**Лабораторна робота №2**

**Тема:** Класи

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

**Хід роботи**

**Завдання 1**

**Завдання:** Вам 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)



Рисунок 1.0



Рисунок 1.1 – Продовження коду з рисунку 1.0

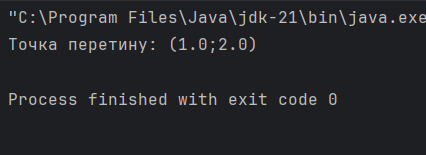


Рисунок 1.2 – Результат знаходження точки перетину

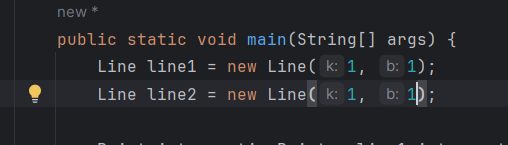


Рисунок 1.3 – Змінив дані аби лінії збігалися

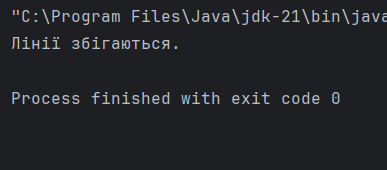


Рисунок 1.4 – Результат якщо лінії збігаються

**Завдання 2**

**Завдання:** Реалізуйте методи класу Segment (відрізок):

Конструктор, в який як параметри передаються координати точок початку і кінця відрізка (використовуйте клас Point).

Переконайтеся, що створений відрізок існує і не є виродженим, що означає, що початок і кінець відрізка не є однією точкою.

Реалізуйте метод double **length**() – повертає довжину сегмента.

Реалізуйте метода Point **middle**() – повертає середню точку сегмента.

Реалізуйте метод Point **intersection**(Segment another) – повертає точку перетину поточного відрізка з іншим

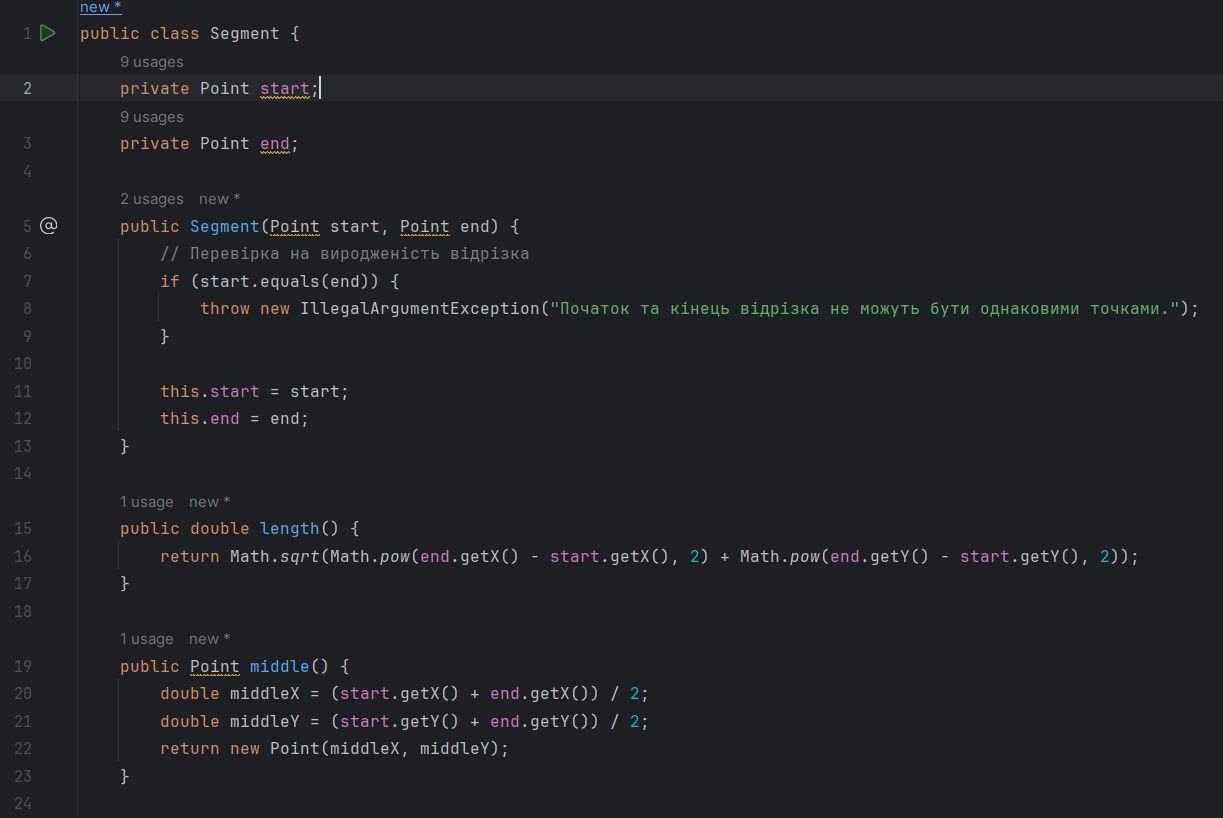
****

Рисунок 2.0

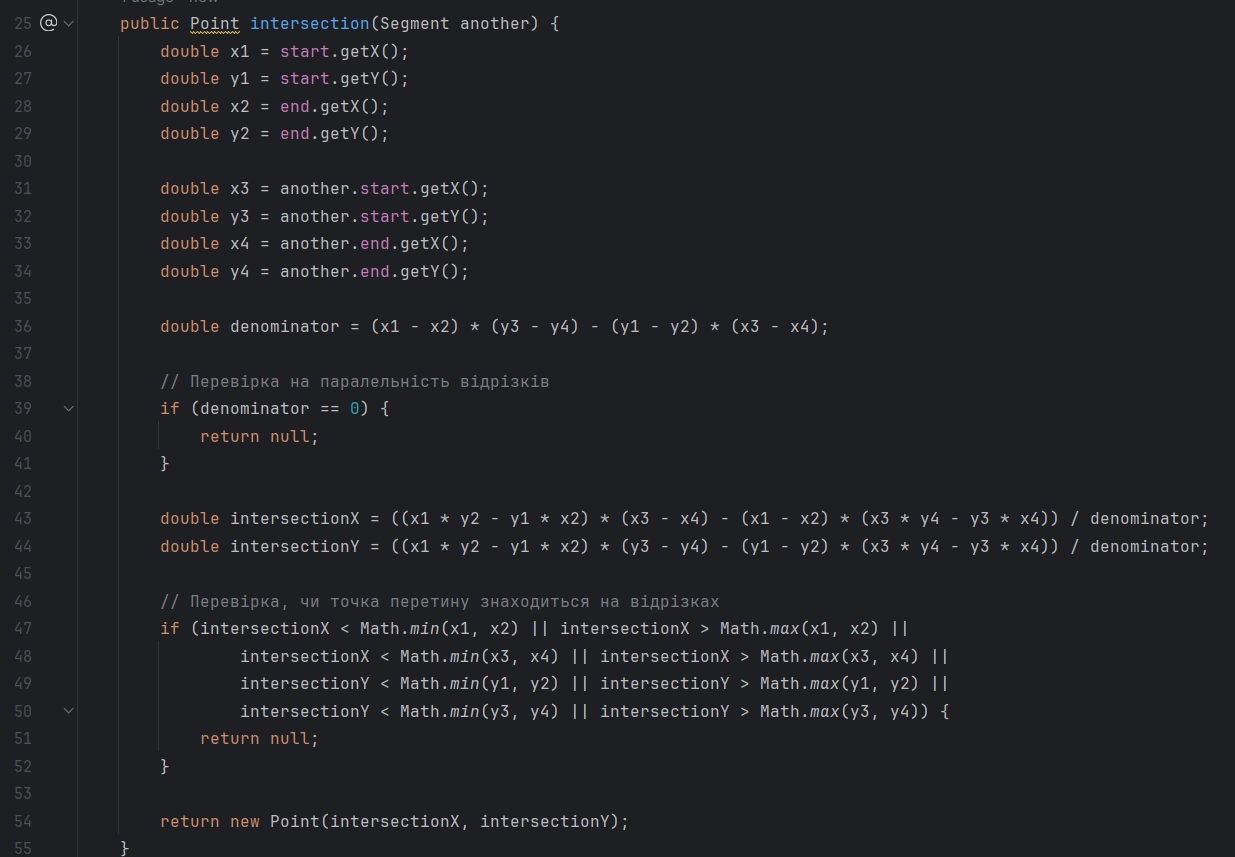


Рисунок 2.1 – Продовження коду з рисунка 2.0

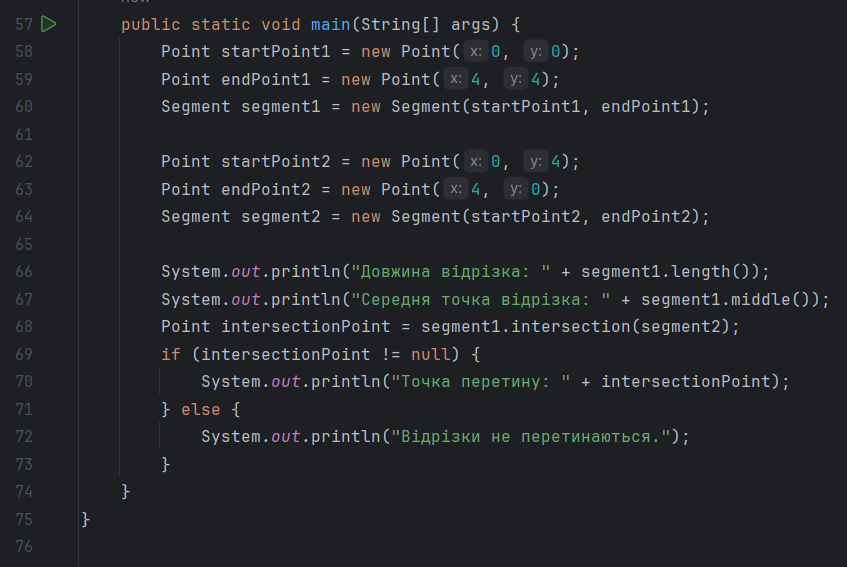


Рисунок 2.2 – Основна функція, продовження коду з рисунка 2.1



Рисунок 2.3 – Продовження коду з рисунка 2.2

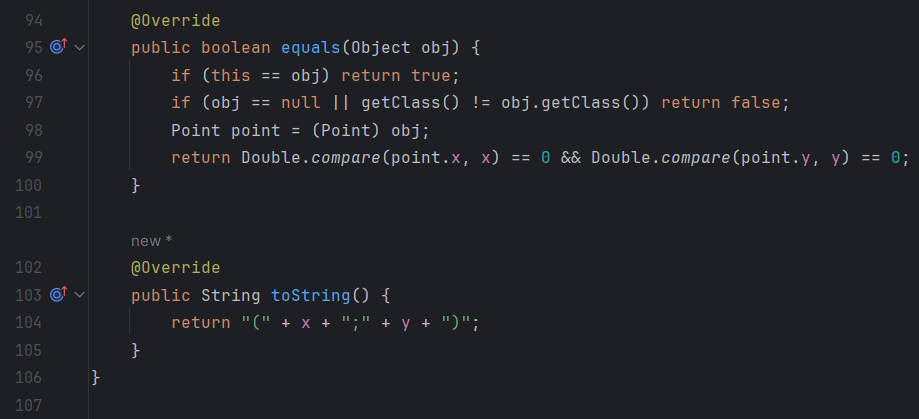


Рисунок 2.4 – Продовження коду з рисунка 2.3

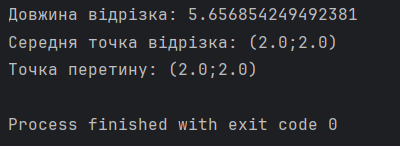


Рисунок 2.5 – Результат звичайного обчислення

Якщо змінити значення координатів так аби це була точка, отримаємо наступне:

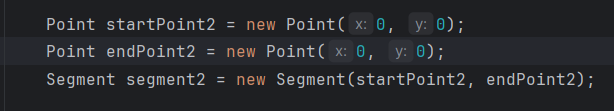


Рисунок 2.6 – Змінили дані

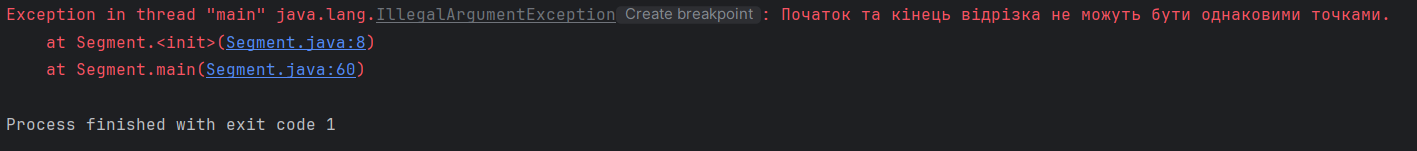


Рисунок 2.7 – Результат

Як бачимо видається помилка, але я додав перевірку на випадок такої ситуації, як результат бачимо текст “Початок та кінець відрізка не можуть бути однаковими точками”.

Також додана перевірка на те, що відрізки не перетинаються.

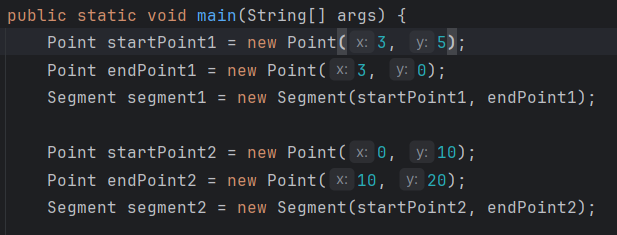


Рисунок 2.8 – Змінені дані

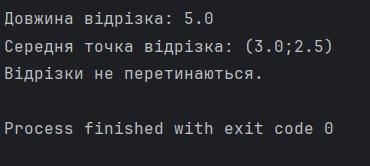


Рисунок 2.9 – Результат.

Невелике пояснення:

Цей код реалізує клас `Segment`, який представляє собою відрізок на площині. Основні методи класу:

1. Конструктор `Segment(Point start, Point end)`: Цей конструктор приймає початкову та кінцеву точки відрізка. Початок і кінець не можуть бути однаковими точками, інакше викидається виняток `IllegalArgumentException`.

2. Метод `double length()`: Цей метод обчислює та повертає довжину відрізка за допомогою формули відстані між двома точками у двовимірному просторі.

3. Метод `Point middle()`: Цей метод обчислює та повертає середню точку відрізка, що є точкою з середини відрізка.

4. Метод `Point intersection(Segment another)`: Цей метод приймає інший відрізок і обчислює точку їх перетину, якщо вона існує. Для цього використовується формула перетину двох прямих у просторі, а також перевірки, чи точка перетину належить обом відрізкам.

Клас `Point` використовується для представлення точок на площині і містить координати x та y.

**Завдання 3**

**Завдання:** Реалізуйте методи класу **Triangle**:

Конструктор, що має як параметри координати трьох вершин (клас Point).

Переконайтеся, що ці точки належать до вершин трикутника.

Перевірте, що створений трикутник існує і не вироджений.

Реалізуйте метод double **area**() – повертає площу трикутника.

Реалізуйте метод Point centroid() – повертає центроїд трикутника.



Рисунок 3.0

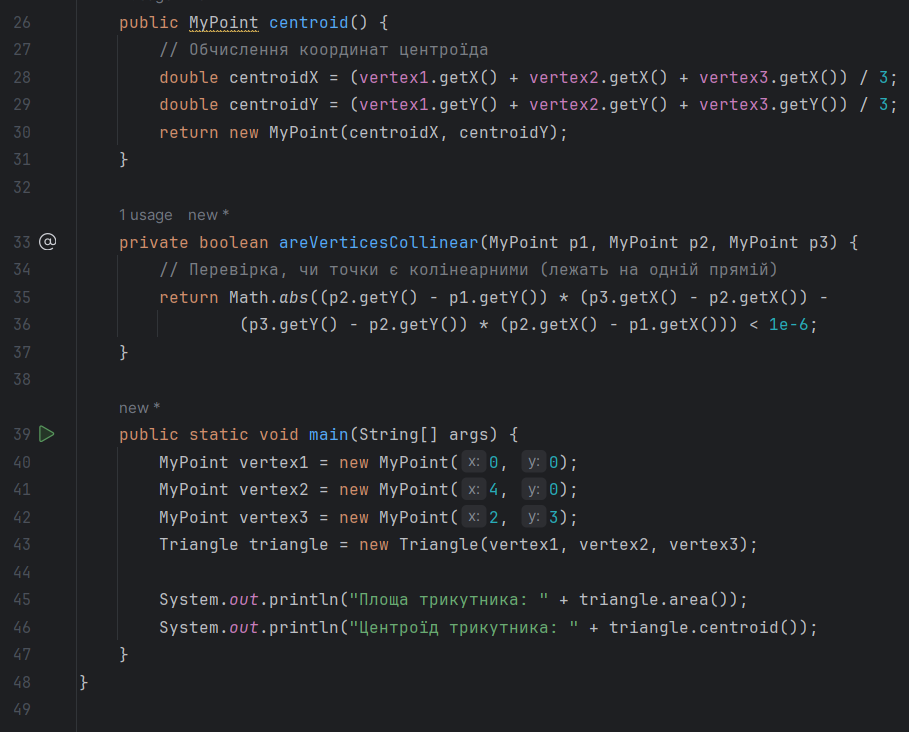


Рисунок 3.1 – Продовження коду з рисунку 3.0



Рисунок 3.2 – Продовження коду з рисунку 3.1

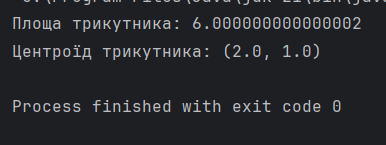


Рисунок 3.3 – Результат роботу коду

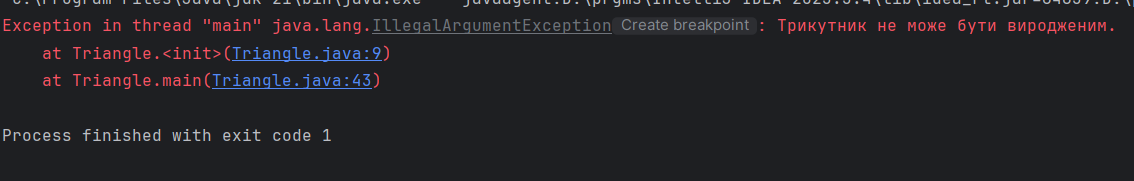


Рисунок 3.4 – Дві точки однакові

Невелике пояснення:

Цей код має клас `Triangle`, який представляє трикутник на площині. Він має такі основні елементи:

1. Конструктор: Конструктор класу приймає три об'єкти класу `MyPoint`, які представляють вершини трикутника. Перш ніж створити трикутник, він перевіряє, чи ці точки не лежать на одній прямій, щоб уникнути створення виродженого трикутника.

2. Метод area(): Цей метод обчислює площу трикутника, використовуючи формулу Герона, що базується на довжинах його сторін.

3. Метод centroid(): Цей метод обчислює центроїд трикутника, який є середнім арифметичним координат вершин.

4. Приватний метод areVerticesCollinear(): Цей метод перевіряє, чи три точки лежать на одній прямій, щоб уникнути створення виродженого трикутника. Він використовує визначник для цього.

5. Метод main(): У цьому методі створюється екземпляр класу `Triangle` з трьома вершинами, і виводиться площа та координати центроїда трикутника.

Цей код виводить коректні значення площі та центроїда трикутника, передбачаючи правильність роботи методів для обчислення цих значень.

**Висновок:** За допомогою теоретичних матеріалів було виконано всі практичні завдання з лабораторної роботи.