

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 № 4
з дисципліни «Основи програмування – 2.
Методології програмування»

«Успадкування та поліморфізм»

Варіант 13

Виконав студент ПІ-13 Жмайло Дмитро Олександрович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вечерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота 4

Успадкування та поліморфізм

Варіант 13

Код програми

Спроектувати клас **“Рухома матеріальна точка”**, який представляє точку, що рухається в певному напрямку і містить методи для визначення координати точки в заданий момент часу t та обчислення відстані від неї до іншої точки. На основі цього класу створити класи-нащадки **“Рухома матеріальна точка x ”**, яка рухається по прямій і її координата визначається як $x = x_0 + a_1 \sin(t)$, $y = 0$, та **“Рухома матеріальна точка (x, y) ”**, яка рухається по площині і її координати визначаються як $x = x_0 + a_1 \sin(t)$, $y = y_0 + a_2 \cos(t)$. Створити k об’єктів класу **“Рухома матеріальна точка x ”** та n об’єктів класу **“Рухома матеріальна точка (x, y) ”** (дані згенерувати випадковим чином). Визначити найбільшу відстань між рухомими матеріальними точками у заданий користувачем момент часу t .

C#

Program.cs

```
using System;

namespace Lab4CSharp
{
    class Program
    {
        static void Main(string[] args)
        {
            int k = Operations.InputNumber("k");
            int n = Operations.InputNumber("n");

            MovingPointParticleX[] pointXArray = new MovingPointParticleX[k];
            for (int i = 0; i < k; i++)
            {
                pointXArray[i] = new MovingPointParticleX();
            }

            MovingPointParticleXY[] pointXYArray = new MovingPointParticleXY[n];
```

```

    for (int j = 0; j < n; j++)
    {
        pointXYArray[j] = new MovingPointParticleXY();
    }

    Console.WriteLine();
    Console.WriteLine("Generated objects in xArray: ");
    Operations.ShowObjects(pointXArray);
    Console.WriteLine();

    Console.WriteLine("Generated objects in xyArray: ");
    Operations.ShowObjects(pointXArray);
    Console.WriteLine();

    double t = Operations.InputTime();

    Console.WriteLine();
    Console.WriteLine("X coordinates of objects in xArray: ");
    Operations.ShowCoordinates(pointXArray, t);
    Console.WriteLine();

    Console.WriteLine("X and Y coordinates of objects in xyArray: ");
    Operations.ShowCoordinates(pointXYArray, t);
    Console.WriteLine();

    double maxDistanceX = Operations.ReturnMaxDistance(pointXArray, out int
firstX, out int secondX, t);

    //double maxDistanceX = Operations.ReturnMaxDistanceX(pointXArray, out int
firstX, out int secondX);

    Console.WriteLine($"Max distance in xArray is {Math.Round(maxDistanceX, 3)}
(between {firstX + 1}) and {secondX + 1}) points");

    double maxDistanceXY = Operations.ReturnMaxDistance(pointXYArray, out int
firstXY, out int secondXY, t);

    //double maxDistanceXY = Operations.ReturnMaxDistanceXY(pointXYArray, out int
firstXY, out int secondXY);

    Console.WriteLine($"Max distance in xyArray is {Math.Round(maxDistanceXY, 3)}
(between {firstXY + 1}) and {secondXY + 1}) points");
}

```

}

}

MovingPointParticle.cs

```
using System;

using System.Collections.Generic;

using System.Text;

namespace Lab4CSharp
{
    abstract class MovingPointParticle
    {
        public abstract double GetCoordinates(double t, out double y);
        public abstract double GetDistance(double x2, double y2);
        public abstract void DisplayObject();
        public abstract void DisplayCoordinates(double t);
    }
}
```

MovingPointParticleX.cs

```
using System;

using System.Collections.Generic;

using System.Text;

namespace Lab4CSharp
{
    class MovingPointParticleX : MovingPointParticle
    {
        private double x;
        private double x0;
        private double a1;
        private double y;

        public double GetX0()
        {
            return x0;
        }

        public double GetA1()
        {
            return a1;
        }

        public double GetX()
        {
            return x;
        }

        public double GetY()
        {
            return y;
        }

        public MovingPointParticleX()
        {

```

```

        int minRandX = -100;

        int maxRandX = 100;

        int minRandA = -10;

        int maxRandA = 10;

        int precision = 10;

        Random rand = new Random();

        x = 0;

        y = 0;

        x0 = Convert.ToDouble(rand.Next(minRandX * precision, maxRandX * precision +
1) / Convert.ToDouble(precision));

        a1 = Convert.ToDouble(rand.Next(minRandA * precision, maxRandA * precision +
1) / Convert.ToDouble(precision));

    }

    public override double GetCoordinates(double t, out double y)
    {
        y = 0;

        x = x0 + a1 * Math.Sin(t);

        return x;
    }

    public override double GetDistance(double x2, double y2)
    {
        return Math.Abs(x2 - x);
    }

    public override void DisplayObject()
    {
        Console.WriteLine($"{ta1 = \t{a1}\tx0 = \t{x0}");
    }

    public override void DisplayCoordinates(double t)
    {
        Console.WriteLine($"{tx is: \t{Math.Round(GetCoordinates(t, out y), 3)}");
    }
}
}

```

MovingPointParticleXY.cs

```
using System;

using System.Collections.Generic;

using System.Text;

namespace Lab4CSharp
{
    class MovingPointParticleXY : MovingPointParticle
    {
        private double x;
        private double y;
        private double x0;
        private double y0;
        private double a1;
        private double a2;

        public double GetX0()
        {
            return x0;
        }

        public double GetY0()
        {
            return y0;
        }

        public double GetA1()
        {
            return a1;
        }

        public double GetA2()
        {
            return a2;
        }

        public double GetX()
        {

```



```

        return x;
    }

    public double GetY()
    {
        return y;
    }

    public MovingPointParticleXY()
    {
        int minRandX = -100;
        int maxRandX = 100;
        int minRandY = -100;
        int maxRandY = 100;
        int minRandA1 = -10;
        int maxRandA1 = 10;
        int minRandA2 = -10;
        int maxRandA2 = 10;
        int precision = 10;

        Random rand = new Random();

        x = 0;
        y = 0;

        x0 = Convert.ToDouble(rand.Next(minRandX * precision, maxRandX * precision +
1) / Convert.ToDouble(precision));

        y0 = Convert.ToDouble(rand.Next(minRandY * precision, maxRandY * precision +
1) / Convert.ToDouble(precision));

        a1 = Convert.ToDouble(rand.Next(minRandA1 * precision, maxRandA1 * precision
+ 1) / Convert.ToDouble(precision));

        a2 = Convert.ToDouble(rand.Next(minRandA2 * precision, maxRandA2 * precision
+ 1) / Convert.ToDouble(precision));
    }

    public override double GetCoordinates(double t, out double y)
    {
        x = x0 + a1 * Math.Sin(t);
        y = y0 + a2 * Math.Cos(t);
        return x;
    }

```

```

    }

    public override double GetDistance(double x2, double y2)
    {
        return (Math.Sqrt(Math.Pow((x2-x),2) + Math.Pow((y2 - y), 2)));
    }

    public override void DisplayObject()
    {
        Console.WriteLine($"\\ta1 = \\t{a1}\\ta2 = \\t{a2}\\tx0 = \\t{x0}\\ty0 = \\t{y0}");
    }

    public override void DisplayCoordinates(double t)
    {
        Console.WriteLine($"\\tx is: \\t{Math.Round(GetCoordinates(t, out y), 3)}\\ty is:
{Math.Round(y, 3)}");
    }
}
}

```

Operations.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Lab4CSharp
{
    class Operations
    {
        public static double InputTime()
        {
            Console.Write("Enter t in seconds: ");
            while (true)
            {
                string input = Console.ReadLine();
                if (double.TryParse(input, out double time) && time > 0)
                {
                    return time;
                }
                else
                {
                    Console.Write("Wrong input. Try again: ");
                }
            }
        }

        public static int InputNumber(string message)
        {
            Console.Write($"Enter {message}: ({message} > 1) ");
            while (true)
            {
                string input = Console.ReadLine();
                if (int.TryParse(input, out int number) && number > 1)
                {
                    return number;
                }
            }
        }
    }
}
```

```

        else
        {
            Console.WriteLine("Wrong input. Try again: ");
        }
    }
}

public static void ShowObjects(MovingPointParticle[] array)
{
    for (int i = 0; i < array.Length; i++)
    {
        Console.WriteLine($"{i + 1}");
        array[i].DisplayObject();
    }
}

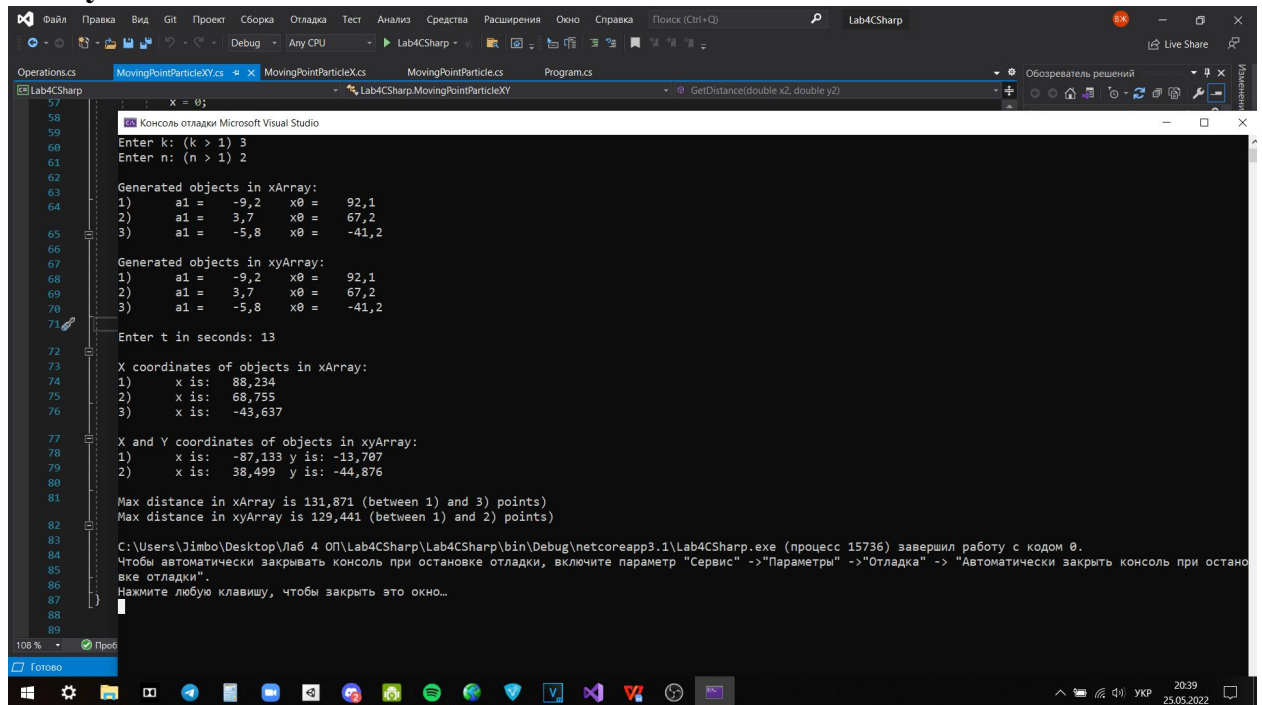
public static void ShowCoordinates(MovingPointParticle[] array, double t)
{
    for (int i = 0; i < array.Length; i++)
    {
        Console.WriteLine($"{i + 1}");
        array[i].DisplayCoordinates(t);
    }
}

public static double ReturnMaxDistance(MovingPointParticle[] array, out int first,
out int second, double t)
{
    first = 0;
    second = 1;
    double maxDistance = array[0].GetDistance(array[1].GetCoordinates(t, out
double y), y);
    for (int o = 0; o < array.Length - 1; o++)
    {
        for (int i = o + 1; i < array.Length; i++)
        {
            double temp = array[o].GetDistance(array[i].GetCoordinates(t, out y),
y);

```

```
        if (temp > maxDistance)
        {
            maxDistance = temp;
            first = o;
            second = i;
        }
    }
}
return maxDistance;
}
}
```

Тестування:



The screenshot shows the Visual Studio Code interface with a C# project named 'Lab4CSharp'. The code in 'MovingPointParticleXY.cs' defines a program that generates two arrays of points, calculates their coordinates at a specific time, and finds the maximum distance between points in each array.

```
57 x = 0;  
58  
59  
60 Enter k: (k > 1) 3  
61 Enter n: (n > 1) 2  
62  
63 Generated objects in xArray:  
64 1) a1 = -9,2 x0 = 92,1  
65 2) a1 = 3,7 x0 = 67,2  
66 3) a1 = -5,8 x0 = -41,2  
67  
68 Generated objects in xyArray:  
69 1) a1 = -9,2 x0 = 92,1  
70 2) a1 = 3,7 x0 = 67,2  
71 3) a1 = -5,8 x0 = -41,2  
72  
73 Enter t in seconds: 13  
74  
75 X coordinates of objects in xArray:  
76 1) x is: 88,234  
77 2) x is: 68,755  
78 3) x is: -43,637  
79  
80 X and Y coordinates of objects in xyArray:  
81 1) x is: -87,123 y is: -13,707  
82 2) x is: 38,499 y is: -44,876  
83  
84 Max distance in xArray is 131,871 (between 1) and 3) points)  
85 Max distance in xyArray is 129,441 (between 1) and 2) points)  
86  
87 C:\Users\Jimbo\Desktop\Лаб 4 ОП\Lab4CSharp\Lab4CSharp\bin\Debug\netcoreapp3.1\Lab4CSharp.exe (процесс 15736) завершил работу с кодом 0.  
88 Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остано  
89 вке отладки".  
Нажмите любую клавишу, чтобы закрыть это окно...
```

The debug console output matches the code's logic, showing the generated points, their coordinates at t=13 seconds, and the calculated maximum distances. The process ended with a return code of 0.

Python

```
from Operations import *
from MovingPointParticle import *

k = input_number("k")
n = input_number("n")

point_x_array = []
for i in range(k):
    point_x_array.append(MovingPointParticleX())

point_xy_array = []
for i in range(n):
    point_xy_array.append(MovingPointParticleXY())

print("\nGenerated objects in x_array: ")
show_objects(point_x_array)

print("\nGenerated objects in xy_array: ")
show_objects(point_xy_array)

print()
t = input_time()

print("\nX and Y coordinates of objects in xArray: ")
show_coordinates(point_x_array, t)

print("\nX and Y coordinates of objects in xyArray: ")
show_coordinates(point_xy_array, t)

max_distance_x, first_x, second_x = return_max_distance(point_x_array)
print(f"\nMax distance in x_array is {round(max_distance_x, 3)} (between {first_x + 1}) and {second_x + 1}) points)")

max_distance_xy, first_xy, second_xy = return_max_distance(point_xy_array)
print(f"Max distance in xy_array is {round(max_distance_xy, 3)} (between {first_xy + 1}) and {second_xy + 1}) points)")
```

Operations.py

```
def input_time():
    time = float(input("Enter t in seconds: "))
    while True:
        if time > 0:
            return time
        else:
            time = float(input("Wrong input. Try again: "))

def input_number(message):
    number = int(input(f"Enter {message}: ( {message} > 0) "))
    while True:
        if number > 0:
            return number
        else:
            number = float(input("Wrong input. Try again: "))

def show_objects(array):
    for i in range(len(array)):
        print(f" {i+1} ", end="")
        array[i].display_object()

def show_coordinates(array, t):
    for i in range(len(array)):
        x, y = array[i].get_coordinates(t)
        print(f" {i+1} )\tx is: \t{round(x, 3)}\ty is: \t{round(y, 3)}")

def return_max_distance(array):
    first = 0
    second = 1
    max_distance = array[0].get_distance(array[0].get_x(), array[0].get_y())
    for o in range(len(array)-1):
        for i in range(len(array)):
            temp_max = array[o].get_distance(array[i].get_x(), array[i].get_y())
            if temp_max > max_distance:
                max_distance = temp_max
                first = o
                second = i
    return max_distance, first, second
```


MovingPointParticle.py

```
from abc import ABC, abstractmethod
import random
import math

class MovingPointParticle(ABC):
    @abstractmethod
    def get_coordinates(self, t):
        pass

    @abstractmethod
    def get_distance(self, x2, y2):
        pass

    @abstractmethod
    def display_object(self):
        pass

class MovingPointParticleX(MovingPointParticle):

    def get_y(self):
        return self.__y

    def get_x0(self):
        return self.__x0

    def get_x(self):
        return self.__x

    def get_a1(self):
        return self.__a1

    def __init__(self):
        min_rand_x = -100
        max_rand_x = 100
        min_rand_a = -10
        max_rand_a = 10
        precision = 10

        self.__x = 0
        self.__y = 0
        self.__x0 = random.randint(min_rand_x * precision, max_rand_x * precision) / precision
        self.__a1 = random.randint(min_rand_a * precision, max_rand_a * precision) / precision

    def get_coordinates(self, t):
        self.__x = self.__x0 + self.__a1 * math.sin(t)
        return self.__x, 0

    def get_distance(self, x2, y2):
        return abs(x2 - self.__x)

    def display_object(self):
        print(f"\ta1 = {self.__a1}\tx0 = {self.__x0}")

class MovingPointParticleXY(MovingPointParticle):

    def get_x0(self):
        return self.__x0
```

```

def get_y0(self):
    return self.__y0

def get_x(self):
    return self.__x

def get_y(self):
    return self.__y

def get_a1(self):
    return self.__a1

def get_a2(self):
    return self.__a2

def __init__(self):
    min_rand_x = -100
    max_rand_x = 100
    min_rand_y = -100
    max_rand_y = 100
    min_rand_a1 = -10
    max_rand_a1 = 10
    min_rand_a2 = -10
    max_rand_a2 = 10
    precision = 10

    self.__x = 0
    self.__y = 0
    self.__x0 = random.randint(min_rand_x * precision, max_rand_x * precision) / precision
    self.__y0 = random.randint(min_rand_y * precision, max_rand_y * precision) / precision
    self.__a1 = random.randint(min_rand_a1 * precision, max_rand_a1 * precision) / precision
    self.__a2 = random.randint(min_rand_a2 * precision, max_rand_a2 * precision) / precision

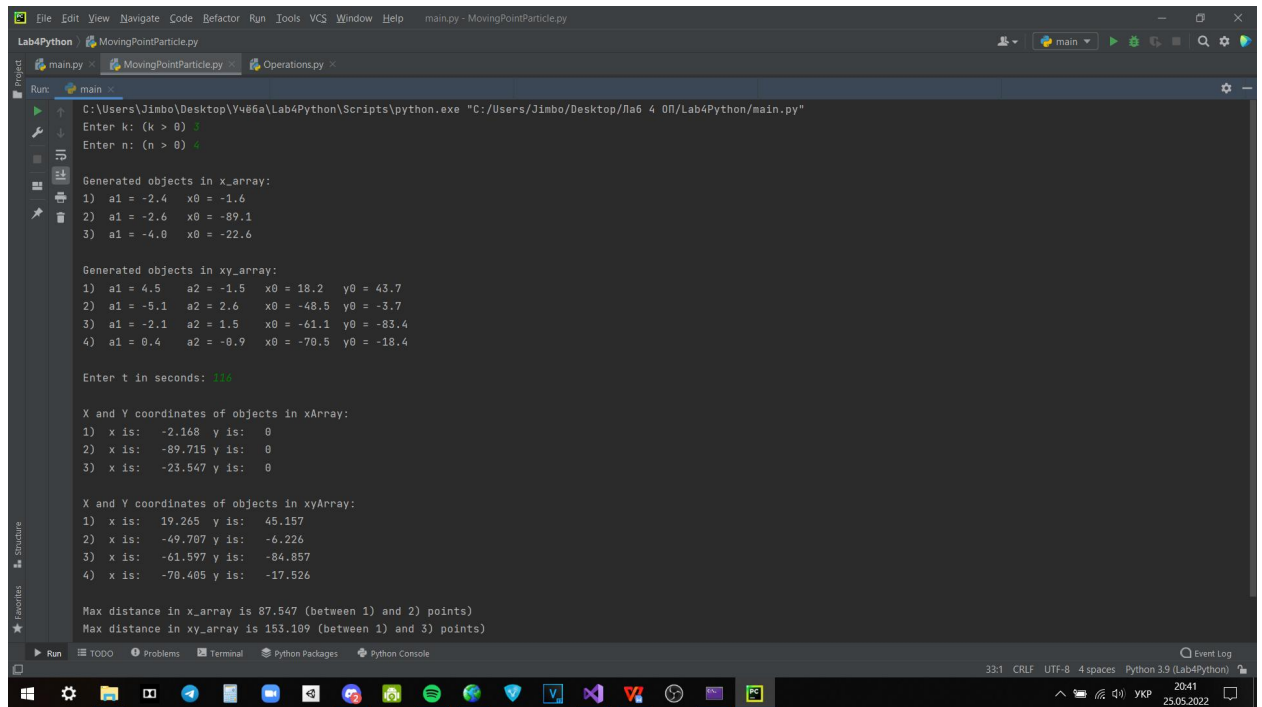
def get_coordinates(self, t):
    self.__x = self.__x0 + self.__a1 * math.sin(t)
    self.__y = self.__y0 + self.__a2 * math.cos(t)
    return self.__x, self.__y

def get_distance(self, x2, y2):
    return math.sqrt((x2 - self.__x) ** 2 + (y2 - self.__y) ** 2)

def display_object(self):
    print(f"\ta1 = {self.__a1}\ta2 = {self.__a2}\tx0 = {self.__x0}\ty0 = {self.__y0}")

```

Тестування:



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help main.py - MovingPointParticle.py
Lab4Python MovingPointParticle.py
main.py MovingPointParticle.py Operations.py
Run: main
C:\Users\Jimbo\Desktop\Учб6a\Lab4Python\Scripts\python.exe "C:/Users/Jimbo/Desktop/Учб6 4 00/Lab4Python/main.py"
Enter k: (k > 0) 3
Enter n: (n > 0) 4

Generated objects in x_array:
1) a1 = -2.4 x0 = -1.6
2) a1 = -2.6 x0 = -89.1
3) a1 = -4.0 x0 = -22.6

Generated objects in xy_array:
1) a1 = 4.5 a2 = -1.5 x0 = 18.2 y0 = 43.7
2) a1 = -5.1 a2 = 2.6 x0 = -48.5 y0 = -3.7
3) a1 = -2.1 a2 = 1.5 x0 = -61.1 y0 = -83.4
4) a1 = 0.4 a2 = -0.9 x0 = -70.5 y0 = -18.4

Enter t in seconds: 135

X and Y coordinates of objects in xArray:
1) x is: -2.168 y is: 0
2) x is: -89.715 y is: 0
3) x is: -23.547 y is: 0

X and Y coordinates of objects in xyArray:
1) x is: 19.265 y is: 45.157
2) x is: -49.707 y is: -6.226
3) x is: -61.597 y is: -84.857
4) x is: -70.405 y is: -17.526

Max distance in x_array is 87.547 (between 1) and 2) points)
Max distance in xy_array is 153.109 (between 1) and 3) points)

Run TODO Problems Terminal Python Packages Python Console
33:1 CRLF UTF-8 4 spaces Python 3.9 (Lab4Python)
2041
25.05.2022
```

Висновки:

На цій лабораторній роботі я застосував на практиці знання щодо таких принципів ООП як успадкування та поліморфізм. Розробив програму, використовуючи мови програмування C# і Python та зробив висновки.