**Звіт до Лабораторної роботи №2**

**Тема:** Класи

**Мета роботи:** Використовуючи теоретичне підґрунтя про об’єктно орієнтоване програмування виконати дії що будуть вказано в завданні до лабораторної роботи.

**Хід роботи**

**Завдання 1.** Необхідно реалізувати метод intersection у класі Line. Він повинен повертати точку перетину двох ліній (клас Point). Якщо лінії збігаються або перетинаються, метод повинен повертати значення null.

Функція, що описує пряму y = k X + b

Користувач вводить значення k та b для двох прямих

Line line1 = new Line(1,1);

Line line2 = new Line(-1,3);

System.out.println(line1.intersection(line2)); // (1;2)

Пакет task\_1 для вирішення Завдання №1 до Лабораторної роботи №2 складається з таких програмних модулів: класи Point, Line та головного класу (з головним модулем) Main.

Клас Point описує об’єкти Точка на площині. Об’єкт точка має атрибути – координати х та у,

Метод-Конструктор цього класу: public Point(double x, double y), отримує координати (х, у) точки в якості параметрів та створює об’єкт Точка.

Метод toString класу Point – виконує вивід точки з визначеними координатами на екран.

Клас Line описує об’єкт Пряма на площині. Об’єкт класу Пряма має атрибутами коефіцієнти k та b, які задають рівняння прямої на площині: y = kx + b

Метод-Конструктор цього класу: public Line(double k, double b) - отримує в якості параметрів коефіцієнти k та b та ініціалізує об’єкт Пряма на площині.

Наступний метод цього класу: public Point intersection(Line other) – знаходить точку перетину 2-х заданих прямих, розраховує її коордиинати та повертає її в головну процедуру main. Так як метод intersection(Line other) відноситься до класу Line, то він викликається після ініціалізації першої прямої line1 за допомогою такого запису: line1.intersection(line2),

де line1 – Перша пряма, для якої застосовується метод intersection, яка позначається в коді самого методу найменуванням this,

а line2 – Друга пряма, яка передається в метод intersection в якості параметра і позначається в коді метода іменем other.

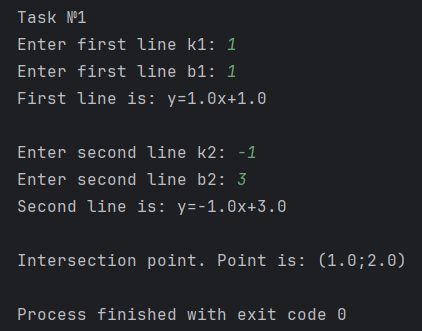
Клас Main, містить головну процедуру main, - головний модуль пакету, який виконує увесь алгоритм завдання і з якого викликаються всі інші класи та їх методи. Перед початком головного модуля main імпортується бібліотека Scaner для діалогу з користувачем. Потім стандартними командами діалогу користувач вводить коефіцієнти k1 та b1 Першої прямої, потім рівняння цієї прямої виводиться на екран у формі: y=kx + b

Далі створюється об’єкт line1 (Перша пряма) класу Line із заданими коефіцієнтами k1, b1

Потім аналогічними командами вводяться коефіцієнти Другої прямої k2, b2, виводиться на екран рівняння Другої прямої. Далі створюється об’єкт line2 (Друга пряма) класу Line із заданими коефіцієнтами k2, b2

Після створення об’єктів line1, line2 – викликається метод: line1.intersection(line2) класу Line, який визначає точку перетину прямих line1 та line2, якщо така існує, і виводить її координати на екран. Якщо прямі не перетинаються – виводиться null.

Результат роботи програми до Завдання 1:



Код програми до Завдання 1:

public class Point {  
 double x, y;  
 public Point(double x, double y){  
 this.x = x;  
 this.y = y;  
 }  
 public String toString(){  
 return "Point is: (" + x + ";" + y + ")";  
 }  
}

public class Line {  
 public double k, b;  
 public Line(double k, double b){  
 this.k = k;  
 this.b = b;  
 }  
 public Point intersection(Line other) {  
 if (this.k == other.k) {  
 return null;  
 }  
 double x = (other.b - this.b) / (this.k - other.k);  
 double y = this.k \* x + this.b;  
 return new Point(x, y);  
 }  
}

public class Main {  
 public static void main(String[] args) {  
 System.*out*.println("Task №1");  
 Scanner scan = new Scanner(System.*in*);  
 System.*out*.print("Enter first line k1: ");  
 double k1 = scan.nextDouble();  
 System.*out*.print("Enter first line b1: ");  
 double b1 = scan.nextDouble();  
 System.*out*.println("First line is: y=" + k1 + "x+" + b1);  
 Line line1 = new Line(k1, b1);  
  
 System.*out*.print("\nEnter second line k2: ");  
 double k2 = scan.nextDouble();  
 System.*out*.print("Enter second line b2: ");  
 double b2 = scan.nextDouble();  
 System.*out*.println("Second line is: y=" + k2 + "x+" + b2);  
 Line line2 = new Line(k2, b2);  
 System.*out*.println("\nIntersection point. " + line1.intersection(line2));  
 }  
}

**Завдання 2.** Необхідно реалізувати методи класу Segment (відрізок):

Конструктор, в який як параметри передаються координати точок початку і кінця відрізка (використовуйте клас Point). Переконайтися, що створений відрізок існує і не є виродженим, що означає, що початок і кінець відрізка не є однією точкою.

Реалізувати метод double length() – повертає довжину сегмента.

Реалізувати метода Point middle() – повертає середню точку сегмента.

Реалізувати метод Point intersection(Segment another) – повертає точку перетину поточного відрізка з іншим.

Пакет task\_2 для вирішення Завдання №2 до Лабораторної роботи №2 складається з таких програмних модулів: класи Point, Line, Segment та головного класу (з головним модулем) Main.

Класи Point та Line із пакету Task\_2 повністю аналогічні класам Point та Line із пакету Task\_1.

Клас Segment описує об’єкт Відрізок на площині. Відрізок має атрибути: firstPoint, secondPoint – початкова і кінцева точки Відрізка.

Метод-Конструктор: public Segment(Point firstPoint, Point secondPoint) – створює об’єкт класу Segment – відрізок із заданими точками firstPoint, secondPoint – тобто його кінцями.

Метод public double length(Point firstPoint, Point secondPoint) – розраховує довжину Відрізка із заданими точками firstPoint, secondPoint - його кінцями, використовуючи їх координати. Після цього довжина відрізка повертається в головну процедуру main.

Метод public Point middle(Point firstPoint, Point secondPoint) – розраховує координати середини відрізка, заданого 2-ма точками. Після розрахунку створюється об’єкт Точка із розрахованими координатами і повертається в головну процедуру main.

Метод public Point intersectSegm(Segment another) - визначає точку перетину 2х відрізків (якщо вона існує) і повертає її в головну процедуру main.

В цьому методі спочатку розраховуються коефіцієнти прямих k та b, яким належать задані відрізки через координати кінців цих відрізків firstPoint(x,y), secondPoint(x,y). Після розрахунку коефіцієнтів прямих k та b створюються об’єкти класу Line – Прямі line1, line2.

Наступною командою:

Point intersectionPoint = line1.intersection(line2);

Викликається метод intersection() класу Line, який знаходить точку перетину 2-х прямих line1, line2 і створюється об’єкт класу Point з іменем intersectionPoint.

Далі виконується перевірка: якщо метод intersection() повернув замість точки перетину – значення null – отже прямі не перетинаються. Тоді метод intersectSegm() також повертає null в головну процедуру.

І остання перевірка: якщо знайдена точка перетину належить обом заданим відрізкам, позначеним в цій умові this та other, - тоді метод повертає точку intersectionPoint в головну процедуру.

Якщо ж знайдена точка перетину не належить хоча б 1-му з 2-х відрізків – метод повертає null

Метод private boolean isPointOnSegment(Point p, Segment segment) – визначає чи належить точка перетину заданому відрізку. Метод отримує в якості параметрів точку перетину p (об’єкт класу Point) та відрізок segment (об’єкт класу Segment).

Далі виконується ряд перевірок:

Чи кордината X точки р лежить вище або рівно найМеншої за координатою X точки відрізка?

Чи кордината X точки р лежить нижче або рівно найБільшої за координатою X точки відрізка?

Виконання цих 2-х умов значить, що точка р за координатою X лежить в межах відрізка.

Чи кордината Y точки р лежить вище або рівно найМеншої за координатою Y точки відрізка?

Чи кордината Y точки р лежить нижче або рівно найБільшої за координатою Y точки відрізка?

Виконання цих 2-х умов значить, що точка р за координатою Y лежить в межах вiдрiзка.

Отже, якщо всі 4 умови, вказані вище, виконуються – точка р належить відрізку і метод isPointOnSegment() повертає значення true до методу intersectSegm(), з якого він викликається.

Якщо хоча б 1 з 4-х умов вище не виконується – значить точка перетину прямих не належить одному з відрізків, і відрізки не перетинаються, хоча прямі по факту і можуть перетинаютися. Тоді метод isPointOnSegment() повертає null до методу intersectSegm(), який також повертає null в процедуру main.

Клас Main, містить головну процедуру main, - головний модуль пакету, який виконує увесь алгоритм завдання і з якого викликаються всі інші класи та їх методи.

Перед початком головного модуля main імпортується бібліотека Scaner для діалогу з користувачем.

Спочатку стандартними командами діалогу користувач вводить координати початку (х1,у1) Першого Відрізка. Створюється об’єкт Точка з цими координатами firstPoint1(х1,у1).

Далі, аналогічними командами – координати кінця Першого Відрізка (х2, у2). Створюється об’єкт Точка з цими координатами secondPoint1(х2,у2)

Потім, аналогічними командами – координати початку Другого Відрізка (х3, у3). Створюється об’єкт Точка з цими координатами firstPoint2(х3,у3)

Накінець, аналогічними командами – координати кінця Другого Відрізка (х4, у4). Створюється об’єкт Точка з цими координатами secondPoint2(х4,у4)

Потім створюються об’єкти Перший та Другий Відрізки:

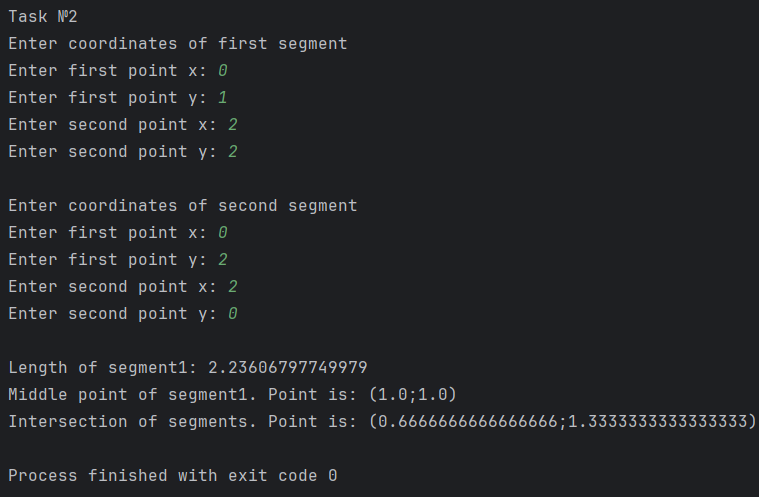
Segment segment1 = new Segment(firstPoint1, secondPoint1);

Segment segment2 = new Segment(firstPoint2, secondPoint2);

Далі розраховується довжина Першого відрізка за допомогою методу length() класу Segment і виводиться на екран. Потім розраховуються координати точки – середини Першого відрізка і виводяться на екран.

Накінець, вираховуються координати точки перетину 2-х відрізків, якщо така існує, і виводяться на екран.

Результат роботи програми до Завдання 2:



Код програми до Завдання 2:

package task\_2;  
import static java.lang.Math.\*;  
  
public class Segment {  
 Point firstPoint, secondPoint;  
 public Segment(Point firstPoint, Point secondPoint){  
 if (firstPoint.x == secondPoint.x && firstPoint.y == secondPoint.y) {  
 System.*out*.println("This Segment NOT exist: start Point and end Point must be different");  
 return;  
 }  
 this.firstPoint = firstPoint;  
 this.secondPoint = secondPoint;  
 }  
 public double length(Point firstPoint, Point secondPoint){  
 return *sqrt*(*pow*((firstPoint.x - secondPoint.x), 2) + *pow*((firstPoint.y - secondPoint.y), 2));  
 }  
  
 public Point middle(Point firstPoint, Point secondPoint){  
 double xMid = (firstPoint.x + secondPoint.x) / 2;  
 double yMid = (firstPoint.y + secondPoint.y) / 2;  
 return new Point(xMid, yMid);  
 }  
 public Point intersectSegm(Segment another){  
// В методі intersection створюютья 2 обєкти -прямі line1,line2, що містять 2 задані відрізки  
// Координати 1-го відрізка firstPoint(x,y), secondPoint(x,y) а другого - another.firstPoint(x,y), another.secondPoint(x,y)  
// Коеф-ти прямих розрах-ся таk: K=(y2-y1)/(x2-x1), b=y1-((y2-y1)/(x2-x1)\*x1)  
// firstPoint(x,y) secondPoint(x,y) - точки початку та кінця 1-го відрізка  
// another.firstPoint(x,y) another.secondPoint(x,y) - точки початку та кінця 2-го відрізка  
 Line line1 = new Line((secondPoint.y - firstPoint.y) / (secondPoint.x - firstPoint.x),  
 firstPoint.y - ((secondPoint.y - firstPoint.y) / (secondPoint.x - firstPoint.x)) \* firstPoint.x);  
 Line line2 = new Line((another.secondPoint.y - another.firstPoint.y) / (another.secondPoint.x - another.firstPoint.x),  
 another.firstPoint.y - ((another.secondPoint.y - another.firstPoint.y) / (another.secondPoint.x - another.firstPoint.x)) \* another.firstPoint.x);  
  
 Point intersectionPoint = line1.intersection(line2);  
 if (intersectionPoint == null) {  
 return null;  
 }  
 if (isPointOnSegment(intersectionPoint, this) && isPointOnSegment(intersectionPoint, another)) {  
 return intersectionPoint;  
 }  
 return null;  
 }  
  
 private boolean isPointOnSegment(Point p, Segment segment) {  
 return *min*(segment.firstPoint.x, segment.secondPoint.x) <= p.x && p.x <= *max*(segment.firstPoint.x, segment.secondPoint.x) &&  
 *min*(segment.firstPoint.y, segment.secondPoint.y) <= p.y && p.y <= *max*(segment.firstPoint.y, segment.secondPoint.y);  
 }  
}

package task\_2;  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 System.*out*.println("Task №2");  
 Scanner scan = new Scanner(System.*in*);  
 System.*out*.print("Enter coordinates of first segment\n");  
 System.*out*.print("Enter first point x: ");  
 double x1 = scan.nextDouble();  
 System.*out*.print("Enter first point y: ");  
 double y1 = scan.nextDouble();  
 Point firstPoint1 = new Point(x1, y1);  
  
 System.*out*.print("Enter second point x: ");  
 double x2 = scan.nextDouble();  
 System.*out*.print("Enter second point y: ");  
 double y2 = scan.nextDouble();  
 Point secondPoint1 = new Point(x2, y2);  
  
 System.*out*.print("\nEnter coordinates of second segment\n");  
 System.*out*.print("Enter first point x: ");  
 double x3 = scan.nextDouble();  
 System.*out*.print("Enter first point y: ");  
 double y3 = scan.nextDouble();  
 Point firstPoint2 = new Point(x3, y3);  
  
 System.*out*.print("Enter second point x: ");  
 double x4 = scan.nextDouble();  
 System.*out*.print("Enter second point y: ");  
 double y4 = scan.nextDouble();  
 Point secondPoint2 = new Point(x4, y4);  
  
 Segment segment1 = new Segment(firstPoint1, secondPoint1);  
 Segment segment2 = new Segment(firstPoint2, secondPoint2);  
  
 System.*out*.println("\nLength of segment1: " + segment1.length(firstPoint1, secondPoint1));  
 System.*out*.println("Middle point of segment1. " + segment1.middle(firstPoint2, secondPoint2));  
 System.*out*.println("Intersection of segments. " + segment1.intersectSegm(segment2));  
 }  
}

**Завдання 3.** Необхідно реалізувати методи класу Triangle:

Конструктор, що має як параметри координати трьох вершин (клас Point). Переконайтися, що ці точки належать до вершин трикутника. Перевірити, що створений трикутник існує і не вироджений.

Реалізувати метод double area() – повертає площу трикутника.

Реалізувати метод Point centroid() – повертає центроїд трикутника.

Пакет task\_3 для вирішення Завдання №3 до Лабораторної роботи №2 складається з таких програмних модулів: класи Point, Triangle, Segment та головного класу (з головним модулем) Main.

Клас Point описує об’єкт Точка на площині з координатами (х,у). Конструктор Point() створює об’єкт Точка і задає їй координати (передані як параметри)

Останній метод toString() виводить координати точки на екран у форматі (х,у)

Клас Triangle описує об’єкт Трикутник на площині. Спочатку створюється сам клас Triangle (Трикутник) з атрибутами – точками Point 3-х вершин а, b, с

Потім описаний метод-Конструктор: public Triangle(Point a, Point b, Point c), який створює об’єкт Трикутник із 3-ма вершинами із заданими координатами, які передаються методу в якості параметрів, та ініціалізуються.

Далі описано метод: public String toString(), який забезпечує вивід вершин об’єкта Трикутник на екран.

Наступний метод: static double area(Segment ab, Segment bc, Segment ac), описує розрахунок площі Трикутника по його сторонах – відрізках класу Segment, які передаються в метод в якості параметрів. Спочатку розраховуються довжини сторін трикутника AB, BC, AC. Далі розраховується півпериметр: p=(AB + BC + AC)/2 Накінець, по формулі Герона розраховується площа Трикутника S і повертається командою return.

Останній метод класу Triangle це метод: Point centroid(Point a, Point b, Point c). Він розраховує координати точки - Центроїда Трикутника (точки О перетину його медіан). Для розрахунку взята медіана АМ від вершини А до середини відрізка ВС.

Спочатку розраховуються координати точки М як середини відрізка ВС:

double Mx = (b.x + c.x)/2; double My = (b.y + c.y)/2;

Створюється об’єкт – точка М з координатами (Мх, Му), інформація про знайдену точку М виводиться на екран.

Далі використовується властивість трикутника, згідно якої Медіани Трикутника перетинаються в одній точці О і діляться цією точкою у відношенні 2:1 (рахуючи від вершини). Отже для розрахунку координат точки О до координат вершини (точки А) (а.х, а.у) треба додати 2/3 довжини медіани АМ, тому координати точки О вираховуються так:

double Ox = a.x + 2.0/3.0\*(Mx - a.x); double Oy = a.y + 2.0/3.0\*(My - a.y);

В цих виразах числа 2.0 та 3.0 вказані в форматі з десятковою крапкою, так як інші змінні виразу мають тип «double»

Після цього створюється об’єкт – Точка О з розрахованими координатами (Ох, Оу), і повертається командою return.

Клас Segment – відрізок, як об’єкт, що задається точками початку і кінця (firstPoint, secondPoint). Відрізок використовується для визначення сторін Трикутника.

Метод-Конструктор public Segment(Point firstPoint, Point secondPoint) – створює об’єкт класу Segment (відрізок) із точками його початку та кінця, що передаються методу як параметри.

Спочатку перевіряється, чи не збігаються початок та кінець відрізка в 1 точку. Якщо не збігаються – тоді ініціалізуються точки початку firstPoint та кінця secondPoint відрізка.

Наступний метод класу Segment – метод length() – визначає довжину відрізка та повертає її командою return.

Клас Main містить головну процедуру public static void main(String[] args), де описано алгоритм вирішення завдання.

Спочатку створюються 3 об’єкти Point (точки) – вершини Трикутника із заданими координатами.

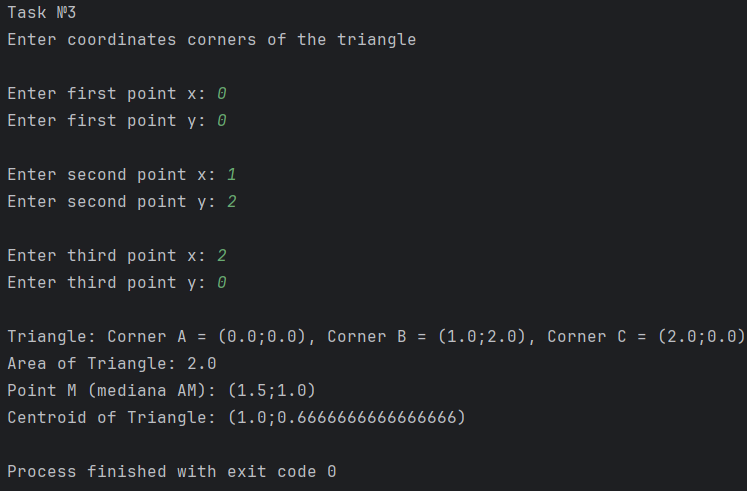
Потім створюється об’єкт Triangle (трикутник), координати точок – вершин Трикутника виводяться на екран.

Далі створюються 3 об’єкти Segment (відрізки) – сторони Трикутника. Потім виконується 2 перевірки: чи не рівна 0 довжина будь-якої сторони, та чи виконується умова існування Трикутника: довжина будь-якої сторони Трикутника має бути менша за суму довжин 2-х інших сторін. Якщо будь-яка з 2х перевірок не виконується – вихід з програми командою return з відповідним повідомленням, що Трикутник не існує.

Якщо Трикутник існує, - спочатку викликається метод area класу Triangle: Triangle.area(ab,bc,ac), який розраховує площу Трикутника та присвоює її змінній sTriangle. Потім площа виводиться на екран.

Далі викликається метод centroid класу Triangle: Triangle.centroid(a, b, c), який визначає координати точки – Центроїда Трикутника, і виводить повідомлення про це на екран. На цьому робота програми завершується.

Результат роботи програми до Завдання 3:



Код програми до Завдання 3:

package task\_3;  
  
public class Point {  
 double x, y;  
 // Клас Point описує обєкт Точка з координатами х,у  
 public Point(double x, double y) {  
// Конструктор Point() створює обєкт Точка і задає їй координати (передані як параметри)  
 this.x = x;  
 this.y = y;  
 }  
  
 // Метод toString() виводить координати обэкту Point (Точка) на екран  
 public String toString() {  
 return "(" + x + ";" + y + ")";  
 }  
}

public class Segment {  
 public Point firstPoint, secondPoint;  
 public Segment(Point firstPoint, Point secondPoint) {  
 if (firstPoint.x == secondPoint.x && firstPoint.y == secondPoint.y) {  
 System.*out*.println("This Segment and Triangle NOT exist: start Point and end Point must be different");  
 }  
 this.firstPoint = firstPoint;  
 this.secondPoint = secondPoint;  
 }  
  
 public double length() {  
 return *sqrt*(*pow*(secondPoint.x - firstPoint.x, 2) + *pow*(secondPoint.y - firstPoint.y, 2));  
 }  
}

public class Triangle {  
 public Point a, b, c;  
 public Triangle(Point a, Point b, Point c){  
 this.a = a;  
 this.b = b;  
 this.c = c;  
 }  
 public String toString(){  
 return ("Corner A = "+ a + ", Corner B = " + b + ", Corner C = "+ c);  
 }  
 public static double area(Segment ab, Segment bc, Segment ac){  
 double AB = ab.length();  
 double BC = bc.length();  
 double AC = ac.length();  
// Півпериметр тр-ка  
 double p = (AB + BC + AC)/2;  
// Площа тр-ка по формулі Герона  
 return Math.*sqrt*(p\*(p-AB)\*(p-BC)\*(p-AC));  
 }  
 public static Point centroid(Point a, Point b, Point c){  
// Координати т.М - середини відрізка BC: Mx=(Bx + Cx)/2, My=(By + Cy)/2  
 double Mx = (b.x + c.x)/2;  
 double My = (b.y + c.y)/2;  
 Point M = new Point(Mx,My);  
 System.*out*.println("Point M (mediana AM): " +M);  
// Координати т.O -Центроїда: Ox=Ax+2/3\*(Mx - Ax)/2, Oy=Ay+2/3\*(My - Ay)/2  
 double Ox = a.x + (Mx - a.x) \* 2.0/3.0;  
 double Oy = a.y + (My - a.y) \* 2.0/3.0;  
 return new Point(Ox,Oy);  
 }  
  
}

public class Main {  
 public static void main(String[] args) {  
// Задання координат вершин Тр-ка: а,в,с  
 System.*out*.println("Task №3");  
 Scanner scan = new Scanner(System.*in*);  
 System.*out*.print("Enter coordinates corners of the triangle\n");  
  
 System.*out*.print("\nEnter first point x: ");  
 double x1 = scan.nextDouble();  
 System.*out*.print("Enter first point y: ");  
 double y1 = scan.nextDouble();  
 Point a = new Point(x1, y1);  
  
 System.*out*.print("\nEnter second point x: ");  
 double x2 = scan.nextDouble();  
 System.*out*.print("Enter second point y: ");  
 double y2 = scan.nextDouble();  
 Point b = new Point(x2, y2);  
  
 System.*out*.print("\nEnter third point x: ");  
 double x3 = scan.nextDouble();  
 System.*out*.print("Enter third point y: ");  
 double y3 = scan.nextDouble();  
 Point c = new Point(x3, y3);  
  
 Triangle abc = new Triangle(a,b,c);  
// Вивід на екран координат вершин Тр-ка: а,в,с  
 System.*out*.println("\nTriangle: " + abc);  
// Створення об"єктів - сторін Тр-ка ав, вс, ас  
 Segment ab = new Segment(a,b);  
 Segment bc = new Segment(b,c);  
 Segment ac = new Segment(a,c);  
// Перевірка, чи рівна 0 будь-яка сторона Тр-ка  
 if ((ab.length()==0)||(bc.length()==0)||(ac.length()==0)){  
 System.*out*.println("This Triangle NOT Exist");  
 return;  
 }  
// Перевірка, чи довжина будь-якої сторони Тр-ка >= Сумі довжин 2-х інших сторін  
 if ((ab.length() >= (bc.length()+ac.length()))||(bc.length() >= (ab.length()+ac.length()))||(ac.length() >= (ab.length()+bc.length()))){  
 System.*out*.println("This Triangle NOT Exist");  
 return;  
 }  
// Площа трикутника sTriangle за 3-ма сторонами  
 double sTriangle = Triangle.*area*(ab,bc,ac);  
 System.*out*.println("Area of Triangle: " + sTriangle);  
// Координати точки-Центроїда Тр-ка  
// Point ooo = Triangle.centroid(a, b, c);  
 System.*out*.println("Centroid of Triangle: " + Triangle.*centroid*(a, b, c));  
 }  
}

**Висновки.** Під час виконання цієї лабораторної роботи я навчився створювати та використовувати класи та об’єкти класів. Я створив:

Клас Point (точка), який містить методи, що задають координати точки на площині;

Клас Line (пряма), який містить метод, що повертає точку перетину двох ліній;

Клас Segment (відрізок), який містить методи, що повертають відрізок на площині за координатами його кінців, довжину відрізка, середню точку відрізка та точку перетину поточного відрізка з іншим на площині;

Клас Triangle (трикутник), який містить методи, що повертають площу трикутника та координати його центроїда.

Посилання на GitHub репозиторій:

<https://github.com/diachenkom/OOP-KB-231-Diachenko-Maksym>