МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
Національний технічний університет України
Факультет інформатики та обчислювальної техніки
«Київський політехнічний інститут імені Ігоря Сікорського»
Кафедра інформаційних систем та технологій

Звіт
з лабораторної роботи № 2
«Поліморфізм. Наслідування. Перегрузка методів»
з дисципліни
«Програмування - 2. Структури даних та алгоритми»

Варіант № 23

Перевірив:
доц. Корнага Ярослав Ігорович

Виконала: Павлова Софія
Студентка гр. ІС-12 , ФІОТ
1 курс,
залікова книжка № ІС-1224

Київ 2022

Лабораторна робота № 2

Тема: Поліморфізм. Наслідування. Перегрузка методів.

Мета: Вивчити приниципи ООП, перегрузку методів.

Обладнання: Персональні комп’ютери.

ЗАВДАННЯ

Завдання 1

Створити додаток, який задовольняє вимогам, наведеним в завданні. Наслідування застосовувати тільки в тих завданнях, в яких воно **логічно обґрунтоване.** Аргументувати належність класу кожного створюваного методу і коректно перевизначити для кожного класу методи *Equals*, *GetHashCode*, *ToString*. При виклику будь-которого методу класу, виводити на екран текстове повідомлення.

Варіант 23:

23. Створити об'єкт класу Одяг, використовуючи клас Куртка, Сорочка, Штани, Взуття. Методи: одягатись, вибирати різні види одягу.

Код:

Program.cs

```
using proga_2_lab_2_1.Models;
using System;

namespace proga_2_lab_2_1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.OutputEncoding = System.Text.Encoding.Default;

            Person[] persons =
            {
                new Person ("Стів Джобс"),
                new Person ("Анджеліна Джолі"),
                new Person ("Джеф Безос")
            };

            Clothing[] clothings =
            {
                new TShirt ("Біла"),
                new TShirt ("Рожева"),
                new Jaket ("Зелена"),
                new Pants ("Сині"),
                new Shoes ("Білі"),
                new Shoes ("Синьо-жовті")
```

```
            };

            Console.WriteLine("ЛЮДИ:");
            for (int i = 0; i < persons.Length; i++)
                Console.WriteLine("\t" + persons[i].Name);

            Console.WriteLine("\n\nГАРДЕРОБ:");
            for (int i = 0; i < clothings.Length; i++)
                Console.WriteLine("\t" + clothings[i].Color + " " +
clothings[i].Name);

            while (true)
            {
                var p = persons[PersonToChose()];
                var cl = clothings[ClothingToChose()];
                p.ToWear(cl);
                Console.WriteLine(p.ToString());
            }
        }

        static int PersonToChose()
        {
            int num;
            Console.WriteLine("\nОбери людину (0/1/2): ");
            num = Convert.ToInt32(Console.ReadLine());
            while (num > 2 || num < 0)
            {
                Console.WriteLine("Такої людини немає!");
                Console.WriteLine("\nОбери людину (0/1/2): ");
                num = Convert.ToInt32(Console.ReadLine());
            }
            return num;
        }
        static int ClothingToChose()
        {
            int num;
            Console.WriteLine("\nОбери річ (0/.../5): ");
            num = Convert.ToInt32(Console.ReadLine());
            while (num > 5 || num < 0)
            {
                Console.WriteLine("Такої речі немає!");
                Console.WriteLine("\nОбери річ (0/.../5): ");
                num = Convert.ToInt32(Console.ReadLine());
            }
            return num;
        }
```

```
    }
}
```

Person.cs

```
using System;

namespace proga_2_lab_2_1.Models
{
    public class Person
    {
        public string Name { get; set; }
        public TShirt TShirt { get; set; }
        public Pants Pants { get; set; }
        public Jaket Jaket { get; set; }
        public Shoes Shoes { get; set; }

        public Person(string name)
        {
            Name = name;
        }

        public void ToWear(Clothing cl)
        {
            if (cl.GetType().Name.ToString() == "TShirt")
            {
                if (TShirt == null)
                {
                    this.TShirt = (TShirt)cl;
                    Console.WriteLine(cl.Color + " " + cl.Name + "
одягнута!");
                }
                else Weared(cl);
            }
            else if (cl.GetType().Name.ToString() == "Jaket")
            {
                if (Jaket == null)
                {
                    this.Jaket = (Jaket)cl;
                    Console.WriteLine(cl.Color + " " + cl.Name + "
одягнута!");
                }
                else Weared(cl);
            }
            else if (cl.GetType().Name.ToString() == "Pants")
```

```
            {
                if (Pants == null)
                {
                    this.Pants = (Pants)cl;
                    Console.WriteLine(cl.Color + " " + cl.Name + "
одягнуті!");
                }
                else Weared(cl);
            }
            else if (cl.GetType().Name.ToString() == "Shoes")
            {
                if (Shoes == null)
                {
                    this.Shoes = (Shoes)cl;
                    Console.WriteLine(cl.Color + " " + cl.Name + "
одягнуті!");
                }
                else Weared(cl);
            }
        }

        private void Weared(Clothing cl)
        {
            Console.WriteLine(cl.Name + " вже надягнут(а/і)! Зняти?
(1/0)");
            int confirm = Convert.ToInt32(Console.ReadLine());
            if (confirm == 1)
            {
                if (cl.GetType().Name.ToString() == "TShirt")
this.TShirt = null;
                else if (cl.GetType().Name.ToString() == "Jaket")
this.Jaket = null;
                else if (cl.GetType().Name.ToString() == "Pants")
this.Pants = null;
                else if (cl.GetType().Name.ToString() == "Shoes")
this.Shoes = null;
                this.ToWear(cl);
            }
        }

        public override string ToString()
        {
            return ("\n" + Name + (TShirt == null ? "" :
"\nФутболка: " + TShirt.Color) + (Pants == null ? "" : "\nШтани: "
+ Pants.Color)
```

```
                + (Jaket == null ? "" : "\nКуртка: " + Jaket.Color)
+ (Shoes == null ? "" : "\nВзуття: " + Shoes.Color));
        }
    }
}
```

Clothing.cs

```
namespace proga_2_lab_2_1.Models
{
    public class Clothing
    {
        public string Color { get; set; }
        public string Name { get; set; }

        protected Clothing(string color, string name)
        {
            Color = color;
            Name = name;
        }

        public override string ToString()
        {
            return Name + " " + Color;
        }
    }
}
```

Jaket.cs

```
namespace proga_2_lab_2_1.Models
{
    class Jaket : Clothing
    {
        public Jaket(string color) : base(color, "куртка") { }
    }
}
```

Pants.cs

```
namespace proga_2_lab_2_1.Models
{
    class Pants : Clothing
```

```
    {
        public Pants(string color) : base(color, "штани") { }
    }
}
```

Shoes.cs

```
namespace proga_2_lab_2_1.Models
{
    class Shoes : Clothing
    {
        public Shoes(string color) : base(color, "кросівки") { }
    }
}
```

TShirt.cs

```
namespace proga_2_lab_2_1.Models
{
    class TShirt : Clothing
    {
        public TShirt(string color) : base(color, "футболка") { }
    }
}
```

Результат:

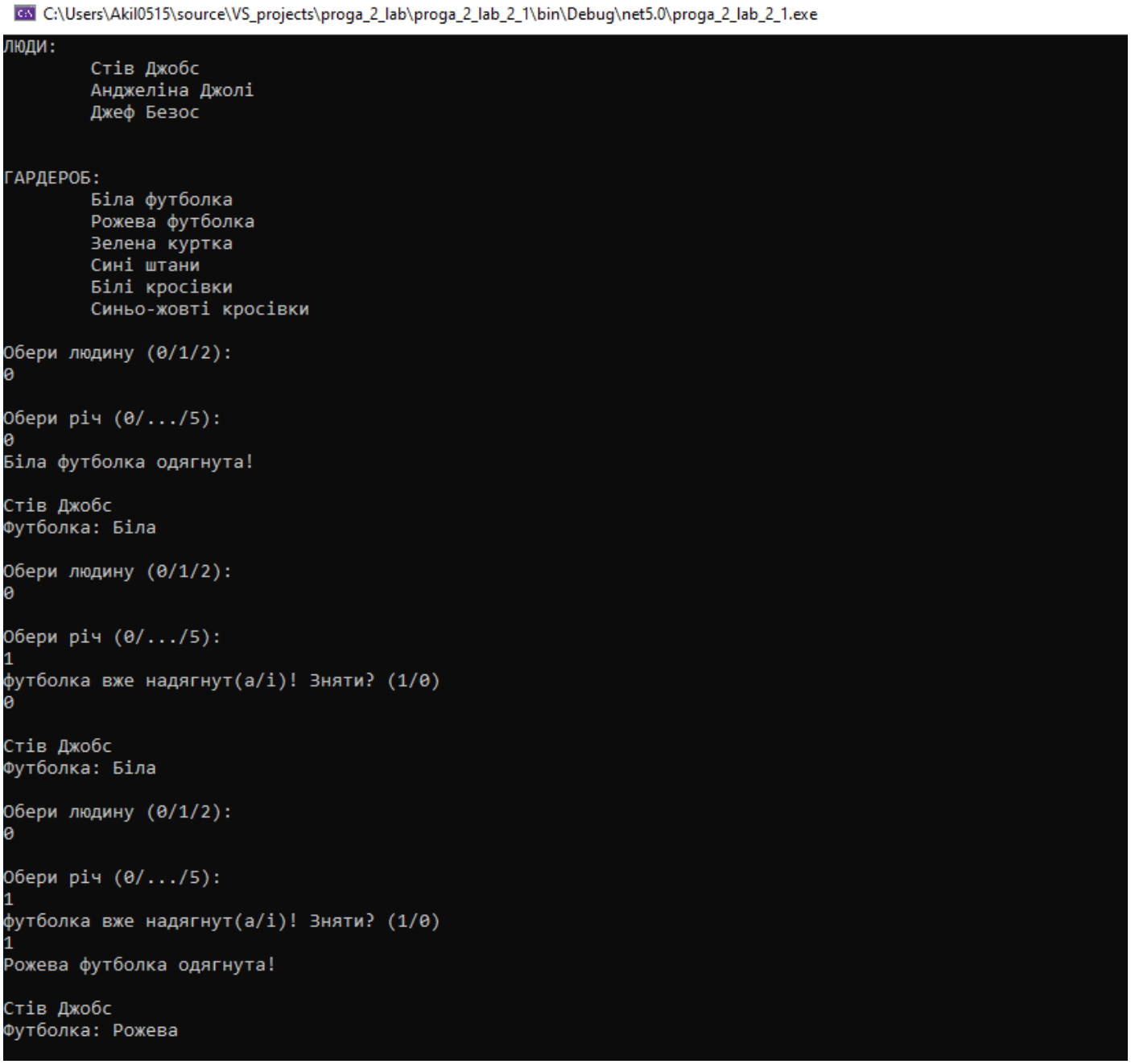

Рис. 1. Виведений результат виконання завдання 1

Завдання 2.

Створити програму, яка задовольняє наступним вимогам:
- Використовувати можливості ООП: класи, наслідування, поліморфізм, інкапсуляція.
- Кожен клас повинен мати змістовну назву та інформативний склад.
- Наслідування має застосовуватися тільки тоді, коли це має сенс.
- Класи повинні бути грамотно розкладені по пакетах.
- Консольне меню повинно бути мінімальним.
-
- Для зберігання параметрів ініціалізації можна використовувати файли.

Варіант 23:

23. Страхування. Визначити ієрархію страхових зобов'язань. Зібрати із зобов'язань дериватив. Підрахувати вартість. Провести сортування зобов'язань в деривативів на основі зменшення ступеня ризику. Знайти зобов'язання в деривативів, відповідне заданому діапазону параметрів.

Код:

Program.cs

```
using proga_2_lab_2_2.Models;
using System;

namespace proga_2_lab_2_2
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.OutputEncoding = System.Text.Encoding.Default;

            InsuranceType[] ins =
            {
                new InsuranceType.Personal(),
                new InsuranceType.Goods(),
                new InsuranceType.Responsibilities(),
                new InsuranceType.Re()
            };

            Zamovnuk[] zam =
            {
                new Zamovnuk("Дмитро Олегович"),
```

```
                new Zamovnuk("Роналдо"),
                new Zamovnuk("Страховик 3"),
                new Zamovnuk("Пан Стефан"),
                new Zamovnuk("Огрядна паня")
            };

            Strahuvalnuk[] strah =
            {
                new Strahuvalnuk("Страховик 1"),
                new Strahuvalnuk("Страховик 2")
            };

            Zabovyazanya[] zabov =
            {
                new Zabovyazanya(zam[1], strah[1], "Перелам правої
ноги", ins[0], 70000),
                new Zabovyazanya(zam[0], strah[0], "Потрапляння
ракети в будинок", ins[1], 50000),
                new Zabovyazanya(zam[3], strah[1], "Втрата роботи",
ins[2], 10000),
                new Zabovyazanya(zam[2], strah[1], "Виплата
страховою компанією 3-х зобов'язань", ins[3], 3000),
                new Zabovyazanya(zam[4], strah[0], "Втрата 20%
акцій власного бізнесу", ins[1], 500000)
            };

            Console.WriteLine("Збір деривативів:\n");
            Zabovyazanya.GetDerivativas();

            Console.WriteLine("\nПідрахунок страхової виплати у
разі виконання страхової умови відповідно типу забов'язань:\n");
            Zabovyazanya.GetFinalCost();

            Console.WriteLine("\nСортування забов'язань на основі
зменшення ступеня ризику:\n");
            Zabovyazanya.Sort();

            Console.WriteLine("\nПошук забов'язань за замовником та
страхувальником:\n");
            Console.WriteLine("Уведіть замовника: ");
            var zamovnuk = Console.ReadLine();
            Console.WriteLine("Уведіть страхувальника: ");
            var strahuvalnuk = Console.ReadLine();
            Zabovyazanya.FindZabovyazanya(zamovnuk, strahuvalnuk);

            Console.WriteLine("\nСтворення нових забов'язань:\n");
```

```
            Console.WriteLine("Виберіть замовника (0/.../4): ");
            foreach (Zamovnuk zam1 in Zamovnuk.Items.Values)
                Console.WriteLine("\t" + zam1);
            Console.WriteLine();
            int z = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Виберіть страхувальника: ");
            foreach (Strahuvalnuk strah1 in
Strahuvalnuk.Items.Values)
                Console.WriteLine("\t" + strah1);
            Console.WriteLine();
            int s = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Уведіть причину страхування: ");
            var r = Console.ReadLine();
            Console.WriteLine("Виберіть тип забов'язання: ");
            foreach (InsuranceType ins1 in
InsuranceType.Items.Values)
                Console.WriteLine("\t" + ins1);
            Console.WriteLine();
            int insur = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Уведіть суму страхування: ");
            int sum = Convert.ToInt32(Console.ReadLine());
            var newZab = new Zabovyazanya(zam[z], strah[s], r,
ins[insur], sum);
            Console.WriteLine(newZab);

            Console.WriteLine("\nДізнайтесь кількість забов'язань
за вибраним типом:\n");
            Console.WriteLine("Виберіть тип забов'язання: ");
            foreach (InsuranceType ins1 in
InsuranceType.Items.Values)
                Console.WriteLine("\t" + ins1);
            Console.WriteLine();
            int i = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("\nКількість забов'язань типу \"" +
ins[i] + "\" = " + InsuranceType.GetCountOfZabovyazannyas(ins[i]));
        }
    }
}
```

Base.cs

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

namespace proga_2_lab_2_2.Models
```

```
{
    public class Base<T> where T : Base<T>
    {
        public static Dictionary<Guid, T> Items = new
Dictionary<Guid, T>();

        public Guid Id { get; private set; }

        public Base()
        {
            Id = Guid.NewGuid();
            Items.Add(Id, (T)this);
        }
    }
}
```

Zabovyazanya.cs

```
using System;

namespace proga_2_lab_2_2.Models
{
    public class Zabovyazanya : Base<Zabovyazanya>
    {
        public int Num;
        public string Reason { get; set; }
        public float StartCost { get; set; }
        public float FinalCost { get; set; }

        // Замовник
        private Guid _zamovnukId;
        public Zamovnuk ZamovnukName
        {
            get { return Zamovnuk.Items[_zamovnukId]; }
            set { _zamovnukId = value.Id; }
        }

        // Страхувальник
        private Guid _strahuvalnukId;
        public Strahuvalnuk StrahuvalnukName
        {
            get { return Strahuvalnuk.Items[_strahuvalnukId]; }
            set { _strahuvalnukId = value.Id; }
        }
```

```
        // Страховий тип
        private Guid _insuranceTypeId;
        public InsuranceType Type
        {
            get { return InsuranceType.Items[_insuranceTypeId]; }
            set { _insuranceTypeId = value.Id; }
        }

        public Zabovyazanya(Zamovnuk zam, Strahuvalnuk strah,
string reason, InsuranceType type, int stcost)
        {
            Num = Zabovyazanya.Items.Count;
            ZamovnukName = zam;
            StrahuvalnukName = strah;
            Reason = reason;
            Type = type;
            StartCost = stcost;
        }

        public static void GetDerivativas()
        {
            foreach (Zabovyazanya zab in Zabovyazanya.Items.Values)
            {
                Console.WriteLine("Я - " + zab.StrahuvalnukName + "
забов'язуюсь \nвиплатити " + zab.Type.InsuranceCost
                    + "-кратну величину базової ціни страхування ("
+ zab.StartCost + ") \nу страховому випадку: " + zab.Reason +
"\n");
            }
        }

        public static void GetFinalCost()
        {
            foreach (Zabovyazanya zab in Zabovyazanya.Items.Values)
            {
                zab.FinalCost = zab.StartCost *
zab.Type.InsuranceCost;
                Console.WriteLine("   Забов'язання № " + zab.Num +
"\n\tСтрахова виплата => " + zab.FinalCost + " грн\n");
            }
        }

        // Сортування на основі зменшення ступеня ризику, тобто за
зменшенням InsuranceCost (за видами забов'язань)
        public static void Sort()
        {
```

```
            Console.WriteLine("-----------------------------------
--------------------------------");
            Console.WriteLine("   Особистісне страхування:");
            DefineAndPrintZabovyazanyaOnType(40);
            Console.WriteLine("-----------------------------------
--------------------------------");
            Console.WriteLine("   Майнове страхування:");
            DefineAndPrintZabovyazanyaOnType(30);
            Console.WriteLine("-----------------------------------
--------------------------------");
            Console.WriteLine("   Страхування видів
відповідальності:");
            DefineAndPrintZabovyazanyaOnType(20);
            Console.WriteLine("-----------------------------------
--------------------------------");
            Console.WriteLine("   Перестрахування:");
            DefineAndPrintZabovyazanyaOnType(10);
            Console.WriteLine("-----------------------------------
--------------------------------");
        }

        private static void DefineAndPrintZabovyazanyaOnType(int i)
        {
            foreach (Zabovyazanya zab in Zabovyazanya.Items.Values)
                if (zab.Type.InsuranceCost == i)
Console.WriteLine(zab + "\n");
        }

        public static void FindZabovyazanya(string zam, string
strah)
        {
            bool check = true;
            foreach (Zabovyazanya zab in Zabovyazanya.Items.Values)
                if (zab.ZamovnukName.Name == zam &&
zab.StrahuvalnukName.Name == strah)
                {
                    Console.WriteLine(zab + "\n");
                    check = false;
                }
            if (check) Console.WriteLine("\nТакого забов'язання не
існує!");
        }

        public override string ToString()
        {
```

```
            return ("\nЗабов'язання № " + Num + ":\n\tЗамовник\t|
" + ZamovnukName + "\n\tСтрахувальник\t|   " + StrahuvalnukName
                + "\n\tТип\t\t|   " + Type + "\n\tПричина\t\t|   "
+ Reason + "\n\tБазова вартість\t|   " + StartCost + " грн");
        }
    }
}
```

InsuranceType.cs

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

namespace proga_2_lab_2_2.Models
{
    public class InsuranceType : Base<InsuranceType>
    {
        public string Name { get; set; }
        public float InsuranceCost { get; set; }
        public int Count { get; set; }

        public class Personal : InsuranceType
        {
            // Уміст забов'язань у типі забов'язння
            public static List<Zabovyazanya> Zabovyazanyas
            {
                get { return Zabovyazanya.Items.Values.Where(zab =>
zab.Type.Name == "Особистісне страхування").ToList(); }
            }
            public Personal()
            {
                Name = "Особистісне страхування";
                InsuranceCost = 40;
            }
        }

        public class Goods : InsuranceType
        {
            // Уміст забов'язань у типі забов'язння
            public static List<Zabovyazanya> Zabovyazanyas
            {
                get { return Zabovyazanya.Items.Values.Where(zab =>
zab.Type.Name == "Майнове страхування").ToList(); }
            }
            public Goods()
```

```
            {
                Name = "Майнове страхування";
                InsuranceCost = 30;
            }
        }

        public class Responsibilities : InsuranceType
        {
            // Уміст забов'язань у типі забов'язння
            public static List<Zabovyazanya> Zabovyazanyas
            {
                get { return Zabovyazanya.Items.Values.Where(zab =>
zab.Type.Name == "Страхування видів відповідальності").ToList(); }
            }
            public Responsibilities()
            {
                Name = "Страхування видів відповідальності";
                InsuranceCost = 20;
            }
        }

        public class Re : InsuranceType
        {
            // Уміст забов'язань у типі забов'язння
            public static List<Zabovyazanya> Zabovyazanyas
            {
                get { return Zabovyazanya.Items.Values.Where(zab =>
zab.Type.Name == "Перестрахування").ToList(); }
            }
            public Re()
            {
                Name = "Перестрахування";
                InsuranceCost = 10;
            }
        }

        public static int GetCountOfZabovyazannyas(InsuranceType
type)
        {
            if (type.GetType().Name.ToString() == "Personal")
return Personal.Zabovyazanyas.Count;
            else if (type.GetType().Name.ToString() == "Goods")
return Goods.Zabovyazanyas.Count;
            else if (type.GetType().Name.ToString() ==
"Responsibilities") return Responsibilities.Zabovyazanyas.Count;
            else return Re.Zabovyazanyas.Count;
```

```
        }

        public override string ToString()
        {
            return Name;
        }
    }
}
```

Strahuvalnuk.cs

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

namespace proga_2_lab_2_2.Models
{
    public class Strahuvalnuk : Base<Strahuvalnuk>
    {
        public string Name { get; set; }

        // Уміст забов'язань у страхувальнику
        public List<Zabovyazanya> Zabovyazanyas
        {
            get { return Zabovyazanya.Items.Values.Where(zab =>
zab.StrahuvalnukName == this).ToList(); }
        }

        public Strahuvalnuk(string name)
        {
            Name = name;
        }

        public override string ToString()
        {
            return Name;
        }
    }
}
```

Zamovnuk.cs

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

```
namespace proga_2_lab_2_2.Models
{
    public class Zamovnuk : Base<Zamovnuk>
    {
        public string Name { get; set; }

        // Уміст забов'язань у замовнику
        public List<Zabovyazanya> Zabovyazanyas
        {
            get { return Zabovyazanya.Items.Values.Where(zab =>
zab.ZamovnukName == this).ToList(); }
        }

        public Zamovnuk(string name)
        {
            Name = name;
        }

        public override string ToString()
        {
            return Name;
        }
    }
}
```

Результат:

Microsoft Visual Studio Debug Console

```
Збір деривативів:

Я - Страховик 2 забов'язуюсь
виплатити 40-кратну величину базової ціни страхування (70000)
у страховому випадку: Перелам правої ноги

Я - Страховик 1 забов'язуюсь
виплатити 30-кратну величину базової ціни страхування (50000)
у страховому випадку: Потрапляння ракети в будинок

Я - Страховик 2 забов'язуюсь
виплатити 20-кратну величину базової ціни страхування (10000)
у страховому випадку: Втрата роботи

Я - Страховик 2 забов'язуюсь
виплатити 10-кратну величину базової ціни страхування (3000)
у страховому випадку: Виплата страховою компанією 3-х зобов'язань

Я - Страховик 1 забов'язуюсь
виплатити 30-кратну величину базової ціни страхування (500000)
у страховому випадку: Втрата 20% акцій власного бізнесу


Підрахунок страхової виплати у разі виконання страхової умови відповідно типу забов'язань:

   Забов'язання № 1
        Страхова виплата => 2800000 грн

   Забов'язання № 2
        Страхова виплата => 1500000 грн

   Забов'язання № 3
        Страхова виплата => 200000 грн

   Забов'язання № 4
        Страхова виплата => 30000 грн

   Забов'язання № 5
        Страхова виплата => 15000000 грн


Сортування забов'язань на основі зменшення ступеня ризику:

----------------------------------------------------------------------
   Особистісне страхування:

Забов'язання № 1:
        Замовник         |   Роналдо
        Страхувальник    |   Страховик 2
        Тип              |   Особистісне страхування
        Причина          |   Перелам правої ноги
        Базова вартість  |   70000 грн

----------------------------------------------------------------------
   Майнове страхування:

Забов'язання № 2:
        Замовник         |   Дмитро Олегович
        Страхувальник    |   Страховик 1
        Тип              |   Майнове страхування
        Причина          |   Потрапляння ракети в будинок
        Базова вартість  |   50000 грн
```

```
Забов'язання № 5:
        Замовник         |   Огрядна паня
        Страхувальник    |   Страховик 1
        Тип              |   Майнове страхування
        Причина          |   Втрата 20% акцій власного бізнесу
        Базова вартість  |   500000 грн

----------------------------------------------------------------------
   Страхування видів відповідальності:

Забов'язання № 3:
        Замовник         |   Пан Стефан
        Страхувальник    |   Страховик 2
        Тип              |   Страхування видів відповідальності
        Причина          |   Втрата роботи
        Базова вартість  |   10000 грн

----------------------------------------------------------------------
   Перестрахування:

Забов'язання № 4:
        Замовник         |   Страховик 3
        Страхувальник    |   Страховик 2
        Тип              |   Перестрахування
        Причина          |   Виплата страховою компанією 3-х зобов'язань
        Базова вартість  |   3000 грн

----------------------------------------------------------------------

Пошук забов'язань за замовником та страхувальником:

Уведіть замовника:
Страховик 3
Уведіть страхувальника:
Страховик 2

Забов'язання № 4:
        Замовник         |   Страховик 3
        Страхувальник    |   Страховик 2
        Тип              |   Перестрахування
        Причина          |   Виплата страховою компанією 3-х зобов'язань
        Базова вартість  |   3000 грн


Створення нових забов'язань:

Виберіть замовника (0/.../4):
        Дмитро Олегович
        Роналдо
        Страховик 3
        Пан Стефан
        Огрядна паня

1
```

```
Виберіть страхувальника:
        Страховик 1
        Страховик 2

0
Уведіть причину страхування:
Красивий голос
Виберіть тип забов'язання:
        Особистісне страхування
        Майнове страхування
        Страхування видів відповідальності
        Перестрахування

0
Уведіть суму страхування:
1000000

Забов'язання № 6:
        Замовник        |   Роналдо
        Страхувальник   |   Страховик 1
        Тип             |   Особистісне страхування
        Причина         |   Красивий голос
        Базова вартість |   1000000 грн

Дізнайтесь кількість забов'язань за вибраним типом:

Виберіть тип забов'язання:
        Особистісне страхування
        Майнове страхування
        Страхування видів відповідальності
        Перестрахування

0

Кількість забов'язань типу "Особистісне страхування" = 2
```

Рис. 1. Виведений результат виконання завдання 2

Висновки:

У ході виконання лабораторної роботи я дізналась, що являє собою поліморфізм, наслідування та перевантаження методів у мові програмування C#, причини та принципи їх реалізації.

Я навчилась на практиці реалізовувати поліморфізм, наслідування та перезапис функцій і розв’язувати за допомогою них поставлені задачі.