**Лабораторна робота №2**

**Класи**

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

# ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

Для виконання лабораторної роботи необхідно використовувати репозиторій на github, що був створений при виконанні завдання до першої лабораторної роботи. Результати роботи необхідно буде завантажити на github для перевірки.

При використанні IDE створити проект під назвою lab02 для розміщення результатів виконання завдань. У випадку використання звичайного текстового редактору створити окрему директорію lab02 для виконання завдань.

Виконайте наступні завдання(кожне завдання має бути виконано в окремому файлі):

**ЗМІСТ**

[1. ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ 1](#_Toc165498818)

[2. ВИКОНАННЯ ЗАВДАНЬ ЛАБОРАТОРНОЇ РОБОТИ 3](#_Toc165498819)

[2.1. Завдання 1 3](#_Toc165498820)

[2.2. Завдання 2 6](#_Toc165498821)

[2.3. Завдання 3 10](#_Toc165498822)

[3. ВИСНОВОК 14](#_Toc165498823)

# ВИКОНАННЯ ЗАВДАНЬ ЛАБОРАТОРНОЇ РОБОТИ

## Завдання 1

|  |  |
| --- | --- |
| 1 | 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)  [додаткова інформація](https://en.wikipedia.org/wiki/Line%E2%80%93line_intersection#Given_two_line_equations) |

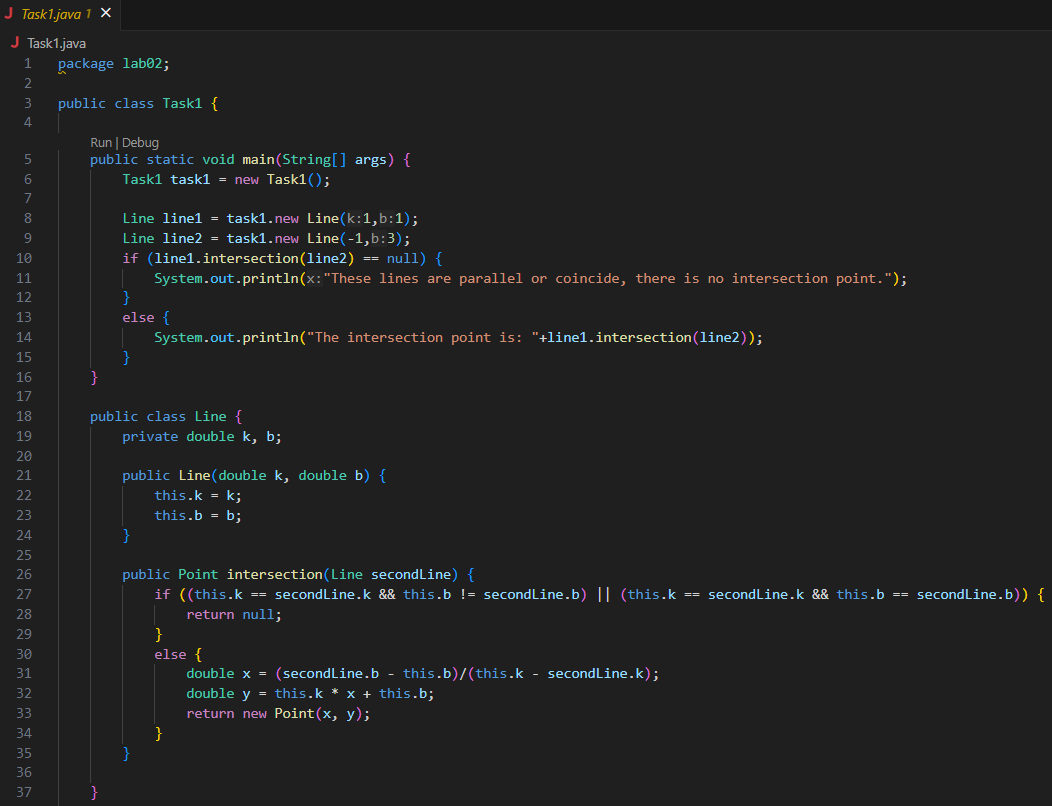
****

Рис. 1. Перше завдання.

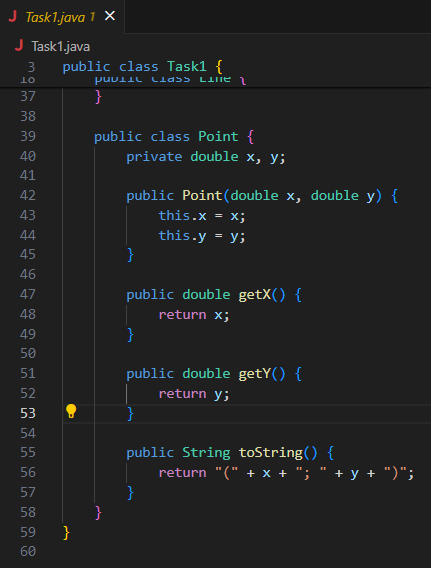
****

Рис. 2. Продовження коду першого завдання.

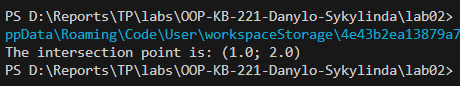
****

Рис. 3. Перевірка роботи коду першого завдання.

Спочатку створюємо клас Line, у його конструкторі визначаємо атрибути k та b (коєфіцієнти для прямої лінії). Це все показано на рисунку 1.

Далі створюємо клас Point, визначаємо атрибути x та y (координати точок по осі Х та по осі У відповідно). Також в цьому класі зробимо два гетери getX() і getY(), які просто повертають значення х та у відповідно, це знадобиться в наступних завданнях. Це все зображено на рисунку 2.

Створивши клас Point, у класі Line тепер створимо метод intersection() типу класу Point, який прийматиме у якості аргументу буде приймати другу лінію(secondLine) типу об’єкта класу Line. Далі робимо перевірку, якщо коєфіцієнт k першої лінії(this.k) дорівнює коєфіцієнту k другої лінії (secondLine.k) і водночас коєфіцієнт b першої лінії(this.b) не дорівнює коєфіцієнту b другої лінії(secondLine.b), або коєфіцієнти k та b у обох лініях однакові, то повернути null. Ця перевірка потрібна, щоб визначити, чи не є ці лінії збіжними або паралельними, тобто, не мають точки перетину між собою. Далі продовжуємо словом else, інакше х = (secondLine.b - this.b)/(this.k - secondLine.k) і y = this.k \* x + this.b та повертаємо створення об’єкту класу Point, який у якості аргументу приймає щойно визначені х та у. Формулу для обчислення координати по осі Х я взяв з довідкового матеріалу, що був наданий до цієї лабораторної роботи. Там також була формула для у, але знаючи всі коєфіцієнти k, b та х, то я можу просто підставити в формулу прямої лінії у = kx + b. Це все показано на рисунку 1.

Також у класі Point перевизначимо вбудований у Java метод toString(), щоб він повертав рядкове представлення об’єкта Point так як ми хочемо. Це зображено на рисунку 2.

Далі в main() створюємо дві змінні типу класу Line і присвоюємо цим зміннам ініціалізацію об’єкта Line з відповідними коєфіцієнтами (для першої лінії k = 1, b = 1 і для другої лінії k = -1, b = 3). Робимо ще перевірку, що якщо для об’єкту line1 викликати метод intersection і передати йому в якості аргументу об’єкт line2 (тобто, мається на увазі цей виклик line1.intersection(line2)) дорівнює null, то вивести повідомлення, що ці лінії збіжні або паралельні, немає точки перетину, інакше вивести повідомлення з результатом виклику методу line1.intersection(line2), що виведе точку перетину між двома лініями. Це все показано на рисунку 1.

Утім коли я запустив код, то з’явилась помилка:

Exception in thread "main" java.lang.Error: Unresolved compilation problem: No enclosing instance of type Task1 is accessible. Must qualify the allocation with an enclosing instance of type Task1 (e.g. x.new A() where x is an instance of Task1). at lab02.Task1.main(Task1.java:8). Виявилося, що ця помилка зазвичай виникає, коли внутрішній клас (у вашому випадку клас Line) намагається бути створеним без екземпляра зовнішнього класу (Task1). У нашому коді клас Line є внутрішнім класом класу Task1, тому він потребує екземпляра зовнішнього класу для створення. Тут було два рішення, або я прибираю клас Task1, тобто, те, що в середині його переносимо код назовні, або створити екземляр Task1() і потім через нього викликати створення об'єктів класу Line. Я обрав другий варіант рішення, оскільки не хотів порушувати свою традицію виконання завдань, а саме створення класу, яке має назву завдання, в якому я вже пишу рішення, тому створив екземпляр і через нього викликав (task1.new Line()) створення об'єктівкласу Line. І все вийшло! Результат виконання коду показано на рисунку 3.

Ця проблема буде з’являтися в наступних завданнях, її я буду вирішувати аналогічним чином.

## Завдання 2

|  |  |
| --- | --- |
| 2 | 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.  [Length-of-a-Line](https://www.wikihow.com/Use-Distance-Formula-to-Find-the-Length-of-a-Line)  [Midpoint](https://www.wikihow.com/Find-the-Midpoint-of-a-Line-Segment#Use-the-Midpoint-Formula)  [two\_points\_on\_each\_line](https://en.wikipedia.org/wiki/Line%E2%80%93line_intersection#Given_two_points_on_each_line) |

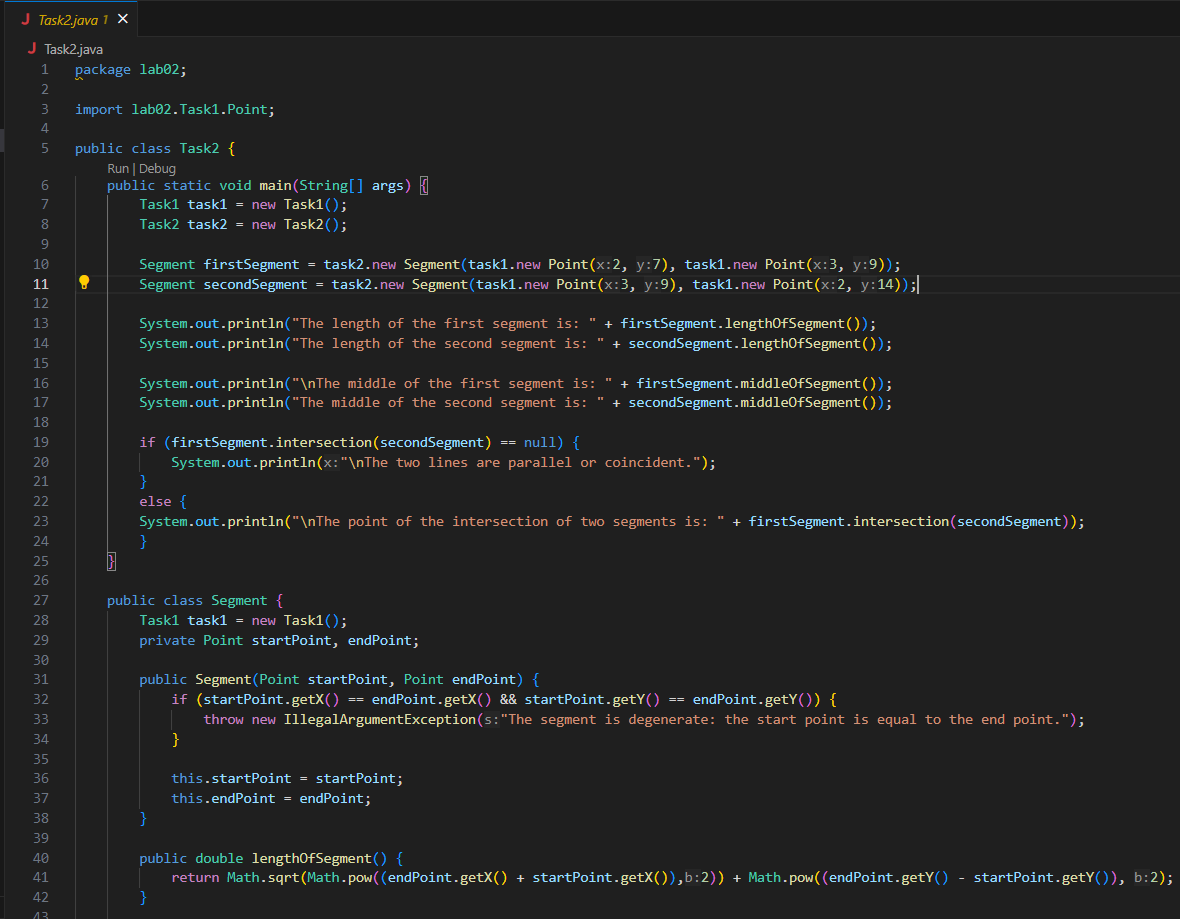


Рис. 4. Друге завдання.

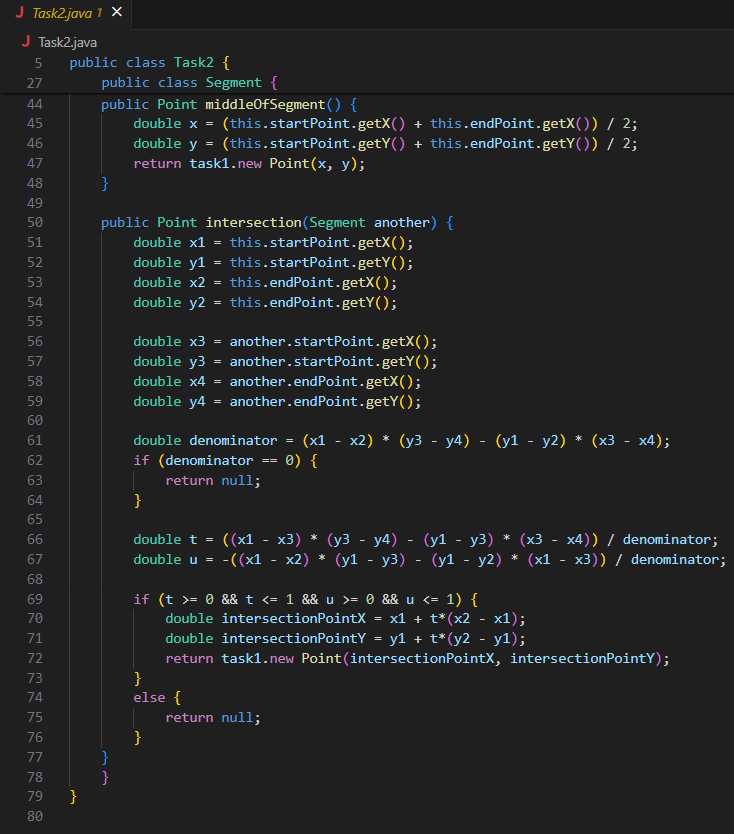


Рис. 5. Продовження коду другого завдання.

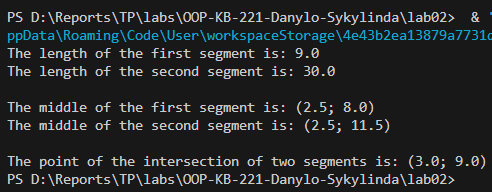


Рис. 6. Результат роботи коду другого завдання.

Імпортуємо клас Point з першого завдання, він нам знадобиться.

Створюємо клас Segment(означає відрізок). У конструкторі він приймає два об’єкти класу Point (початкова та кінцева точка відрізка). Далі в конструкторі робимо перевірку цих параметрів startPoint.getX() == endPoint.getX() && startPoint.getY() == endPoint.getY(), перевіряємо, чи не збігається початок відрізку з кінцем, тобто, чи не є виродженим відрізком. Якщо так, то викликаємо виключення IllegalArgumentException з відповідним повідомленням. Потім визначаємо атрибути. Це показано на рисунку 4.

Далі напишемо метод lengthOfSegment() для обрахування довжини відрізка, який буде повертати результат обчислення за формулою, яка була подана в посиланні, яке в подане в методичці до цієї лабораторної роботи. Це зображено на рисунку 4.

Потім реалізуємо метод Point middleOfSegment(), який буде визначати точку середини відрізку. За формулами, що подані в посиланні, що є в методичці, визначаємо координати точки середини відрізку по осі Х та по осі У. І повертаємо створення об’єкта класу Point(), передавши в параметри щойно знайдені х та у. Це показано на рисунку 4.

Реалізуємо метод Point intersection(Segment another), який отримуємо в якості аргументу отримує другий відрізок і повертає точку перетину з першим відрізком. Спочатку створимо змінні х1, х2, х3, х4 та у1, у2, у3, у4 і присвоїмо координати по осі Х та осі У з обох відрізків для зручності розуміння коду.

Щоб знайти точку перетину відрізків необхідно скористатися шляхом знаходження параметрів (t та u) Безьє першого ступеня (цей спосіб був представлений в посиланні, що є в методичці, також інші дані, що будуть далі використовуватися взяті з цього посилання). Але спочатку знайдемо знаменник дробу (x1 - x2) \* (y3 - y4) - (y1 - y2) \* (x3 - x4), який є при обрахуванні параметра t. І далі напишемо для нього таку умову, що, якщо дорівнює 0, то повернути null (означає, що відрізки або збіжні, або паралельні, тобто, не перетинаються).

Знайдемо параметри Безьє першого ступеня t та u за формулами, що були подані в посиланні. Далі пишемо таку перевірку, що якщо t >= 0 && t <= 1 && u >= 0 && u <= 1 виконується, то знаходимо точку перетину по осі Х (intersectionPointX) та по осі У(intersectionPointУ) за формулами, що подані за посиланням. І повертаємо створення об’єкту класу Point, якому передаємо щойно знайдені координати. Якщо умова не спрацювала, то повертаємо null, тобто, два відрізки не мають точки перетину. Цей код показано на рисунку 5.

Далі в main() створюємо дві змінні і присвоюємо їм два об’єкти класу Segment, куди в якості параметрів передаємо два об’єкти(точки) класу Point кожному, яким в якості параметрів передаємо бажані координати відповідно. Потім для кожного з відрізків викликаємо методи lengthOfSegment() та middleOfSegment() і виводимо повідомлення з результатом. Далі як у попередньому завданні, я перевіряю метод intersection(), чи не дорівнює його результат null, якщо так, то виводжу повідомлення, що відрізки паралельні або збіжні. Тобто, означає, що не перетинаються. Якщо не null, то виводжу в повідомленні результат роботи методу intersection(). Цей код щодо цього показано на рисунку 4. Результат роботи коду завдання показано на рисунку 6.

## Завдання 3

|  |  |
| --- | --- |
| 3 | 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.  [Existence\_of\_a\_triangle](https://en.wikipedia.org/wiki/Triangle#Existence_of_a_triangle)  [Computing\_the\_area\_of\_a\_triangle](https://en.wikipedia.org/wiki/Triangle#Computing_the_area_of_a_triangle)  [Centroid](https://en.wikipedia.org/wiki/Centroid) |

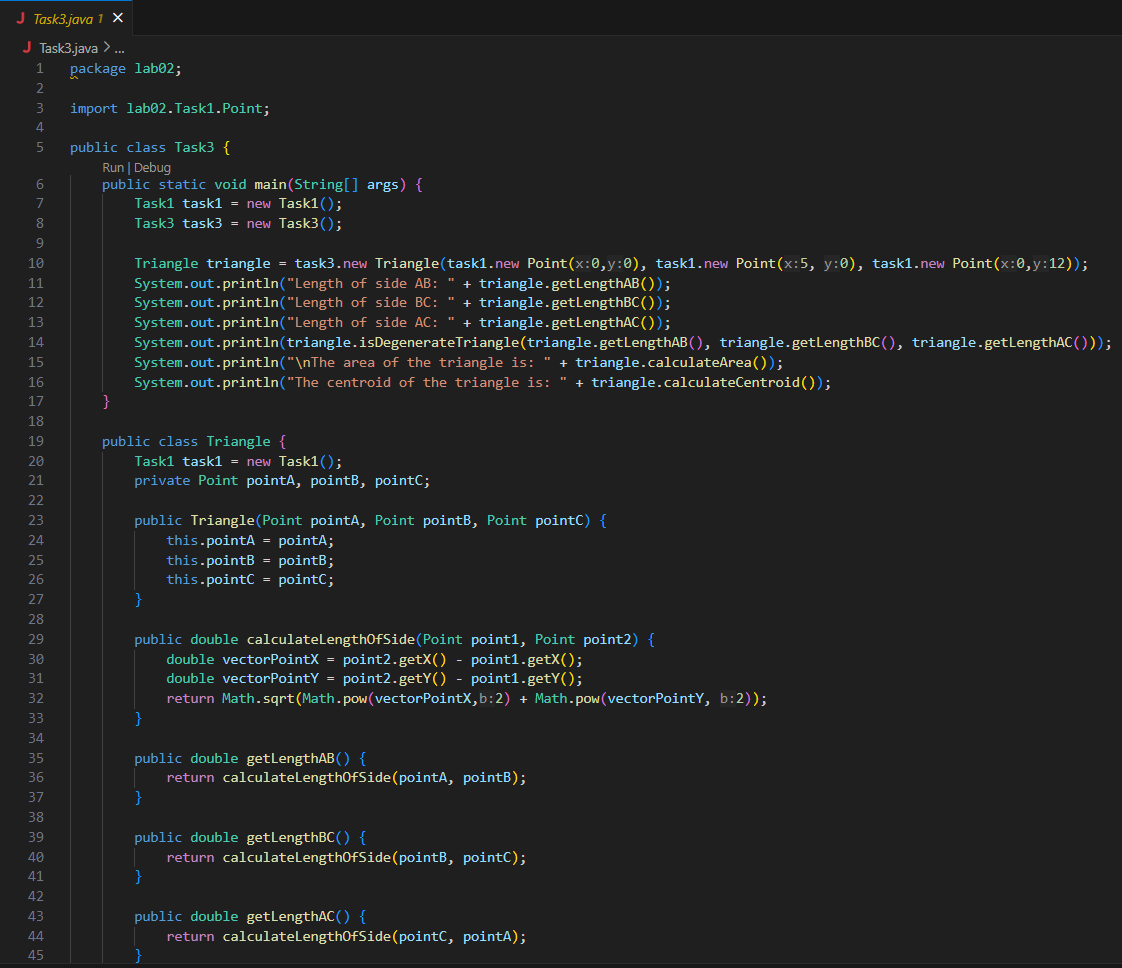


Рис. 7. Третє завдання.



Рис. 8. Продовження коду третього завдання.

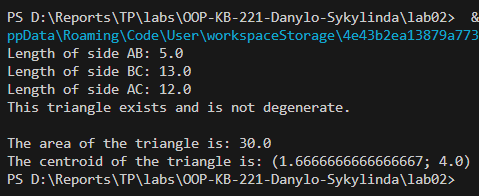


Рис. 9. Результат роботи коду третього завдання.

Спочатку імпортуємо клас Point з першого завдання. Далі створюємо клас Triangle. Його конструктор буде приймати у якості аргументу три точки(три об’єкти класу Point), які є вершинами трикутника. І також потім визначаємо атрибути.

Потім реалізуємо метод calculateLengthOfSide(), який приймає дві точки(два об’єкти класу Point) і за формулами знаходить довжину сторони(довжину вектора). Ці формули на фото нижче.



Рис. 10. Формули для знаходження довжини сторін, знаючи координати двох точок.

І цей метод повертає результат формули, що справа на рисунку 10.

Далі я створюю методи-гетери, які просто будуть повертати виклики функції calculateLengthOfSide() з відповідними аргументами для кожної з сторін. Три сторони – три методи-гетери. Це все, що було до цього, показано на рисунку 7.

Потім я реалізовую метод isDegenerateTriangle(), у якому я прописую перевірку, що якщо (a + b > c && b + c > a && c + a > b) не спрацює, то кидає виключення IllegalArgumentException() з повідомленням «Трикутник не існує та/або є виродженим». Якщо спрацює умова, то повернути повідомлення, що «Цей трикутний існує і не є виродженим.»

Тепер реалізовуємо метод calculateArea(), який буде обраховувати площу за формулою Герона, що на фото нижче.



Рис. 11. Формула знаходження площі трикутника, де p – півпериметр, a, b, c – сторони трикутника.

Спочатку знаходимо півпериметр, далі вже площу трикутника і повертаємо її як результат.

Далі створимо метод Point calculateCentroid(), який буде обраховувати центроїд (точка) трикутника. Його координати обраховуються за формулами, що подані в посиланні, що є методичці. Повертаємо як результат створення об’єкту класу Point, якому в якості параметра передані щойно знайдені параметри точки. Цей код щодо цього показаний на рисунку 8.

І тепер в main() створюємо об’єкт класу Triangle і передаємо йому в якості аргументів три точки(три об’єкти класу Point) з координатами. Далі ми виводимо сторони, щоб переконатися в правильності обчислення, виводимо результати перевірки на існування та виродженість трикутника, результати обрахунку площі та обрахування точки центроїда трикутника. Результати роботи коду показано на рисунку 9.

# ВИСНОВОК

Я ознайомився з поданим матеріалом, що був наданий в методичці, зрозумів краще основи ООП. Я виконав всі завдання лабораторної роботи. Як результат, покращив своє розуміння щодо концепції ООП, щодо принципів рішення задач, також навчився краще вирішувати проблеми з помилками, що можуть виникнути під час написання коду. Завантажив всі виконані завдання та звіт лабораторної роботи на Github та Moodle.