Федеральное государственное бюджетное образовательное учреждение

высшего образования

РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА и ГОСУДАРСТВЕННОЙ СЛУЖБЫ

при ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

НИЖЕГОРОДСКИЙ ИНСТИТУТ УПРАВЛЕНИЯ

Факультет управления  
Кафедра информатики и информационных технологий  
Направление (профиль) подготовки: 09.03.03 Прикладная информатика

ОТЧЕТ

ПО ЛЕКЦИИ №4

по предмету «Программирование»

Выполнил студент группы ИБ-321

Клинов Павел Александрович

Нижний Новгород

2024

***Содержание***

[Задание 1](#Задание1): ……………….……………….……………….……………….……………….……………….………………………………………. 3

[Задание 2](#Задание2): ..……………….……………….……………….……………….……………….……………….……………………………………. 4

[Задание 3](#Задание3): ……………….……………….……………….……………….……………….……………….………………………………………. 5

[Задание 4](#Задание4): ……………….……………….……………….……………….……………….……………….………………………………………. 6

[Задание 5](#Задание5): ..……………….……………….……………….……………….……………….……………….……………………………………. 7

[Задание 6](#Задание6): ……………….……………….……………….……………….……………….……………….………………………………………. 8

[Задание 7](#Задание7): ……………….……………….……………….……………….……………….……………….………………………………………. 9

[Задание 8](#Задание8): ..……………….……………….……………….……………….……………….……………….………………………………….. 14

[Задание 9](#Задание9): ……………….……………….……………….……………….……………….……………….……………………………………. 15

[Вывод](#Вывод): ……………….……………….……………….……………….……………….……………….…………………………………….…… 17

**Цель работы**

Выполнить консольный ввод-вывод, сделать примеры с спецификаторами формата в Java. Реализовать примеры с арифметическими операции, с инкрементами, а также с поразрядными операциями. Выполнить операции сравнения и описать приоритет операций

**Задание 1.**  Проверить, а также реализовать перевод печатающей каретки (головки) на следующую стоку без использования println.

**public class Program {**

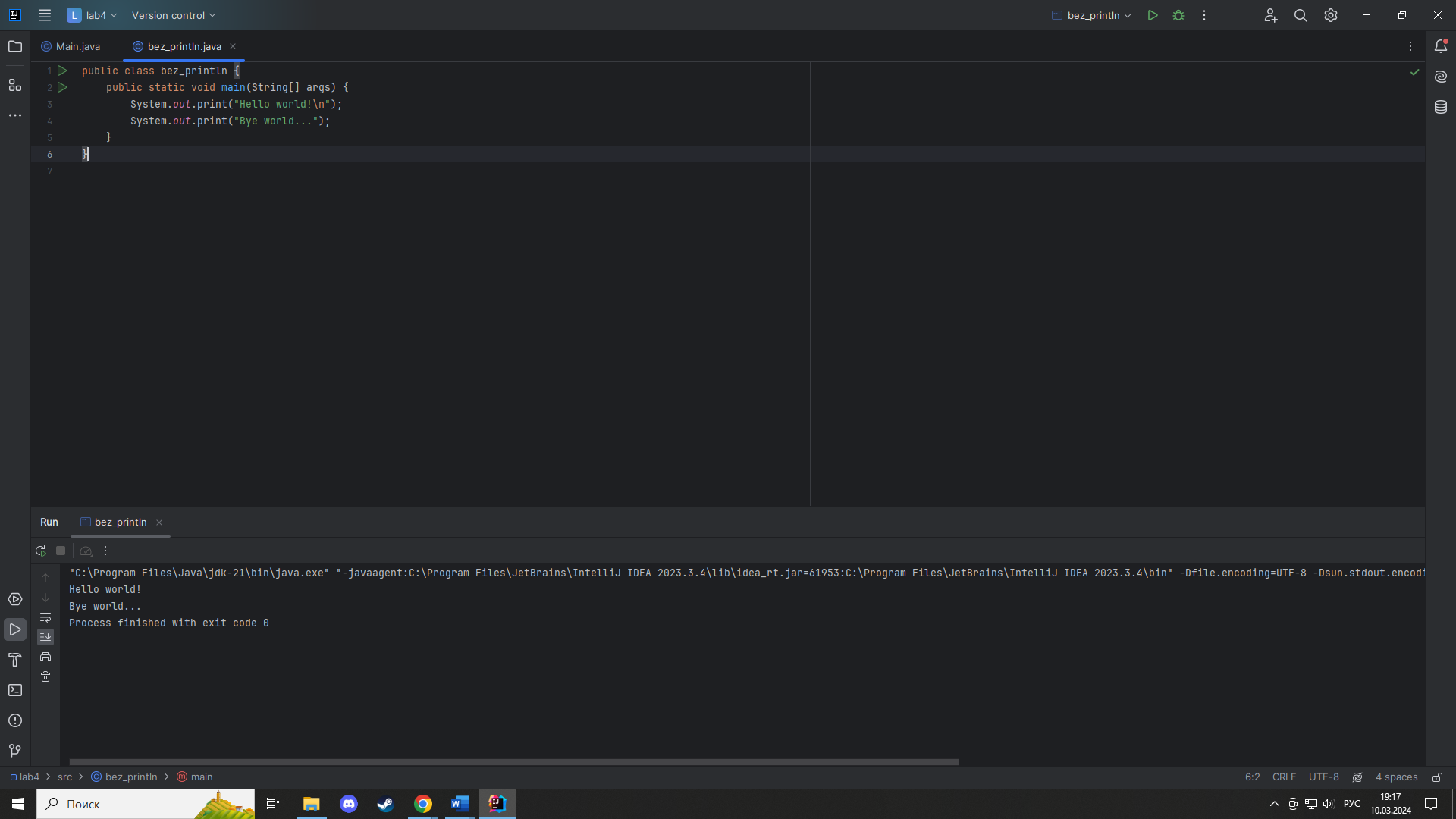
**public static void main(String[] args) {**

**System.out.println("Hello world!");**

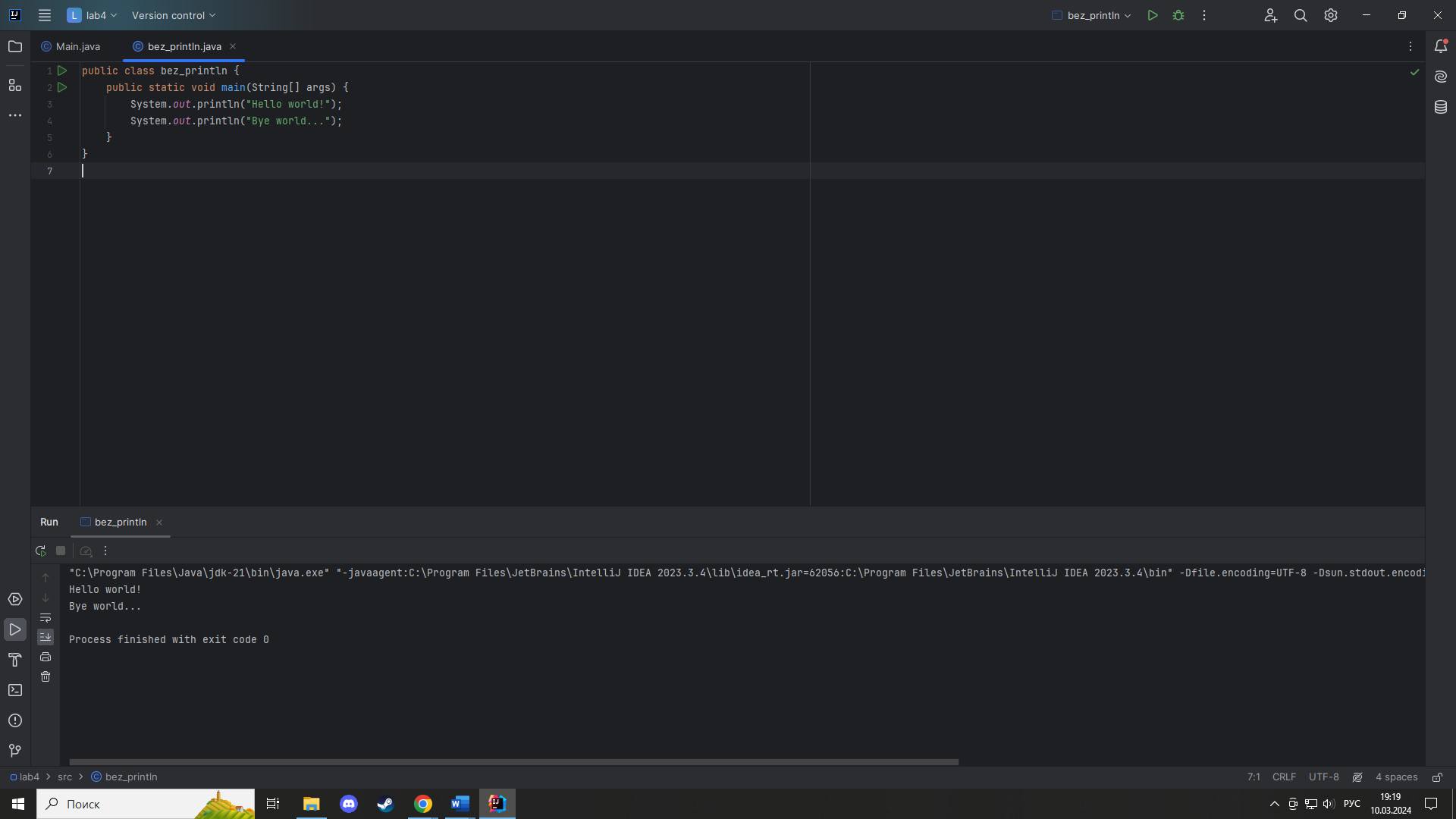
**System.out.println("Bye world...");**

**}**

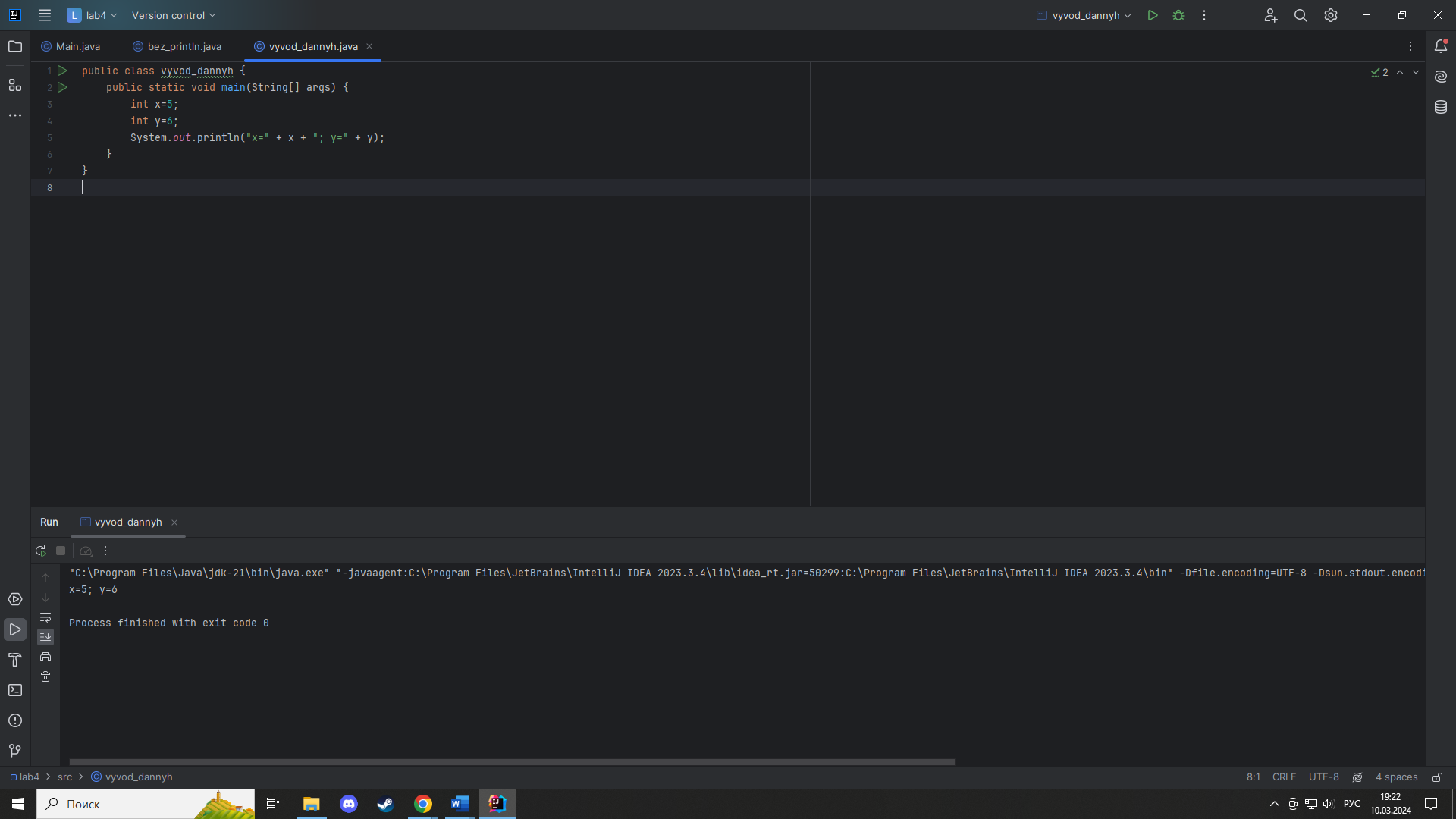
**}**



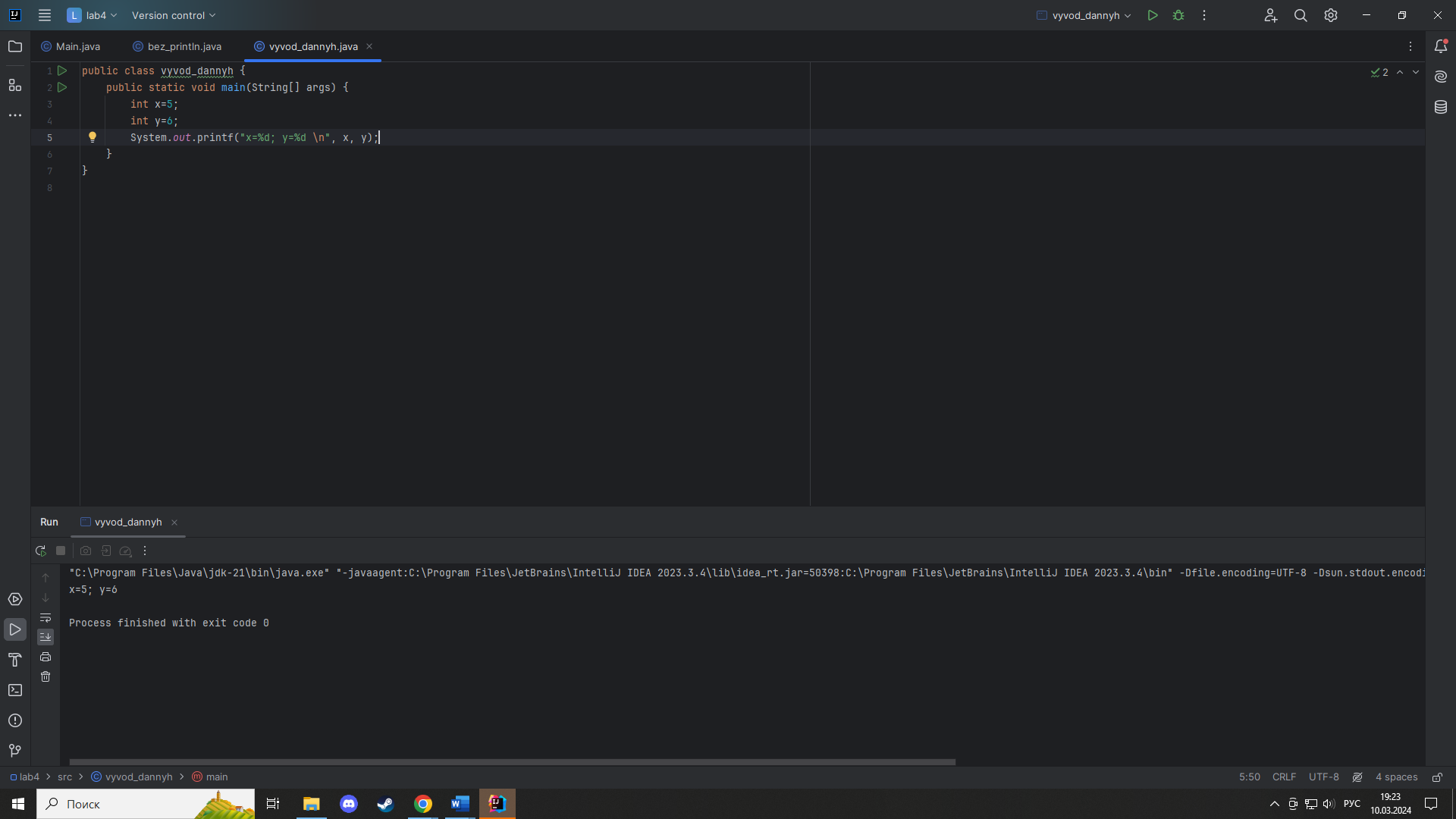
Я использовал \n, чтобы выполнить перенос строки без println.



**Задание 2.** Вывод данных (не строк!)



В Java есть также функция для **форматированного** вывода, унаследованная от языка С: System.out.printf(). С ее помощью мы можем переписать предыдущий пример следующим образом:

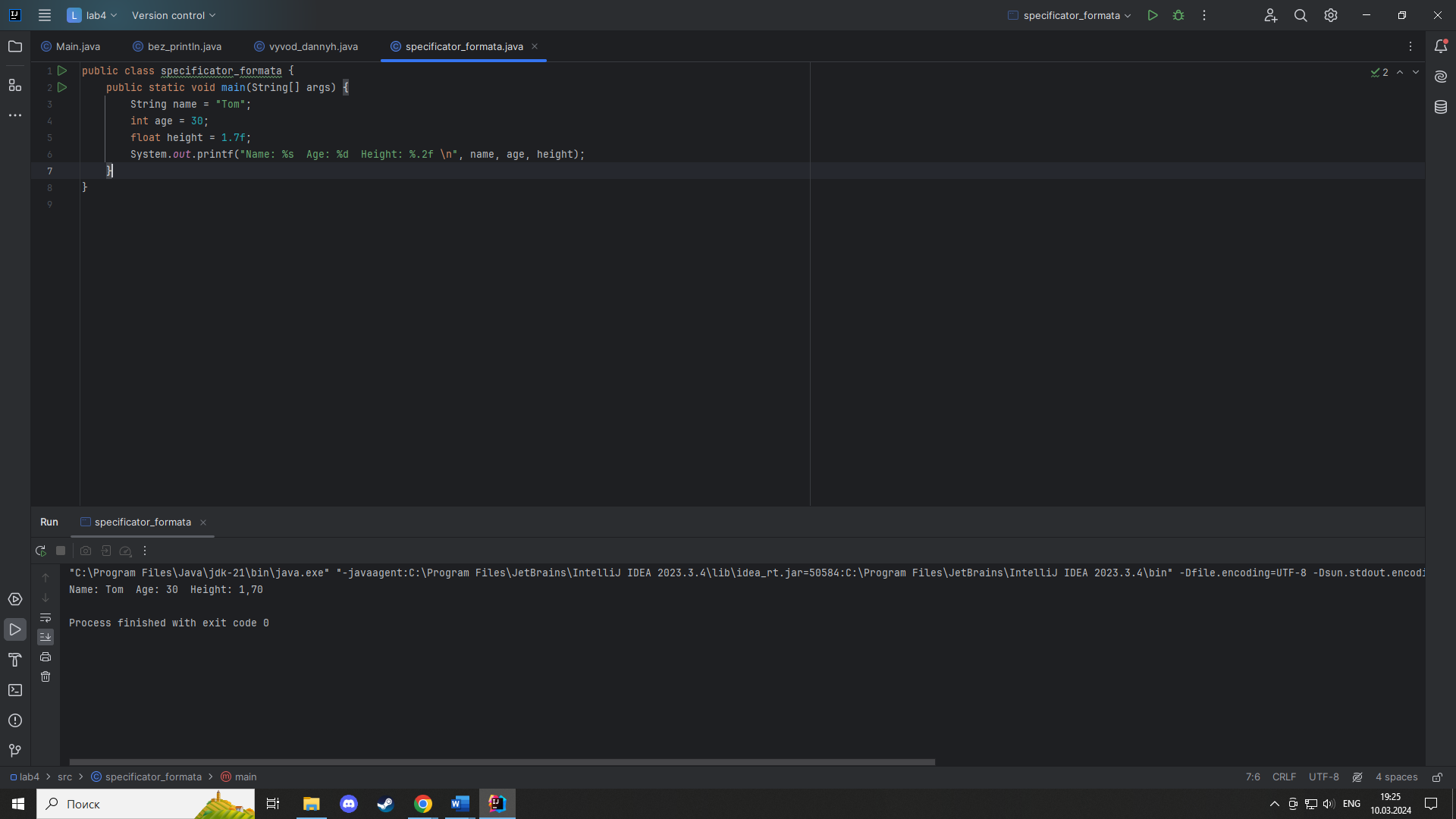


В данном случае символы %d обозначают спецификатор формата, **вместо которого подставляет один из аргументов.** В данном случае у нас только два аргумента, поэтому вместо первого %d подставляет значение переменной x, а вместо второго - значение переменной y. Сама буква d означает, что данный спецификатор будет использоваться для вывода целочисленных значений.

**Задание 3.** Спецификаторы формата в Java

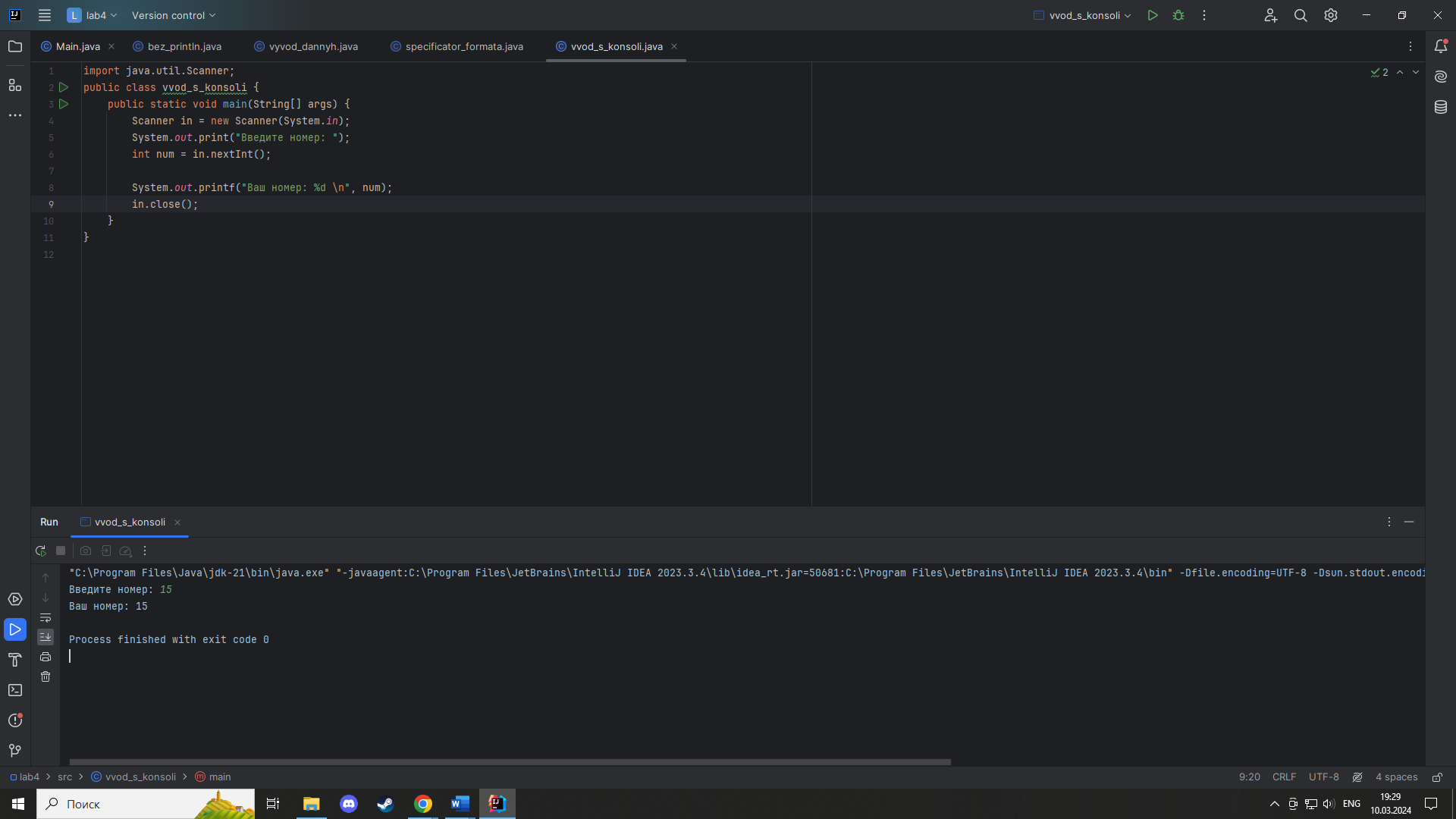
Спецификаторы формата начинаются с символа процента (%) и заканчиваются “символом типа”, который указывает тип данных (int, float и т. Д.), Которые будут преобразованы основным способом представления данных (десятичный, шестнадцатеричный и т. Д.). Общий синтаксис спецификатора формата:

**% [flags] [width] [.precision] [argsize] typechar**

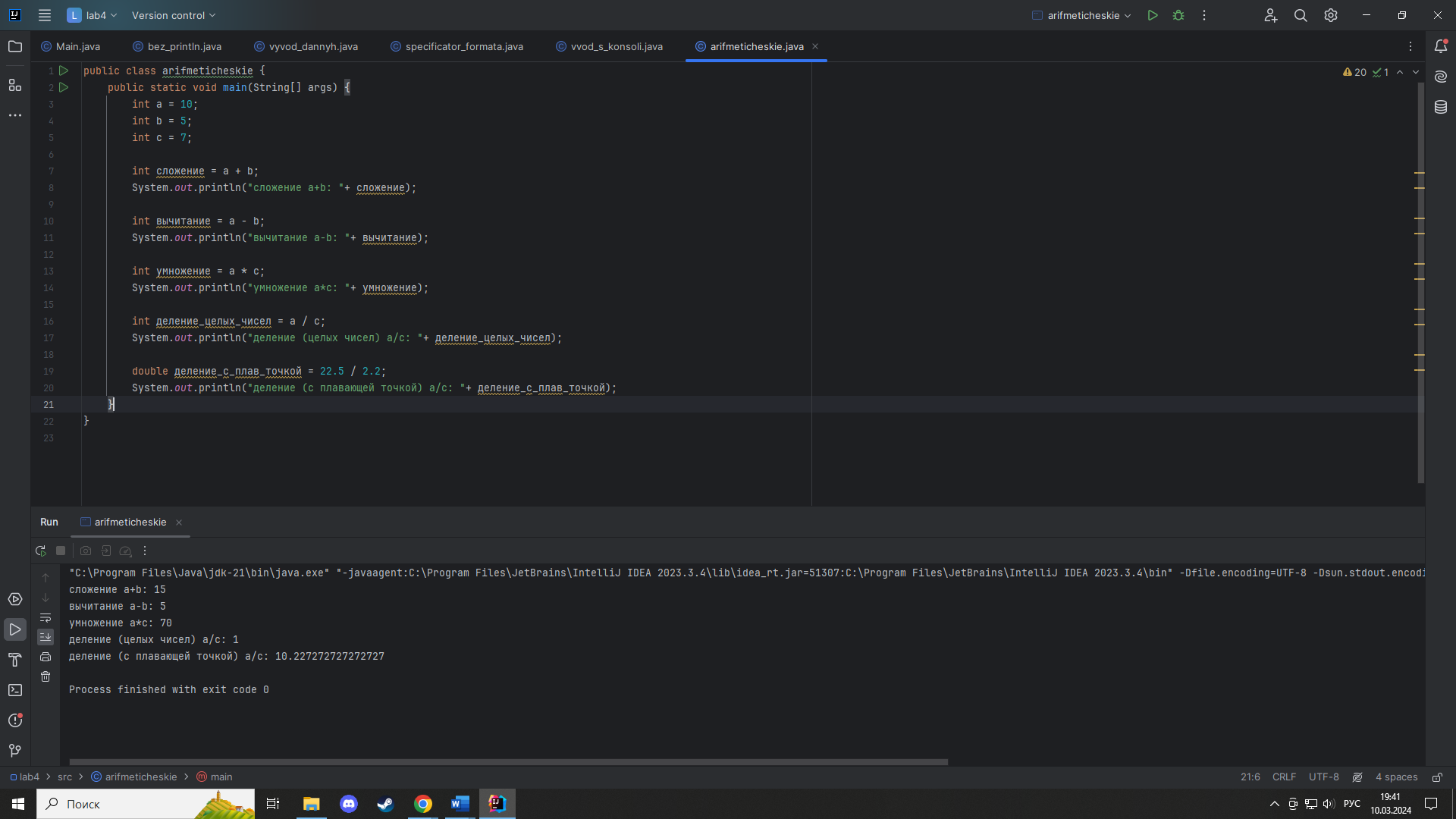


**Задание 4.** Ввод с консоли

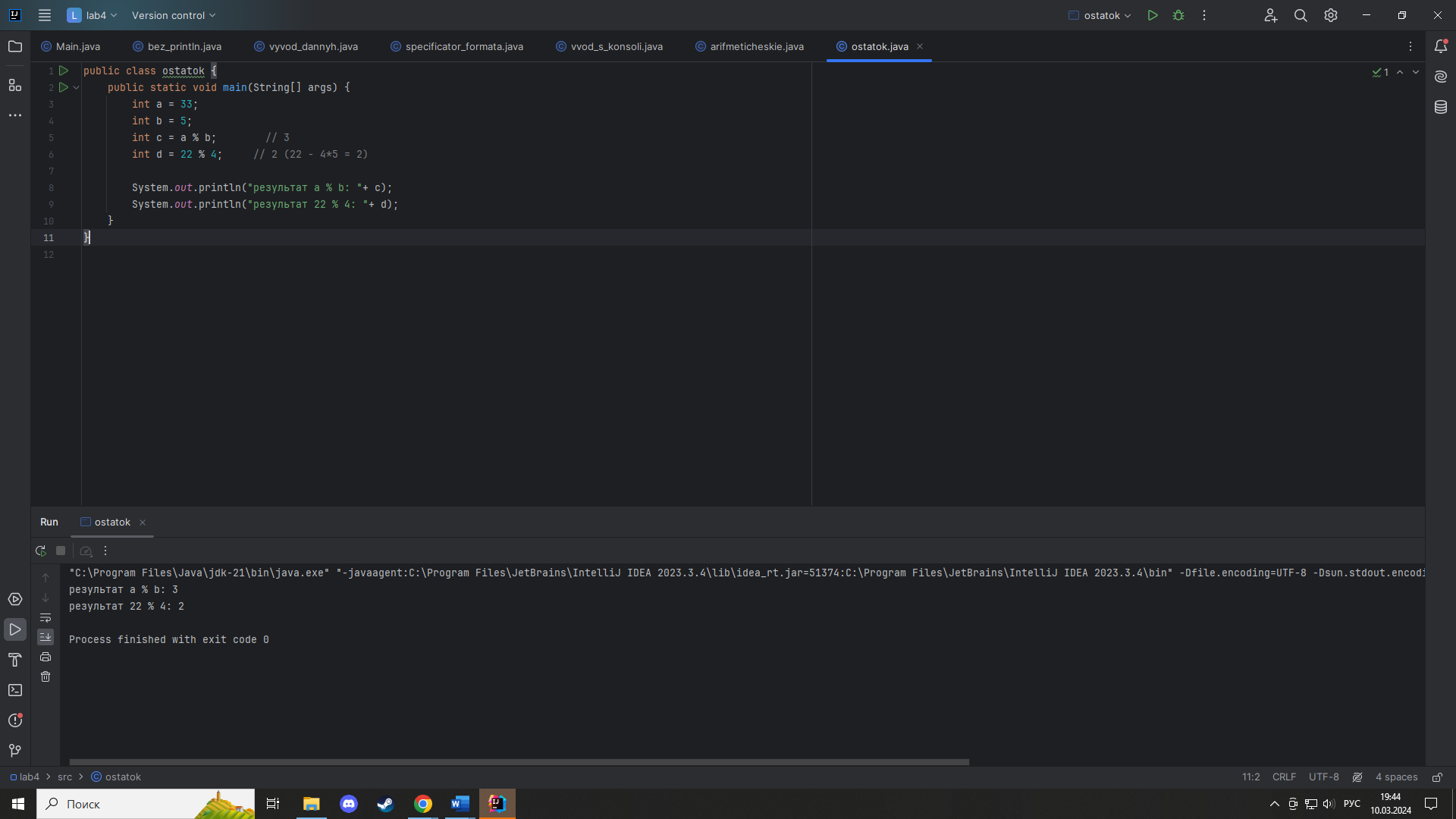
Для получения ввода с консоли в классе System определен объект in. Однако непосредственно через объект System.in не очень удобно работать, поэтому, как правило, используют класс **Scanner, который, в свою очередь использует System.in**.



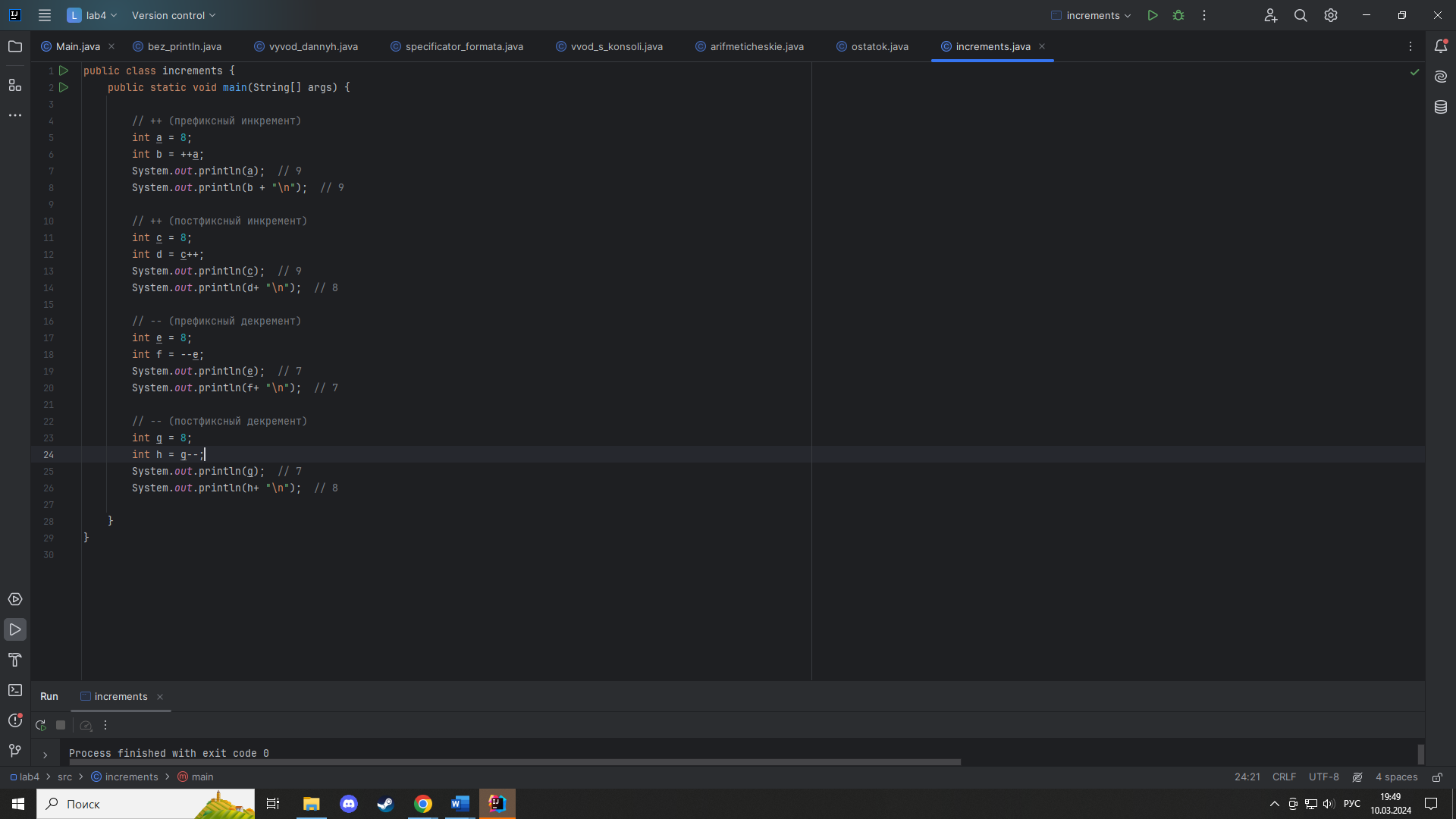
**Задание 5.** Арифметические. Операции: сложения(+), вычитания(-), умножения(\*), деления(/) двух чисел. Операция получения остатка от деления двух чисел

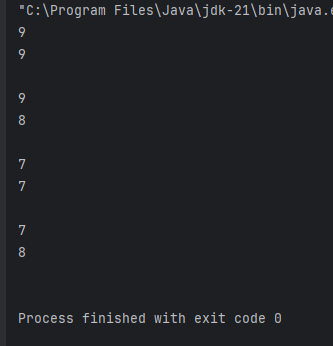


**Операция получения остатка от деления двух чисел:**



**Задание 6.** Инкременты





**Приоритет арифметических операций**

Одни операции имеют больший приоритет, чем другие, и поэтому выполняются вначале. Операции в **порядке уменьшения приоритета**:

++ (постфиксный инкремент), -- (постфиксный декремент)

++ (префиксный инкремент), -- (префиксный декремент)

\* (умножение), / (деление), % (остаток от деления)

+ (сложение), - (вычитание)

**Ассоциативность операций**

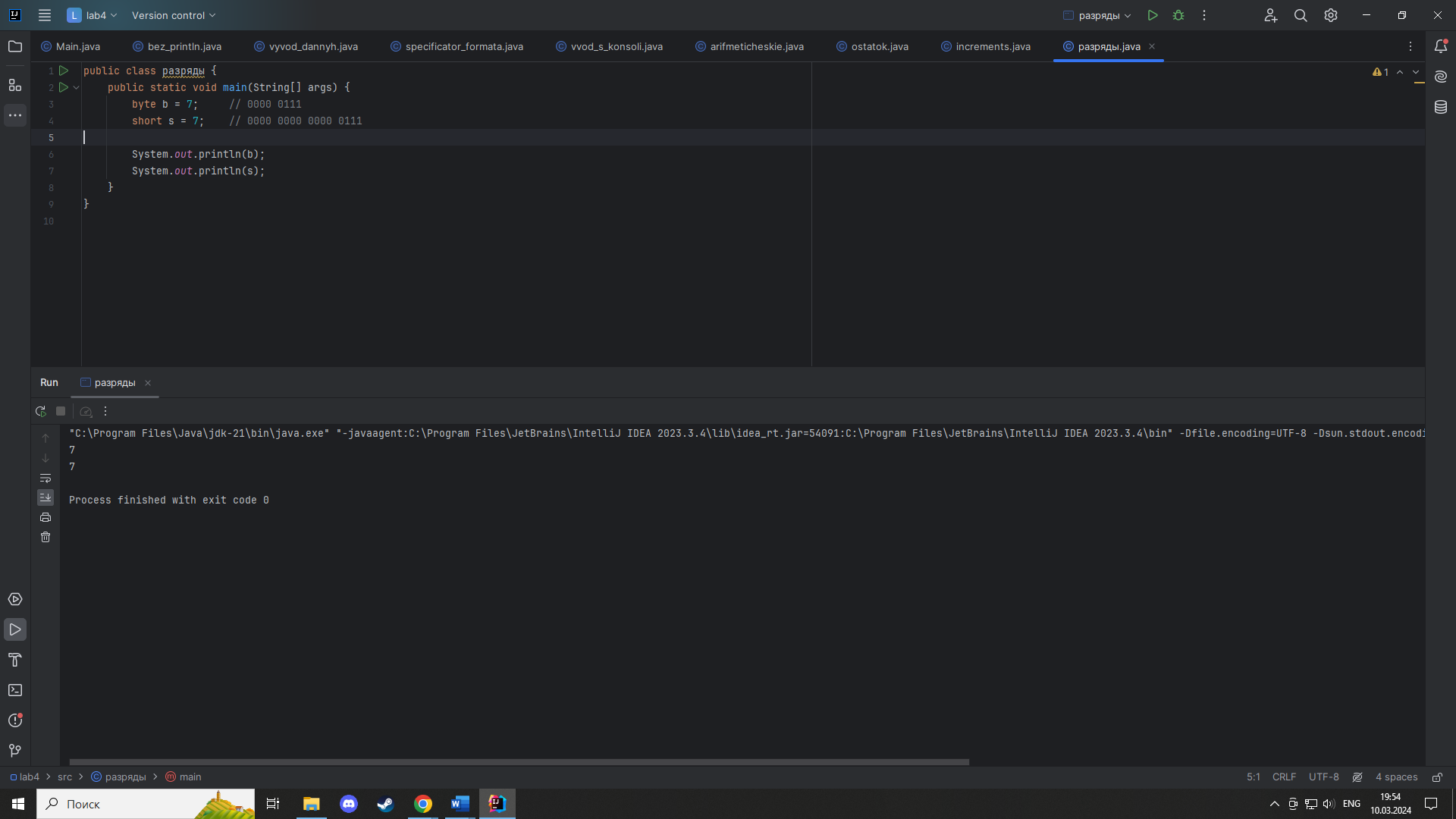
**Левоассоциативные** операторы, которые выполняются слева направо

**Правоассоциативные** операторы, которые выполняются справа налево

**Задание 7.** Поразрядные операции – операции над целыми числами

Каждое число имеет определенное двоичное представление. Например, число 4 в двоичной системе 100, а число 5 - 101 и так далее.

К примеру, возьмем следующие переменные:

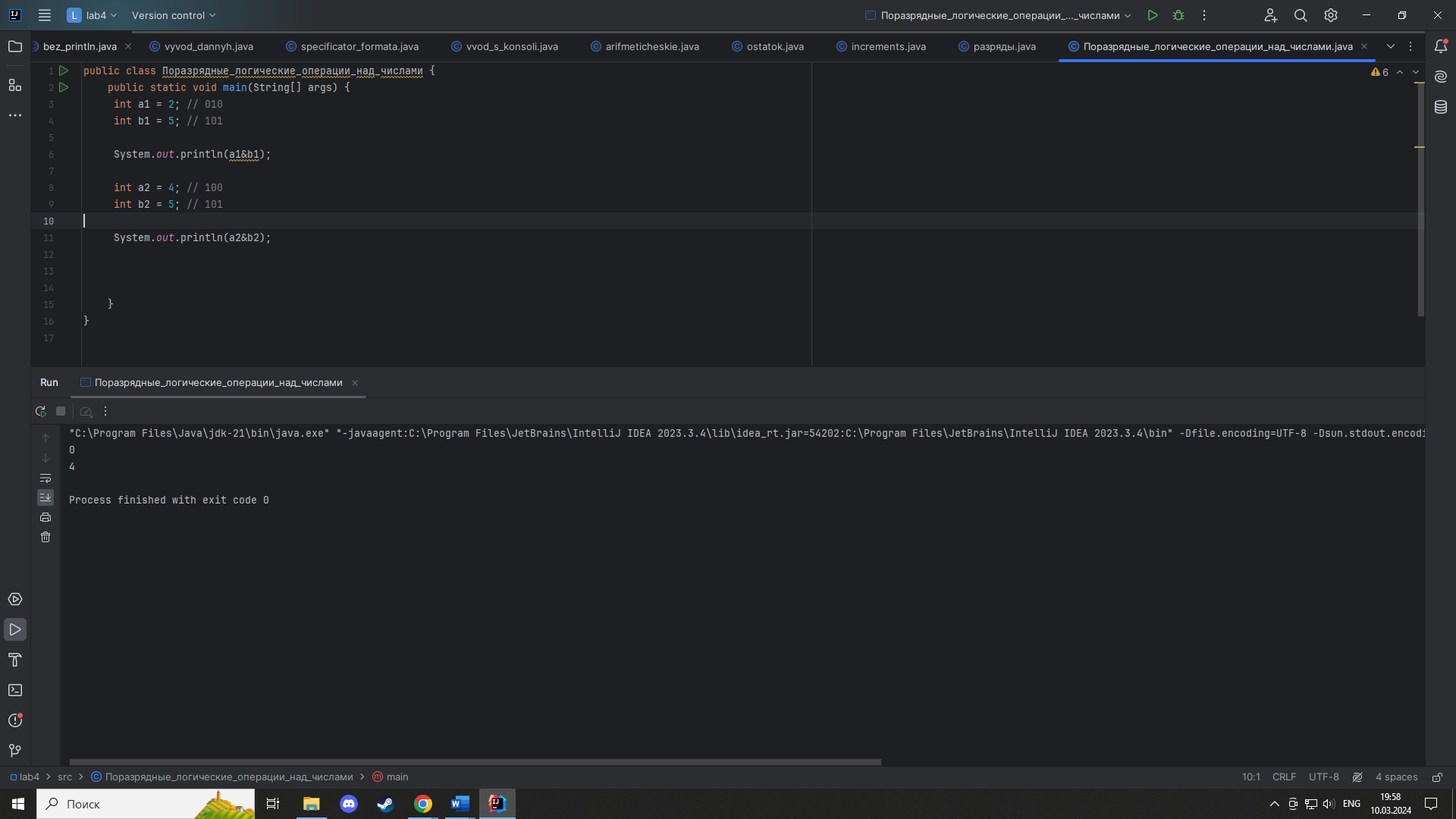


Тип byte занимает 1 байт или 8 бит, соответственно представлен 8 разрядами. Поэтому значение переменной b в двоичном коде будет равно 00000111.

**Поразрядные логические операции над числами**

**& (логическое умножение)**

Умножение производится поразрядно, и если у обоих операндов значения разрядов равно 1, то операция возвращает 1, иначе возвращается число 0.



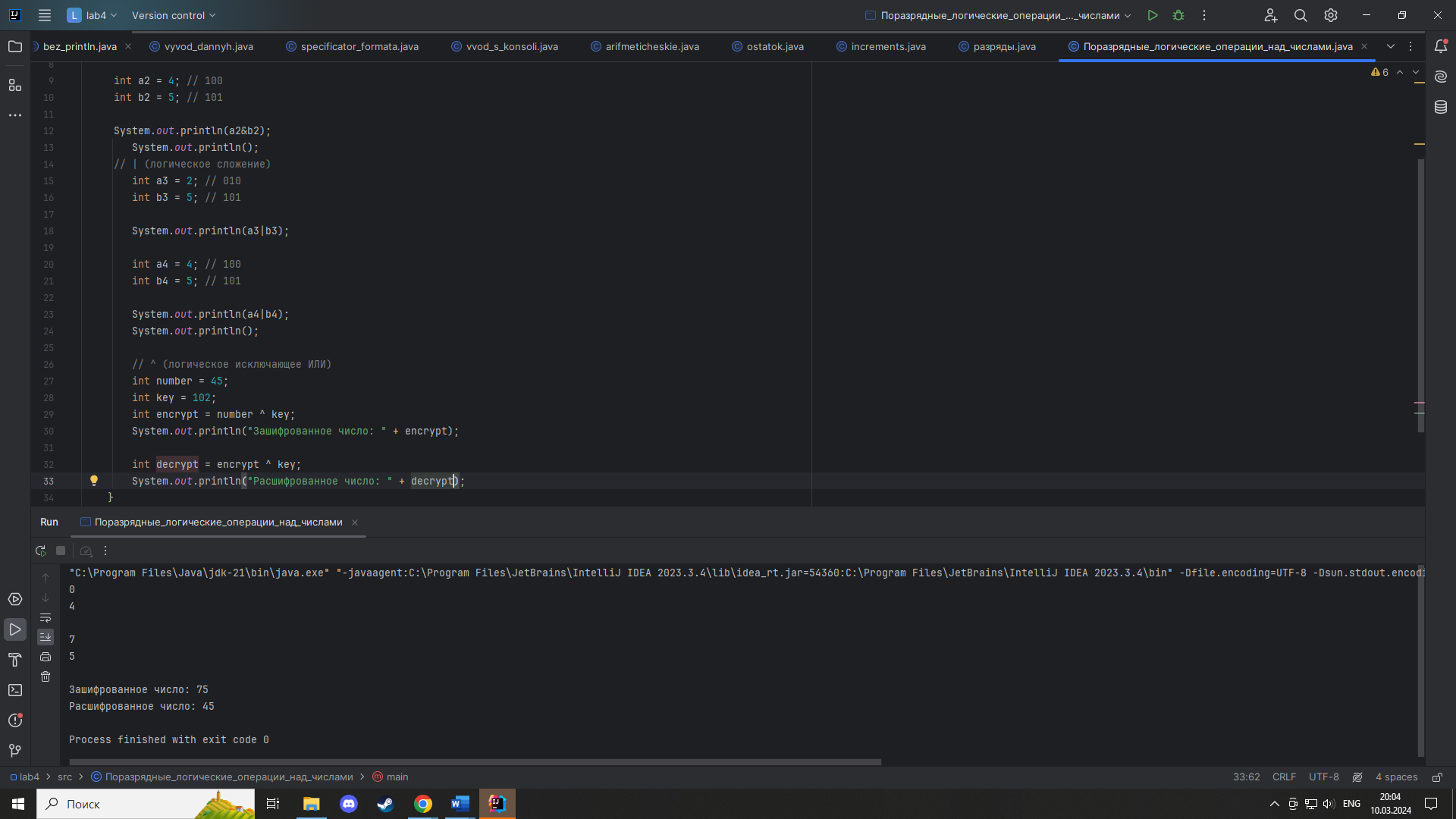
В первом случае у нас два числа 2 и 5.

2 в двоичном виде представляет число 010, а 5 - 101. Поразрядное умножение чисел (0\*1, 1\*0, 0\*1) дает результат 000.

Во втором случае у нас вместо двойки число 4, у которого в первом разряде 1, так же, как и у числа 5, поэтому здесь результатом операции (1\*1, 0\*0, 0 \*1) = 100 будет число 4 в десятичном формате.

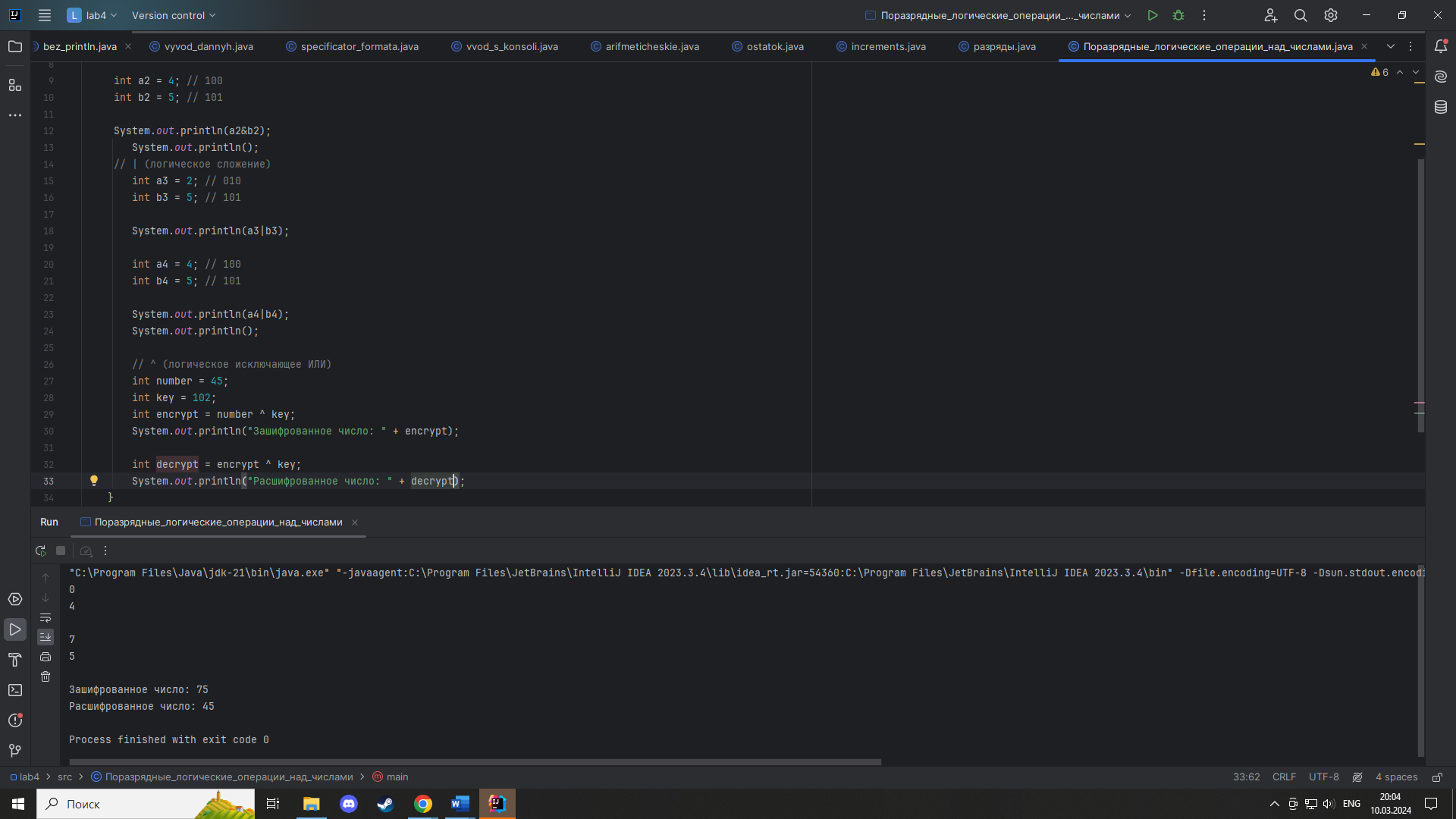
**| (логическое сложение)**

Данная операция также производится по двоичным разрядам, но теперь возвращается единица, если хотя бы у одного числа в данном разряде имеется единица (операция "логическое ИЛИ").



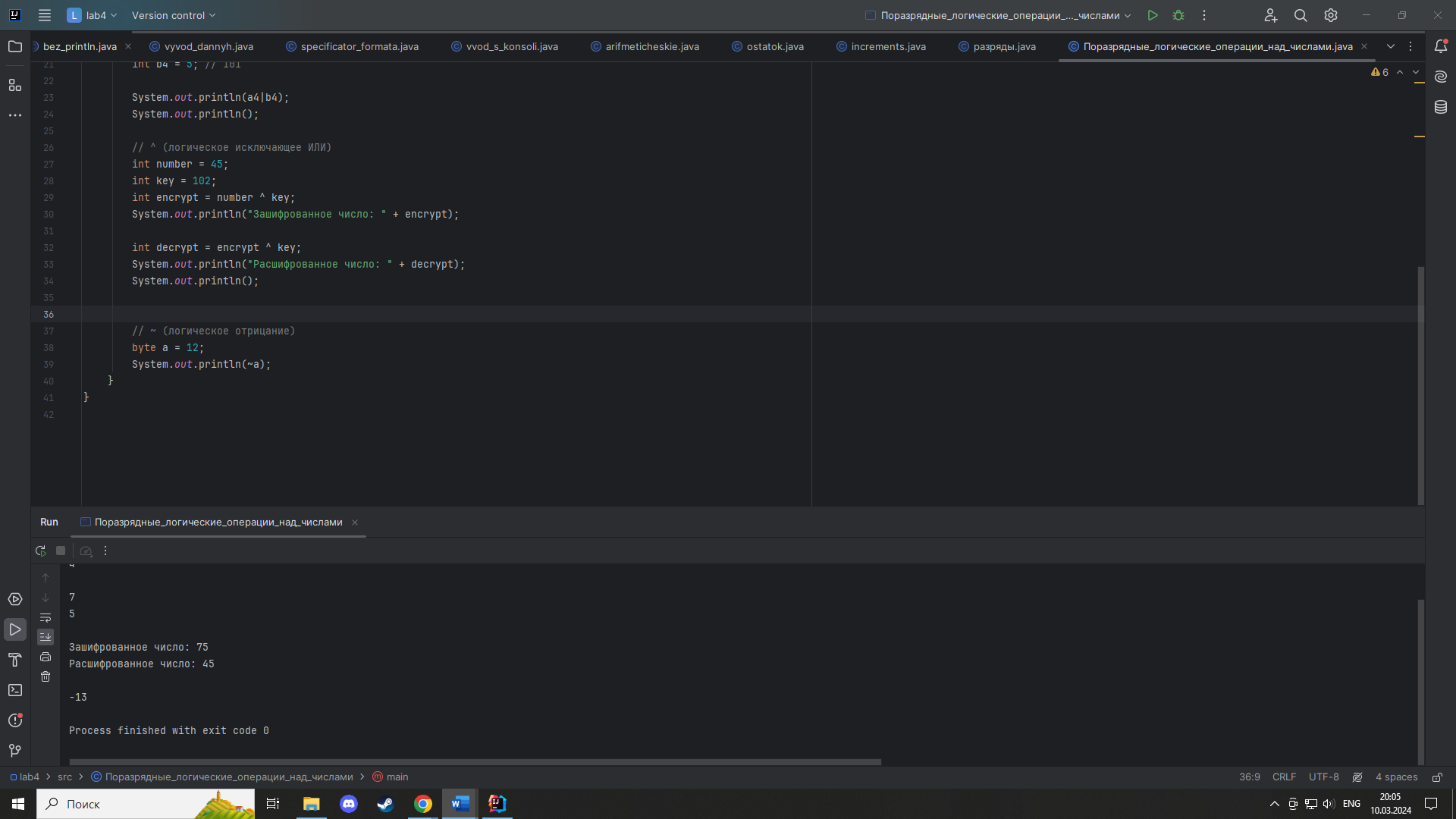
**^ (логическое исключающее ИЛИ)**

Здесь также производятся поразрядные операции. Если у нас значения текущего разряда у обоих чисел разные, то возвращается 1, иначе возвращается 0. Например, результатом выражения 9^5 будет число 12. А чтобы расшифровать число, мы применяем обратную операцию к результату.



**~ (логическое отрицание)**

Поразрядная операция, которая инвертирует все разряды числа: если значение разряда равно 1, то оно становится равным нулю, и наоборот.



**Задание 8.** Операции сравнения (отношений)

В операциях сравнения сравниваются два операнда, и возвращается значение типа boolean - true, если выражение верно, и false, если выражение неверно.

**(==) -** сравнивает два операнда на равенство и возвращает true (если операнды равны) и false (если операнды не равны);

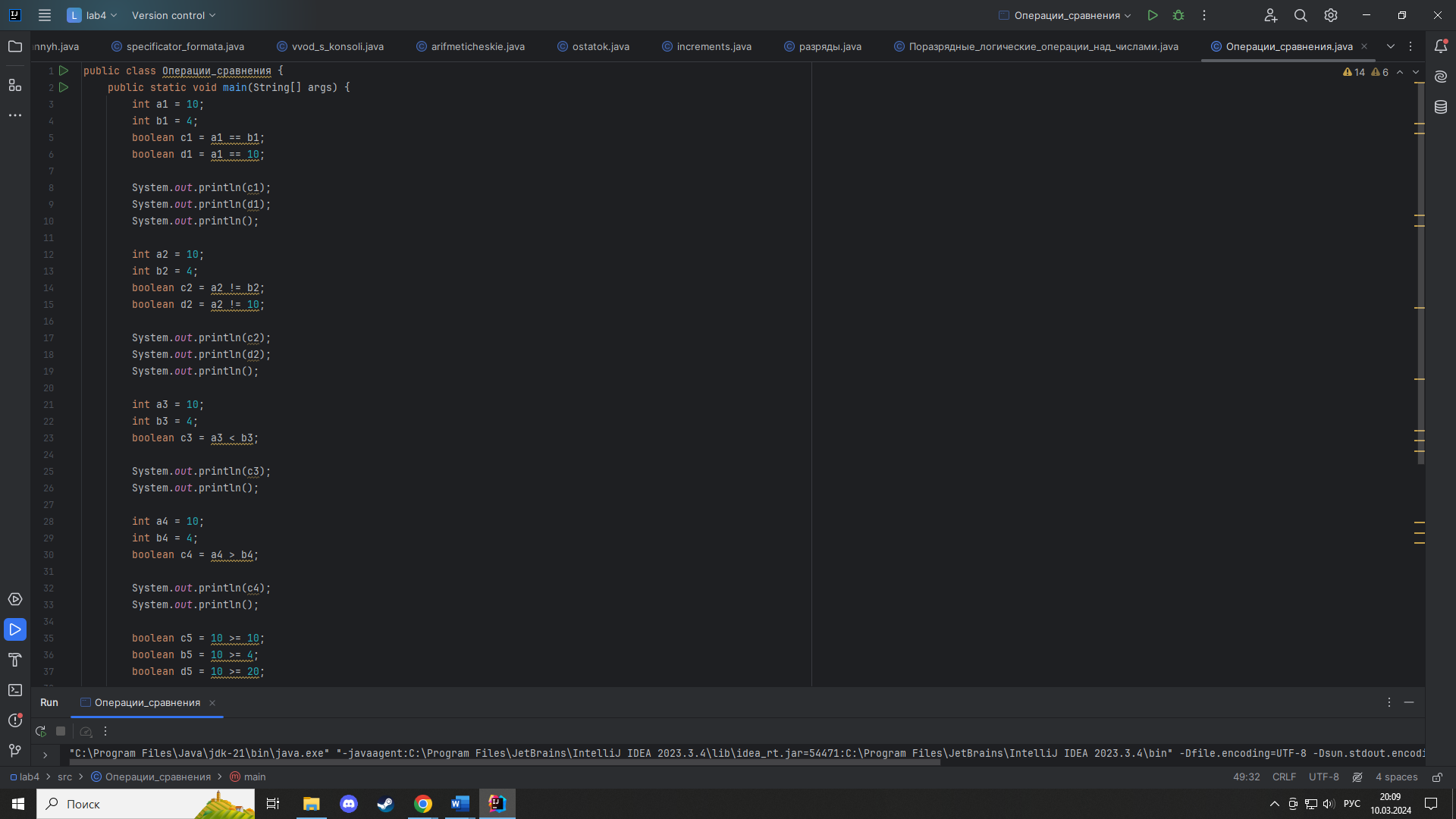
**(!=)** - сравнивает два операнда и возвращает true, если операнды НЕ равны, и false, если операнды равны;

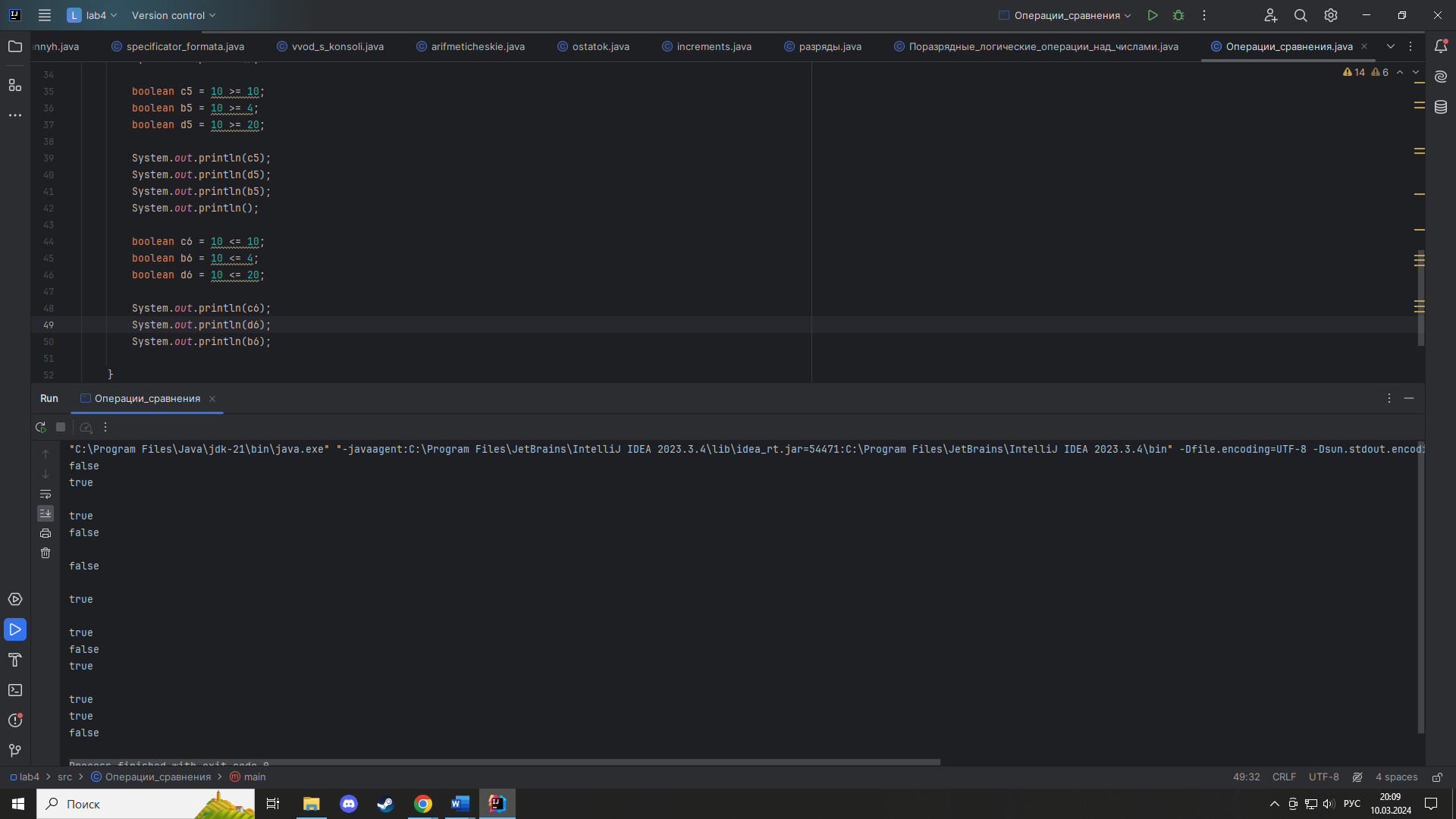
**(<)** - меньше чем;

**(>)** - больше чем;

**(>=)** - больше или равно;

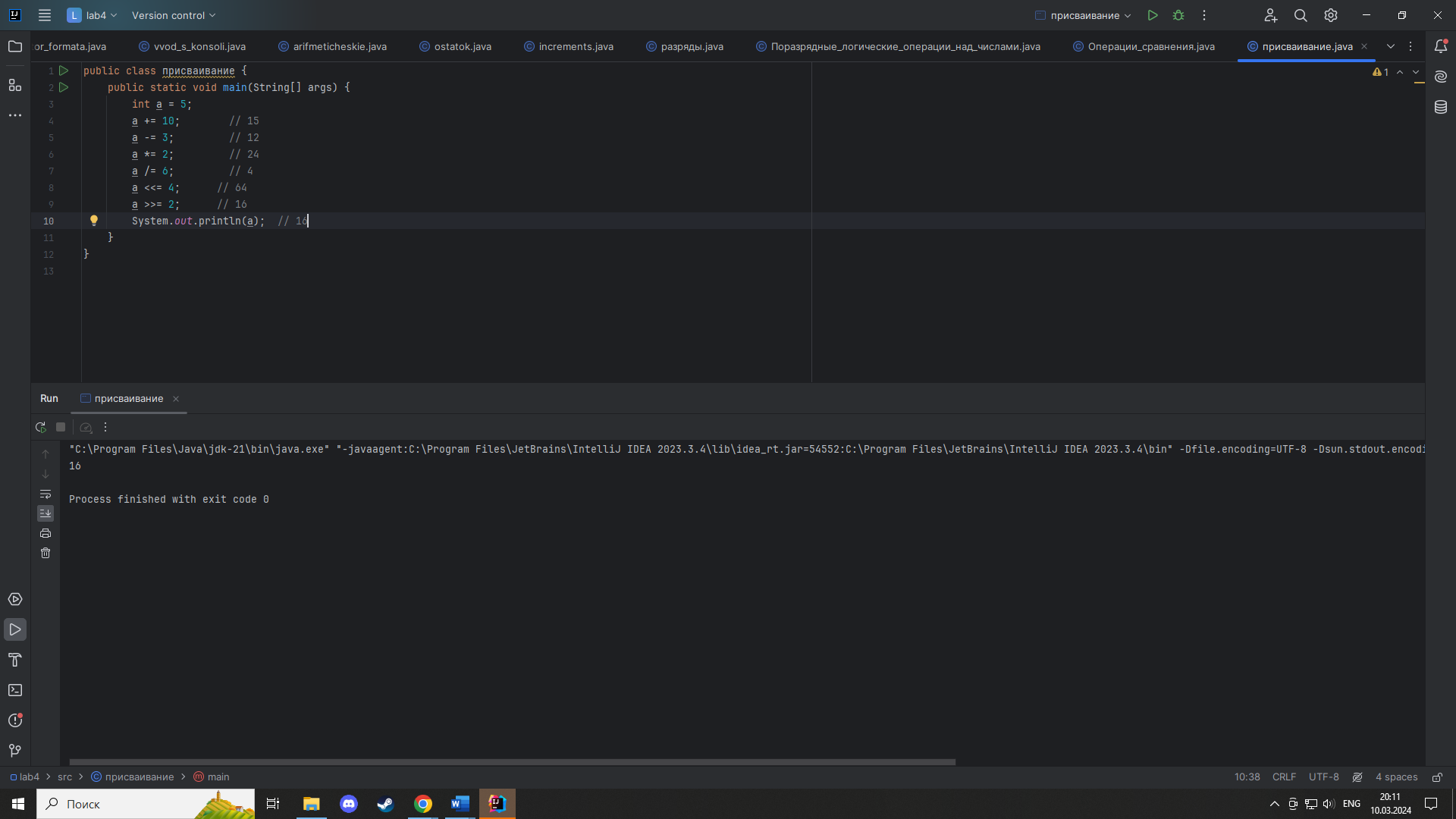
**(<=)** - меньше или равно.





**Задание 9.** Операции присваивания и приоритет операций

* **(=)** - просто приравнивает одно значение другому: c=b;
* **(+=)** - c+=b; (переменной c присваивается результат сложения c и b)
* **(-=)** - c-=b; (переменной c присваивается результат вычитания b из c)
* **(\*=)** - c\*=b; (переменной c присваивается результат произведения c и b)
* **(/=)** - c/=b; (переменной c присваивается результат деления c на b)
* **(%=)** - c%=b; (переменной c присваивается остаток от деления c на b)
* **(&=)** - c&=b; (переменной c присваивается значение c&b)
* **(|=)** - c|=b; (переменной c присваивается значение c|b)
* **(^=)** - c^=b; (переменной c присваивается значение c^b)
* **(<<=)** - c<<=b; (переменной c присваивается значение c<<b)
* **(>>=)** - c>>=b; (переменной c присваивается значение c>>b)
* **(>>>=)** - c>>>=b; (переменной c присваивается значение c>>>b)



**Приоритет операций**

При работе с операциями важно понимать их приоритет, который можно описать следующей таблицей:



Чем выше оператор в этой таблице, тем больше его приоритет. При этом скобки повышают приоритет операции, используемой в выражении

**Вывод**: Я выполнил консольный ввод-вывод, сделал примеры с спецификаторами формата в Java. Реализовал примеры с арифметическими операции, с инкрементами, а также с поразрядными операциями. Выполнил операции сравнения и описал приоритет операций