**Лабораторна робота 3**

**Минко Ярослав**

1.   
using System;

using System.Collections.Generic;

public class LivingOrganism

{

public double Energy { get; set; }

public int Age { get; set; }

public double Size { get; set; }

public LivingOrganism(double energy, int age, double size)

{

Energy = energy;

Age = age;

Size = size;

}

}

public class Animal : LivingOrganism, IReproducible, IPredator

{

public string Species { get; set; }

public int OffspringCount { get; set; }

public Animal(double energy, int age, double size, string species, int offspringCount)

: base(energy, age, size)

{

Species = species;

OffspringCount = offspringCount;

}

public void Reproduce()

{

Console.WriteLine($"{Species} тварина розмножується.");

}

public void Hunt(LivingOrganism prey)

{

Console.WriteLine($"{Species} тварина полює на {prey.GetType().Name}.");

}

}

public class Plant : LivingOrganism, IReproducible

{

public string Type { get; set; }

public Plant(double energy, int age, double size, string type)

: base(energy, age, size)

{

Type = type;

}

public void Reproduce()

{

Console.WriteLine($"{Type} рослина розмножується.");

}

}

public class Microorganism : LivingOrganism, IReproducible

{

public string MicrobeType { get; set; }

public Microorganism(double energy, int age, double size, string microbeType)

: base(energy, age, size)

{

MicrobeType = microbeType;

}

public void Reproduce()

{

Console.WriteLine($"{MicrobeType} мікроорганізм розмножується.");

}

}

public interface IReproducible

{

void Reproduce();

}

public interface IPredator

{

void Hunt(LivingOrganism prey);

}

public class Ecosystem

{

public List<LivingOrganism> Organisms { get; set; }

public Ecosystem()

{

Organisms = new List<LivingOrganism>();

}

public void AddOrganism(LivingOrganism organism)

{

Organisms.Add(organism);

}

public void SimulateEcosystem()

{

foreach (var organism in Organisms)

{

Console.WriteLine($"Характеристики {organism.GetType().Name}:");

Console.WriteLine($"Енергія: {organism.Energy}");

Console.WriteLine($"Вік: {organism.Age}");

Console.WriteLine($"Розмір: {organism.Size}");

if (organism is IReproducible)

{

((IReproducible)organism).Reproduce();

}

if (organism is IPredator)

{

foreach (var prey in Organisms)

{

if (prey != organism)

{

((IPredator)organism).Hunt(prey);

}

}

}

Console.WriteLine();

}

}

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Минко Ярослав");

Ecosystem ecosystem = new Ecosystem();

Animal lion = new Animal(100, 5, 2, "Лев", 2);

Animal zebra = new Animal(50, 3, 1, "Зебра", 1);

Plant tree = new Plant(20, 10, 5, "Дерево");

Microorganism bacteria = new Microorganism(5, 1, 0.01, "Бактерія");

ecosystem.AddOrganism(lion);

ecosystem.AddOrganism(zebra);

ecosystem.AddOrganism(tree);

ecosystem.AddOrganism(bacteria);

ecosystem.SimulateEcosystem();

}

}

2.   
using System;

using System.Collections.Generic;

public class Computer

{

public string IPAddress { get; set; }

public int Power { get; set; }

public string OperatingSystem { get; set; }

public Computer(string ipAddress, int power, string os)

{

IPAddress = ipAddress;

Power = power;

OperatingSystem = os;

}

}

public class Server : Computer, IConnectable

{

public int MaxConnections { get; set; }

public List<Computer> ConnectedComputers { get; }

public Server(string ipAddress, int power, string os, int maxConnections)

: base(ipAddress, power, os)

{

MaxConnections = maxConnections;

ConnectedComputers = new List<Computer>();

}

public void Connect(Computer computer)

{

if (ConnectedComputers.Count < MaxConnections)

{

ConnectedComputers.Add(computer);

Console.WriteLine($"Сервер {IPAddress} підключив комп'ютер {computer.IPAddress}.");

}

else

{

Console.WriteLine($"Помилка: Сервер {IPAddress} досягнув ліміту з'єднань.");

}

}

public void Disconnect(Computer computer)

{

if (ConnectedComputers.Contains(computer))

{

ConnectedComputers.Remove(computer);

Console.WriteLine($"Сервер {IPAddress} відключив комп'ютер {computer.IPAddress}.");

}

else

{

Console.WriteLine($"Помилка: Сервер {IPAddress} не підключений до комп'ютера {computer.IPAddress}.");

}

}

}

public class Workstation : Computer, IConnectable

{

public Workstation(string ipAddress, int power, string os)

: base(ipAddress, power, os)

{

}

public void Connect(Computer computer)

{

Console.WriteLine($"Робоча станція {IPAddress} підключилась до сервера {computer.IPAddress}.");

}

public void Disconnect(Computer computer)

{

Console.WriteLine($"Робоча станція {IPAddress} відключилась від сервера {computer.IPAddress}.");

}

}

public class Router : Computer, IConnectable

{

public List<Computer> ConnectedComputers { get; }

public Router(string ipAddress, int power, string os)

: base(ipAddress, power, os)

{

ConnectedComputers = new List<Computer>();

}

public void Connect(Computer computer)

{

ConnectedComputers.Add(computer);

Console.WriteLine($"Маршрутизатор {IPAddress} підключив комп'ютер {computer.IPAddress}.");

}

public void Disconnect(Computer computer)

{

if (ConnectedComputers.Contains(computer))

{

ConnectedComputers.Remove(computer);

Console.WriteLine($"Маршрутизатор {IPAddress} відключив комп'ютер {computer.IPAddress}.");

}

else

{

Console.WriteLine($"Помилка: Маршрутизатор {IPAddress} не підключений до комп'ютера {computer.IPAddress}.");

}

}

}

public interface IConnectable

{

void Connect(Computer computer);

void Disconnect(Computer computer);

}

public class Network

{

public List<Computer> Computers { get; }

public Network()

{

Computers = new List<Computer>();

}

public void AddComputer(Computer computer)

{

Computers.Add(computer);

}

public void SimulateNetwork()

{

foreach (var computer in Computers)

{

if (computer is IConnectable)

{

var connectableComputer = (IConnectable)computer;

if (Computers.Count > 1)

{

var random = new Random();

var targetComputer = Computers[random.Next(Computers.Count)];

if (targetComputer != computer)

{

connectableComputer.Connect(targetComputer);

connectableComputer.Disconnect(targetComputer);

}

}

}

}

}

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Минко Ярослав");

Network network = new Network();

Server server1 = new Server("192.168.0.1", 4, "Windows Server", 3);

Server server2 = new Server("192.168.0.2", 4, "Windows Server", 2);

Workstation workstation1 = new Workstation("192.168.0.3", 2, "Windows 10");

Workstation workstation2 = new Workstation("192.168.0.4", 2, "Windows 10");

Router router = new Router("192.168.0.5", 2, "Router OS");

network.AddComputer(server1);

network.AddComputer(server2);

network.AddComputer(workstation1);

network.AddComputer(workstation2);

network.AddComputer(router);

network.SimulateNetwork();

}

}

3.  
using System;

using System.Collections.Generic;

public class Дорога

{

public double Довжина { get; set; }

public double Ширина { get; set; }

public int КількістьСмуг { get; set; }

public int РівеньТрафіку { get; set; }

}

public class ТранспортнийЗасіб : IDriveable

{

public double Швидкість { get; set; }

public double Розмір { get; set; }

public string Тип { get; set; }

public void Рухатися()

{

Console.WriteLine($"Транспортний засіб типу {Тип} рухається зі швидкістю {Швидкість} км/год.");

}

public void Зупинитися()

{

Console.WriteLine($"Транспортний засіб типу {Тип} зупинився.");

}

}

public interface IDriveable

{

void Рухатися();

void Зупинитися();

}

public class РухТранспорту

{

public void РухТранспортуНаДорозі(ТранспортнийЗасіб засіб, Дорога дорога)

{

Console.WriteLine($"Транспортний засіб типу {засіб.Тип} рухається на дорозі довжиною {дорога.Довжина} км зі швидкістю {засіб.Швидкість} км/год.");

}

}

public class Симуляція

{

public List<ТранспортнийЗасіб> ТранспортніЗасоби { get; set; }

public Симуляція(List<ТранспортнийЗасіб> транспортніЗасоби)

{

ТранспортніЗасоби = транспортніЗасоби;

}

public void МоделюватиТрафік(Дорога дорога)

{

Console.WriteLine($"Моделювання трафіку на дорозі. Довжина дороги: {дорога.Довжина} км, кількість смуг: {дорога.КількістьСмуг}");

foreach (var засіб in ТранспортніЗасоби)

{

РухТранспорту рухТранспорту = new РухТранспорту();

рухТранспорту.РухТранспортуНаДорозі(засіб, дорога);

}

}

public void ОптимізуватиТрафік(Дорога дорога)

{

Console.WriteLine($"Оптимізація трафіку на дорозі. Поточний рівень трафіку: {дорога.РівеньТрафіку}");

}

}

class Program

{

static void Main()

{

Console.WriteLine("Минко Ярослав");

var дорога = new Дорога { Довжина = 100, КількістьСмуг = 3, РівеньТрафіку = 4 };

var авто1 = new ТранспортнийЗасіб { Тип = "Автомобіль", Швидкість = 90 };

var авто2 = new ТранспортнийЗасіб { Тип = "Автомобіль", Швидкість = 80 };

var автобус = new ТранспортнийЗасіб { Тип = "Автобус", Швидкість = 60 };

var транспортніЗасоби = new List<ТранспортнийЗасіб> { авто1, авто2, автобус };

var симуляція = new Симуляція(транспортніЗасоби);

симуляція.МоделюватиТрафік(дорога);

симуляція.ОптимізуватиТрафік(дорога);

Console.ReadLine();

}

}