**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное автономное образовательное учреждение высшего образования «Уральский федеральный университет имени первого Президента России Б.Н. Ельцина

Институт радиоэлектроники и информационных технологий – РТФ

Центр ускоренного обучения

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7**

**По дисциплине «Программирование»**

Студент группы РИЗ-140938у Бабенко М.А.

Преподаватель: Архипов Н.А.

**Екатеринбург, 2024 г.**

**Лабораторная работа N7**

**«Введение в наследование»**

Цель: получить представление о механизме наследования в языке программирования Java.

**Задания для самостоятельной работы**

**Задание 1.** Напишите программу, в которой есть суперкласс с приватным текстовым полем, конструктором с текстовым параметром и где переопределен метод toString (). На основе суперкласса путем наследования создается подкласс. У него появляется еще одно приватное текстовое ноле. Также подкласс должен иметь версии конструктора с одним и двумя текстовыми аргументами, а еще в нем должен быть переопределен метод toString (). В обоих классах метод toString () переопределяется так, что он возвращает строку с названием класса и значение текстового поля или текстовых полей.

В proj71.java создается суперкласс SuperClass, в котором имеется приватное текстовое поле. Конструктор этого класса принимает строку **stirng\_field**.

Результат работы программы и ее код показан на рисунке 1.

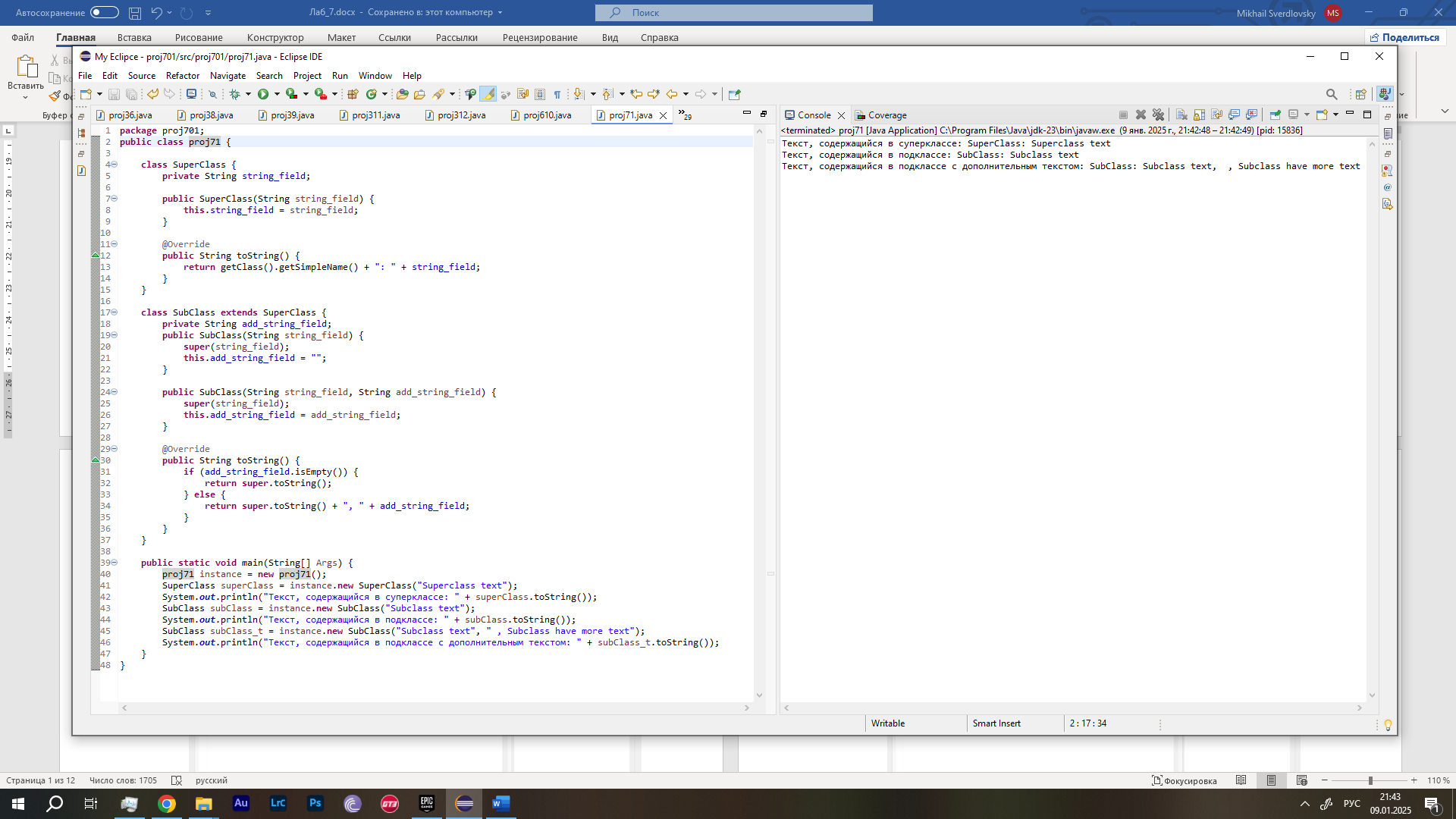


Рисунок 1 – Код и результат работы программы proj71

**Задание 2**. Напишите программу с классом, в котором есть закрытое статическое целочисленное ноле с начальным нулевым значением. В классе должен быть описан статический метод, при вызове которого отображается текущее значение статического поля, после чего значение поля увеличивается на единицу.

В классе proj62 создается закрытое статическое поле **num** с значением 0. Также создается метод для вывода значения поля с комментарием, после чего, в этом же методе, значение статического поля инкрементируется. В главном методе вводится цикл **for** для обращения к полю **num**. Результат работы программы и ее код показан на рисунке 2.

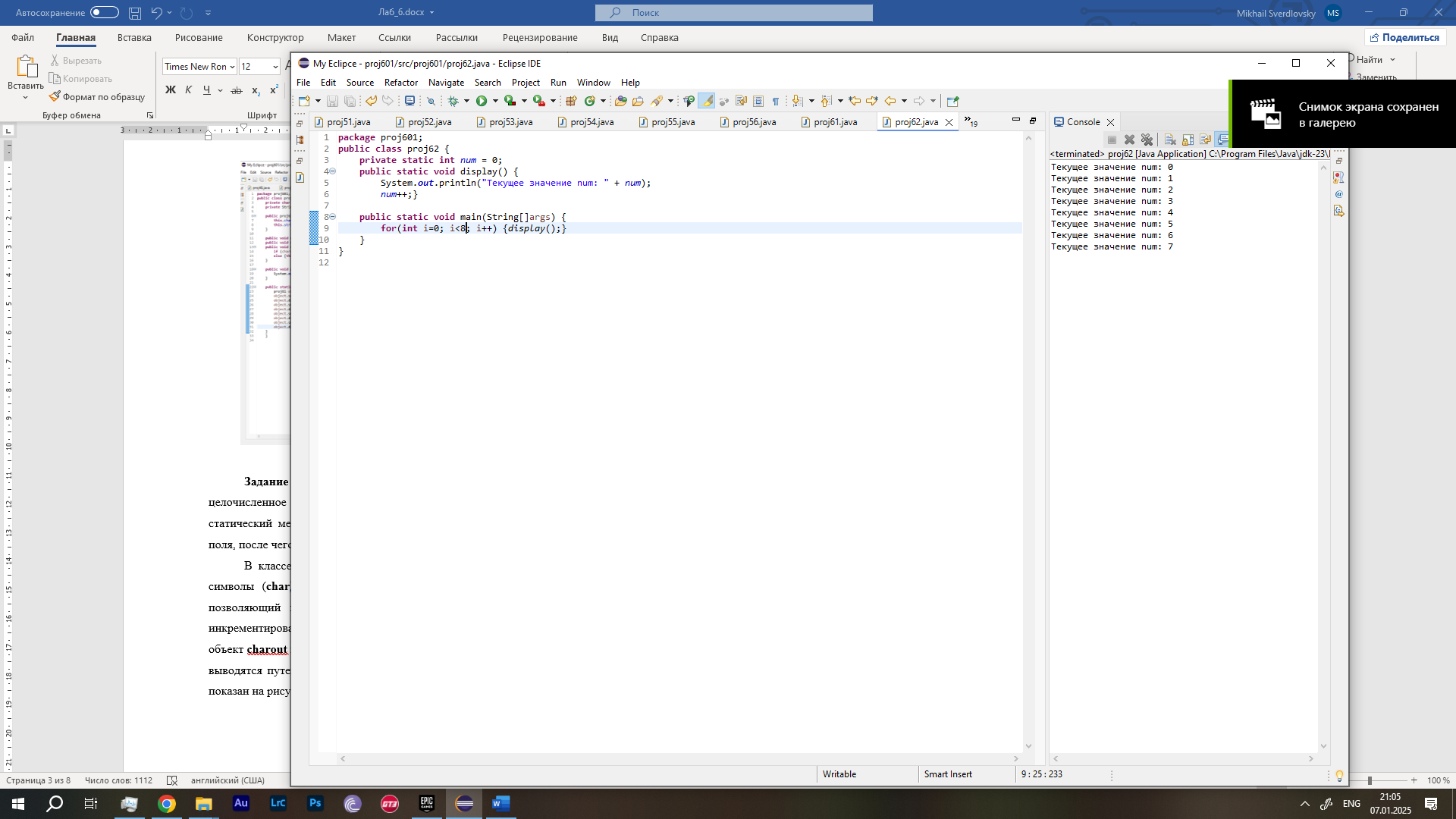


Рисунок 2 – Код и результат работы программы proj62

**Задание 3**. Напишите программу с классом, в котором есть статические методы, которым можно передавать произвольное количество целочисленных аргументов целочисленный массив). Методы, на основании переданных аргументов или массива, позволяют вычислить: наибольшее значение, наименьшее значение, а также среднее значение из набора чисел.

В классе proj63 определяются статические методы min и max, предназначенные для поиска наименьшего и наибольшего числа. Для этого изначально им присваиваются минимально и максимально возможные значения для типа **int**, и, благодаря использованию конструкции **each-for**, происходит сравнение различных аргументов между собой. Оба метода возвращают собой результаты вычислений типа int – **min** и **max**. Также определяется метод для поиска среднего среди какого-либо количества аргументов, а изначально переменной **Snums** присваивается значение 0, а затем он принимает значение суммы всех аргументов делится на их количество. Метод возвращает значение переменной **Snums**. В главном методе выводятся числа, которые были определены для всех методов и результаты вычислений. Результат работы программы и ее код показан на рисунке 3.

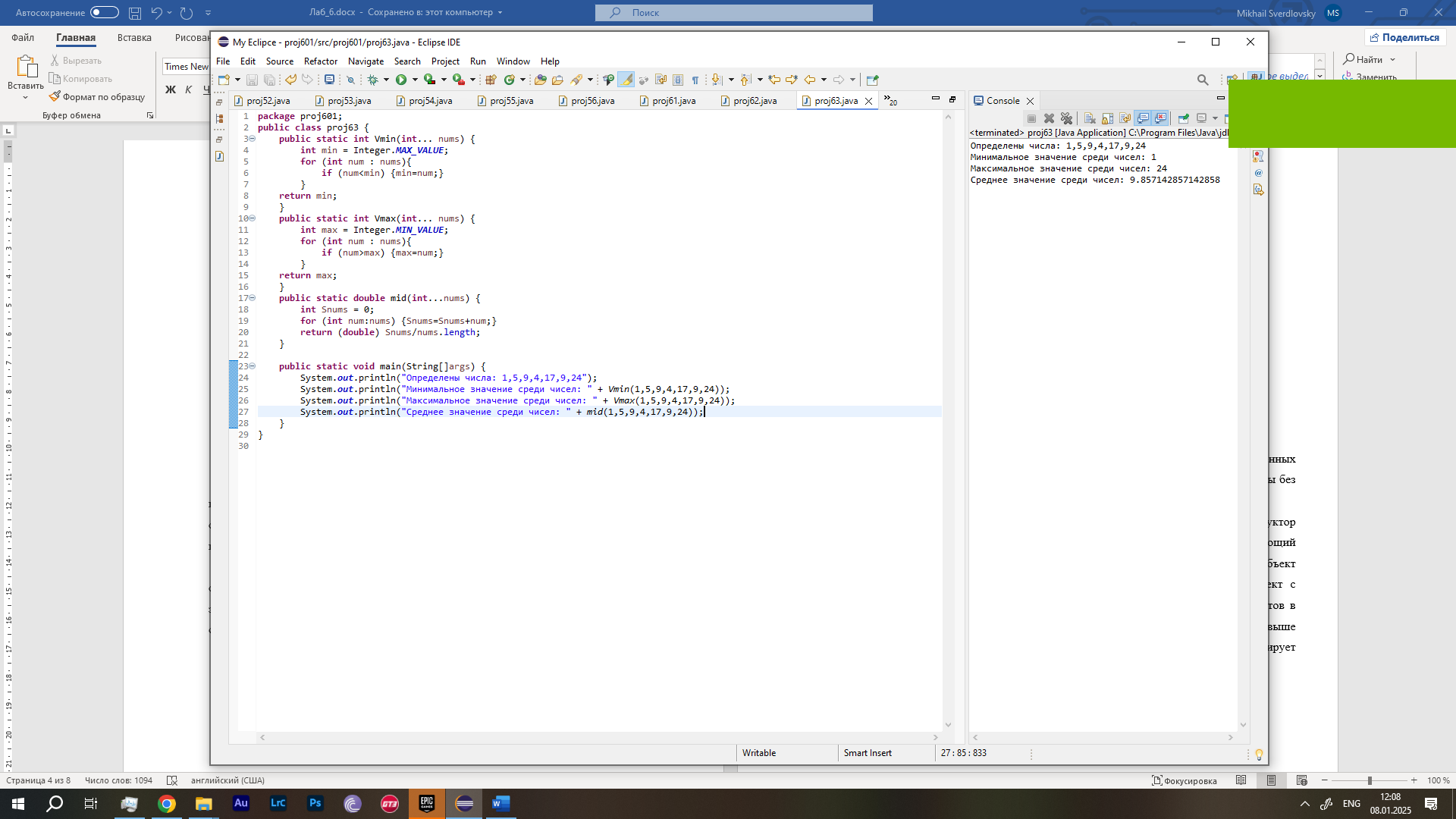


Рисунок 3 – Код и результат работы программы proj63

**Задание 4**. Напишите программу, в которой описан статический метод для вычисления двойного факториала числа, переданного аргументом методу. По определению, двойной факториал числа п (обозначается как n!!) — это произведение через одно всех чисел, не больших числа п. То есть n!! = п \* (n - 2) \* (п - 4)\* ... (последний множитель равен 1 для нечетного п и равен 2 для четного n). Например, 6!! = 6 х 4 х 2 = 48 и 5!! = 5 х 3 х 1 = 15. Предложите версию метода без рекурсии и с рекурсией.

В классе proj64 определяются два метода. Первый метод описывает вычисление двойного факториала без рекурсии (with\_no\_rec), в котором значение факториала fact по умолчанию равно 1. Если переданный аргумент равен 0, метод возвращает 1. В других случаях через цикл for реализуется вычисление значения факториала, значение которого возвращается в результате выполнения метода. Второй метод описывает вычисление двойного факториала с рекурсией (with\_rec), в котором значение факториала fact по умолчанию равно 1. Если переданный аргумент равен 0, или меньше 0, метод возвращает 1. В других случаях результатом работы метода будет последовательное рекурсивное вычисление двойного факториала.

В главном методе реализуется ввод аргумента при помощи консоли. Если было введено отрицательное значение аргумента, будет выведено соответствующее сообщение . Если значение аргумента положительное, в консоль будет выведен результат вычисления двумя методами. Результат работы программы и ее код показан на рисунке 4.

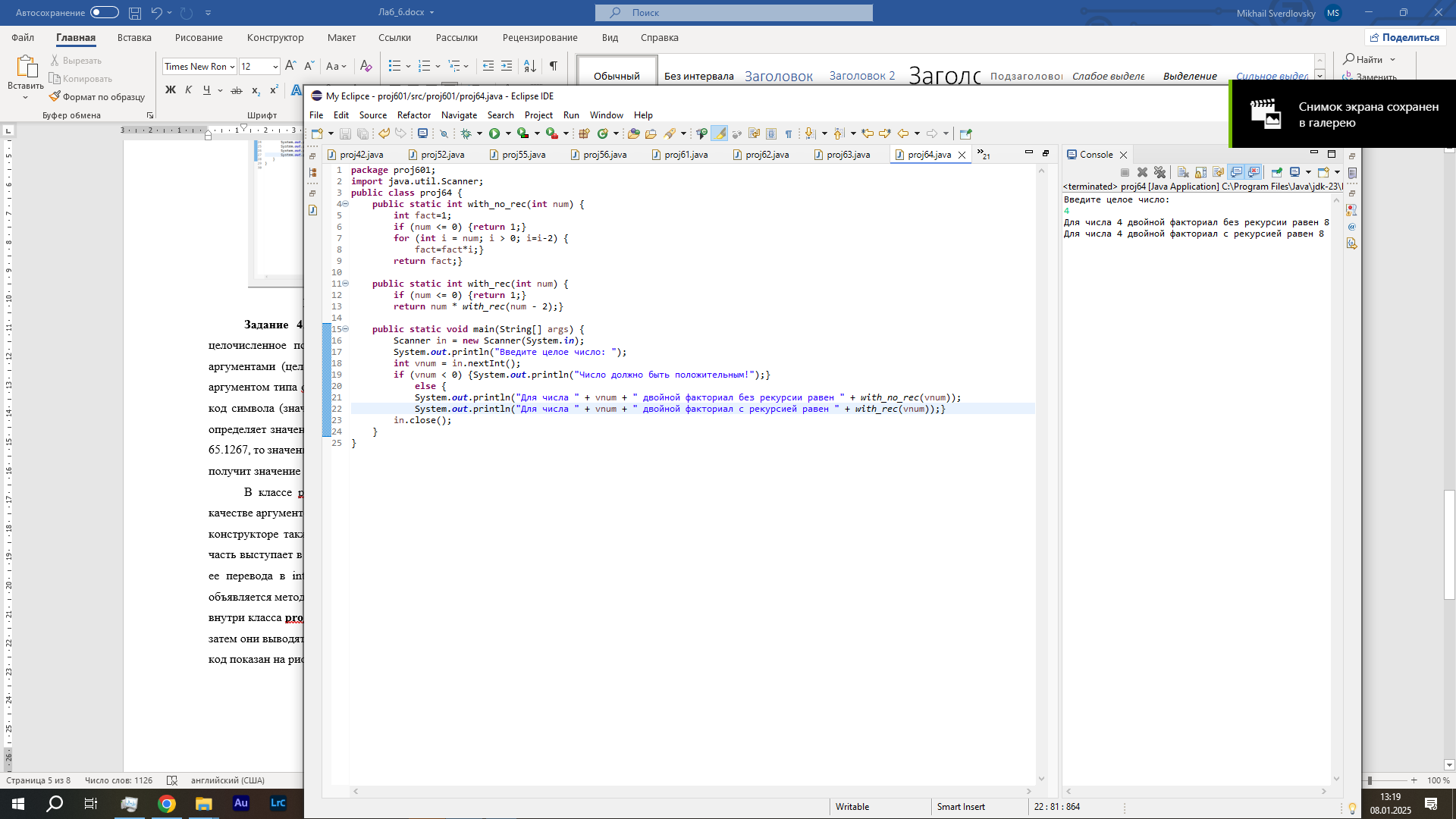


Рисунок 4 – Код и результат работы программы proj64

**Задание 5**. Напишите программу со статическим методом, которым вычисляется сумма квадратов натуральных чисел 12 + 22 + 32 + ... + n2. Число п передается аргументом методу. Предложите версию метода с рекурсией и без рекурсии. Для проверки результата можно использовать формулу 12 + 22 +32+…+n2 =(n+l) (2n + 1)/6.

В классе proj65 объявляется три метода. Первый метод реализует вычисление суммы квадратных чисел без рекурсии (**with\_no\_rec**). По умолчанию значение переменной **Snum** является 0, затем происходит вычисление, реализованное при помощи цикла **for**. Результатом работы метода является возвращение значения суммы квадратных чисел - **Snum**. Второй метод реализует вычисление суммы квадратных чисел с рекурсией (with\_rec). В случае если значение переданного аргумента является 0, результатом выполнения метода является возврат 1. В других случаях результатом работы является возвращение значения рекурсивного вычисления суммы предыдущих аргументов и ее суммы с квадратом текущего. В третьем методе (check) реализована проверка прошлых вычислений по формуле, описанной выше. Результатом его работы является возврат вычисленного значения.

В главном методе реализуется ввод аргумента при помощи консоли. Если было введено значение аргумента, не являющегося натуральным числом, будет выведено соответствующее сообщение. Если значение аргумента соответствует условиям, в консоль будет выведен результат вычисления двумя методами и результат проверки. Результат работы программы и ее код показан на рисунке 5.

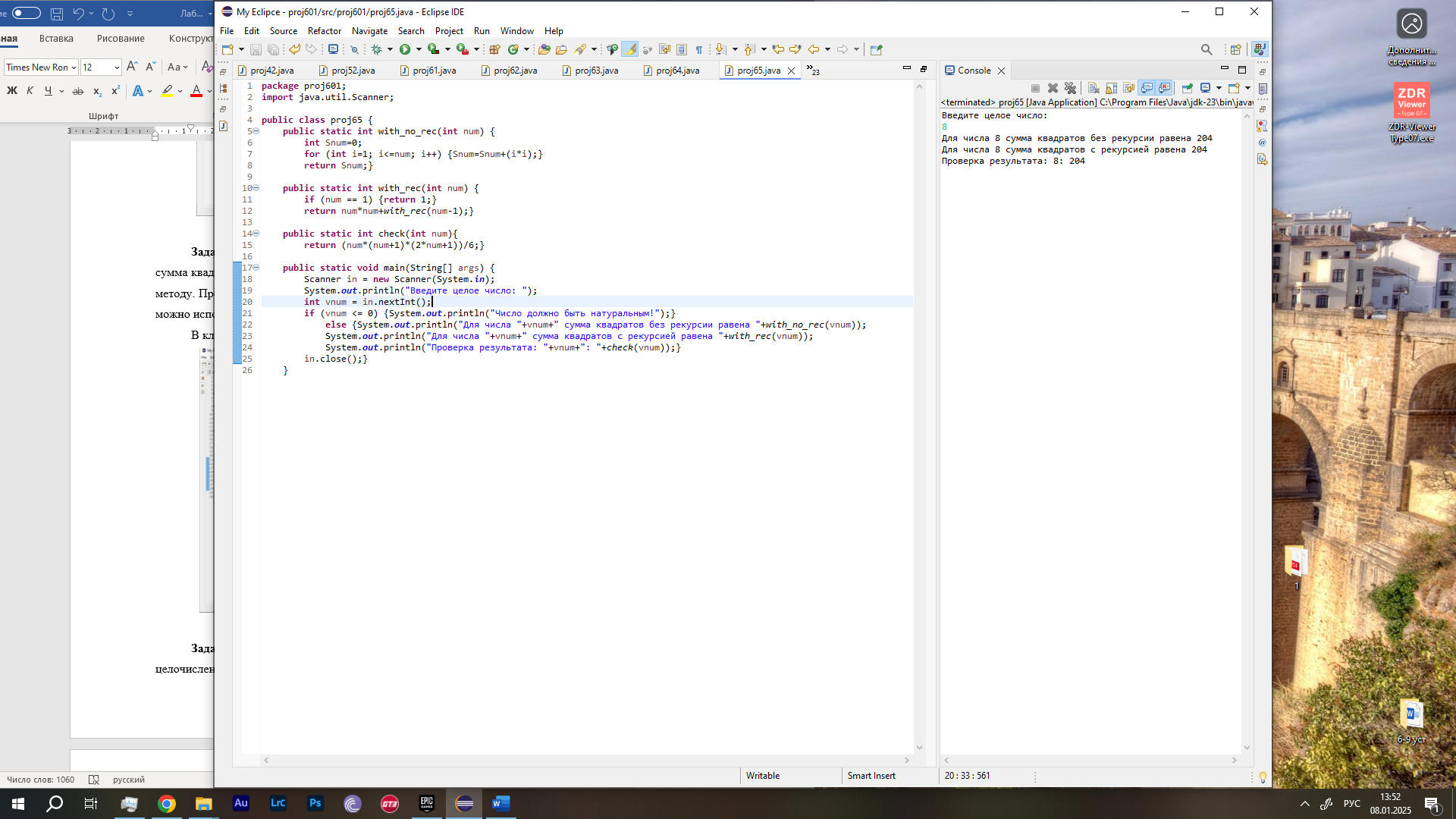


Рисунок 5 – Код и результат работы программы proj65

**Задание 6**. Напишите программу со статическим методом, которому аргументом передается целочисленный массив и целое число. Результатом метод возвращает ссылку на новый массив, который получается из исходного массива (переданного первым аргументом

методу), если в нем взять несколько начальных элементов. Количество элементов, которые нужно взять из исходного массива, определяются вторым аргументом метода. Если второй

аргумент метода больше длины массива, переданного первым аргументом, то методом создается копия исходного массива и возвращается ссылка на эту копию.

В классе proj66 создается статический метод, принимающий массив и аргумент (целое число) – **mass**. Если количество элементов массива больше его длинны, результатом выполнения метода будет возвращение копии массива, реализованную через функцию Arrays.copyOf (копирование массива с возможностью изменения его размера). В других случаях результатом работы метода будет возвращение создаваемого массива, содержащего **num** элементов.

В главном методе определяется значение старого массива и создаются новые с различным количеством элементов. Результаты выводятся в консоль. Результат работы программы и ее код показан на рисунке 6.

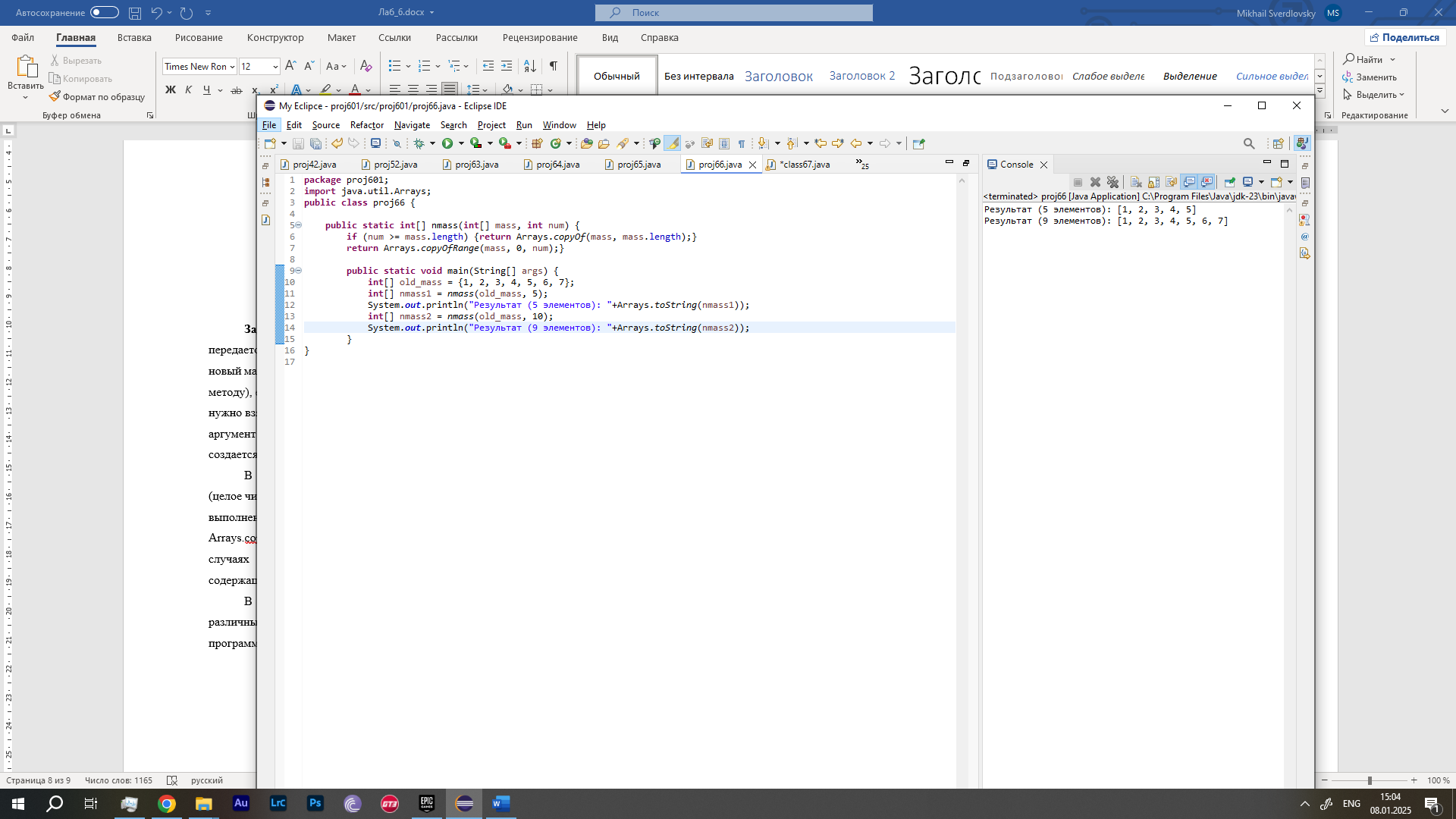


Рисунок 6 – Код и результат работы программы proj66

**Задание 7**. Напишите программу со статическим методом, аргументом которому передастся символьный массив, а результатом возвращается ссылка на целочисленным массив, состоящий из кодов символов из массива-аргумента.

В классе proj67 создается метод **char\_mass\_do**, преобразующий массив символов в целочисленный массив кодов этих символов. Для этого внутри метода создается новый целочисленный массив размером с количество элементов массива символов. Далее, с помощью цикла **for**, каждому целочисленному элементу массива присваивается код символа при помощи конвертации char в int. Результатом вычислений метода будет возвращение массива **symb\_mass**. В главном методе создается массив символов, и, затем, создается массив кодов символов, который содержит в себе результат выполнения метода **char\_mass\_do** с **char\_mass** в качестве аргумента. Далее выводится массив символов и массив кодов символов при помощи метода **Arrays.toString**, встроенного в **java.util.Arrays**. Результат работы программы и ее код показан на рисунке 7.

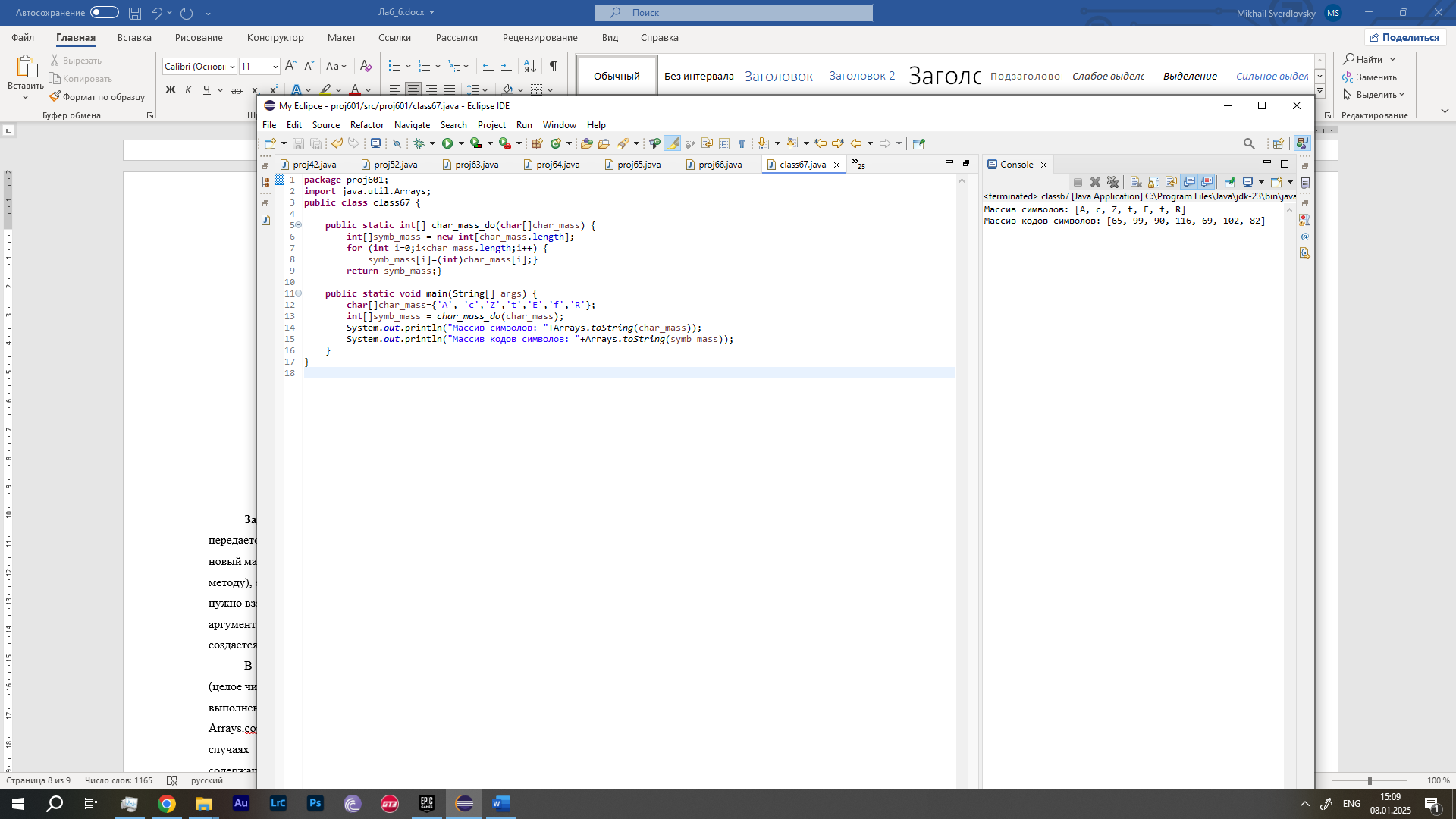


Рисунок 7 – Код и результат работы программы proj67

**Задание 8**. Напишите программу со статическим методом, аргументом которому передается целочисленный массив, а результатом возвращается среднее значение для элементов массива (сумма значений элементов, деленная на количество элементов в массиве).

В классе proj68 создается метод **int\_avr**, который в качестве аргумента принимает элементы массива **int\_mass** и возвращает значение типа double. Если длинна массива равна 0 (массив пустой), в результате выполнения программы будет возвращаться значение 0. Если массив не пустой, создается целочисленная переменная snum, которой по умолчанию присваивается значение 0. Далее, с помощью цикла конструкции each-for, поочередно складываются элементы массива, и, после этого, разделяются на длину массива (количество элементов). В главном методе инициализируется целочисленный массив **int\_mass**. Затем, при помощи метода **Arrays.toString**, встроенного в **java.util.Arrays**, выводится целочисленный массив, и следом выводится результат получения среднего числа методом **int\_avr**, в качестве аргумента имеющего **int\_mass**. Результат работы программы и ее код показан на рисунке 8.

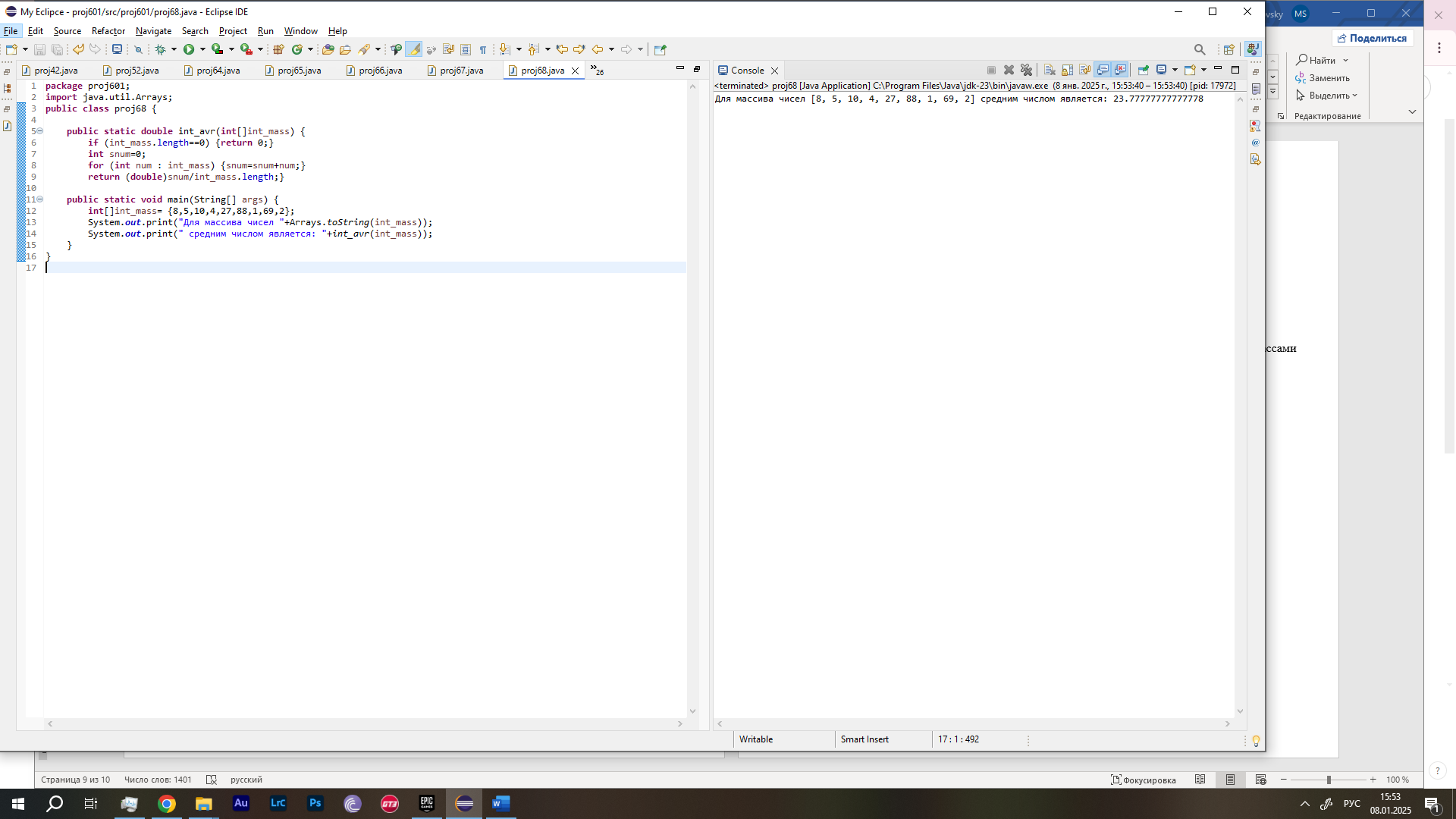


Рисунок 8 – Код и результат работы программы proj68

**Задание 9**. Напишите программу со статическим методом, аргументом которому передается одномерный символьный массив. В результате вызова метода элементы массива попарно меняются местами: первый — с последним, второй — с предпоследним и так далее.

В классе proj69 создается метод **swap\_char\_mass**, возвращающий после выполнения массив символов. В методе инициализируются номера символов (**numl** – принимает значение 0 и **numr** – принимает значение на 1 меньше чем длинна массива). С помощью функции цикла **while** реализуется замена мест нахождения символов с использованием инициализированной в теле цикла переменной **tmp**, а также инкрементирование или декрементирование счетчика символов. Результатом работы метода является возвращение преобразованного массива **mass**. В главном методе происходит инициализация массива char\_mass и его вывод в консоль с помощью метода **Arrays.toString**, встроенного в **java.util.Arrays.** Затем, на экран с помощью **Arrays.toString** выводится массив chars\_mass, который получил значение, возвращаемое при выполнении **swap\_char\_mass** с **chars\_mass** в качестве аргумента. Результат работы программы и ее код показан на рисунке 9.

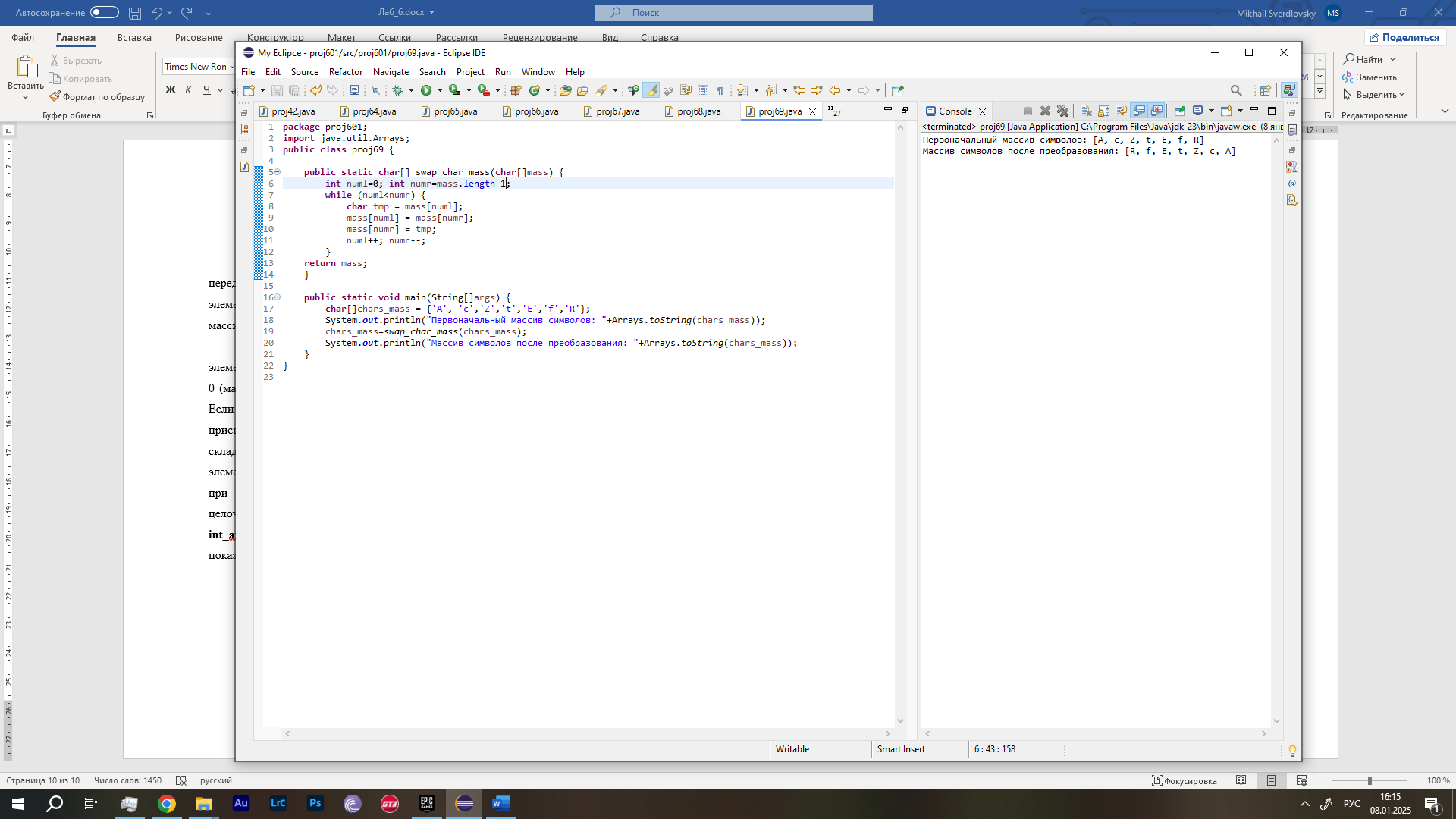


Рисунок 9 – Код и результат работы программы proj69

**Задание 10**. Напишите программу со статическим методом, аргументом которому передается произвольное количество целочисленных аргументов. Результатом метод возвращает массив из двух элементов: это значения наибольшего и наименьшего значений среди аргументов, переданных методу.

В классе proj610 создается метод **check\_nums**, который возвращает после своего выполнения массив целочисленных данных, а в качестве аргумента принимает неопределенное количество целочисленных аргументов. Создается массив **numchecked**, для хранения минимального и максимального значения чисел; создаются переменные max и min, принимающие значения предельно малого и предельно большого значения для этого типа данных. Далее с помощью цикла **for** реализуется поочередное сравнение переменных с аргументами, результат записывается в массив **numchecked**. В главном методе объявляется массив данных **nums**, который выступает в роли аргументов для предыдущего метода, он выводится на экран с помощью **Arrays.toString.** Затем массив **nums** принимает значения, которые возвращает после завершения работы метод **check\_nums**, и минимальные и максимальные значения выводятся в консоль. Результат работы программы и ее код показан на рисунке 10.

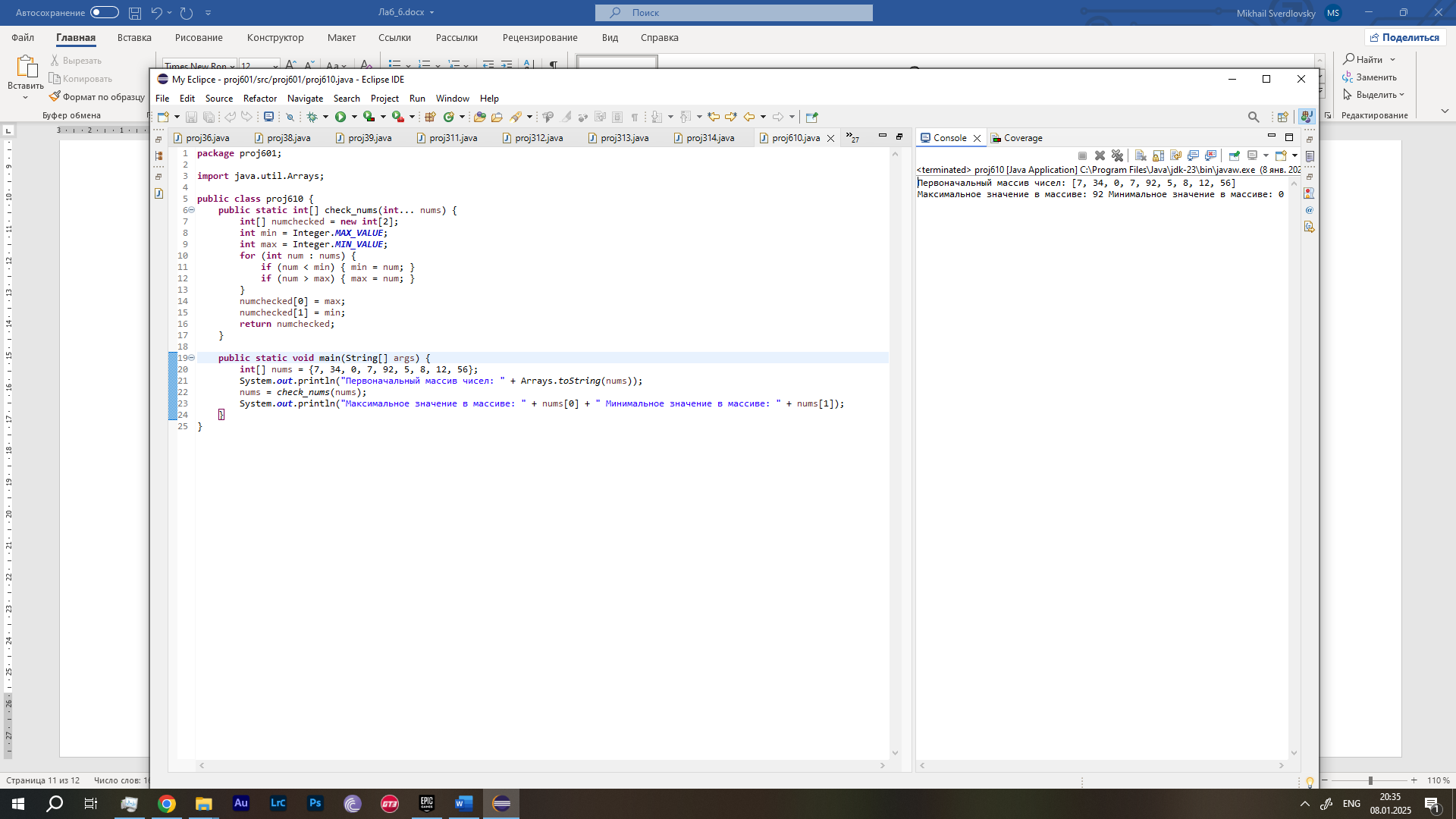


Рисунок 10 – Код и результат работы программы proj610

**Вывод**

В результате лабораторной работы было пройдено введение в работу с классами Java.