Міністерство освіти і науки України
Національний технічний університет України „КПІ”
Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки
інформації та управління

ЗВІТ

з комп’ютерного практикуму № 2
на тему :

„ ПРИВЕДЕННЯ ТИПІВ. ПЕРЕЗАВАНТАЖЕННЯ ОПЕРАТОРІВ ”

Виконав студент *ІП-63 Семченко Андрій Олегович* _____
(№ групи, прізвище, ім’я, по батькові)

Прийняв *Ліщук К. І.* _____
(посада, прізвище, ім’я, по батькові)

Київ 2018

ЗМІСТ

1 МЕТА РОБОТИ

Мета роботи – вивчення засобів мови C# для програмування перезавантаження операцій для класів.

2 ПОСТАНОВКА ЗАДАЧІ

Создать абстрактный класс AirCraft. На его основе реализовать классы Plane (самолет), Helicopter (вертолет) и Freighter (грузовой самолет). Классы должны иметь возможность задавать и получать параметры средств передвижения (цена, максимальная скорость, год выпуска, грузоподъемность, вместимость пассажиров и т.д.). Наряду с общими полями и методами, каждый класс должен содержать и специфичные для него поля.

Создать класс Airport, содержащий массив/параметризованную коллекцию объектов этих классов в динамической памяти.

Предусмотреть возможность вывода характеристик объектов списка.

3 ТЕКСТИ ПРОГРАМНОГО КОДУ

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.Serialization.Formatters.Binary;
using System.Text;
using System.Threading.Tasks;


namespace lab02
{
   abstract class AirCraft : Object
   {
      protected int Cost { get; set; }
      protected double Max_speed_meters_per_sec { get; set; }
      protected string Assembly_date { get; set; }
      protected int Carrying_capacity_kg { get; set; }
      protected int Capacity_persons { get; set; }
      public Object DeepCopy()
      {
         using (var ms = new MemoryStream())
         {
            var formatter = new BinaryFormatter();
            formatter.Serialize(ms, this);
            ms.Position = 0;


            return (Object)formatter.Deserialize(ms);
         }
      }
   }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace lab02
```

```
{
    class Airport
    {
        public Airport()
        {
            AirCrafts = new List<AirCraft>();
            InitiallySubscribeToEvents();
        }
        public Airport(List<AirCraft> craftsList)
        {
            AirCrafts = craftsList;
            InitiallySubscribeToEvents();
        }
        public override string ToString()
        {
            string str = "AirCrafts: ";
            foreach (AirCraft function in AirCrafts)
            {
                str += function.ToString() + "; ";
            }


            return str;
        }
        public static bool operator ==(Airport firstAirport, Airport secondAirport)
        {
            return firstAirport.Equals(secondAirport);
        }
        public static bool operator !=(Airport firstAirport, Airport secondAirport)
        {
            return !firstAirport.Equals(secondAirport);
        }
        public List<AirCraft> AirCraftsList
        {
            get
            {
                return AirCrafts;
            }
            set
            {
                AirCrafts.Clear();
                Changed(this, new MyEventArgs("Aircrafts list has been changed"));
```

```
            AirCrafts.Clear();
            for (int i = 0; i < value.Count; i++)
            {
                AirCrafts.Add(value[i]);
            }
        }
    }
    public void OnEventTriggered(object sender, MyEventArgs e)
    {
        Console.WriteLine(e.toString());
    }
    public void Add(AirCraft function)
    {
        AirCrafts.Add(function);
        Added?.Invoke(this, new MyEventArgs("Added " + function.ToString()));
    }
    public void RemoveAt(int index)
    {
        AirCrafts.RemoveAt(index);
        Deleted(this, new MyEventArgs("Deleted craft[" + index.ToString() + "]"));
    }
    public void Remove(Object obj)
    {
        AirCrafts.Remove((AirCraft)obj);
        Deleted(this, new MyEventArgs("Deleted " + ((AirCraft)obj).ToString()));
    }
    public override bool Equals(object obj)
    {
        List<AirCraft> other = ((Airport)obj).AirCraftsList;
        if (other.Count != AirCrafts.Count)
        {
            return false;
        }


        for (int i = 0; i < AirCrafts.Count; i++)
        {
            if (!AirCrafts[i].Equals(other[i]))
            {
                return false;
            }
        }
```

```
            return true;
        }
        public override int GetHashCode()
        {
            int hash = 0;
            foreach (var element in AirCrafts)
            {
                hash += element.GetHashCode();
            }
            return hash / 120;
        }
        public object DeepCopy()
        {
            Airport newAirport = new Airport();
            foreach (var craft in AirCrafts)
            {
                newAirport.Add(craft);
            }
            return newAirport;
        }
        private List<AirCraft> AirCrafts;
        private delegate void AirportEventHandler(object sender, MyEventArgs e);
        private event AirportEventHandler Added;
        private event AirportEventHandler Deleted;
        private event AirportEventHandler Changed;
        private void InitiallySubscribeToEvents()
        {
            Added += OnEventTriggered;
            Deleted += OnEventTriggered;
            Changed += OnEventTriggered;
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace lab02
{
```

```
class Freighter: AirCraft
{
    private int minimal_pilot_count { get; set; }
    private double cargo_hold_volume_m3 { get; set; }
    public override bool Equals(Object other)
    {
        if (other == null || other.GetType() != typeof(Freighter))
        {
            return false;
        }
        Freighter other_freighter = (Freighter)other;
        return other_freighter.minimal_pilot_count == minimal_pilot_count &&
            other_freighter.cargo_hold_volume_m3 == cargo_hold_volume_m3 &&
            other_freighter.Carrying_capacity_kg == Carrying_capacity_kg &&
            other_freighter.Capacity_persons == Capacity_persons;
    }
    public static bool operator == (Freighter freighter1, Freighter freighter2)
    {
        if (freighter1 == null)
        {
            throw new ArgumentNullException(nameof(freighter1));
        }
        return freighter1.Equals(freighter2);
    }
    public static bool operator !=(Freighter freighter1, Freighter freighter2)
    {
        if (freighter1 == null)
        {
            throw new ArgumentNullException(nameof(freighter1));
        }
        return !freighter1.Equals(freighter2);
    }
    public override int GetHashCode()
    {
        return minimal_pilot_count + (int)cargo_hold_volume_m3 + Carrying_capacity_kg + Capacity_persons;
    }
    public override string ToString()
    {
          return String.Format("Freighter max speed {0} m/s, costs {1}$, minimal {2} pilots expected, {3}m3 cargo
section volume",
            Max_speed_meters_per_sec,
            Cost, minimal_pilot_count,
```

```
            cargo_hold_volume_m3);
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace lab02
{
    class Helicopter: AirCraft
    {
        private int propeller_blades_count { get; set; }
        private int propeller_length { get; set; }
        public override bool Equals(Object other)
        {
            if (other == null || other.GetType() != typeof(Helicopter))
            {
                return false;
            }
            Helicopter other_helicopter = (Helicopter)other;
            return other_helicopter.propeller_blades_count == propeller_blades_count &&
                other_helicopter.propeller_length == propeller_length &&
                other_helicopter.Carrying_capacity_kg == Carrying_capacity_kg &&
                other_helicopter.Capacity_persons == Capacity_persons;
        }
        public static bool operator ==(Helicopter helicopter1, Helicopter helicopter2)
        {
            if (helicopter1 == null)
            {
                throw new ArgumentNullException(nameof(helicopter1));
            }
            return helicopter1.Equals(helicopter2);
        }
        public static bool operator !=(Helicopter helicopter1, Helicopter helicopter2)
        {
            if (helicopter1 == null)
            {
                throw new ArgumentNullException(nameof(helicopter1));
```

```
            }
            return !helicopter1.Equals(helicopter2);
        }
        public override int GetHashCode()
        {
            return propeller_blades_count + propeller_length + Carrying_capacity_kg + Capacity_persons;
        }
        public override string ToString()
        {
            return String.Format("Helicopter max speed {0} m/s, costs {1}$, contains {2} propeller blades, each {3} meters lenght",
                Max_speed_meters_per_sec,
                Cost, propeller_blades_count,
                propeller_length);
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace lab02
{
    class MyEventArgs : System.EventArgs
    {
        public MyEventArgs(String str)
        {
            Info = str;
        }
        public String toString()
        {
            return Info;
        }
        private String Info;
    }
}
using System;

namespace lab02
```

```
{
    class Plane : AirCraft
    {
        private string owner_company_name { get; set; }
        private int life_jackets_count { get; set; }
        private int required_stewardess_count { get; set; }
        public override bool Equals(Object other) {
            if (other == null || other.GetType() != typeof(Plane))
            {
                return false;
            }
            Plane other_plane = (Plane)other;
            return other_plane.required_stewardess_count == required_stewardess_count &&
                other_plane.Carrying_capacity_kg == Carrying_capacity_kg &&
                other_plane.Capacity_persons == Capacity_persons;
        }
        public static bool operator == (Plane plane1, Plane plane2)
        {
            if (plane1 == null)
            {
                throw new ArgumentNullException(nameof(plane1));
            }
            return plane1.Equals(plane2);
        }
        public static bool operator != (Plane plane1, Plane plane2)
        {
            if (plane1 == null)
            {
                throw new ArgumentNullException(nameof(plane1));
            }
            return !plane1.Equals(plane2);
        }
        public override int GetHashCode()
        {
            return required_stewardess_count + Carrying_capacity_kg + Capacity_persons;
        }
        public override string ToString()
        {
            return String.Format("Plane owned by {0}, minimal {1} stewardess required," +
                " max speed {2} m/s, costs {3}$", owner_company_name,
                required_stewardess_count, Max_speed_meters_per_sec, Cost);
        }
```

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


namespace lab02
{
    class Program
    {
        static void Main(string[] args)
        {
            Airport airport = new Airport();
            airport.Add(new Freighter());
            airport.Add(new Helicopter());
            airport.Add(new Plane());
            System.Console.WriteLine(airport.ToString());
            airport.RemoveAt(0);
            System.Console.WriteLine(airport.ToString());
        }
    }
}
```

4 ВИСНОВКИ

В ході лабораторної роботи було розглянуто принципи побудови об’єктно-орієнтованих програм. Також були розглянуті такі засоби реалізації поліморфізму у мові програмування C# як перевантаження операторів. В рамках завдання була створена колекція даних, що сповіщує про зміни своїх елементів та свого розміру за допомогою подій. Було придбано навички об’єктно-орієнтованого проектування засобами мови програмування C#.

В ході перевірки програми помилок знайдено не було.