**Практична робота № 6**

**Інтерфейси**

**Завдання 1**

interface IDemo

{

void Show();

double Length();

int X { get; }

int Y { get; }

int this[int i] { get; set; }

}

class DemoPoint : IDemo

{

protected int x;

protected int y;

public DemoPoint(int x, int y)

{

this.x = x; this.y = y;

}

public void Show()

{

Console.WriteLine($"Точка на плоскости: ({x}, {y})");

}

public double Length()

{

return Math.Sqrt(x \* x + y \* y);

}

public int X => x;

public int Y => y;

public int this[int i]

{

get

{

if (i == 0) return x;

if (i == 1) return y;

throw new Exception("недопустимое значение индекса");

}

set

{

if (i == 0) x = value;

else if (i == 1) y = value;

else throw new Exception("недопустимое значение индекса");

}

}

}

class DemoShape : DemoPoint, IDemo

{

protected int z;

public DemoShape(int x, int y, int z) : base(x, y)

{

this.z = z;

}

// реалізація методу, оголошеного в інтерфейсі, з приховуванням однойменного

public new void Show()

{

Console.WriteLine("точка в пространстве: ({0}, {1}, {2})", x, y,

z);

}

// реалізація методу, оголошеного в інтерфейсі, з приховуванням однойменного

public new double Length()

{

return Math.Sqrt(x \* x + y \* y + z \* z);

}

// реалізація індексатора, оголошеного в інтерфейсі, з приховуванням

// однойменного індексатора з базового класу

public new int this[int i]

{

get

{

if (i == 0) return x;

else if (i == 1) return y;

else if (i == 2) return z;

else throw new Exception("недопустимое значение индекса");

}

set

{

if (i == 0) x = value;

else if (i == 1) y = value;

else if (i == 2) z = value;

else throw new Exception("недопустимое значение индекса");

}

}

}

using System;

namespace Task\_1

{

class Program

{

static void Main()

{

// створення масиву інтерфейсних посилань

IDemo[] a = new IDemo[4];

// заповнення масиву

a[0] = new DemoPoint(0, 1);

a[1] = new DemoPoint(-3, 0);

a[2] = new DemoShape(3, 4, 0);

a[3] = new DemoShape(0, 5, 6);

// перегляд масиву

foreach (IDemo x in a)

{

x.Show();

Console.WriteLine("Dlina={0:f2}", x.Length());

Console.WriteLine("x=" + x.X);

x[1] += x[0];

Console.Write("новые коордынаты - ");

x.Show();

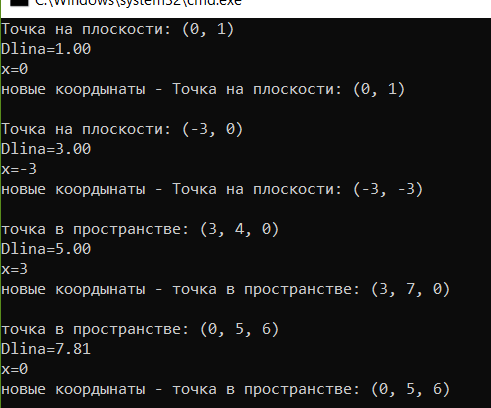
Console.WriteLine();

}

}

}

}



**Завдання 2-3**

namespace Lab6

{

interface IMeasurable

{

double Perimeter();

double Area();

}

}

namespace Lab6

{

interface ICircumcircleIncircle

{

double R { get; }

double r { get; }

}

}

using System;

namespace Lab6

{

class Square: IMeasurable, ICircumcircleIncircle, IComparable<Square>

{

private readonly double a;

public double Perimeter()

{

return 4 \* a;

}

public double Area()

{

return a \* a;

}

public double R

{

get => a / Math.Sqrt(2);

}

public double r

{

get => a/ 2;

}

public int CompareTo(Square a)

{

if (a == null) return -1;

return Perimeter().CompareTo(a.Perimeter());

}

public Square(double a)

{

this.a = a;

}

public double Sidea

{

get => a;

}

public override string ToString()

{

return $"Info about square : {nameof(a)}: {a}, {nameof(r)}: {r}, {nameof(R)}: {R}";

}

}

}

using System;

using System.Collections.Generic;

namespace Lab6

{

class Program

{

static void Main(string[] args)

{

var square = new Square(5);

Console.WriteLine(square.ToString());

Console.WriteLine($"P = {square.Perimeter()}");

Console.WriteLine($"S = {square.Area()}");

var squar = new List<Square>

{

square, new Square(33), new Square(11), new Square(99)

};

squar.Sort();

Console.WriteLine("All sorted squares:");

foreach (Square item in squar)

{

Console.WriteLine(item.ToString());

}

}

}

}

