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»

«Наслідування та поліморфізм»

Варіант 3

Виконав студент ІП-14 Басан Антон Сергійович

(шифр, прізвище, ім'я, по батькові)

Перевірив Вітковська Ірина Іванівна

( прізвище, ім'я, по батькові)

Київ 202 2

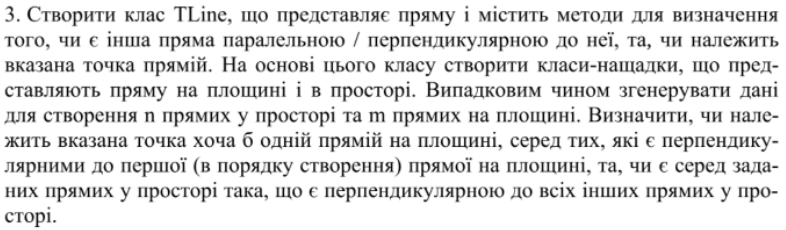
**Лабораторна робота 5**

**Наслідування та поліморфізм**

*Варіант 3*

*Мета:* вивчити механізми створення і використання класів та об’єктів

*Задача:*



**Розв’язання**

*Постановка задачі*

Результатом розв’язку задачі є перевірка прямих на площині та в просторі на вказані властивості, що є об’єктами відповідних класів-наслідників класу Line. Перевірка заданих умов відбуватиметься за допомогою методів створеного класу.

*Код та його випробування (С#)*

Program.cs

using System;  
using static Lab5.Func;  
  
namespace Lab5  
{  
 internal class Program  
 {  
 public static void Main(string[] args)  
 {  
 */\*  
 Random rand = new Random();  
 int n = 4;  
 Line2D[] lines2D = new Line2D[n];  
 for (int i = 0; i < n; i++)  
 lines2D[i] = new Line2D(new int[]{rand.Next(0, 10), rand.Next(0, 10)},   
 new int[]{rand.Next(0, 10), rand.Next(0, 10)});  
 \*/* Line2D[] lines2D = {  
 new Line2D(new int[]{4, 5}, new int[]{3, 2}),  
 new Line2D(new int[]{1, 6}, new int[]{5, 7}),  
 new Line2D(new int[]{8, 5}, new int[]{5, 6}),  
 new Line2D(new int[]{1, 8}, new int[]{2, 3})  
 };  
 LinesOut(lines2D, "\nGenerated 2D lines:");  
 int[] point2D = {2, 7};  
 int index = 0;  
 if(PointCheck(lines2D, point2D, ref index))  
 Console.WriteLine( $"Point M {PointString(point2D)} belongs to line #{index + 1}: {lines2D[index]}, " +  
 $"which is perpendicular to line #1: {lines2D[0]}");  
 */\*  
 Line3D[] lines3D = new Line3D[n];  
 for (int i = 0; i < n; i++)  
 lines3D[i] = new Line3D(new int[]{rand.Next(0, 10), rand.Next(0, 10), rand.Next(0, 10)},   
 new int[]{rand.Next(0, 10), rand.Next(0, 10), rand.Next(0, 10)});  
 \*/* Line3D[] lines3D = {  
 new Line3D(new int[]{4, 5, 2}, new int[]{3, 2, 3}),  
 new Line3D(new int[]{8, 5, 2}, new int[]{5, 6, 2}),  
 new Line3D(new int[]{4, 7, 2}, new int[]{3, 4, 3}),  
 new Line3D(new int[]{4, 3, 2}, new int[]{3, 0, 3}),  
 };  
 LinesOut(lines3D, "\nGenerated 3D lines:");  
 if(LineCheck(lines3D, ref index))  
 Console.WriteLine($"Line #{index + 1}: {lines3D[index]} is perpendicular to all other lines");  
 }  
 }  
}

Func.cs

using System;  
using System.Linq;  
  
namespace Lab5  
{  
 public class Func  
 {  
 *//перевірка прямих масиву на першу вказану властивість* public static bool PointCheck(Line[] lines, int[] point, ref int index)  
 {  
 for (int i = 1; i < lines.Length; i++)  
 if (lines[i].IsPerpendicular(lines[0]))  
 if (lines[i].PointBelongs(point))  
 {  
 index = i;  
 return true;  
 }  
 return false;  
 }  
   
 *//перевірка прямих масиву на другу вказану властивість* public static bool LineCheck(Line[] lines, ref int index)  
 {  
 for (int i = 0; i < lines.Length; i++)  
 {  
 bool exists = false;  
 foreach (var line in lines)  
 {  
 if (lines[i] == line) continue;  
 exists = lines[i].IsPerpendicular(line);  
 }  
 if (exists)  
 {  
 index = i;  
 return true;  
 }  
 }  
 return false;  
 }  
   
 *//виведення масиву прямих* public static void LinesOut(Line[] lines, string comment)  
 {  
 Console.WriteLine(comment);  
 for (int i = 0; i < lines.Length; i++)  
 Console.WriteLine($"Line #{i + 1}:\t{lines[i]}");  
 Console.WriteLine();  
 }  
   
 *//перетворення координат точки у рядок* public static string PointString(int[] point)  
 {  
 return $"({string.Join("; ", point.Select(x => x.ToString()).ToArray())})";  
 }  
 }  
}

Line.cs

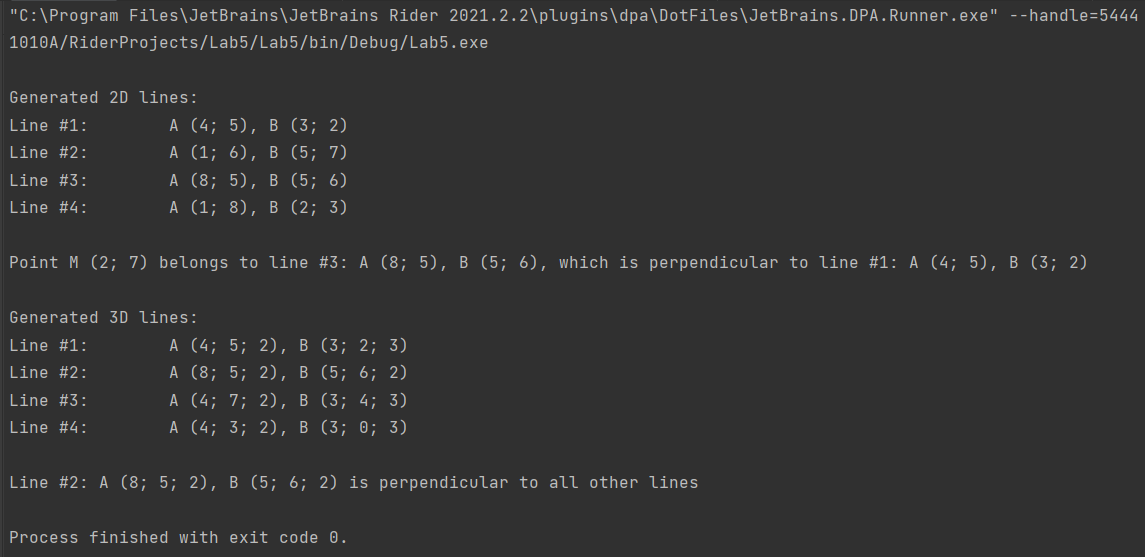
using System.Linq;  
  
namespace Lab5  
{  
 public abstract class Line  
 {  
 protected readonly int[] coordinatesA;  
 protected readonly int[] coordinatesB;  
  
 *//конструктор з параметрами* protected Line(int[] coordinatesA, int[] coordinatesB)  
 {  
 this.coordinatesA = coordinatesA;  
 this.coordinatesB = coordinatesB;  
 }  
   
 *//зведення прямої до вектора* public int[] VectorCoordinates()  
 {  
 int[] vectorCoordinates = new int[coordinatesA.Length];  
 for (int i = 0; i < coordinatesA.Length; i++)  
 vectorCoordinates[i] = coordinatesB[i] - coordinatesA[i];  
 return vectorCoordinates;  
 }  
   
 *//представлення атрибутів об'єкта у вигляді рядка* public override string ToString()  
 {  
 return $"A ({string.Join( "; ", coordinatesA.Select(x => x.ToString()).ToArray())}), " +  
 $"B ({string.Join( "; ", coordinatesB.Select(x => x.ToString()).ToArray())})";  
 }  
   
 *//перевірка паралельності двох ліній* public abstract bool IsParallel(Line line);  
   
 *//перевірка перпендикулярності двох ліній* public abstract bool IsPerpendicular(Line line);  
   
 *//перевірка належності точки до прямої* public abstract bool PointBelongs(int[] coordinatesP);  
 }  
}

Line2D.cs

namespace Lab5  
{  
 public class Line2D : Line  
 {  
 public Line2D(int[] coordinatesA, int[] coordinatesB)   
 : base(coordinatesA, coordinatesB) { }  
   
 public override bool IsParallel(Line line)  
 {  
 return VectorCoordinates()[0] / line.VectorCoordinates()[0] ==   
 VectorCoordinates()[1] / line.VectorCoordinates()[1];  
 }  
   
 public override bool IsPerpendicular(Line line)  
 {  
 return VectorCoordinates()[0] \* line.VectorCoordinates()[0] +  
 VectorCoordinates()[1] \* line.VectorCoordinates()[1] == 0;  
 }  
   
 public override bool PointBelongs(int[] coordinatesP)  
 {  
 return IsParallel(new Line2D(coordinatesA, coordinatesP));  
 }  
 }  
}

Line3D.cs

namespace Lab5  
{  
 public class Line3D : Line  
 {  
 public Line3D(int[] coordinatesA, int[] coordinatesB)   
 : base(coordinatesA, coordinatesB) { }  
  
 public override bool IsParallel(Line line)  
 {  
 return VectorCoordinates()[0] / line.VectorCoordinates()[0] ==   
 VectorCoordinates()[1] / line.VectorCoordinates()[1] &&   
 VectorCoordinates()[0] / line.VectorCoordinates()[0] ==   
 VectorCoordinates()[2] / line.VectorCoordinates()[2];  
 }  
   
 public override bool IsPerpendicular(Line line)  
 {  
 return VectorCoordinates()[0] \* line.VectorCoordinates()[0] +  
 VectorCoordinates()[1] \* line.VectorCoordinates()[1] +  
 VectorCoordinates()[2] \* line.VectorCoordinates()[2]== 0;  
 }  
   
 public override bool PointBelongs(int[] coordinatesP)  
 {  
 return IsParallel(new Line3D(coordinatesA, coordinatesP));  
 }  
 }  
}

**

*Код та його випробування (Python)*

main.py

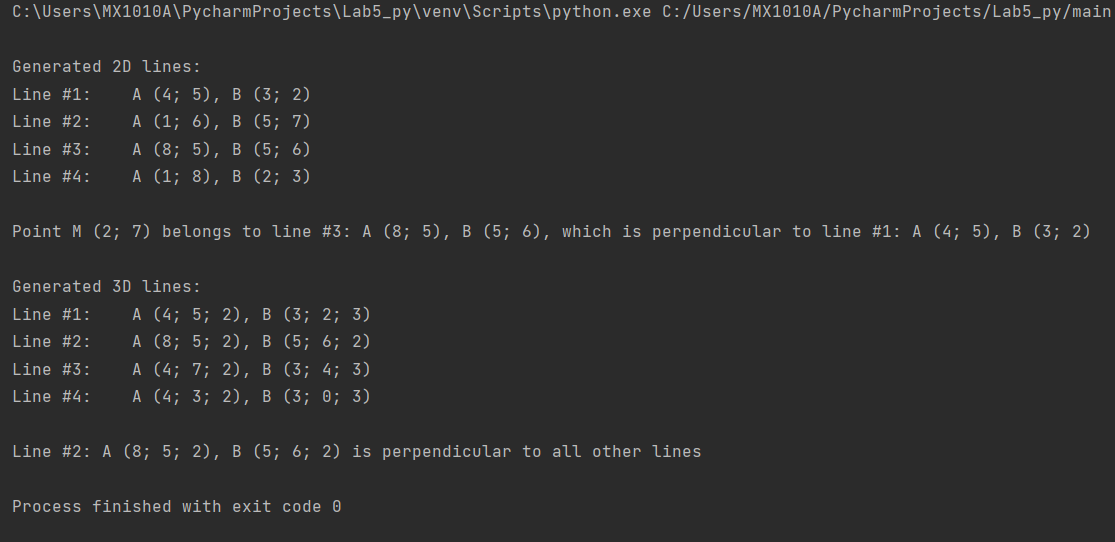
import random as rand  
from classes import Line2D, Line3D  
import func  
  
'''  
n = 4  
lines2D = []  
for i in range(n - 1):  
 lines2D.append(Line2D([rand.randint(0, 10), rand.randint(0, 10)], [rand.randint(0, 10), rand.randint(0, 10)]))  
'''  
  
lines2D = [  
 Line2D([4, 5], [3, 2]),  
 Line2D([1, 6], [5, 7]),  
 Line2D([8, 5], [5, 6]),  
 Line2D([1, 8], [2, 3])  
]  
func.lines\_out(lines2D, "\nGenerated 2D lines:")  
point2D = [2, 7]  
result = func.point\_check(lines2D, point2D)  
if result[0]:  
 print(f"Point M {func.point\_string(point2D)} belongs to line #{result[1] + 1}: {lines2D[result[1]]}, "  
 f"which is perpendicular to line #1: {lines2D[0]}")  
  
'''  
lines3D = []  
for i in range(n - 1):  
 lines3D.append(Line3D([rand.randint(0, 10), rand.randint(0, 10), rand.randint(0, 10)],  
 [rand.randint(0, 10), rand.randint(0, 10), rand.randint(0, 10)]))  
'''  
  
lines3D = [  
 Line3D([4, 5, 2], [3, 2, 3]),  
 Line3D([8, 5, 2], [5, 6, 2]),  
 Line3D([4, 7, 2], [3, 4, 3]),  
 Line3D([4, 3, 2], [3, 0, 3]),  
]  
func.lines\_out(lines3D, "\nGenerated 3D lines:")  
result = func.line\_check(lines3D)  
if result[0]:  
 print(f"Line #{result[1] + 1}: {lines3D[result[1]]} is perpendicular to all other lines")

func.py

# перевірка прямих списку на першу вказану властивість  
def point\_check(lines, point):  
 for i in range(len(lines)):  
 if lines[i].is\_perpendicular(lines[0]):  
 if lines[i].point\_belongs(point):  
 index = i  
 return True, index  
 return False, 0  
  
# перевірка прямих списку на другу вказану властивість  
def line\_check(lines):  
 for i in range(len(lines)):  
 exists = False  
 for line in lines:  
 if lines[i] == line:  
 continue  
 exists = lines[i].is\_perpendicular(line)  
 if exists:  
 index = i  
 return True, index  
 return False, 0  
  
# виведення масиву прямих  
def lines\_out(lines, comment):  
 print(comment)  
 for i in range(len(lines)):  
 print(f"Line #{i + 1}:\t{lines[i]}")  
 print()  
  
  
# перетворення координат точки у рядок  
def point\_string(point):  
 point\_str = [str(i) for i in point]  
 return f"({'; '.join(point\_str)})"

classes.py

import abc  
  
  
class Line:  
 coordinatesA = []  
 coordinatesB = []  
  
 # конструктор з параметрами  
 def \_\_init\_\_(self, coordinates\_a, coordinates\_b):  
 self.coordinatesA = coordinates\_a  
 self.coordinatesB = coordinates\_b  
  
 # зведення прямої до вектора  
 def vector\_coordinates(self):  
 vector\_coords = []  
 for i in range(len(self.coordinatesA)):  
 vector\_coords.append(self.coordinatesB[i] - self.coordinatesA[i])  
 return vector\_coords  
  
 # представлення атрибутів об'єкта у вигляді рядка  
 def \_\_str\_\_(self):  
 coords\_a = [str(i) for i in self.coordinatesA]  
 coords\_b = [str(i) for i in self.coordinatesB]  
 return f"A ({'; '.join(coords\_a)}), B ({'; '.join(coords\_b)})"  
  
 # перевірка паралельності двох ліній  
 @abc.abstractmethod  
 def is\_parallel(self, line):  
 pass  
  
 # перевірка перпендикулярності двох ліній  
 @abc.abstractmethod  
 def is\_perpendicular(self, line):  
 pass  
  
 # перевірка належності точки до прямої  
 @abc.abstractmethod  
 def point\_belongs(self, line):  
 pass  
  
  
class Line2D(Line):  
 def is\_parallel(self, line):  
 return self.vector\_coordinates()[0] / line.vector\_coordinates()[0] == \  
 self.vector\_coordinates()[1] / line.vector\_coordinates()[1]  
  
 def is\_perpendicular(self, line):  
 return self.vector\_coordinates()[0] \* line.vector\_coordinates()[0] + \  
 self.vector\_coordinates()[1] \* line.vector\_coordinates()[1] == 0  
  
 def point\_belongs(self, coordinates\_p):  
 return self.is\_parallel(Line2D(self.coordinatesA, coordinates\_p))  
  
  
class Line3D(Line):  
 def is\_parallel(self, line):  
 return self.vector\_coordinates()[0] / line.vector\_coordinates()[0] == \  
 self.vector\_coordinates()[1] / line.vector\_coordinates()[1] and \  
 self.vector\_coordinates()[0] / line.vector\_coordinates()[0] == \  
 self.vector\_coordinates()[2] / line.vector\_coordinates()[2]  
  
 def is\_perpendicular(self, line):  
 return self.vector\_coordinates()[0] \* line.vector\_coordinates()[0] + \  
 self.vector\_coordinates()[1] \* line.vector\_coordinates()[1] + \  
 self.vector\_coordinates()[2] \* line.vector\_coordinates()[2] == 0  
  
 def point\_belongs(self, coordinates\_p):  
 return self.is\_parallel(Line3D(self.coordinatesA, coordinates\_p))

******

***Висновок:*** під час виконання завдання були досліджені механізми створення класів та об’єктів, крім того були набуті практичні навички розробки програм, які працюють з наслідуванням класів (а саме програми для перевірки взаємного положення прямих на площині та в просторі) на компільованій мові С# та інтерпретованій Python. Результати обох програм та власноручних обчислень збігаються.