

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп'ютерних систем та інфокомунікацій

Кафедра комп'ютерних систем, мереж і кібербезпеки (503)

Лабораторна робота № 4

Наслідування та поліморфізм. Інтерфейси.

з дисципліни

Технології програмування

XAI.503.525a.03O.123-Комп'ютерна інженерія, ПЗ №9629619

Виконав студент гр.	525a	Литвиненко А.В.
11.11.2022	(№	(П.І.Б.)
	групи)	

(підпис, дата)

Перевірив	канд. техн. наук, доцент
-----------	--------------------------

Є. В. Бабешенко

(підпис, дата)

(П.І.Б.)

Харків – 2022

Тема роботи: Наслідування та поліморфізм. Інтерфейси.

Мета роботи: Вивчити парадигми «наслідування» та «поліморфізм». Освоїти можливості мови С#, призначені для реалізації даних парадигм. Вивчити призначення інтерфейсів. Використати на практиці upcast та downcast.

Варіант 5

Задача 1

Частина 1. Постановка завдання

Умова:

Завдання 1

Реалізувати у основному класі з лабораторної роботи №3 інтерфейс IComparable. Продемонструвати можливість сортування масиву екземплярів таких класів за допомогою методу Array.Sort.

Реалізувати для основного класу з лабораторної роботи №3 не менше трьох helper-класів, які реалізують інтерфейс IComparer і призначені для сортування за різними полями (або комбінаціями полів). Продемонструвати можливість сортування екземплярів класів, використовуючи метод Array.Sort та відповідний helper-клас.

Умова з додатка:

5. Населений пункт, місто, селище міського типу

Частина 2. Схема класу

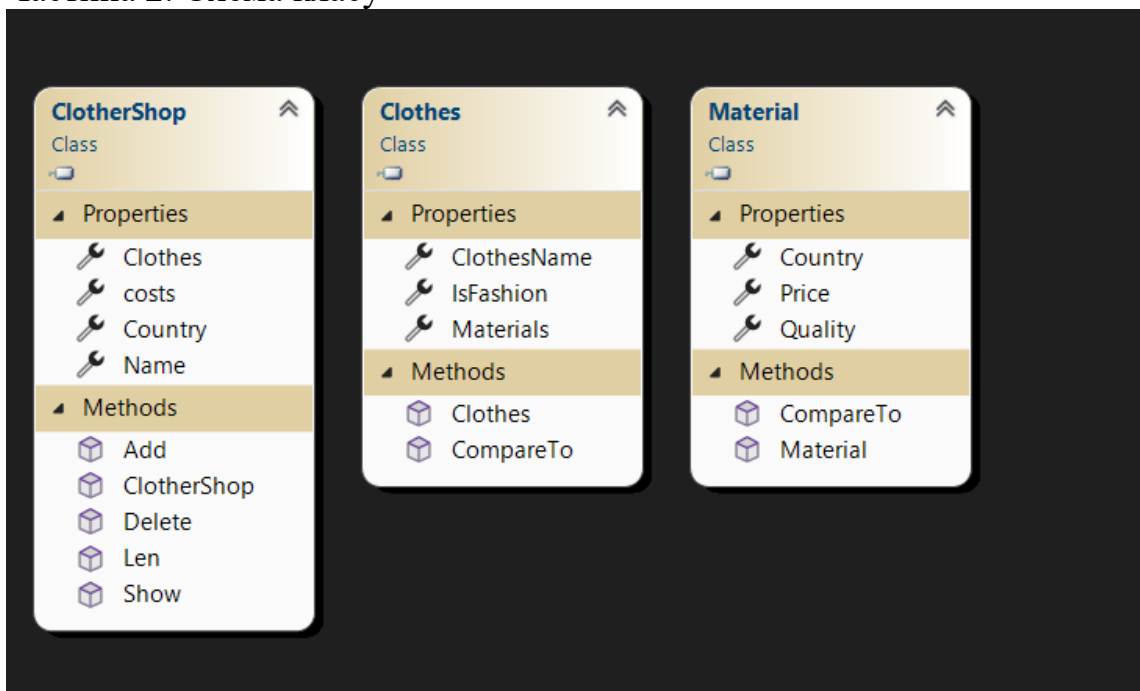


Рисунок 1 - Схема

Частина 3. Текст програми

Відповідно до розробленого алгоритму в середовищі Microsoft Visual Studio була написана програма, яка наведена нижче.

Program.cs

```
using Task02;

namespace Task02_App
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Material Cotton = new Material(50, 100, "Ukraine");
            Material Silk = new Material(90, 50, "Poland");
            Material Leather = new Material(100, 100, "USA");

            Material[] forCoat = new Material[3];
            forCoat[0] = Cotton;
            forCoat[1] = Leather;
            forCoat[2] = Silk;

            Material[] forJeens = new Material[2];
            forJeens[0] = Silk;
            forJeens[1] = Cotton;

            Clothes Coat = new Clothes(forCoat, "Premium Coat", true);
            Clothes Jeans = new Clothes(forJeens, "Cool Jeans", false);
            Clothes Pants = new Clothes(forJeens, "Pants from hell", true);
            Clothes[] goods = new Clothes[3];
            goods[0] = Coat;
            goods[1] = Jeans;
            goods[2] = Pants;

            Console.WriteLine("Unsorted array: ");
            foreach (Material item in Jeans.Materials)
            {
                Console.WriteLine(item.Price);
            }
            Console.WriteLine();
            Array.Sort(Jeans.Materials);
            Console.WriteLine("Sorted array: ");
            foreach (Material item in Jeans.Materials)
            {
                Console.WriteLine(item.Price);
            }
            Console.WriteLine();

            Console.WriteLine("Unsorted array: ");
            foreach (Clothes item in goods)
            {
                Console.WriteLine(item.ClothesName);
            }
            Console.WriteLine();

            Array.Sort(goods);

            Console.WriteLine("Sorted array: ");
            foreach (Clothes item in goods)
```

```

        {
            Console.WriteLine(item.ClothesName);
        }
        Console.WriteLine();
    }
}

```

Class1.cs

```

using System.Data;

namespace Task02
{
    public class Clothes : IComparable<Clothes>
    {
        public Clothes(Material[] materials, string clothesName, bool isFashion)
        {
            Materials = materials;
            ClothesName = clothesName;
            IsFashion = isFashion;
        }

        public Material[] Materials { get; set; }
        public string ClothesName { get; set; }
        public bool IsFashion { get; set; }

        public int CompareTo(Clothes other)
        {
            return ClothesName.CompareTo(other.ClothesName);
        }
    }

    public class Material : IComparable<Material>
    {
        public Material(int price, int quality, string country)
        {
            Price = price;
            Quality = quality;
            Country = country;
        }

        public int Price { get; set; }
        public int Quality { get; set; }
        public string Country { get; set; }

        public int CompareTo(Material other)
        {
            return Price.CompareTo(other.Price);
        }
    }

    public class ClothShop
    {
        public ClothShop(Clothes[] clothes, string name, string country, int costs)
        {
            Clothes = clothes;
            Name = name;
            Country = country;
            this.costs = costs;
        }

        public int Len()
    }
}

```

```

{
    int length = 0;
    for (int i = 0; i < Clothes.Length; i++)
    {
        if (Clothes[i] == null)
        {
            break;
        }
        length++;
    }
    return length;
}

public void Add(Clothes obj)
{
    Clothes[Len()] = obj;
}

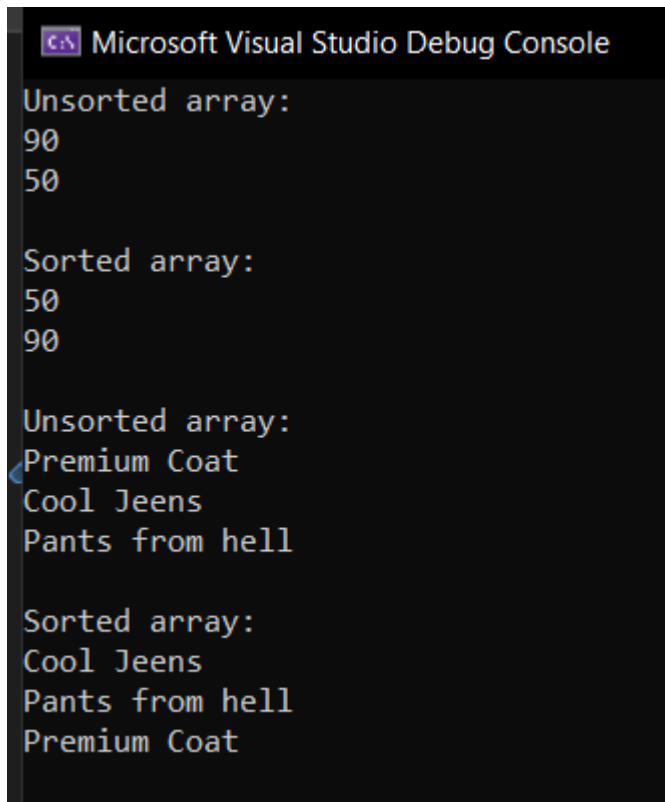
public void Delete()
{
    Clothes[Len()-1] = null;
}

public void Show()
{
    // define length
    for(int i = 0; i < Clothes.Length; ++i)
    {
        if (Clothes[i] == null)
        {
            break;
        }
        Console.Write(Clothes[i].ClothesName + "\n");
    }
    Console.WriteLine();
}

public Clothes[] Clothes { get; set; }
public string Name { get; set; }
public string Country { get; set; }
public int costs { get; set; }
}
}

```

Скриншот тестування:



```
Microsoft Visual Studio Debug Console

Unsorted array:
90
50

Sorted array:
50
90

Unsorted array:
Premium Coat
Cool Jeans
Pants from hell

Sorted array:
Cool Jeans
Pants from hell
Premium Coat
```

The screenshot displays the Microsoft Visual Studio Debug Console with a dark background and light-colored text. It shows two test cases for an array sorting function. The first test case uses numerical values: the unsorted array contains 90 and 50, while the sorted array contains 50 and 90. The second test case uses string values: the unsorted array contains 'Premium Coat', 'Cool Jeans', and 'Pants from hell', while the sorted array contains 'Cool Jeans', 'Pants from hell', and 'Premium Coat'. The strings are sorted alphabetically.

Задача 2.1/2.2

Частина 1. Постановка завдання

Умова:

Завдання 2.1

Відповідно до індивідуального варіанта завдання (додаток А) створіть бібліотеку класів, в якій побудуйте ієрархію класів, враховуючи такі вимоги:

- базовий клас, що знаходиться на вершині ієрархії, має бути реалізований у вигляді абстрактного класу;
- у кожного класу має бути конструктор за замовчуванням та кілька конструкторів з параметрами;
- конструктори з параметрами повинні перевіряти коректність переданих параметрів та генерувати виняток, якщо параметри некоректні;
- конструктори похідних класів мають викликати конструктори базових;
- у кожного класу має бути реалізований метод, який повертає максимум інформації про нього (значення всіх полів) у вигляді рядка; даний метод має бути визначений як `virtual` та перевизначений у похідних класах;
- усі поля повинні мати специфікатор доступу `private` або `protected`; якщо потрібна можливість зміни/завдання значення, необхідно реалізувати властивості з специфікатором доступу `public`;
- у властивостях необхідно перевіряти значення, що присвоюються їм, при спробі присвоїти некоректне значення необхідно згенерувати виняток.

Завдання 2.2

Створіть масив типу «базовий клас», помістіть до масиву екземпляри всіх похідних класів ієрархії, використовуючи `upcast`. Дані для створення екземплярів запитайте у користувача. У циклі продемонструйте можливість виклику всіх методів/звернення до властивостей кожного елемента цього масиву, використовуючи `downcast`. Передбачте можливість оброблення всіх можливих винятків (як генерованих у бібліотеці класів, так і стандартних).

Частина 2. Схема класу

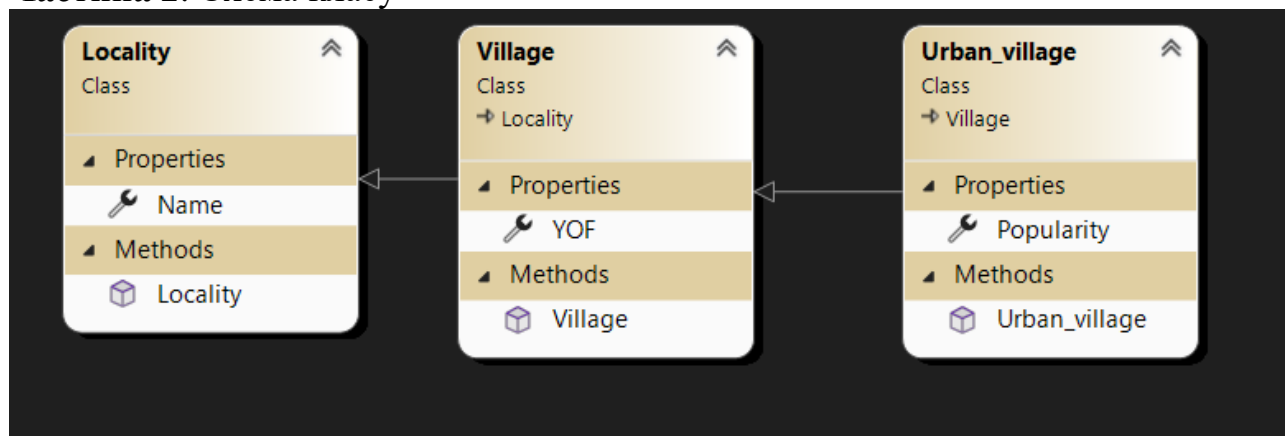


Рисунок 2

Частина 4. Текст програми

Program.cs

```
namespace Task_2_1
{
    internal class Task_2_1
    {
        static void Main(string[] args)
        {
            // input data

            string name;
            int YOF;
            int money;
            int popularity;

            Console.WriteLine("Enter name: ");
            name = Console.ReadLine();

            Locality Markivka = new Locality(name);

            Console.WriteLine("Enter name: ");
            name = Console.ReadLine();

            Console.WriteLine("Enter year of foundation: ");
            YOF = int.Parse(Console.ReadLine());

            Village Bilokyrakina = new Village(name, YOF);

            Console.WriteLine("Enter name: ");
            name = Console.ReadLine();

            Console.WriteLine("Enter year of foundation: ");
            YOF = int.Parse(Console.ReadLine());

            Console.WriteLine("Enter popularity: ");
            popularity = int.Parse(Console.ReadLine());

            Urban_village Bilovodsk = new Urban_village(name, YOF, popularity);

            Locality[] cities = { Markivka, Bilokyrakina, Bilovodsk };

            Village tmpV;
            Urban_village tmpUV;

            Console.WriteLine("\n\n");
            for(int i = 0; i < 3; ++i)
            {
                if(i == 0)
                {
                    Console.WriteLine("<LOCATION> Name: " + cities[i].Name);
                }
                else if (i == 1)
                {
                    tmpV = (Village)cities[i];
                    Console.WriteLine("<VILLAGE> Name: " + tmpV.Name + "\nYear of
foundation: " + tmpV.YOF);
                }
            }
        }
    }
}
```



```

    }
    else if(i == 2)
    {
        tmpUV = (Urban_village)cities[i];
        Console.WriteLine("<URBAN VILLAGE> Name: " + tmpUV.Name + "\nYear
of foundation: " + tmpUV.YOF + "\nPopularity: " + tmpUV.Popularity);
    }
}
}
}
}

```

Class1.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Task_2_1
{
    class Locality
    {
        public Locality(string name)
        {
            Name = name;
        }

        public string Name { get; set; }
    }

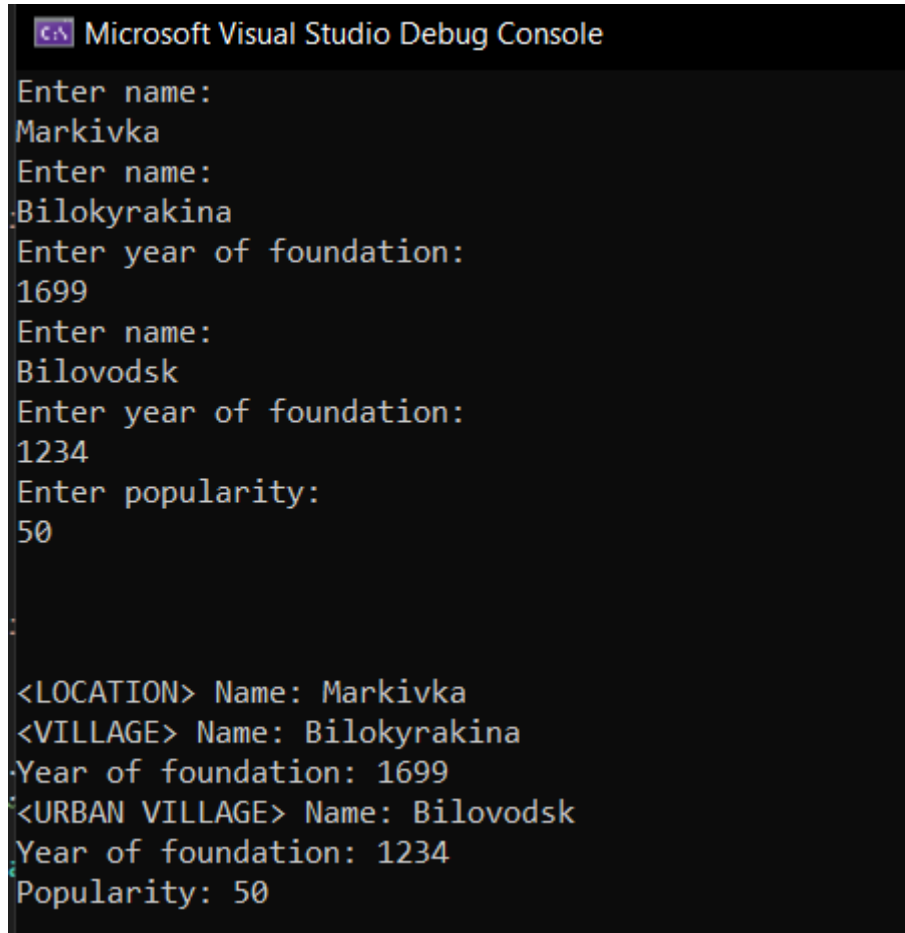
    class Village : Locality
    {
        public Village(string name, int year_of_foundation) :base(name){
            if (year_of_foundation > 0 && year_of_foundation < 2023)
            {
                YOF = year_of_foundation;
            }
            else
            {
                throw new ArgumentException();
            }
        }
        public int YOF { get; set; }
    }

    class Urban_village : Village
    {
        public Urban_village(string name, int year_of_foundation, int popularity) :
base(name, year_of_foundation)
        {
            if(popularity >= -100 && popularity <= 100)
            {
                Popularity = popularity;
            }
            else
            {
                throw new ArgumentException();
            }
        }
    }
}

```

```
        public int Popularity { get; set; }  
    }  
}
```

Скриншот тестування:



```
Microsoft Visual Studio Debug Console

Enter name:
Markivka
Enter name:
Bilokyrakina
Enter year of foundation:
1699
Enter name:
Bilovodsk
Enter year of foundation:
1234
Enter popularity:
50

:

<LOCATION> Name: Markivka
<VILLAGE> Name: Bilokyrakina
Year of foundation: 1699
<URBAN VILLAGE> Name: Bilovodsk
Year of foundation: 1234
Popularity: 50
```

Рисунок 6 – скришот тестування

Висновки

Під час цієї лабораторної роботи я вивчив такі парадигми як наслідування та поліморфізм. Освоїв нові можливості мови C# для реалізації даних парадигм, вивчив призначення інтерфейсів та вперше використав у своєму коді на практиці upcast та downcast.