**Лабораторна робота № 5 : перевантаження, абстракція, поліморфізм  
Минко Ярослав**

**Завдання 1**using System;

using System.Collections.Generic;

abstract class Vehicle

{

public int Speed { get; set; }

public int Capacity { get; set; }

public abstract void Move();

}

class Human

{

public int Speed { get; set; }

public void Move()

{

Console.WriteLine("Human is walking.");

}

}

class Car : Vehicle

{

public Car()

{

Speed = 60;

Capacity = 5;

}

public override void Move()

{

Console.WriteLine("Car is driving.");

}

}

class Bus : Vehicle

{

public Bus()

{

Speed = 40;

Capacity = 30;

}

public override void Move()

{

Console.WriteLine("Bus is moving.");

}

}

class Train : Vehicle

{

public Train()

{

Speed = 100;

Capacity = 200;

}

public override void Move()

{

Console.WriteLine("Train is moving on tracks.");

}

}

class TransportNetwork

{

private List<Vehicle> vehicles;

public TransportNetwork()

{

vehicles = new List<Vehicle>();

}

public void AddVehicle(Vehicle vehicle)

{

vehicles.Add(vehicle);

}

public void MoveAllVehicles()

{

foreach (var vehicle in vehicles)

{

vehicle.Move();

}

}

}

class Route

{

public string StartPoint { get; set; }

public string EndPoint { get; set; }

public Route(string startPoint, string endPoint)

{

StartPoint = startPoint;

EndPoint = endPoint;

}

}

class ExtendedTransportNetwork : TransportNetwork

{

public void CalculateOptimalRoute(Route route)

{

Console.WriteLine($"Calculating optimal route from {route.StartPoint} to {route.EndPoint}.");

}

public void BoardPassengers(int numberOfPassengers)

{

Console.WriteLine($"Boarding {numberOfPassengers} passengers.");

}

public void DisembarkPassengers(int numberOfPassengers)

{

Console.WriteLine($"Disembarking {numberOfPassengers} passengers.");

}

}

class Program

{

static void Main()

{

Console.WriteLine("Минко Ярослав");

Car car = new Car();

Bus bus = new Bus();

Train train = new Train();

Human person = new Human();

ExtendedTransportNetwork transportNetwork = new ExtendedTransportNetwork();

transportNetwork.AddVehicle(car);

transportNetwork.AddVehicle(bus);

transportNetwork.AddVehicle(train);

transportNetwork.MoveAllVehicles();

Route route = new Route("City A", "City B");

transportNetwork.CalculateOptimalRoute(route);

transportNetwork.BoardPassengers(20);

transportNetwork.MoveAllVehicles();

transportNetwork.DisembarkPassengers(15);

}

}

**Завдання 2**

using System;

class MathOperations

{

public static T Add<T>(T a, T b)

{

return (dynamic)a + (dynamic)b;

}

public class ArrayAdder

{

public static T[] Add<T>(T[] a, T[] b)

{

if (a.Length != b.Length)

throw new ArgumentException("Масиви повинні бути однакової довжини.");

T[] result = new T[a.Length];

for (int i = 0; i < a.Length; i++)

{

result[i] = AddValues(a[i], b[i]);

}

return result;

}

private static T AddValues<T>(T a, T b)

{

dynamic dynamicA = a;

dynamic dynamicB = b;

return dynamicA + dynamicB;

}

public class MatrixAdder

{

public static T[,] Add<T>(T[,] a, T[,] b)

{

if (a.GetLength(0) != b.GetLength(0) || a.GetLength(1) != b.GetLength(1))

{

throw new ArgumentException("Матриці повинні мати однакові розміри.");

}

T[,] result = new T[a.GetLength(0), a.GetLength(1)];

for (int i = 0; i < a.GetLength(0); i++)

{

for (int j = 0; j < a.GetLength(1); j++)

{

result[i, j] = AddValues(a[i, j], b[i, j]);

}

}

return result;

}

private static T AddValues<T>(T a, T b)

{

dynamic dynamicA = a;

dynamic dynamicB = b;

return dynamicA + dynamicB;

}

public static T Subtract<T>(T a, T b)

{

dynamic aa = a, bb = b;

return aa - bb;

}

public class ArraySubtractor

{

public static T[] Subtract<T>(T[] a, T[] b)

{

if (a.Length != b.Length)

throw new ArgumentException("Масиви повинні бути однакової довжини.");

T[] result = new T[a.Length];

for (int i = 0; i < a.Length; i++)

{

result[i] = SubtractValues(a[i], b[i]);

}

return result;

}

private static T SubtractValues<T>(T a, T b)

{

dynamic dynamicA = a;

dynamic dynamicB = b;

return dynamicA - dynamicB;

}

public class MatrixSubtractor

{

public static T[,] Subtract<T>(T[,] a, T[,] b)

{

if (a.GetLength(0) != b.GetLength(0) || a.GetLength(1) != b.GetLength(1))

throw new ArgumentException("Матриці повинні мати однакові розміри.");

T[,] result = new T[a.GetLength(0), a.GetLength(1)];

for (int i = 0; i < a.GetLength(0); i++)

{

for (int j = 0; j < a.GetLength(1); j++)

{

result[i, j] = SubtractValues(a[i, j], b[i, j]);

}

}

return result;

}

private static T SubtractValues<T>(T a, T b)

{

dynamic dynamicA = a;

dynamic dynamicB = b;

return dynamicA - dynamicB;

}

public static T Multiply<T>(T a, T b)

{

dynamic aa = a, bb = b;

return aa \* bb;

}

public class ArrayMultiplier

{

public static T[] Multiply<T>(T[] a, T[] b)

{

if (a.Length != b.Length)

throw new ArgumentException("Масиви повинні бути однакової довжини.");

T[] result = new T[a.Length];

for (int i = 0; i < a.Length; i++)

{

result[i] = MultiplyValues(a[i], b[i]);

}

return result;

}

private static T MultiplyValues<T>(T a, T b)

{

dynamic dynamicA = a;

dynamic dynamicB = b;

return dynamicA \* dynamicB;

}

public class MatrixMultiplier

{

public static T[,] Multiply<T>(T[,] a, T[,] b)

{

if (a.GetLength(1) != b.GetLength(0))

throw new ArgumentException("Кількість стовпців у першій матриці має дорівнювати кількості рядків у другій матриці.");

T[,] result = new T[a.GetLength(0), b.GetLength(1)];

for (int i = 0; i < a.GetLength(0); i++)

{

for (int j = 0; j < b.GetLength(1); j++)

{

result[i, j] = MultiplyValues(a, b, i, j);

}

}

return result;

}

private static T MultiplyValues<T>(T[,] a, T[,] b, int row, int col)

{

dynamic sum = 0;

for (int k = 0; k < a.GetLength(1); k++)

{

dynamic dynamicA = a[row, k];

dynamic dynamicB = b[k, col];

sum += dynamicA \* dynamicB;

}

return sum;

}

}

class Program

{

static void Main()

{

Console.WriteLine(MathOperations.Add(5, 3));

int[] array1 = { 1, 2, 3 };

int[] array2 = { 4, 5, 6 };

var resultArray = MathOperations.Add(array1, array2);

Console.WriteLine(string.Join(", ", resultArray));

int[,] matrix1 = { { 1, 2 }, { 3, 4 } };

int[,] matrix2 = { { 5, 6 }, { 7, 8 } };

var resultMatrix = MathOperations.Add(matrix1, matrix2);

Console.WriteLine($"[{resultMatrix[0, 0]}, {resultMatrix[0, 1]}]");

Console.WriteLine($"[{resultMatrix[1, 0]}, {resultMatrix[1, 1]}]");

}

}

}

}

}

}

}

}