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 № 2**

з дисципліни « Програмування мовою Java »

*назва дисципліни*

на тему: «РОЗРОБКА ЗАСТОСУНКІВ МОВОЮ ПРОГРАМУВАННЯ JAVA. ТИПИ ДАНИХ»

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](#_Toc145352814)

[ВИКОНАННЯ РОБОТИ 4](#_Toc145352815)

[Загальні відомості 4](#_Toc145352816)

[Завдання 1.1. 4](#_Toc145352817)

[Завдання 1.2. 4](#_Toc145352818)

[Завдання 1.3. 6](#_Toc145352819)

[Завдання 1.4. 7](#_Toc145352820)

[Завдання 1.5. 7](#_Toc145352821)

[ВИСНОВОК 9](#_Toc145352822)

[Контрольні питання 10](#_Toc145352823)

**Мета роботи:** отримання практичних навичок роботи з простими типами даних.

# **ПОСТАНОВКА ЗАВДАННЯ**

Вивчити прості типи в Java.

# **ЗМІСТ ЗВІТУ**

1. Постановка завдання.
2. Встановити завдання, які вирішуються в лабораторній роботі
3. Коротко описати особливості цілочисельних типів.
4. Коротко описати властивості дійсних типів.
5. Коротко описати особливості булева типу.
6. Коротко описати поняття констант, керуючих символів і таблиці UNICODE.
7. Виконати завдання 2.1 – 2.7.
8. Висновки по роботі.

**ПОРЯДОК ВИКОНАННЯ РОБОТИ**

1. Прочитати теоретичний матеріал
2. Вивчити структуру типів
3. Вивчити цілочисельні типи даних
4. Вивчити дійсні типи даних
5. Вивчити логічний тип даних
6. Поняття констант
7. Керуючі символи
8. Особливості цілочисленного типу char
9. 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

На додаток до int, мова програмування Java підтримує сім інших примітивних типів даних. Примітивний тип заздалегідь визначається мовою і називається зарезервованим ключовим словом. Примітивні значення не поділяють стан з іншими примітивними значеннями. Мова програмування Java підтримує вісім примітивних типів даних:

У цьому розділі обговорюється правила і умови іменування змінних, основні типи даних (примітивні типи,), значення за замовчуванням та літерали.

* **byte**:Тип даних байта-це 8-розрядне ціле число з двома знаками зі знаком. Він має мінімальне значення -128 і максимальне значення 127 (включно). Байтовий тип даних може бути корисним для збереження пам'яті у великих масивах, де економія пам'яті насправді має значення. Вони також можуть бути використані замість int, де їх обмеження допомагають уточнити ваш код; той факт, що діапазон змінної обмежений, може служити формою документації.
* **short**: короткий тип даних-це 16-розрядне ціле число з двома знаковими знаками. Він має мінімальне значення -32,768 і максимальне значення 32,767 (включно). Як і у випадку з байтом, застосовуються ті ж рекомендації: ви можете використовувати короткий для збереження пам’яті у великих масивах у ситуаціях, коли економія пам’яті насправді має значення.
* **int**: За замовчуванням тип даних int є 32-розрядним цілим числом із двома знаками зі знаком, яке має мінімальне значення -231 і максимальне значення 231-1. У Java SE 8 і пізніших версіях можна використовувати тип даних int для представлення 32-розрядного цілого числа без знака, яке має мінімальне значення 0 і максимальне значення 232-1. Використовуйте клас Integer, щоб використовувати тип даних int як ціле число без знака. Додаткову інформацію див. У розділі Класи чисел. Статичні методи, такі як compareUnsigned, divideUnsigned тощо, були додані до класу Integer для підтримки арифметичних операцій для цілих чисел без знака.
* **long**: довгий тип даних-це 64-розрядне ціле число з двома доповненнями. Підписаний лонг має мінімальне значення -263 і максимальне значення 263-1. У Java SE 8 та пізніших версіях можна використовувати довгий тип даних для представлення 64-розрядної беззнакової довжини, яка має мінімальне значення 0 і максимальне значення 264-1. Використовуйте цей тип даних, коли вам потрібен ширший діапазон значень, ніж ті, які надає int. Клас Long також містить методи, такі як compareUnsigned, divideUnsigned тощо для підтримки арифметичних операцій протягом довгого беззнаку.
* **char**: Тип даних char-це один 16-розрядний символ Unicode. Він має мінімальне значення "\ u0000" (або 0) і максимальне значення "\ uffff" (або 65 535 включно).
* **float**: Тип даних з плаваючою системою є одноточною 32-розрядною плаваючою точкою IEEE 754. Його діапазон значень виходить за рамки цього обговорення, але він зазначений у розділі Типи, формати та значення з плаваючою комою Специфікації мови Java. Як і в рекомендаціях щодо байтів і коротких, використовуйте плаваючу (замість подвійної), якщо вам потрібно зберегти пам'ять у великих масивах чисел з плаваючою комою. Цей тип даних ніколи не слід використовувати для точних значень, таких як валюта. Для цього вам потрібно буде використовувати клас java.math.BigDecimal. Numbers and Strings охоплює BigDecimal та інші корисні класи, що надаються платформою Java.
* **double**: Подвійний тип даних-це 64-розрядна плаваюча крапка з подвійною точністю IEEE 754. Його діапазон значень виходить за рамки цього обговорення, але він зазначений у розділі Типи, формати та значення з плаваючою комою Специфікації мови Java. Для десяткових значень цей тип даних зазвичай є вибором за замовчуванням. Як згадувалося вище, цей тип даних ніколи не слід використовувати для точних значень, таких як валюта.
* **boolean**: Булевий тип даних має лише два можливих значення: true та false. Використовуйте цей тип даних для простих прапорів, які відстежують істинні/хибні умови. Цей тип даних представляє один біт інформації, але його "розмір" не є чимось точно визначеним.

Константи

**final** - це модифікатор, що дозволяє оголошувати константні поля в класі. Якщо у вас є певна властивість проектованого вами об'єкта, значення якого не буде змінюватися, то ви можете скористатися цим модифікатором. Будь-яка спроба перевизначити значення поля з модифікатором final призводить до викиду винятку.

**Static** поле: існує на рівні класу, а не об'єкта; Для кожного класу створюється тільки один екземпляр статичної змінної (зміною класу).

При описі статичних змінних повинні бути виконані наступні правила:

* статичні змінні повинні бути оголошені поза всяким методу;
* статичні змінні можуть бути ініційовані при їх оголошенні або блоком коду, зазначеного ключовим словом static, і також розташованим поза методу.

Керуючі символи

У програмування на Java символ, якому передує знак зворотної косої риски (\), називається керуючої послідовністю і має особливе значення для компілятора.

Символ переходу на новий рядок (\ n) часто використовується в наших прикладах в System.out.println (), оператор переносить на наступний рядок після надрукованій рядки.

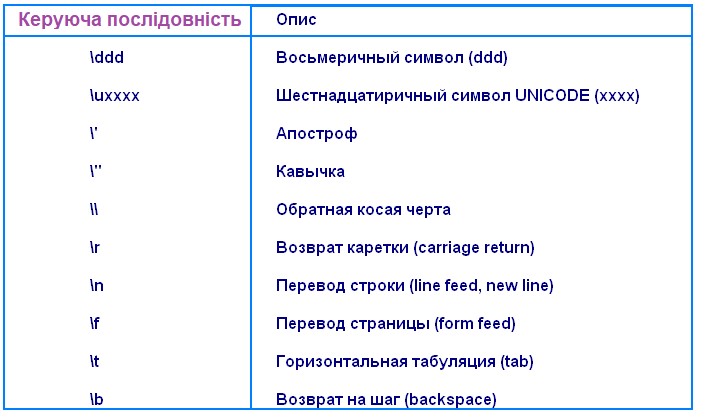


Рисунок 1 – керуючі послідовності, використовувані в Java

Таблиця UNICODE

Це стандартизований набір символів та їх кодів, який призначений для представлення текстової інформації в різних мовах та підтримки різноманітних символів і символьних систем. Вона включає в себе широкий спектр символів, включаючи букви, цифри, пунктуаційні знаки, математичні символи, символи валют, і багато інших. Кожен символ у таблиці Unicode має свій унікальний числовий код, який називається кодом Unicode.

Unicode розроблена з метою створення єдиного і універсального стандарту для кодування символів у всьому світі, незалежно від мови або платформи. Це дозволяє комп'ютерам та програмам правильно обробляти і відображати тексти, що містять символи різних мов та писемностей.

Основна перевага таблиці Unicode полягає в тому, що вона дозволяє одночасно підтримувати різні мови та культури, що робить її важливим інструментом для міжнародного програмування, розробки веб-сайтів, обробки текстів та багато інших застосувань.

**Завдання 2.1** Розробити, відлагодити та запустити програму для обчислень (рис. 2 в “Методичних вказівках до виконання лабораторної роботи 2”). Назви файлу-звіту, проектів робити як в прикладі завдання 1-3). (рис. 2)

1. Проаналізуйте помилку та повідомлення

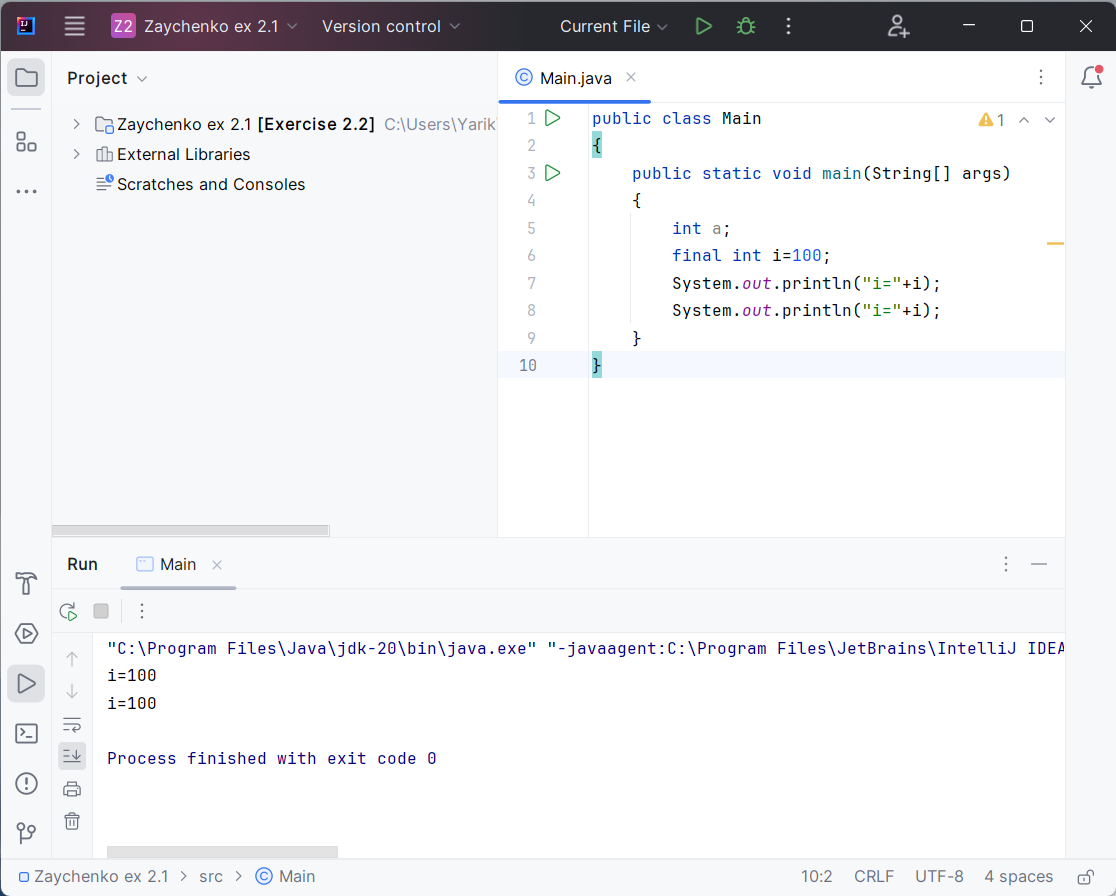


Рисунок 2 – Результат компіляції коду до завдання 2.1 (а)

На рисунку 2 середовище розробки показало повідомлення про те, що змінна а не ініціалізована та ні разу не використовується у програмі.

Ключове слово **final** вказує на те, що значення змінної **i** не може бути змінено після того, як воно було присвоєно. **i** ініціалізована значенням 100, і це значення не може бути змінено в подальших операціях в коді.

Програма зкомпілювалась без помилок та закінчила процес з кодом 0.

1. Розробити, відлагодити та запустити програму для обчислень. Проаналізуйте помилку та повідомлення (рис.3).

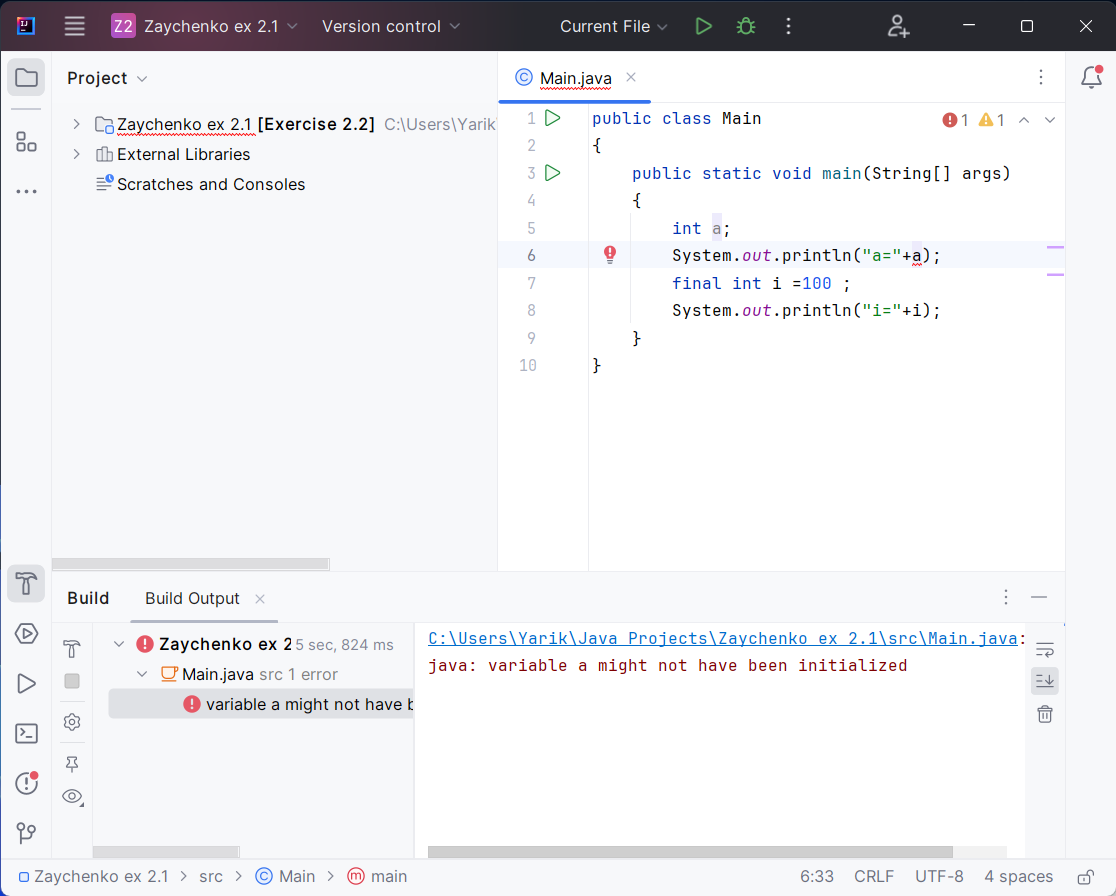


Рисунок 3 – Результат компіляції коду до завдання 2.1 (б)

Компілятор видає помилку «змінна **a** могла бути не ініціалізована», та не пропускає через цю помилку програму на виконання. Обов'язкова ініціалізація локальних змінних визначена [специфікацією Java](https://docs.oracle.com/javase/specs/jls/se8/html/jls-16.html).

1. Розробити, відлагодити та запустити програму для обчислень. Проаналізуйте помилку та повідомлення (рис.4).

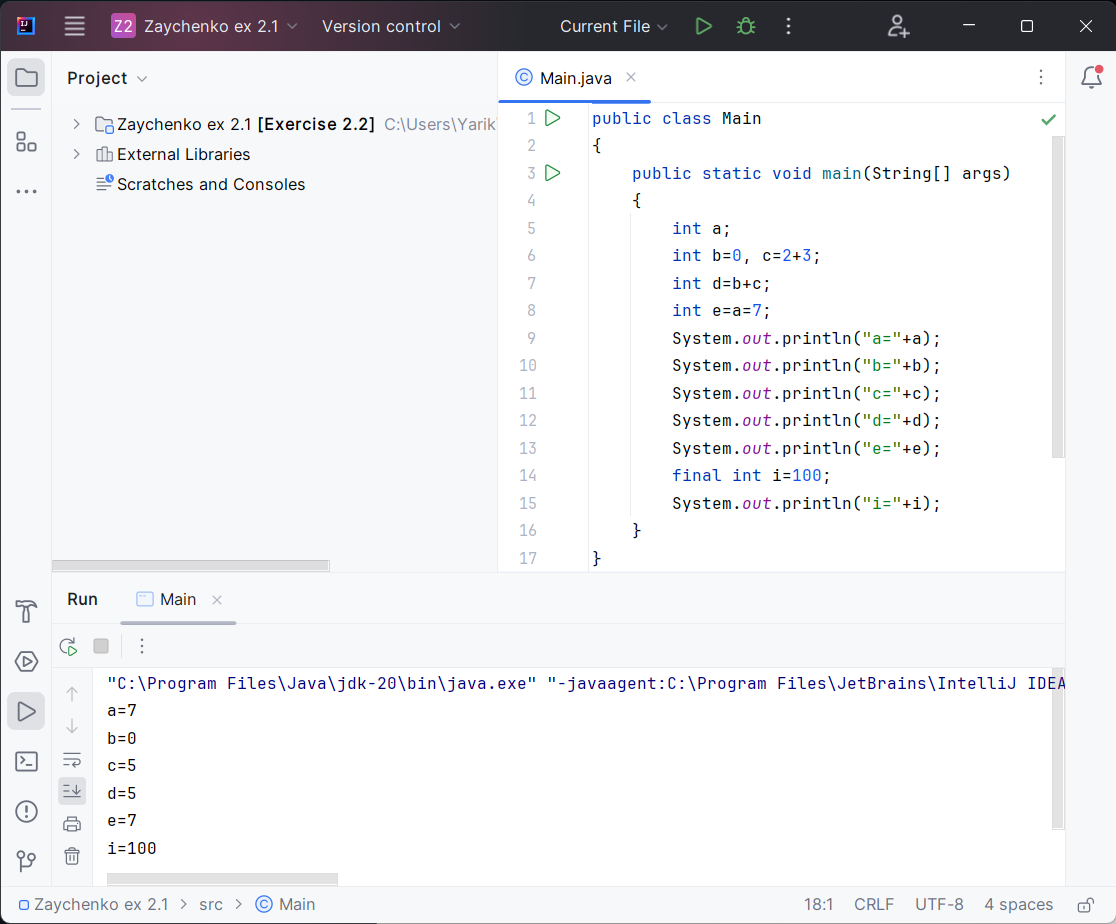


Рисунок 4 – Результат компіляції коду до завдання 2.1 (в)

Помилок не виникає, оскільки всі змінні в програмі ініціалізовані правильно. Значення змінної **і** змінити вже не можна через використання **final**.

**Завдання 2.2** (приклад після рис. 2)

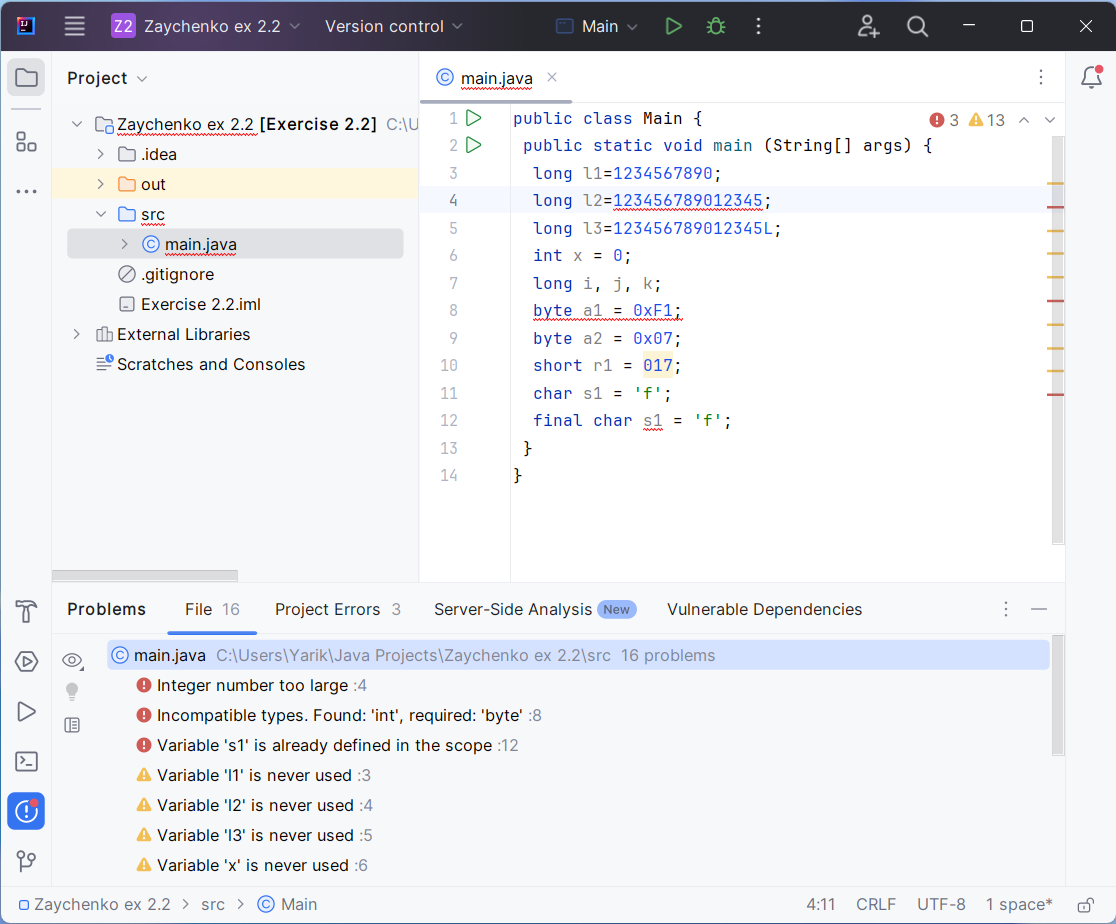
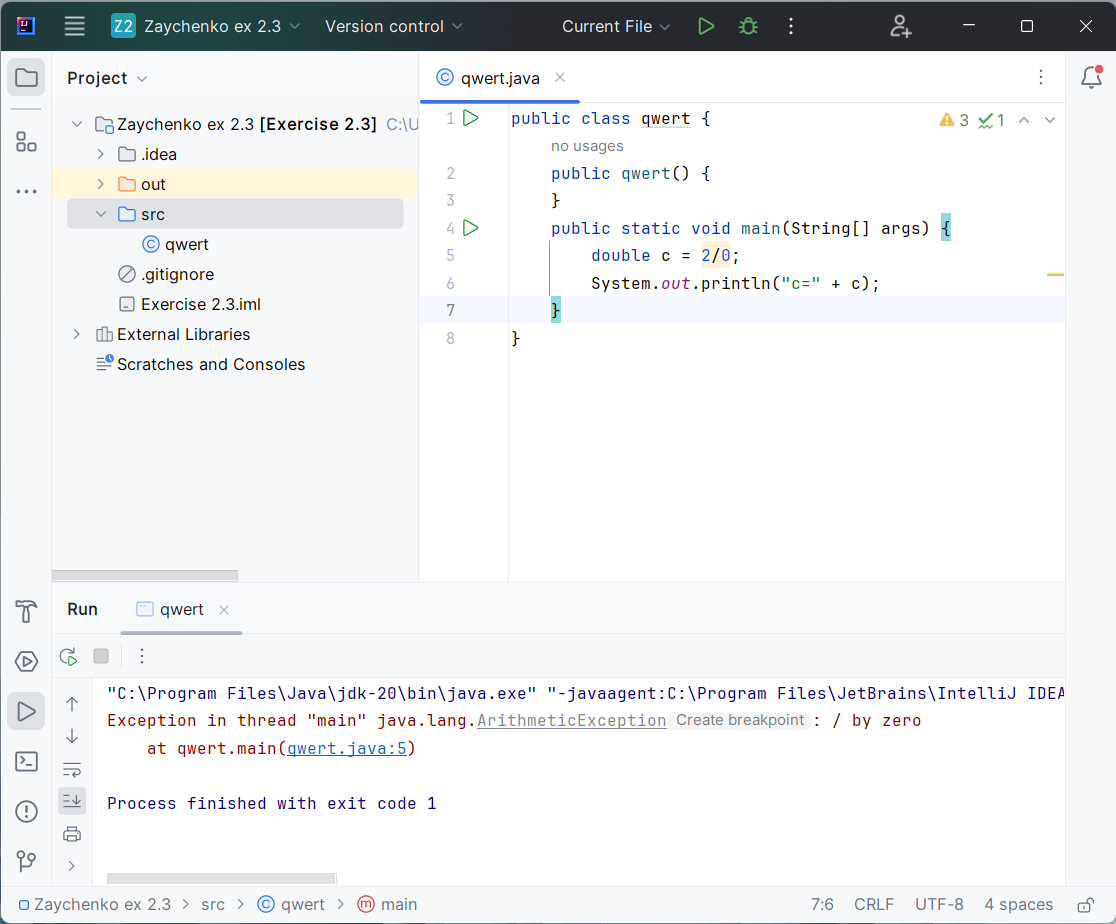


Рисунок 5 – Результат компіляції коду до завдання 2.2

Змінна l2 – значення занадто велике для типу данних long. Значення змінної a1 – занадто велике для типу даних byte. Змінна s1 – вже визначена в області класу Main.

**Завдання 2.3** (приклад перед рис. 4)

Рисунок 6 – Результат компіляції коду до завдання 2.3

Ділення на нуль є математично некоректною операцією, і воно призводить до помилки під час виконання програми. У Java така помилка називається "ArithmeticException," і вона виникає, коли програма намагається виконати неможливу математичну операцію.

**Завдання 2.4** (рис. 4)

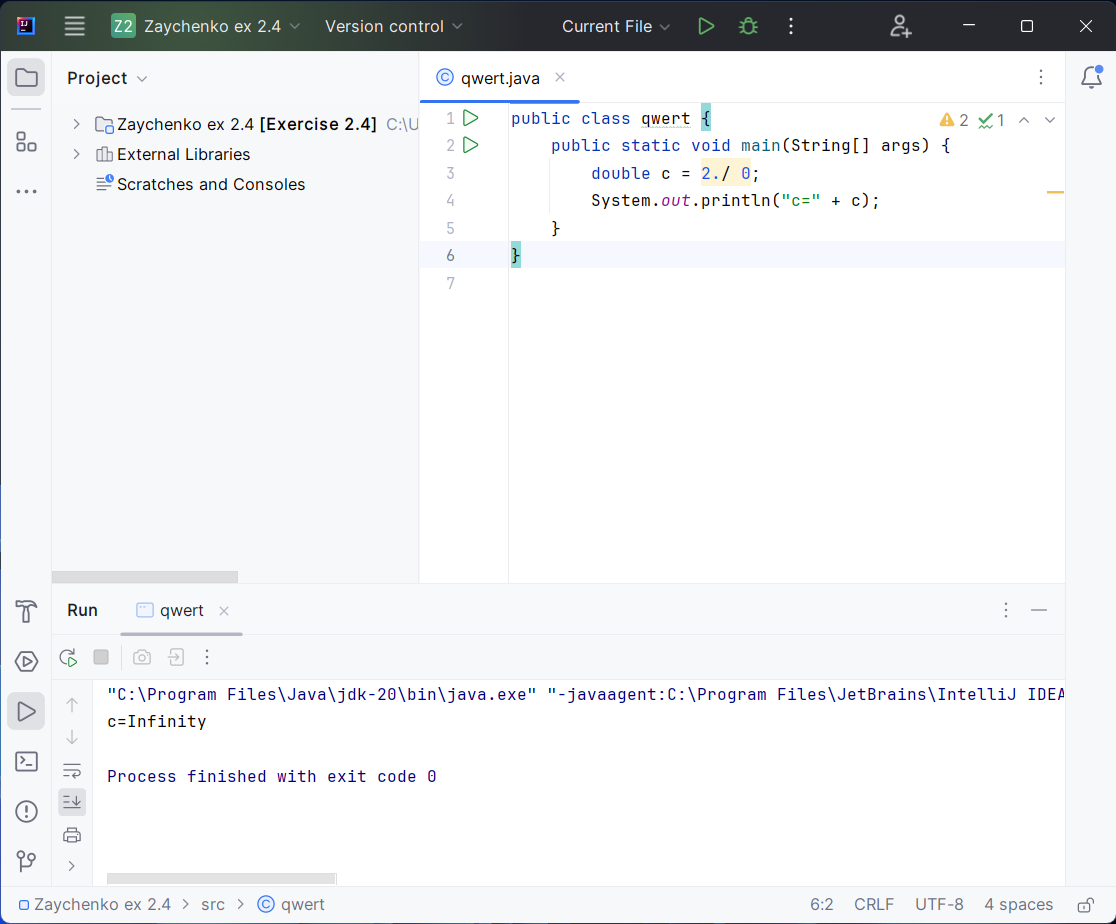


Рисунок 7 – Результат компіляції коду до завдання 2.4

Використано поділ числа на нуль у форматі подвійного числа (double). У відміну від цілочисельного поділу, випадок поділу десяткового числа на нуль у Java не призводить до помилки виключення, але замість цього встановлюється результат як "Infinity" (безкінечність) для додатного числа або "-Infinity" для від'ємного числа.

**Завдання 2.5** (рис. 5)

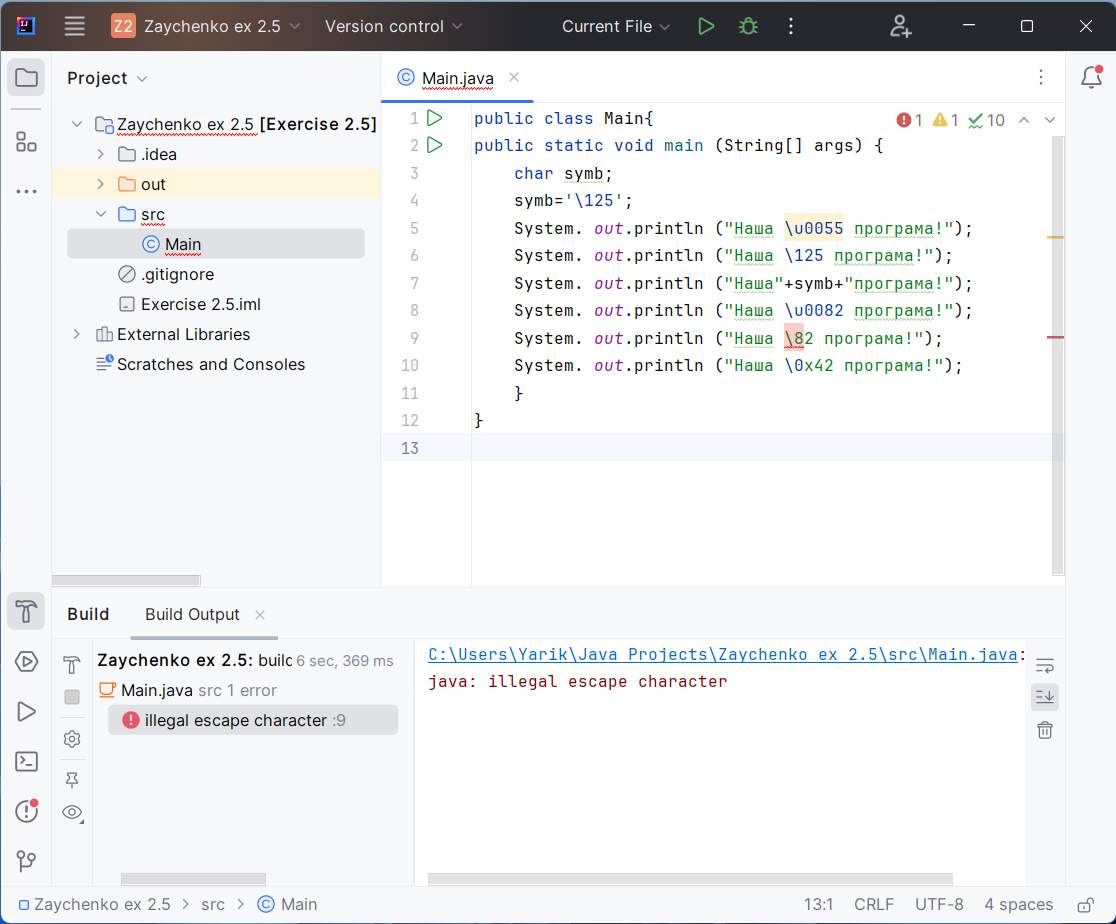
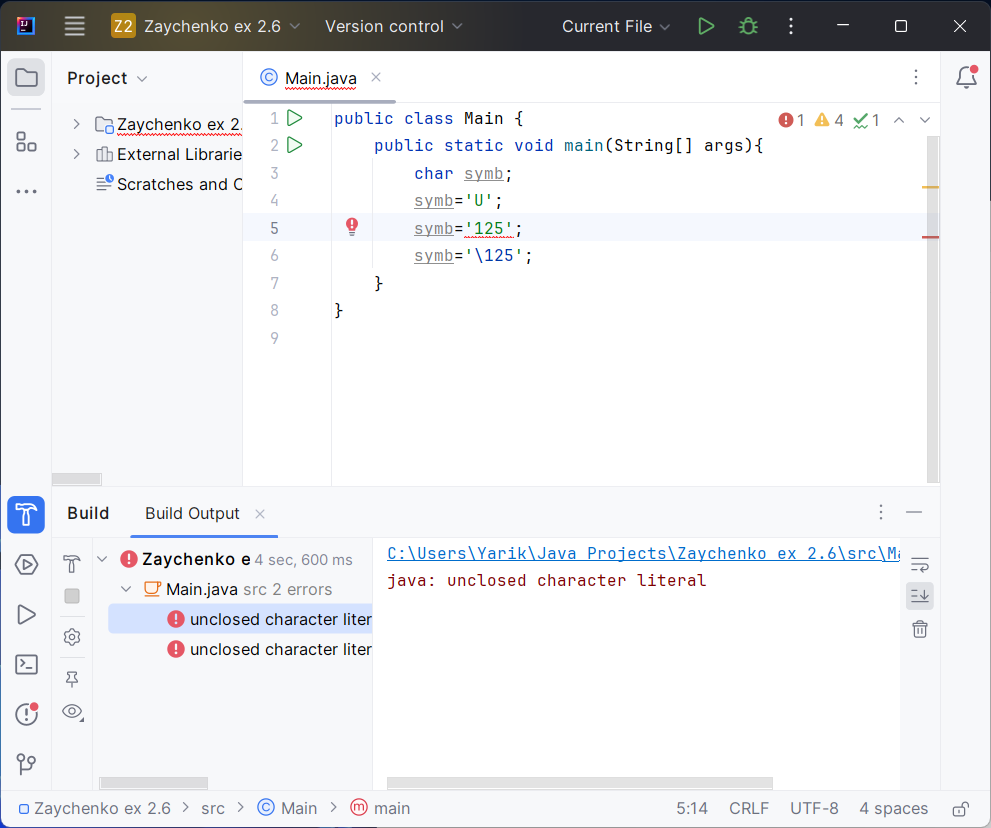


Рисунок 8 – Результат компіляції коду до завдання 2.5

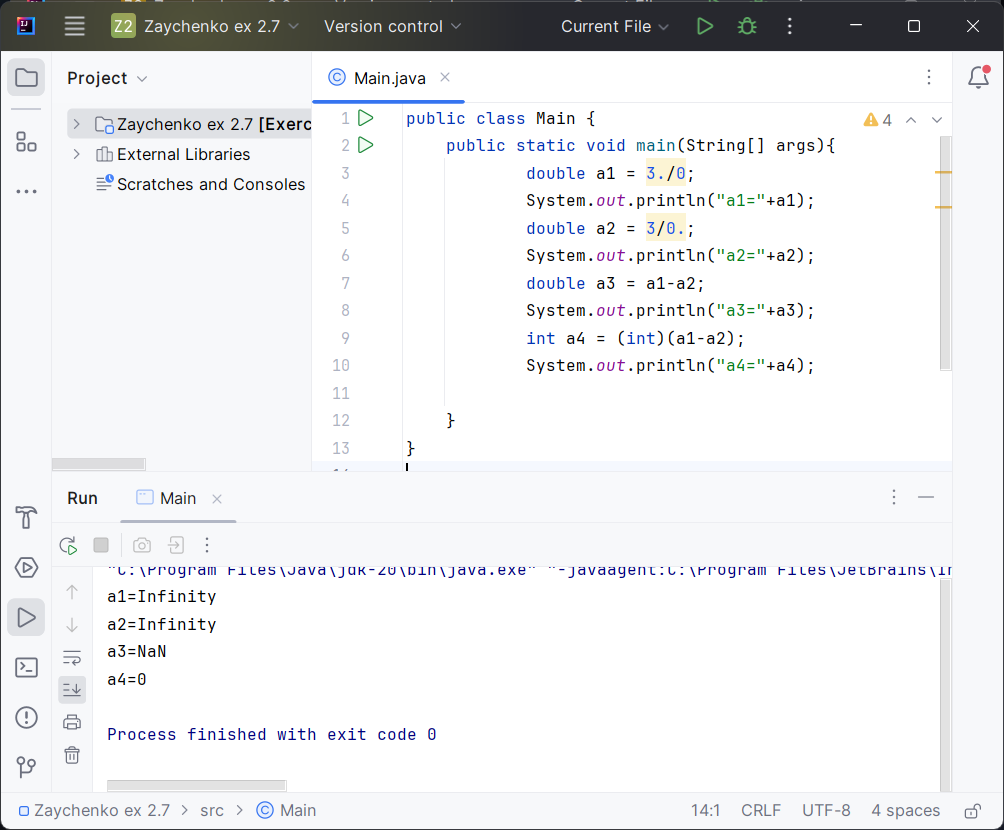
Проблема в коді полягає в некоректному використанні спроб уникнути спеціальних символів інтерпретації рядків. У коді використовуються спеціальні послідовності, які не розпізнаються компілятором, і це призводить до помилки "java: illegal escape character" (незаконний символ екранування). Щоб виправити цю проблему, потрібно виправити послідовності екранування в коді.

**Завдання 2.6** (рис. 6)

Рисунок 9 – Результат компіляції коду до завдання 2.6

Код має проблеми, оскільки відбувається присвоєння змінній symb рядкового значення та восьмирічного коду символа, але змінна визначена як символ типу char. Це може призвести до помилок компіляції.

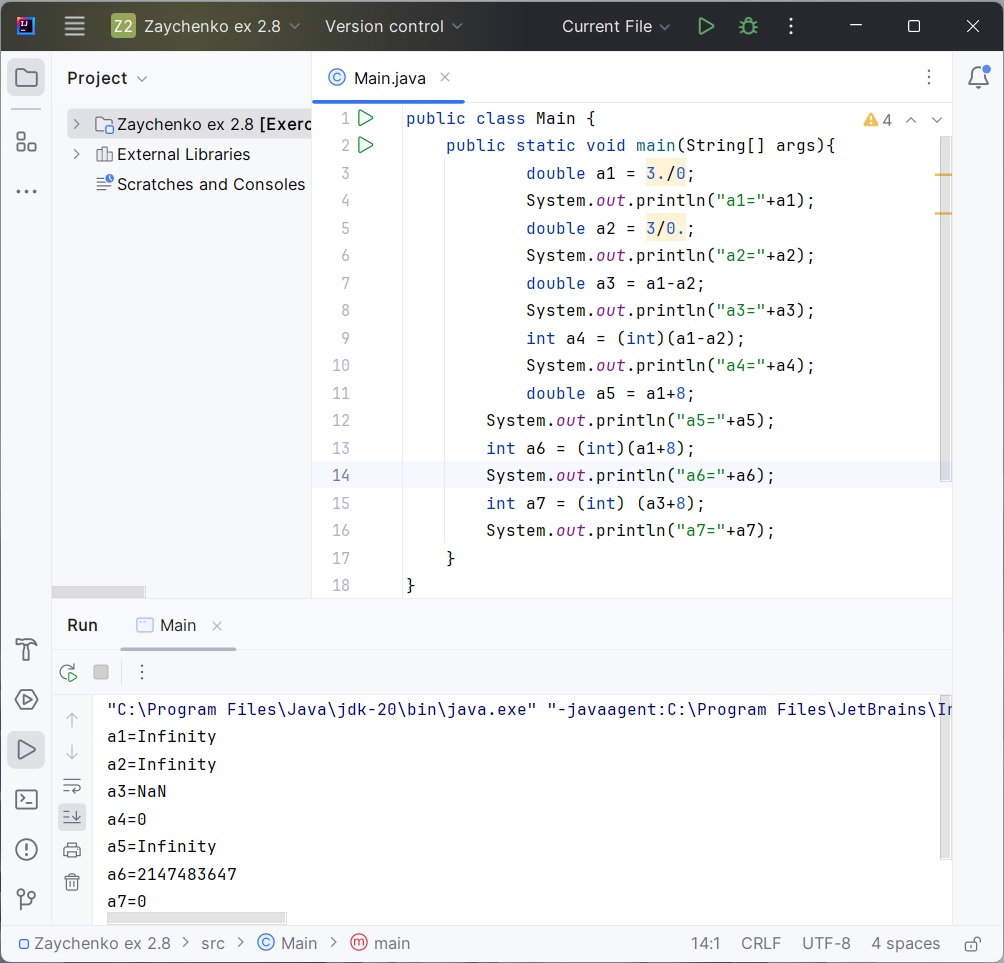
**Завдання 2.7** (рис. 8)

Рисунок 10 – Результат компіляції коду до завдання 2.7

Код виконує ділення чисел на нуль, що викликає специфічну поведінку:

* a1 отримує значення "Infinity". Це трапляється через ділення числа 3 на 0.0. У математиці ділення на нуль є нескінченністю, і в Java це представляється як "Infinity".
* a2 також отримує значення "Infinity". Тут число 3 ділиться на 0.0, що також призводить до нескінченності.
* a3 отримує значення "NaN" (Not-a-Number). Це трапляється, коли віднімають одну нескінченність від іншої. В математиці така операція не визначена, тому результат є "NaN".
* a4 отримує значення 0. Оскільки "NaN" не може бути приведений до цілого числа, то при спробі такого перетворення отримується 0.

Отже, ділення на нуль у плаваючій точці (floating-point arithmetic) призводить до спеціальних значень, таких як "Infinity" і "NaN", які вказують на неможливість виконати певні математичні операції.

**Завдання 2.8** (рис. 8) Рисунок 11 – Результат компіляції коду до завдання 2.8

Код використовує ділення на нуль у плаваючій точці та інші математичні операції.

Змінна a1 отримує значення "Infinity" через ділення 3 на 0.0. Аналогічно, змінна a2 також отримує значення "Infinity" через ділення 3 на 0.0.

Змінна a3 отримує значення "NaN" (Not-a-Number), оскільки віднімається одна нескінченність від іншої.

Змінна a4 дорівнює 0, оскільки "NaN" не може бути приведений до цілого числа.

Змінна a5 отримує значення "Infinity" через додавання 8 до нескінченності, а змінна a6 також отримує значення "Infinity" через додавання 8 до нескінченності.

Змінна a7 отримує значення "NaN" через додавання "NaN" до 8.

Отже, результати цих операцій вказують на те, що ділення на нуль та математичні операції з нескінченностями та "NaN" призводять до спеціальних значень у плаваючій точці, які вказують на неможливість виконання певних математичних операцій.

# **ВИСНОВОК**

У ході виконання цієї лабораторної роботи я отримано практичні навички роботи з простими типами в мові програмування Java. Вивчив особливості цілочисельних та властивості дійсних типів даних. Ознайомився з булевим типом даних. Вивчив поняття констант, керуючих символів та таблиці символів UNICODE.

# **Контрольні питання**

1. Що таке сувора типізація?

У Java сувора типізація (strong typing) означає, що мова має сильну систему типів, і всі змінні та вирази повинні відповідати строго визначеним типам даних. Це означає, що ви не можете виконувати операції між різними типами без явного приведення типів. Сувора типізація допомагає запобігти помилкам в час виконання, пов'язаним з несумісністю типів даних.

Наприклад, в Java ви не можете безпосередньо додавати рядок і ціле число, як це можливо у деяких інших мовах програмування. Вам спочатку потрібно перетворити число в рядок або навпаки, явно вказавши, яке приведення типів ви хочете використовувати.

Сувора типізація допомагає збільшити надійність програм та полегшує їх розуміння, адже вона змушує програмістів бути більш уважними щодо типів даних та перевірки сумісності при написанні коду.

1. Які особливості використання цілочисельних типів?

Цілочисельні типи (наприклад, `int`, `byte`, `short`, `long`) використовуються для зберігання цілих чисел без десяткової частини та підтримують арифметичні операції, інкремент та декремент. Кожен тип має обмежений діапазон значень і витратує різну кількість пам'яті, залежно від типу.

1. Які особливості використання дійсних типів?

Дійсні типи (наприклад, `float` і `double`) використовуються для зберігання чисел з плаваючою точкою, таких як десяткові дроби, з різною точністю та витратою пам'яті. Вони підтримують математичні операції та дозволяють обробляти дробові числа в програмах.

1. Які особливості використання булева типу?

Має лише два можливих значення – true чи falseПредставляє один біт інформації, але його «розмір» не є чимось точно визначеним. Використовується для позначення простих міток які відстежують хибні або істинні умови.

1. Які керуючі символи ви знаєте?

На лекції було вивчено такі керуючі символи:

* Вісімковий символ \ddd
* Шістнадцятковий символ UNICODE \uxxxx
* Апостроф – \`
* Кавички – \”
* Зворотна коса риска – \\
* Повернення каретки – \r
* Переведення рядка – \n
* Переведення сторінки – \f
* Горизонтальна табуляція – \t
* Повернення на крок – \b

1. Ділення на нуль.

В Java заборонено ділити на нуль тільки в цілочисельних обчисленнях. Ділення на нуль не приведе до помилки при використанні в дійсних типах.