.

У тілі блоку ітератора можуть зустрічатися дві конструкції:

- **`yield return`** формує значення, що видається на черговій ітерації;
- **`yield break`** сигналізує про завершення ітерації.

Ключове слово **`yield`** має спеціальне значення для компілятора лише в цих конструкціях.

Код блоку ітератора виконується не так, як звичайні блоки. Компілятор формує службовий об'єкт-нумератор, при виклику методу **`MoveNext`** якого виконується код блоку ітератора, що видає чергове значення за допомогою ключового слова `yield`. Наступний виклик методу `MoveNext` об'єкту-нумератора відновлює виконання блоку ітератора з моменту, на якому він був призупинений в попередній раз.

Індивідуальні завдання:

(Для виконання індивідуальних завдань № варінта є порядковим номером прізвища студента в списку групи. Усі проекти завантажити у власні репозиторії на Git Hub та виконати їх збірку/build у середовищі Travis CI. Посилання на репозиторії проектів вказати у звіті та обов'язково долучати скріни успішної збірки в Travis CI.)

1. Розробити консольний застосунок згідно свого варіанту. Клас повинен містити не менше 5 полів. Набір та типи полів придумайте самі. Передбачити обробку виключень. Масив об'єктів класу повинен містити не менше 10 записів.

(2 бала)

№ варіанту	Зміст завдання
1	А). Створіть масив об'єктів класу **Тварина**. Реалізуйте інтерфейс *IComparable* для порівняння тварин за вагою в методі *CompareTo()*. Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння тварин не тільки за вагою, але і за зростом. В). Реалізуйте інтерфейс *IEnumerable. В*иведіть на консоль список тварин, впорядкований за вагою.
2	А). Створіть масив об'єктів класу **Робочий**. Реалізуйте інтерфейс *IComparable* для порівняння робочих за віком в методі *CompareTo()*. Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння робочих не тільки за віком, але і за зарплатою. В). Реалізуйте інтерфейс *IEnumerable. В*иведіть на консоль список робочих, впорядкований за зарплатою.
3	А). Створіть масив об'єктів класу **Співробітник**. Реалізуйте інтерфейс *IComparable* для порівняння співробітників за віком в методі *CompareTo()*. Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння співробітників не тільки за віком, але і за стажем роботи на цьому підприємстві. В). Реалізуйте інтерфейс *IEnumerable. В*иведіть на консоль список співробітників, впорядкований за стажем роботи.
4	А). Створіть масив об'єктів класу **Виріб**. Реалізуйте інтерфейс *IComparable* для порівняння виробів за вагою в методі *CompareTo()*. Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння виробів за

№ варіанту	Зміст завдання
	ціною і за якістю (створити свою шкалу якості). В). Реалізуйте інтерфейс *IEnumerable. В*иведіть на консоль список виробів, впорядкований за ціною.
5	А). Створіть масив об'єктів класу **Організація**. Реалізуйте інтерфейс *IComparable* для порівняння організацій за кількістю співробітників в методі *CompareTo().* Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння організацій за кількістю співробітників і за рейтингом успішності (створити свою шкалу). В). Реалізуйте інтерфейс *IEnumerable.* Виведіть на консоль список організацій , впорядкований за рейтингом.
6	А). Створіть масив об'єктів класу **Журнал**. Реалізуйте інтерфейс *IComparable* для порівняння журналів за ціною в методі *CompareTo().* Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння журналів за кількістю сторінок і за рейтингом продажів (створити свою шкалу). В). Реалізуйте інтерфейс *IEnumerable.* Виведіть на консоль список журналів, впорядкований за рейтингом продажів.
7	А). Створіть масив об'єктів класу **Дерево**. Реалізуйте інтерфейс *IComparable* для порівняння порід дерев за ціною в методі *CompareTo().* Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння дерев за висотою і за ціною. В). Реалізуйте інтерфейс *IEnumerable.* Виведіть на консоль список порід дерев, впорядкований за ціною.
8	А). Створіть масив об'єктів класу **Місто**. Реалізуйте інтерфейс *IComparable* для порівняння міст за територією в методі *CompareTo().* Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння міст за територією і за кількістю населення. В). Реалізуйте інтерфейс *IEnumerable.* Виведіть на консоль список міст, впорядкований за кількістю населення.
9	А). Створіть масив об'єктів класу **Товар**. Реалізуйте інтерфейс *IComparable* для порівняння товарів за ціною в методі *CompareTo().* Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння товарів за ціною і за розмірами. В). Реалізуйте інтерфейс *IEnumerable.* Виведіть на консоль список товарів, впорядкований за ціною.
10	А). Створіть масив об'єктів класу **Документ**. Реалізуйте інтерфейс *IComparable* для порівняння документів за кількістю сторінок в методі *CompareTo().* Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння документів за кількістю сторінок і за датою створення. В). Реалізуйте інтерфейс *IEnumerable.* Виведіть на консоль список документів, впорядкований за кількістю сторінок.
11	А). Створіть масив об'єктів класу **Автомобіль**. Реалізуйте інтерфейс *IComparable* для порівняння автомобілів за ціною в методі *CompareTo().* Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння автомобілів за ціною і за потужністю двигуна. В). Реалізуйте інтерфейс *IEnumerable.* Виведіть на консоль список автомобілів, впорядкований за ціною і потужністю двигуна.
12	А). Створіть масив об'єктів класу **Викладач**. Реалізуйте інтерфейс *IComparable* для порівняння викладачів за зарплатою в методі *CompareTo().*

№ варіанту	Зміст завдання
	Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння викладачів за зарплатою і за стажем роботи. В). Реалізуйте інтерфейс *IEnumerable*. Виведіть на консоль список викладачів, впорядкований за зарплатою і за стажем роботи.
13	А). Створіть масив об'єктів класу **Овочі**. Реалізуйте інтерфейс *IComparable* для порівняння овочів за ціною в методі *CompareTo()*. Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння овочів за ціною і за кількістю ящиків на складі. В). Реалізуйте інтерфейс *IEnumerable*. Виведіть на консоль список овочів, впорядкований за ціною.
14	А). Створіть масив об'єктів класу **Книга**. Реалізуйте інтерфейс *IComparable* для порівняння книг за ціною в методі *CompareTo()*. Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння книг за ціною і за кількістю сторінок. В). Реалізуйте інтерфейс *IEnumerable*. Виведіть на консоль список книг, впорядкований за ціною.
15	А). Створіть масив об'єктів класу **Меблі**. Реалізуйте інтерфейс *IComparable* для порівняння **Меблів** за ціною в методі *CompareTo()*. Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння меблів за ціною і за габаритами. В). Реалізуйте інтерфейс *IEnumerable*. Виведіть на консоль список меблів, впорядкований за ціною.
16	А). Створіть масив об'єктів класу **Будинок**. Реалізуйте інтерфейс *IComparable* для порівняння будинків за ціною в методі *CompareTo()*. Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння будинків за ціною і за площею. В). Реалізуйте інтерфейс *IEnumerable*. Виведіть на консоль список будинків, впорядкований спочатку за ціною, потім за площею.
17	А). Створіть масив об'єктів класу **Літак**. Реалізуйте інтерфейс *IComparable* для порівняння літаків за кількістю годин нальоту в методі *CompareTo()*. Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння літаків за кількістю годин нальоту і за кількістю пасажирських місць. В). Реалізуйте інтерфейс *IEnumerable*. Виведіть на консоль список літаків, впорядкований спочатку за кількістю годин нальоту, потім за кількістю пасажирських місць.
18	А). Створіть масив об'єктів класу **Операційна система (ОС)**. Реалізуйте інтерфейс *IComparable* для порівняння ОС за ціною в методі *CompareTo()*. Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння ОС за ціною і за роками виходу у продаж. В). Реалізуйте інтерфейс *IEnumerable*. Виведіть на консоль список ОС, впорядкований спочатку за ціною, потім за роками виходу у продаж.
19	А). Створіть масив об'єктів класу **Комп'ютер**. Реалізуйте інтерфейс *IComparable* для порівняння Комп'ютерів за ціною в методі *CompareTo()*. Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння комп'ютерів за ціною і за розмірами жорсткого диска. В). Реалізуйте інтерфейс *IEnumerable*. Виведіть на консоль список комп'ютерів, впорядкований спочатку за ціною, потім за розмірами жорсткого диска.

№ варіанту	Зміст завдання
20	А). Створіть масив об'єктів класу **Збірка**. Реалізуйте інтерфейс *IComparable* для порівняння збірок за ціною в методі *CompareTo().* Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння збірок за ціною і за кількістю сторінок. В). Реалізуйте інтерфейс *IEnumerable.* Виведіть на консоль список збірок, впорядкований за ціною.
21	А). Створіть масив об'єктів класу **Квартира**. Реалізуйте інтерфейс *IComparable* для порівняння квартир за ціною в методі *CompareTo().* Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння квартирза ціною і за кількістю кімнат. В). Реалізуйте інтерфейс *IEnumerable.* Виведіть на консоль список квартир, впорядкований спочатку за ціною, потім за площею.
22	А). Створіть масив об'єктів класу **Птахоферма**. Реалізуйте інтерфейс *IComparable* для порівняння порід птахів за вагою в методі *CompareTo().* Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння тварин не тільки за вагою, але і за ціною їх м'яса за 1 кг. В). Реалізуйте інтерфейс *IEnumerable. В*иведіть на консоль список тварин, впорядкований за вагою.
23	А). Створіть масив об'єктів класу **Країна**. Реалізуйте інтерфейс *IComparable* для порівняння країн за площею території в методі *CompareTo().* Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння країн за площею і за кількістю населення. В). Реалізуйте інтерфейс *IEnumerable.* Виведіть на консоль список країн, впорядкований за кількістю населення.
24	А). Створіть масив об'єктів класу **Кущі**. Реалізуйте інтерфейс *IComparable* для порівняння порід фруктових кущів за ціною в методі *CompareTo().* Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння кущів за середньою врожайністю з 1 куща даного виду і за ціною. В). Реалізуйте інтерфейс *IEnumerable.* Виведіть на консоль список порід саджанців фруктових кущів, впорядкований за ціною.
25	А). Створіть масив об'єктів класу **Гаджет**. Реалізуйте інтерфейс *IComparable* для порівняння смартфонів за ціною в методі *CompareTo().* Б). Реалізуйте в класі інтерфейс *IComparer* для порівняння смартфонів за ціною і за обсягом пам'яті. В). Реалізуйте інтерфейс *IEnumerable.* Виведіть на консоль список гаджетів, впорядкований спочатку за ціною, потім за обсягом пам'яті.

2. Розробити консольний застосунок для роботи з базою даних, що зберігається у текстовому файлі (початковий масив не менше 10 записів). Структура бази даних описується класом згідно вашого варіанта. Для ідентифікації спроби введення з клавіатури некоректних даних описати виключення. Проект повинен містити засоби для додавання, редагування, знищення записів, інформації з файла на екран та запису змін до файла. Меню програми реалізувати по натисненню на певні клавіші: наприклад, Enter – вихід, n - пошук, p – редагування тощо. Для реалізації сортування використати вбудовані інтерфейси.

(3 бала)

№ варіанта	Клас	Поля	Сортування за 1 параметром	Сортування за 2 параметрами
1.	Користувачі локальної мережі коледжу	Прізвище, група, курс, логін, пароль, дата реєстрації, номер студентського квитка.	Номер студентського квитка	Курс, дата реєстрації
2.	Камера схову	Прізвище, дата здачі, термін зберігання, інвентарний номер та назва предмета	Інвентарний номер	Дата здачі, термін зберігання
3.	Склад товарів	Інвентарний номер, назва товару, вага, ціна, кількість	Вага	Ціна, кількість
4.	Каса продажу квитків	Назва пункту, час відправлення, дата відправлення, час прибуття, дата прибуття, ціна квитка	Час відправлення	Дата прибуття, ціна
5.	Успішність студентів	Прізвище, номер групи, номер залікової книжки, оцінки з трьох предметів	Номер залікової книжки	Номер групи, оцінки з одного предмету (оберіть самі на вибір)
6	Асортимент виробів	Номер виробу, назва, вага, собівартість виробництва, ціна продажу	Ціна продажу	Вага, собівартість виробництва
7	Метеорологічні дані	Дата, місто, атмосферний тиск, температура повітря, швидкість вітру	Дата	Температура повітря, швидкість вітру
8	Склад автозапчастин	Код товару, назва, марка авто, ціна, кількість	Ціна	Ціна, кількість
9	Пацієнти сімейного лікаря	Номер карти, Прізвище, ім’я, по-батькові, рік народження, стать, кількість дітей	Рік народження	Рік народження, кількість дітей
10	Викладачі університету	Номер посвідчення, Прізвище, ім’я, по-батькові, кафедра, дата народження, науковий ступінь, стаж	Дата народження	Стаж, дата народження
11	Власники квартир ОСББ	Прізвище, номер будинку, номер квартири, стать, сума боргу	Сума боргу	Номер будинку, сума боргу
12	Плейлист	Назва пісні, виконавець, альбом, рік випуску, тривалість пісні	Рік випуску	Рік випуску, тривалість пісні
13	Колекція фільмів	Назва, прізвище режисера, рік випуску, кіностудія, тривалість фільму, бюджет	Рік випуску	Тривалість фільму, бюджет

№ варіанта	Клас	Поля	Сортування за 1 параметром	Сортування за 2 параметрами
14	ЖРЕПи міста	Назва, адреса, прізвище начальника, кількість підзвітних будинків, кількість працівників, заборгованість мешканців підзвітних будинків	Заборгованість мешканців підзвітних будинків	Кількість підзвітних будинків, кількість працівників
15	Абоненти кабельної мережі	Номер договору, Прізвище, ім’я, адреса, борг, сума абонплати	Номер договору	Борг, сума абонплати
16.	Бібліотека	Інвентарний номер, автор, назва, кількість сторінок, рік видання	Рік видання	Інвентарний номер, кількість сторінок
17.	Асортимент мобільних телефонів	Модель, виробник, рік випуску, діагональ екрана, обсяг пам’яті, ціна.	Ціна	Обсяг пам’яті, рік випуску
18.	Розклад руху літаків	Номер рейсу, тип літака, напрямок руху, кількість місць, кількість вільних місць.	Номер рейсу	Кількість місць, кількість вільних місць
19.	Колекція компакт-дисків	Інвентарний номер, назва альбому, об'єм диску, тип, дата запису.	Дата запису	Інвентарний номер, дата запису
20.	Записна книжка	Прізвище, ім'я, по-батькові, дата народження, телефон, електронна пошта.	Дата народження	Дата народження, телефон
21.	Бібліотека школи	Інвентарний номер, назва, клас, рік видання, кількість сторінок	Інвентарний номер	Рік видання, кількість сторінок
22.	Розклад пар	Номер пари, дата, предмет, прізвище викладача, форма заняття.	Дата	День тижня, номер пари
23.	Список файлів	ім'я файла, розширення, розмір, дата створення, атрибути.	Розмір	Дата створення, розмір
24.	Архів програмного забезпечення	Назва програми, операційна система, розмір дистрибутиву програми, дата запису, ціна	Дата запису	Ціна, розмір файла
25.	Рахунки банку	Номер рахунку, прізвище, ім'я, дата останньої операції, сума вкладу	Сума вкладу	Дата операції, номер рахунку