Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 5 з дисципліни

«Основи програмування-2.

Методології програмування»

«Успадкування та поліморфізм»

Варіант 3

Виконав студент ІП-11 Веремчук Ігор Ігорович

(шифр, прізвище, ім'я, по батькові)

Перевірив

( прізвище, ім'я, по батькові)

Київ 2022

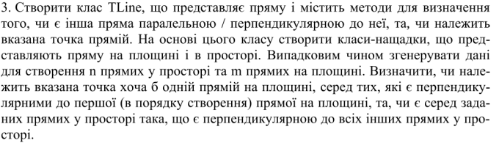
**Лабораторна робота №5**

**Успадкування та поліморфізм**

**Мета**

****

**Завдання(варіант 3):**



**Постановка задачі**

Для виконання поставленої задачі створимо абстрактні класи “Point”, з яких складається “Line”, їх двовивимірні та тривімірні представлення та клас для роботи з лініями “LineWorker”, кожен з них помістимо в окремий файл.

**Програма на С#**

Файл 1:

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Lab5Sharp  
{  
 internal static class Program  
 {  
 static void Main(string[] args)  
 {  
 Point2D point = new Point2D(new Random().Next(-5, 6), new Random().Next(-5, 6));  
 Line[] line2Ds = LineWorker.GetRandomLine2DArray(100);  
 var perpendicularToFirst = line2Ds.Where(l => l.IsPerpendicularTo(line2Ds.First())).ToArray();  
 var linesWithPoint = perpendicularToFirst.Where(l => l.ContainsPoint(point)).ToArray();  
  
 Line[] randomLine3Ds = LineWorker.GetRandomLine3DArray(3);  
 Line[] line3Ds = {new Line3D((1, 2, 3), (2, 3, 4)), new Line3D((1, 1, -2)), new Line3D((2, 2, -4))};  
  
 LineWorker.PrintLineArray(line2Ds, "2D Lines");  
 LineWorker.PrintLineArray(perpendicularToFirst, "Lines, perpendicular to first");  
 LineWorker.PrintLineArray(linesWithPoint, $"Lines, perpendicular to first with point: {point}");  
 LineWorker.PrintLineArray(randomLine3Ds, "Random 3D Lines");  
 Console.WriteLine($"Random 3D Line Perpendicular to all:\n{LineWorker.PerpendicularToAll(randomLine3Ds)}");  
 LineWorker.PrintLineArray(line3Ds, "3D Lines");  
 Console.WriteLine($"3D Line Perpendicular to all:\n{LineWorker.PerpendicularToAll(line3Ds)}");  
 Console.ReadLine();  
 }  
 }  
}

Файл 2:

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Lab5Sharp  
{  
 internal static class LineWorker  
 {  
 public static Line? PerpendicularToAll(Line[] lines)  
 {  
 for (int indToCheck = 0; indToCheck < lines.Length; indToCheck++)  
 {  
 Line perpLine = lines[indToCheck];  
 bool perpendicular =  
 lines.Where((l, j) => indToCheck != j).All(l => lines[indToCheck].IsPerpendicularTo(l));  
 if (perpendicular) return perpLine;  
 }  
  
 return null;  
 }  
  
 public static void PrintLineArray(Line[] lines, string prePrint = "")  
 {  
 Console.WriteLine(prePrint);  
 Array.ForEach(lines, Console.WriteLine);  
 Console.WriteLine();  
 }  
  
 public static Line2D GetRandomLine2D(int randMin = -5, int randMax = 6)  
 {  
 Random random = new Random();  
 var a = (random.Next(randMin, randMax), random.Next(randMin, randMax));  
 var b = (random.Next(randMin, randMax), random.Next(randMin, randMax));  
 return new Line2D(a, b);  
 }  
  
 public static Line3D GetRandomLine3D(int randMin = -5, int randMax = 6)  
 {  
 Random random = new Random();  
 var a = (random.Next(randMin, randMax), random.Next(randMin, randMax), random.Next(randMin, randMax));  
 var b = (random.Next(randMin, randMax), random.Next(randMin, randMax), random.Next(randMin, randMax));  
 return new Line3D(a, b);  
 }  
  
 public static Line2D[] GetRandomLine2DArray(int size)  
 {  
 var lines = new Line2D[size];  
 for (int i = 0; i < size; i++)  
 {  
 lines[i] = GetRandomLine2D();  
 }  
  
 return lines;  
 }  
  
 public static Line3D[] GetRandomLine3DArray(int size)  
 {  
 var lines = new Line3D[size];  
 for (int i = 0; i < size; i++)  
 {  
 lines[i] = GetRandomLine3D();  
 }  
  
 return lines;  
 }  
 }  
}

Файл 3:

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Lab5Sharp  
{  
 internal abstract class Point { }  
  
 internal class Point2D : Point  
 {  
 public int X { get; protected set; }  
 public int Y { get; protected set; }  
  
 public Point2D(int x = 0, int y = 0)  
 {  
 X = x;  
 Y = y;  
 }  
  
 public Point2D((int x, int y) coordinates)  
 {  
 X = coordinates.x;  
 Y = coordinates.y;  
 }  
  
 public override string ToString() => $"({X}, {Y})";  
  
  
 public static bool operator ==(Point2D point1, Point2D point2)  
 {  
 return point1.X == point2.X && point1.Y == point2.Y;  
 }  
  
 public static bool operator !=(Point2D point1, Point2D point2)  
 {  
 return point1.X != point2.X || point1.Y != point2.Y;  
 }  
   
 }  
  
 internal class Point3D : Point2D  
 {  
 public int Z { get; protected set; }  
  
 public Point3D(int x = 0, int y = 0, int z = 0) : base(x, y)  
 {  
 Z = z;  
 }  
  
 public Point3D((int x, int y, int z) coordinates)  
 {  
 X = coordinates.x;  
 Y = coordinates.y;  
 Z = coordinates.z;  
 }  
  
 public override string ToString() => $"({X}, {Y}, {Z})";  
   
   
 public static bool operator ==(Point3D point1, Point3D point2)  
 {  
 return point1.X == point2.X && point1.Y == point2.Y && point1.Z == point2.Z;  
 }  
  
 public static bool operator !=(Point3D point1, Point3D point2)  
 {  
 return point1.X != point2.X || point1.Y != point2.Y || point1.Z != point2.Z;  
 }  
 }  
}

Файл 4:

namespace Lab5Sharp  
{  
 internal abstract class Line  
 {  
 public abstract bool IsParallelTo(Line line);  
 public abstract bool IsPerpendicularTo(Line line);  
 public abstract bool ContainsPoint(Point point);  
 }  
}

Файл 5:

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Lab5Sharp  
{  
 internal class Line2D : Line  
 {  
 public Point2D A { get; private set; }  
 public Point2D B { get; private set; }  
 public Point2D VectorCoord => new Point2D(B.X - A.X, B.Y - A.Y);  
  
 public Line2D()  
 {  
 A = new Point2D();  
 B = new Point2D(1);  
 }  
  
 public Line2D(Point2D b)  
 {  
 A = new Point2D();  
 B = b;  
 }  
  
 public Line2D(Point2D a, Point2D b) : this(b)  
 {  
 if (a != b) A = a;  
 }  
  
 public Line2D((int x, int y) b)  
 {  
 A = new Point2D();  
 B = new Point2D(b);  
 }  
  
 public Line2D((int x, int y) a, (int x, int y) b) : this(b)  
 {  
 if(a != b) A = new Point2D(a);  
 }  
  
 public override bool IsParallelTo(Line line)  
 {  
 return line is Line2D line2D && line2D.VectorCoord.X != 0 && line2D.VectorCoord.Y != 0 &&  
 (decimal) VectorCoord.X / line2D.VectorCoord.X == (decimal) VectorCoord.Y / line2D.VectorCoord.Y;  
 }  
  
 public override bool IsPerpendicularTo(Line line)  
 {  
 return line is Line2D line2D &&  
 VectorCoord.X \* line2D.VectorCoord.X + VectorCoord.Y \* line2D.VectorCoord.Y == 0;  
 }  
  
 public override bool ContainsPoint(Point point)  
 {  
 if (point is not Point2D point2D) return false;  
 Line2D lineWithPoint = new Line2D(A, point2D);  
 return IsParallelTo(lineWithPoint);  
 }  
  
 public override string ToString() => $"A: {A}\tB: {B}";  
 }  
}

Файл 6:

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace Lab5Sharp  
{  
 internal class Line3D : Line  
 {  
 public Point3D A { get; private set; }  
 public Point3D B { get; private set; }  
 public Point3D VectorCoord => new Point3D(B.X - A.X, B.Y - A.Y, B.Z - A.Z);  
  
 public Line3D()  
 {  
 A = new Point3D();  
 B = new Point3D(1);  
 }  
  
 public Line3D(Point3D b)  
 {  
 A = new Point3D();  
 B = b;  
 }  
  
 public Line3D(Point3D a, Point3D b) : this(b)  
 {  
 if (a != b) A = a;  
 }  
  
 public Line3D((int x, int y, int z) b)  
 {  
 A = new Point3D();  
 B = new Point3D(b);  
 }  
  
 public Line3D((int x, int y, int z) a, (int x, int y, int z) b) : this(b)  
 {  
 if (a != b) A = new Point3D(a);  
 }  
  
 public override bool IsParallelTo(Line line)  
 {  
 return line is Line3D line3D &&  
 line3D.VectorCoord.X != 0 && line3D.VectorCoord.Y != 0 && line3D.VectorCoord.Z != 0 &&  
 (decimal) VectorCoord.X / line3D.VectorCoord.X == (decimal) VectorCoord.Y / line3D.VectorCoord.Y &&  
 (decimal) VectorCoord.X / line3D.VectorCoord.X == (decimal) VectorCoord.Z / line3D.VectorCoord.Z;  
 }  
  
 public override bool IsPerpendicularTo(Line line)  
 {  
 return line is Line3D line3D &&  
 VectorCoord.X \* line3D.VectorCoord.X + VectorCoord.Y \* line3D.VectorCoord.Y +  
 VectorCoord.Z \* line3D.VectorCoord.Z == 0;  
 }  
  
 public override bool ContainsPoint(Point point)  
 {  
 if (point is not Point3D point3D) return false;  
 Line3D lineWithPoint = new Line3D(A, point3D);  
 return IsParallelTo(lineWithPoint);  
 }  
  
 public override string ToString() => $"A: {A}\tB: {B}";  
 }  
}

**Програма на python**

Файл 1:

from os import system  
from line\_worker import \*  
  
  
def main():  
 point = Point2D(randint(-5, 6), randint(-5, 6))  
 line2d\_list = get\_random\_line2d\_list(50)  
 random\_line3d = get\_random\_line3d\_list(5)  
 perpendicular\_to\_first = [l for l in line2d\_list if l.is\_perpendicular\_to(line2d\_list[0])]  
 lines\_with\_point = [l for l in perpendicular\_to\_first if l.contains\_point(point)]  
 line3d\_list = [Line3D(Point3D(1, 2, 3), Point3D(2, 3, 4)),  
 Line3D(Point3D(0, 0, 0), Point3D(1, 1, -2)), Line3D(Point3D(0, 0, 0), Point3D(2, 2, -4))]  
  
 print\_list(line2d\_list, "2D Lines: ")  
 print\_list(perpendicular\_to\_first, "Lines, perpendicular to first")  
 print\_list(lines\_with\_point, f"Lines, perpendicular to first with point: {point}")  
 print\_list(random\_line3d, "Random 3D Lines: ")  
 print(f"Random 3D Line Perpendicular to all:\n{perpendicular\_to\_all(random\_line3d)}")  
 print\_list(line3d\_list, "3D Lines: ")  
 print(f"3D Line Perpendicular to all:\n{perpendicular\_to\_all(line3d\_list)}")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()  
 system('pause')

Файл 2:

from Line import \*  
from Point import \*  
from random import randint  
  
  
def print\_list(lst, pre\_print=""):  
 print(pre\_print)  
 for item in lst:  
 print(item)  
 print()  
  
  
def perpendicular\_to\_all(lines: list[Line]):  
 for ind\_to\_check in range(len(lines)):  
 perp\_line = lines[ind\_to\_check]  
 perpendicular = True  
 for j in range(len(lines)):  
 if ind\_to\_check != j:  
 if not lines[ind\_to\_check].is\_perpendicular\_to(lines[j]):  
 perpendicular = False  
 break  
 if perpendicular:  
 return perp\_line  
  
  
def get\_random\_line2d(rand\_min=-5, rand\_max=6) -> Line2D:  
 a = Point2D(randint(rand\_min, rand\_max), randint(rand\_min, rand\_max))  
 b = Point2D(randint(rand\_min, rand\_max), randint(rand\_min, rand\_max))  
 return Line2D(a, b)  
  
  
def get\_random\_line3d(rand\_min=-5, rand\_max=6) -> Line3D:  
 a = Point3D(randint(rand\_min, rand\_max), randint(rand\_min, rand\_max), randint(rand\_min, rand\_max))  
 b = Point3D(randint(rand\_min, rand\_max), randint(rand\_min, rand\_max), randint(rand\_min, rand\_max))  
 return Line3D(a, b)  
  
  
def get\_random\_line2d\_list(size) -> list[Line2D]:  
 return [get\_random\_line2d() for i in range(size)]  
  
  
def get\_random\_line3d\_list(size) -> list[Line3D]:  
 return [get\_random\_line3d() for i in range(size)]

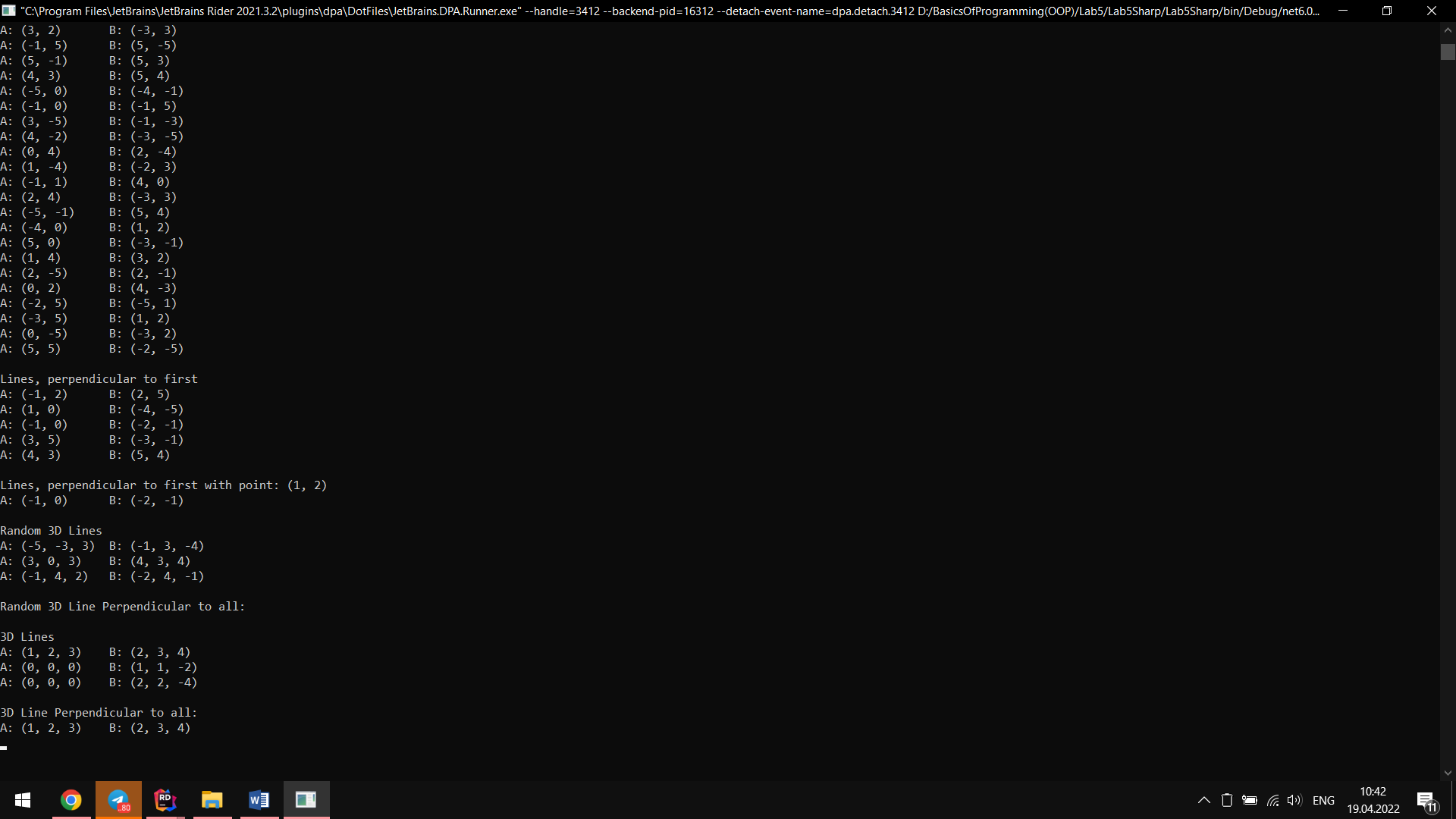
Файл 3:

from abc import ABC  
  
  
class Point(ABC):  
 pass  
  
  
class Point2D(Point):  
 def \_\_init\_\_(self, x=0, y=0):  
 self.x = x  
 self.y = y  
  
 def \_\_str\_\_(self):  
 return f"({self.x}, {self.y})"  
  
 def \_\_eq\_\_(self, other):  
 return self.x == other.x and self.y == other.y  
  
  
class Point3D(Point2D):  
 def \_\_init\_\_(self, x=0, y=0, z=0):  
 super().\_\_init\_\_(x, y)  
 self.z = z  
  
 def \_\_str\_\_(self):  
 return f"({self.x}, {self.y}, {self.z})"  
  
 def \_\_eq\_\_(self, other):  
 return self.x == other.x and self.y == other.y and self.z == other.z

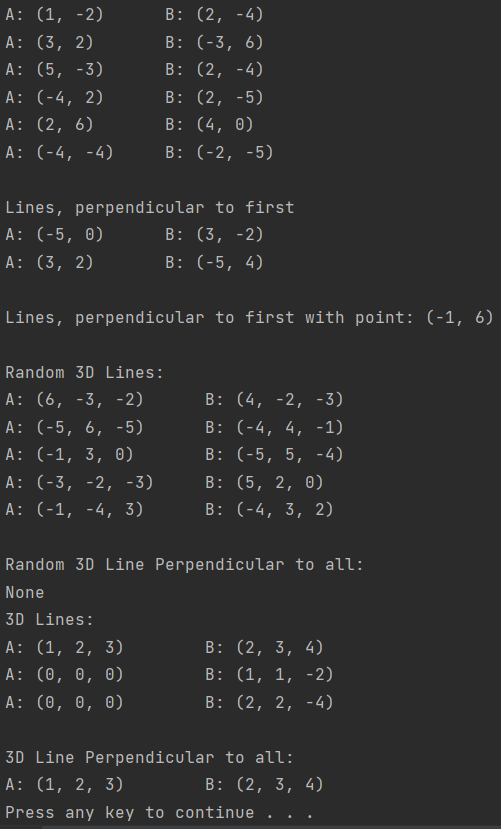
Файл 4:

from abc import ABC, abstractmethod  
from Point import \*  
  
  
class Line(ABC):  
 @abstractmethod  
 def is\_parallel\_to(self, line):  
 pass  
  
 @abstractmethod  
 def is\_perpendicular\_to(self, line):  
 pass  
  
 @abstractmethod  
 def contains\_point(self, point):  
 pass  
  
  
class Line2D(Line):  
 def \_\_init\_\_(self, a: Point2D, b: Point2D):  
 if a != b:  
 self.A = a  
 self.B = b  
 self.vector\_coord = Point2D(b.x - a.x, b.y - a.y)  
  
 def is\_parallel\_to(self, line):  
 line: Line2D = line  
 return line.vector\_coord.x != 0 and line.vector\_coord.y != 0 and \  
 self.vector\_coord.x / line.vector\_coord.x == self.vector\_coord.y / line.vector\_coord.y  
  
 def is\_perpendicular\_to(self, line):  
 line: Line2D = line  
 return self.vector\_coord.x \* line.vector\_coord.x + self.vector\_coord.y \* line.vector\_coord.y == 0  
  
 def contains\_point(self, point):  
 point: Point2D = point  
 line\_with\_point = Line2D(self.A, point)  
 return self.is\_parallel\_to(line\_with\_point)  
  
 def \_\_str\_\_(self):  
 return f"A: {self.A}\t\tB: {self.B}"  
  
  
class Line3D(Line):  
 def \_\_init\_\_(self, a: Point3D, b: Point3D):  
 if a != b:  
 self.A = a  
 self.B = b  
 self.vector\_coord = Point3D(b.x - a.x, b.y - a.y, b.z - a.z)  
  
 def is\_parallel\_to(self, line):  
 line: Line3D = line  
 return line.vector\_coord.x != 0 and line.vector\_coord.y != 0 and line.vector\_coord.z != 0 and \  
 self.vector\_coord.x / line.vector\_coord.x == \  
 self.vector\_coord.y / line.vector\_coord.y == self.vector\_coord.z / line.vector\_coord.z  
  
 def is\_perpendicular\_to(self, line):  
 line: Line3D = line  
 return self.vector\_coord.x \* line.vector\_coord.x + self.vector\_coord.y \* line.vector\_coord.y + \  
 self.vector\_coord.z \* line.vector\_coord.z == 0  
  
 def contains\_point(self, point):  
 point: Point3D = point  
 line\_with\_point = Line3D(self.A, point)  
 return self.is\_parallel\_to(line\_with\_point)  
  
 def \_\_str\_\_(self):  
 return f"A: {self.A}\t\tB: {self.B}"

**Виконання на C#**



**Виконання на python**



**Висновок:**

На лабораторній роботі я опанував технологію роботи з класами та об’єктами, наслідування та поліморфізму, виконавши поставлене завдання на мовах програмування C# та python.