МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

**Лабораторна робота № 4**

з дисципліни « Програмування мовою Java »

*назва дисципліни*

на тему: «РОЗРОБКА ЗАСТОСУНКІВ МОВОЮ ПРОГРАМУВАННЯ

JAVA. ОПЕРАТОРИ: ПОБІТОВІ, ПОРІВНЯННЯ, ЛОГІЧНІ»

Виконав: студент 2 курсу групи № 622п

освітньої програми

121 інженерія програмного забезпечення

(шифр і назва ОП)

Зайченко Ярослав Ігорович

(прізвище й ініціали студента)

Прийняв: доцент

Симбірський Генадій Дмитрович

(посада, науковий ступінь, прізвище й ініціали)

Кількість балів:

Харків – 2023

ЗМІСТ

[**ПОСТАНОВКА ЗАВДАННЯ 3**](#_Toc148216875)

[**ЗМІСТ ЗВІТУ 3**](#_Toc148216876)

[**ПОРЯДОК ВИКОНАННЯ РОБОТИ 3**](#_Toc148216877)

[**ВИКОНАННЯ РОБОТИ 4**](#_Toc148216878)

[**ЗАГАЛЬНІ ВІДОМОСТІ 4**](#_Toc148216879)

[**ВИСНОВОК 11**](#_Toc148216880)

[**КОНТРОЛЬНІ ПИТАННЯ 12**](#_Toc148216881)

Мета роботи: отримання практичних навичок роботи з операторами мови програмування Java: побітові, порівняння, логічні.

ПОСТАНОВКА ЗАВДАННЯ

Навчитися працювати з операторами: побітові, порівняння, логічні

# **ЗМІСТ ЗВІТУ**

1. Постановка завдання.
2. Встановити завдання, які вирішуються в лабораторній роботі
3. Коротко описати побітові оператори.
4. Коротко описати оператори порівняння.
5. Коротко описати логічні оператори.
6. Висновки по роботі.

ПОРЯДОК ВИКОНАННЯ РОБОТИ

1. Прочитати теоретичний матеріал
2. Вивчити побітові оператори
3. Вивчити оператори порівняння
4. Вивчити логічні оператори
5. Зробити висновки по роботі.

# **ВИКОНАННЯ РОБОТИ**

## **Загальні відомості**

Мова програмування – Java

Операційна система – Windows 11

Процесор – Apple Silicon M1 Pro with 4 CPU cores

Тип компілятора – IntelliJ IDEA 2023

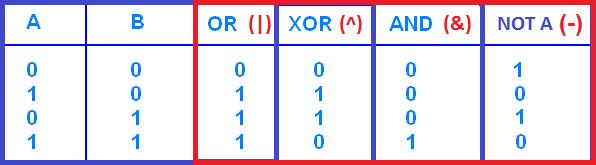
Побітові оператори

Працюючи з будь-якими цілочисельними типами, можна застосовувати операції, безпосередньо обробні біти, з яких складаються цілі числа. Це означає, що для визначення стану окремих бітів числа можна використовувати маски.

|  |  |
| --- | --- |
| Оператор | Результат |
| ~ | Поразрядна унарна операція НЕ (доповнення) |
| & | Поразрядна логічна операція І (AND, побітова кон'юнкція) |
| | | Поразрядна логічна операція АБО (OR, побітова  диз'юнкція) |
| ^ | Поразрядна логічна операція виключаюче АБО (XOR) |
| >> | Зсув вправо (арифметичний Зсув) |
| >>> | Зсув вправо із заповненням нулями (беззнаковий зсув) |
| << | Зсув вліво |

|  |  |
| --- | --- |
| Оператор | Результат |
| &= | Порозрядна логічна операція І (AND, побітова кон'юнкція) з присвоюванням |
| |= | Поразрядна логічна операція АБО (OR, побітова диз'юнкція) з присвоюванням |
| ^= | Поразрядна логічна операція виключає АБО (XOR) з присвоюванням |
| >>= | Зсув вправо (арифметичний Зсув) з присвоюванням |
| >>>= | Зсув вправо із заповненням нулями (беззнаковий Зсув) з присвоюванням |
| <<= | Зсув вліво з присвоюванням |

Результати виконання побітових логічних операцій:



Оператори порівняння

Результатом виконання операторів порівняння виявляється логічне значення. Найчастіше оператори порівняння застосовуються в виразах, керуючих умовним оператором if і різними операторами циклу.

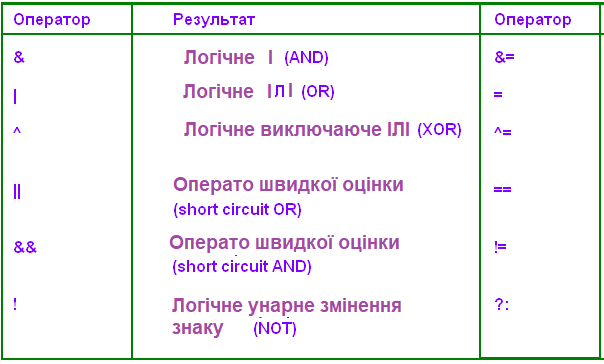
Допускається порівнювати значення будь-яких типів за допомогою операцій == і ! =.

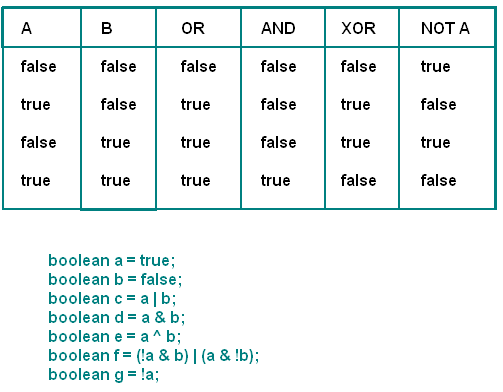
Порівняння за допомогою операцій упорядкування допускається тільки для числових типів даних.



Логічні оператори

Boolean в Java - це логічний тип даних. Змінна типу boolean може приймати всього два значення - це true або false. Ці два значення позначаються в інших мовах і часто видаються на екран як 1 і 0, але все ж не рівні цих значень: наприклад, вираз boolean b = 0; приведе до помилки при компіляції програми. Але ми можемо порівняти змінні або виконати логічну операцію з типом даних boolean:





Завдання 4.0**.** Знайдіть у документації по Java опис методу Integer.*toString*(a,2).

Метод `Integer.toString(a, 2)` використовується для перетворення цілого числа `a` у рядок у двійковій системі числення.

Параметр `a` - це ціле число, яке бажають перетворити.

Другий параметр, `2`, вказує, що необхідно перетворити число в двійкову систему числення.

Наприклад:

int number = 10;

String binaryString = Integer.toString(number, 2);

У цьому прикладі `binaryString` буде рядком, що містить двійкове представлення числа 10, тобто "1010".

Завдання 4.1**.** Розробити, відлагодити та запустити програму для обчислень (рис. 1 в “Методичних вказівках до виконання лабораторної роботи 4”).

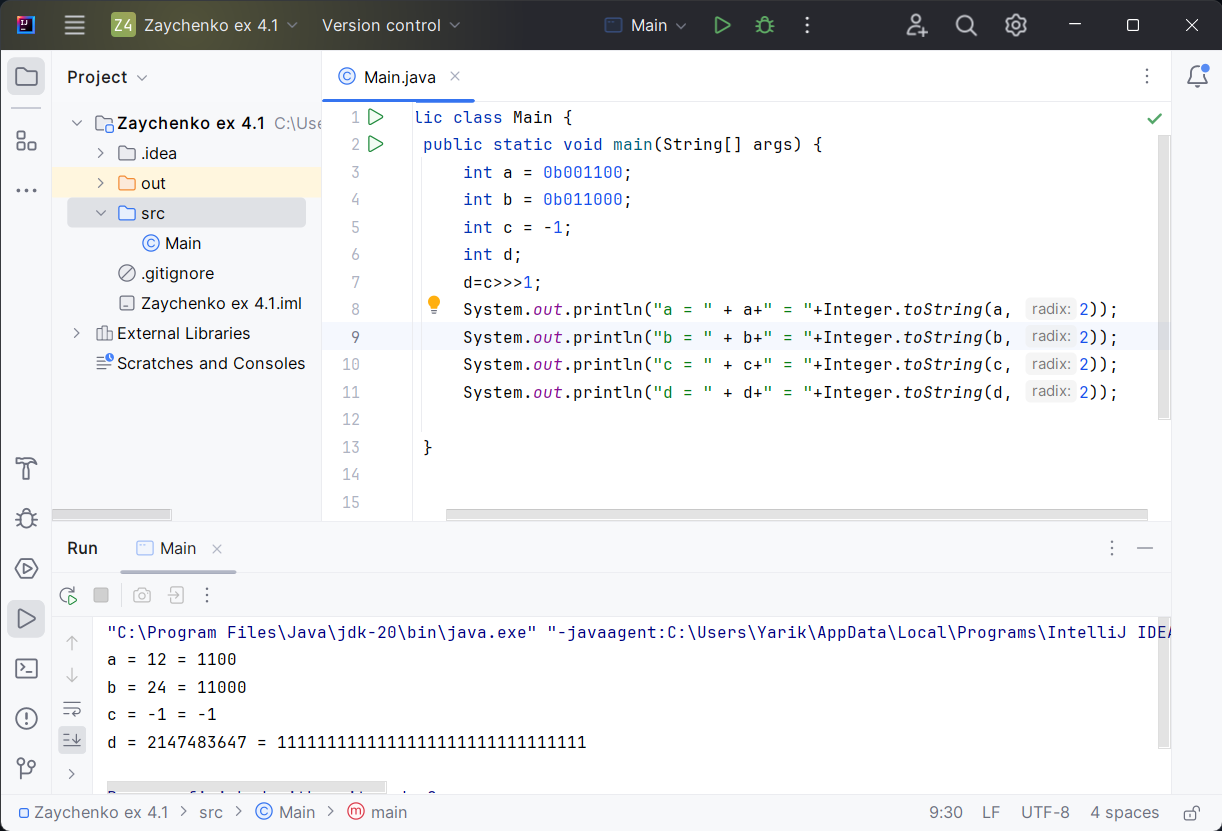


Рисунок 1 – Використання бітових операторів мови програмування Java

Завдання 4.2

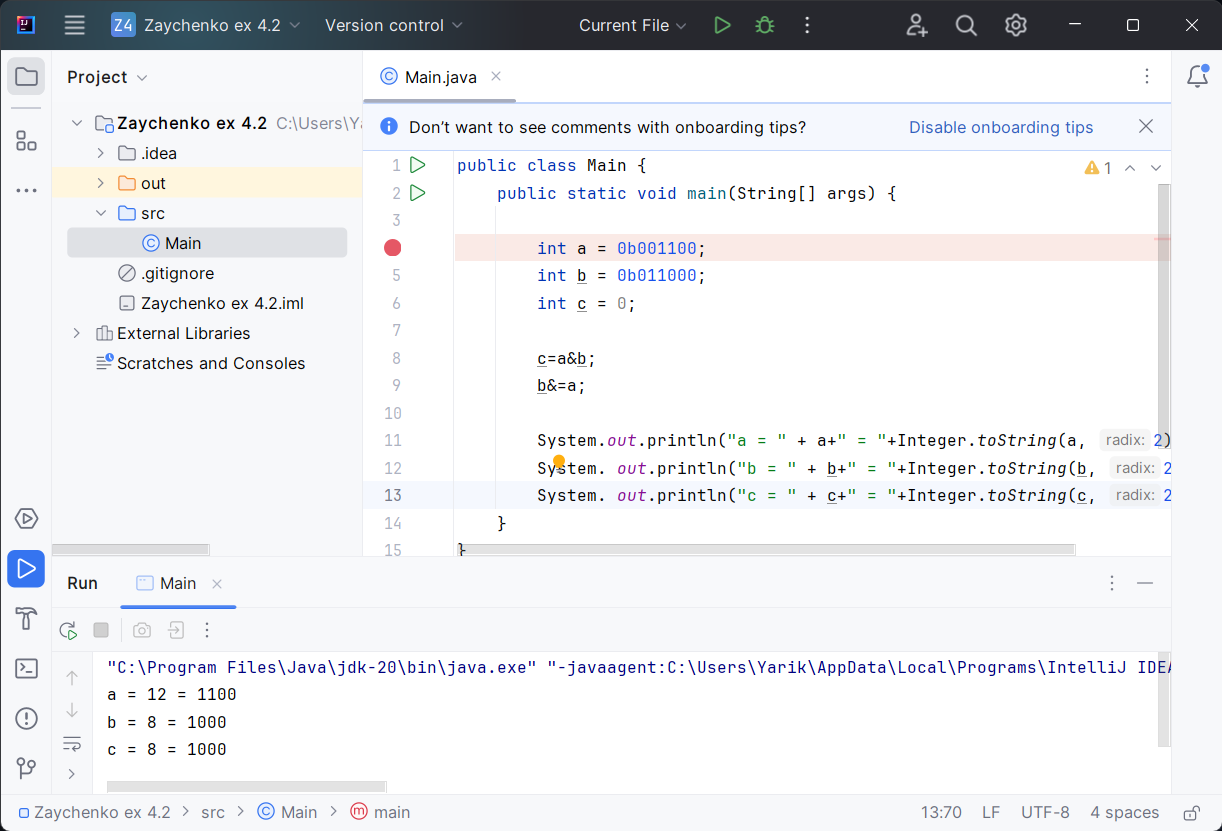
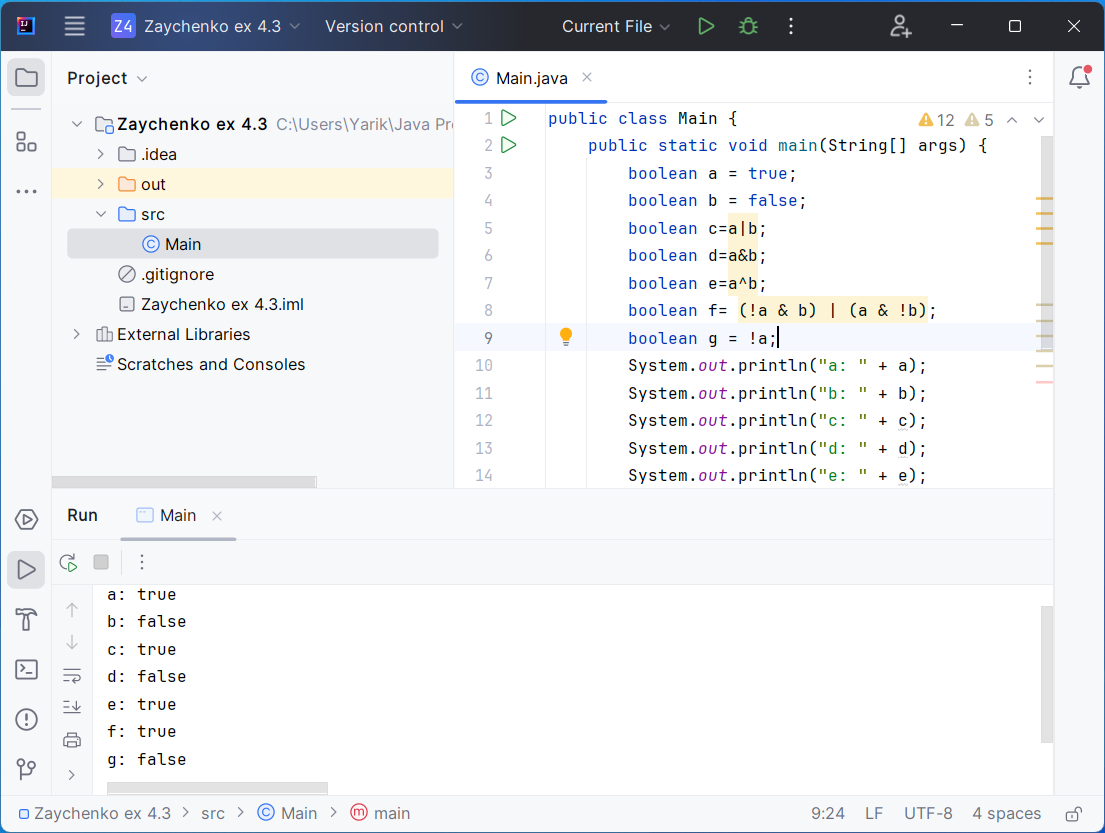


Рисунок 2 – Використання побітових операторів мови програмування Java

Завдання 4.3

Рисунок 3 – використання булевих (логічних) операторів

Завдання 4.4

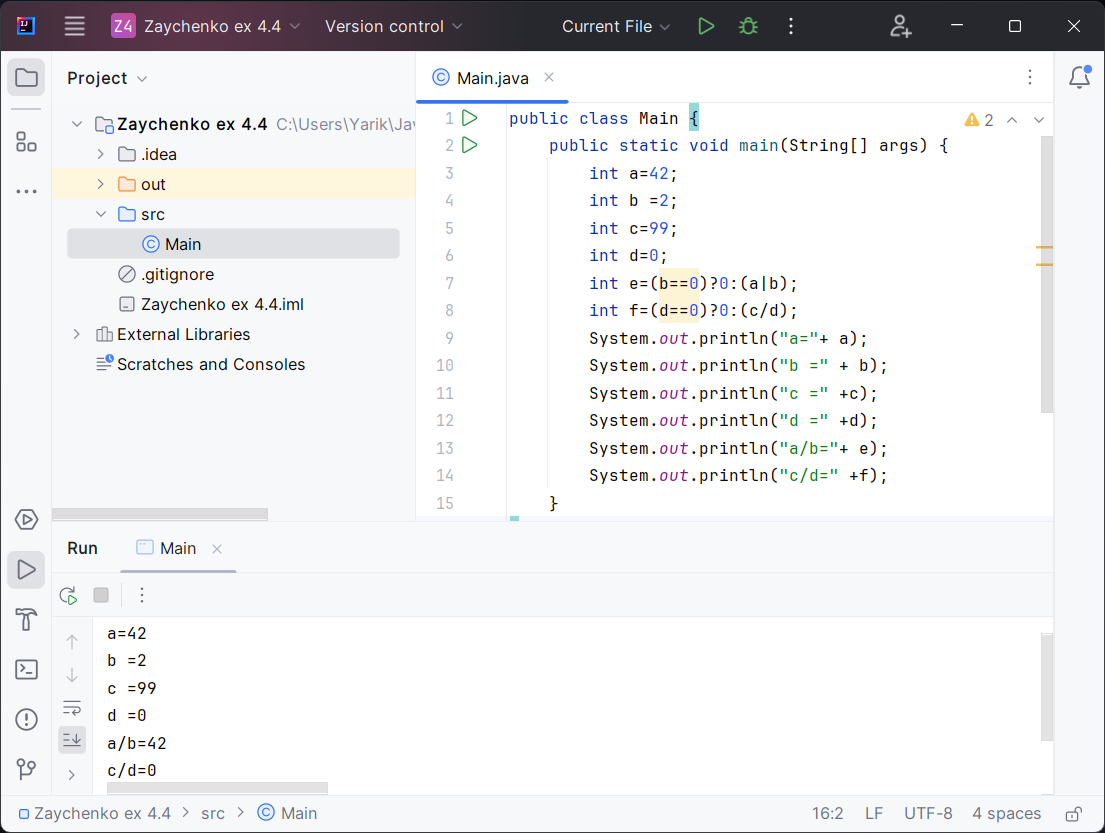


Рисунок 4 –використання тернарних операторів

Завдання 4.5. Вирішити квадратне рівняння використовуючи тернарний оператор

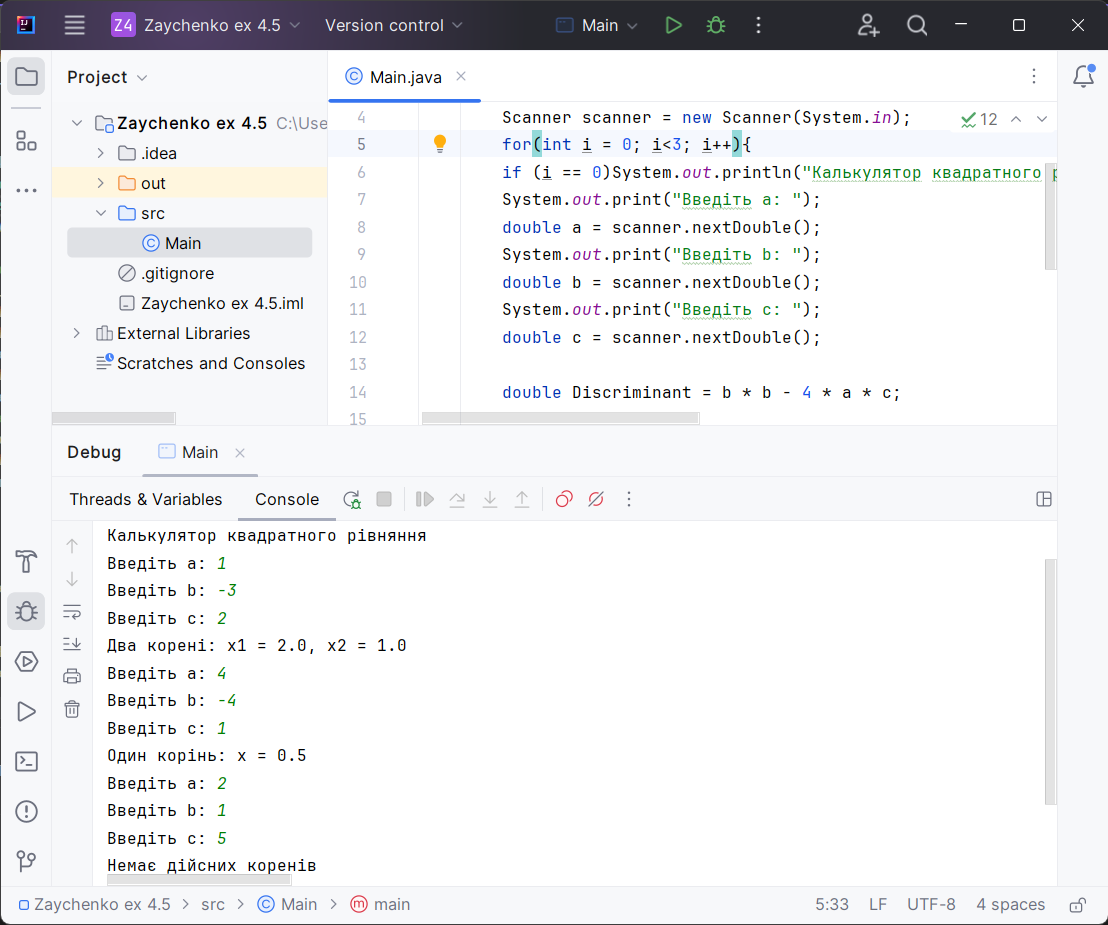


Рисунок 5 – Відладка коду до задачі 4.5

Завдання 4.6. Є 5 цілочисельних змінних. Необхідно знайти найбільшу

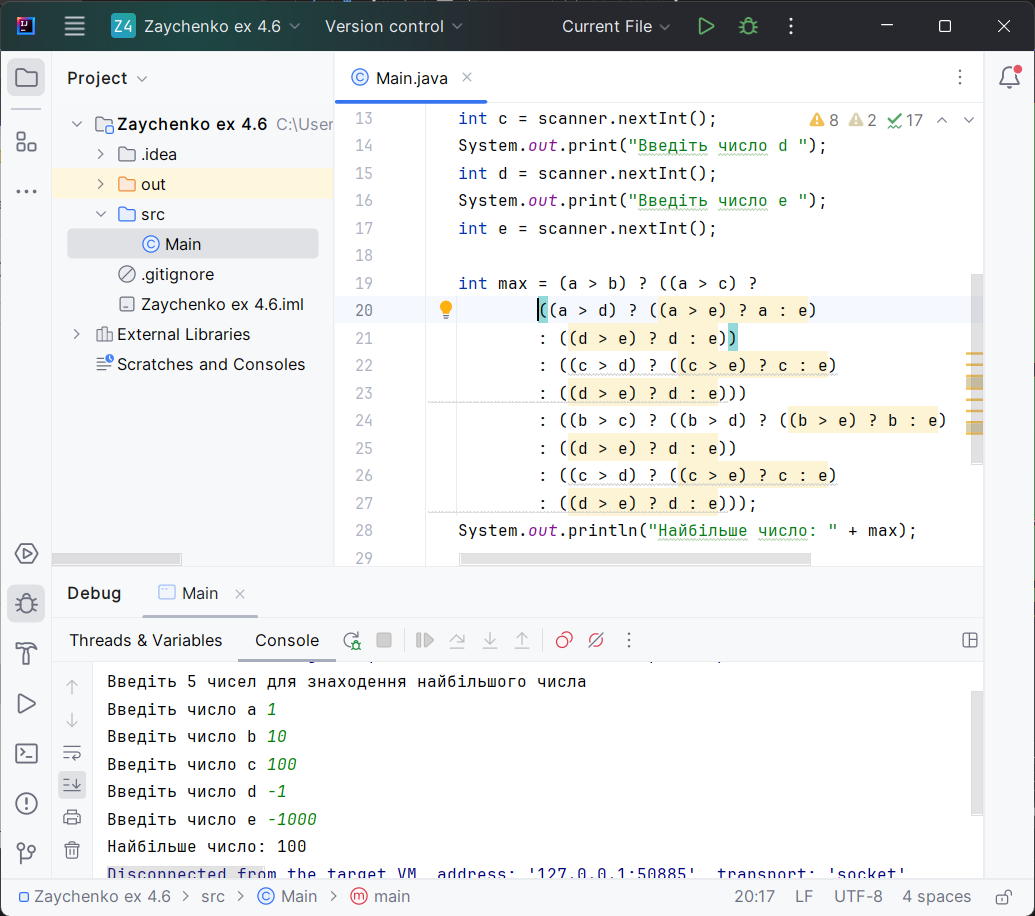


Рисунок 6 – Відладка коду до задачі 4.6

Завдання 4.7. Є 5 цілочисельних змінних. Необхідно знайти дві набільші

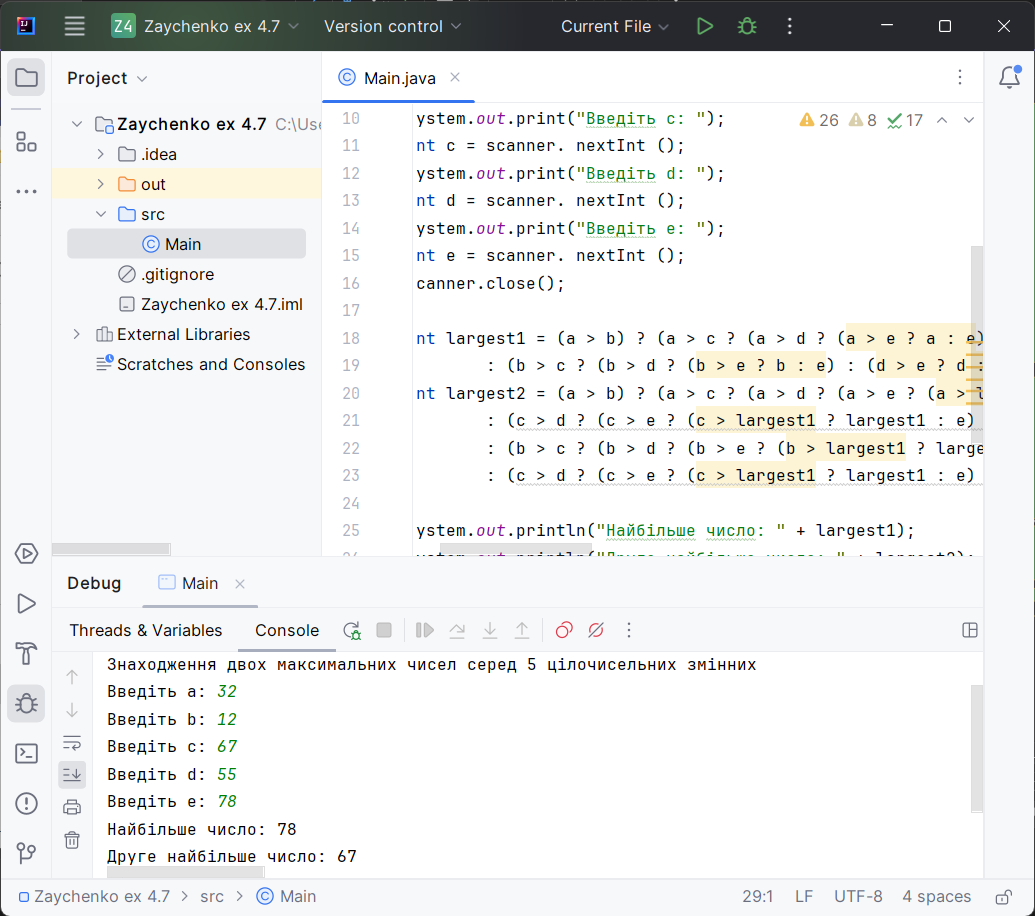


Рисунок 7 – Відладка коду до задачі 4.7

# **ВИСНОВОК**

Отримано практичні навички роботи з операторами мови.

# **Контрольні питання**

1. Які побітові оператори ви знаєте?

В Java існують наступні побітові оператори:

* Побітовий І (&): Виконує операцію "І" над кожним бітом двох чисел.
* Побітовий АБО (|): Виконує операцію "АБО" над кожним бітом двох чисел.
* Побітовий виключаюче АБО (^): Виконує операцію "виключаюче АБО" над кожним бітом двох чисел.
* Зсув вліво (<<): Здвигає біти числа вліво на вказану кількість позицій.
* Зсув вправо (>>): Здвигає біти числа вправо на вказану кількість позицій з знакозбереженням (для знакових цілих) або з заповненням нулями (для беззнакових цілих).
* Зсув вправо з заповненням нулями (>>>): Здвигає біти числа вправо на вказану кількість позицій, заповнюючи нові позиції нулями.

1. Які оператори порівняння ви знаєте?

* У Java існують наступні оператори порівняння, які використовуються для порівняння значень:
* `==` (рівність): Порівнює два значення на рівність. Повертає `true`, якщо значення рівні, і `false`, якщо ні.
* `!=` (не рівність): Порівнює два значення на нерівність. Повертає `true`, якщо значення не рівні, і `false`, якщо вони рівні.
* `>` (більше): Порівнює, чи перше значення більше за друге. Повертає `true`, якщо перше значення більше, і `false` в іншому випадку.
* `<` (менше): Порівнює, чи перше значення менше за друге. Повертає `true`, якщо перше значення менше, і `false` в іншому випадку.
* `>=` (більше або рівне): Порівнює, чи перше значення більше або рівне другому. Повертає `true`, якщо перше значення більше або рівне, і `false` в іншому випадку.
* `<=` (менше або рівне): Порівнює, чи перше значення менше або рівне другому. Повертає `true`, якщо перше значення менше або рівне, і `false` в іншому випадку.

Ці оператори дозволяють порівнювати значення та визначати, чи вони відповідають певним умовам.

1. Які логічні оператори ви знаєте?

* У Java існують три основних логічних оператори:
* `&&` (логічне І): Повертає `true`, якщо обидва операнди є `true`. В іншому випадку повертає `false`.
* `||` (логічне АБО): Повертає `true`, якщо хоча б один з операндів є `true`. Якщо обидва операнди `false`, то повертає `false`.
* `!` (логічне НЕ): Цей оператор використовується для інвертування значення операнду. Наприклад, `!true` повертає `false`, і навпаки.

Ці логічні оператори використовуються для створення складних умовних виразів і логічних операцій в програмах на Java.

1. Що таке пріоритетність операторів?

Пріоритетність операторів в мові програмування визначає, який оператор буде виконуватися першим у виразі з кількома операторами. Наприклад, в математичних виразах множення та ділення мають вищий пріоритет, ніж додавання та віднімання, тому вираз "5 \* 2 + 3" обчислиться як "10 + 3" і результат буде 13. Пріоритетність операторів можна змінювати за допомогою дужок, вказуючи, які операції повинні виконуватися першими.