41. Write a program to find the square, cube of the given decimal number

CODE:

import java.util.Scanner;

public class SquareAndCube {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Getting the decimal number from the user

System.out.print("Enter a decimal number: ");

double number = scanner.nextDouble();

// Calculating and displaying the square

double square = findSquare(number);

System.out.println("Square of " + number + " is: " + square);

// Calculating and displaying the cube

double cube = findCube(number);

System.out.println("Cube of " + number + " is: " + cube);

scanner.close();

}

public static double findSquare(double number) {

return Math.pow(number, 2);

}

public static double findCube(double number) {

return Math.pow(number, 3);

}

}

OUTPUT:

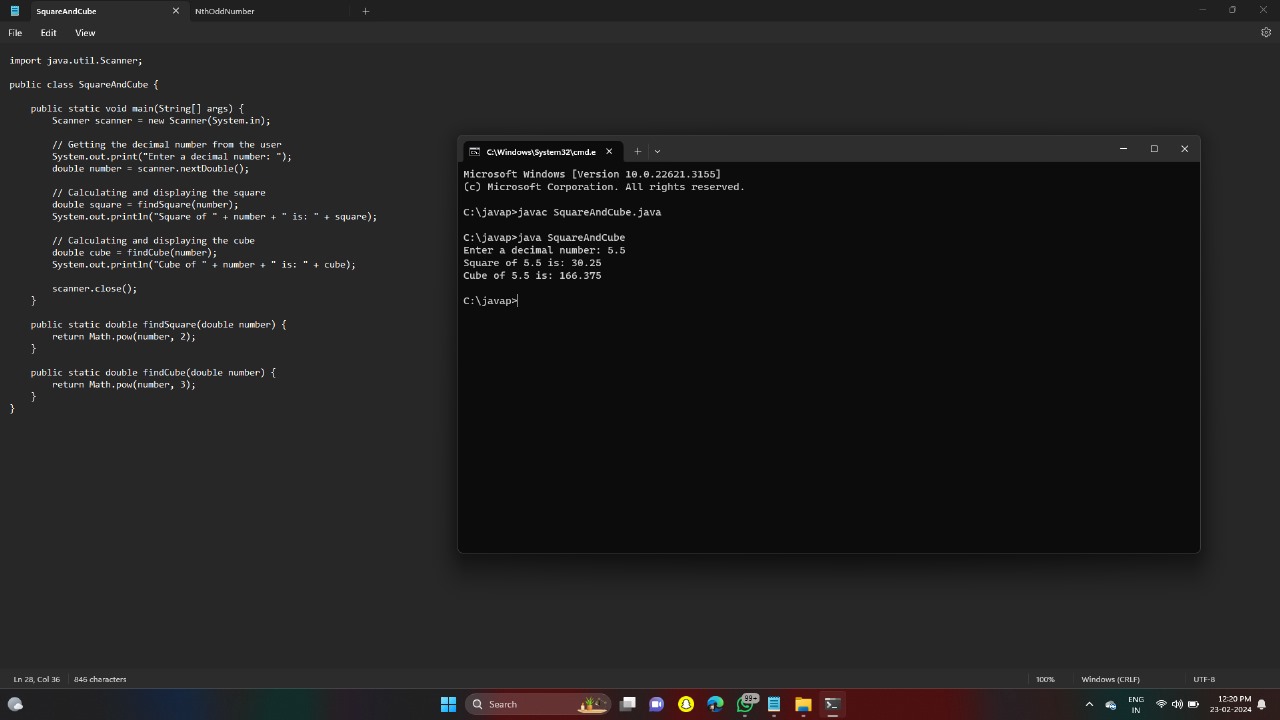
C:\javap>javac SquareAndCube.java

C:\javap>java SquareAndCube

Enter a decimal number: 5.5

Square of 5.5 is: 30.25

Cube of 5.5 is: 166.375



42. Find the nth odd number after n odd number

CODE:

import java.util.Scanner;

public class NthOddNumber {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Getting the value of n and m from the user

System.out.print("Enter the value of m (number of initial odd numbers): ");

int m = scanner.nextInt();

System.out.print("Enter the value of n (position of the odd number): ");

int n = scanner.nextInt();

// Calculating and displaying the nth odd number after m odd numbers

int nthOddNumber = findNthOddNumber(m, n);

System.out.println("The " + n + "th odd number after " + m + " odd numbers is: " + nthOddNumber);

scanner.close();

}

public static int findNthOddNumber(int m, int n) {

return m \* 2 + n \* 2 - 1;

}

}

OUTPUT:

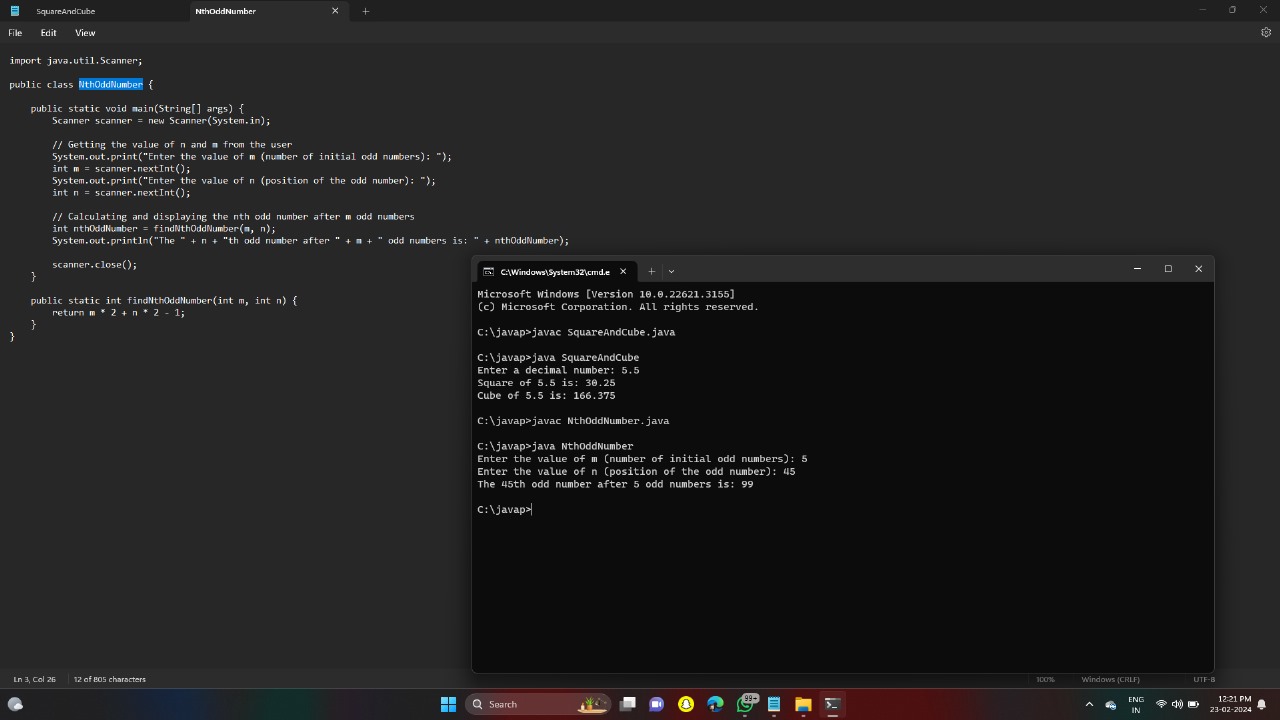
C:\javap>javac NthOddNumber.java

C:\javap>java NthOddNumber

Enter the value of m (number of initial odd numbers): 5

Enter the value of n (position of the odd number): 45

The 45th odd number after 5 odd numbers is: 99



43. Program to find whether the given number is Armstrong number or not

CODE:

import java.util.Scanner;

public class ArmstrongNumber {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number: ");

int number = scanner.nextInt();

scanner.close();

if (isArmstrong(number))

System.out.println(number + " is an Armstrong number.");

else

System.out.println(number + " is not an Armstrong number.");

}

public static boolean isArmstrong(int num) {

int originalNum, remainder, result = 0, n = 0;

originalNum = num;

// Count number of digits

for (; originalNum != 0; originalNum /= 10, ++n);

originalNum = num;

// Calculate sum of nth power of individual digits

for (; originalNum != 0; originalNum /= 10) {

remainder = originalNum % 10;

result += Math.pow(remainder, n);

}

// Check if number is Armstrong

if (result == num)

return true;

else

return false;

}

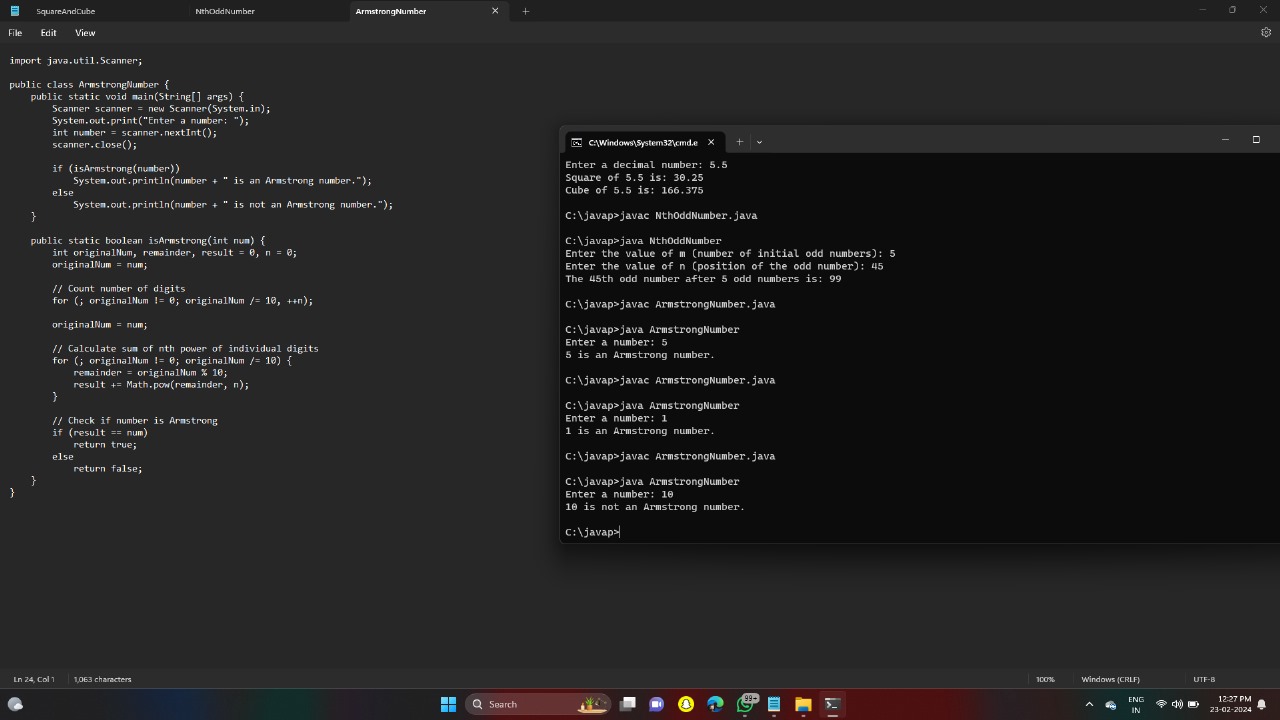
}

OUTPUT:

C:\javap>java ArmstrongNumber

Enter a number: 10

10 is not an Armstrong number.



44. Write a program to find the sum of digits of N digit number (sum should be single digit)

CODE:

import java.util.Scanner;

public class SumOfDigits {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter an N-digit number: ");

int number = scanner.nextInt();

scanner.close();

int sum = calculateSingleDigitSum(number);

System.out.println("Sum of digits until single digit: " + sum);

}

public static int calculateSingleDigitSum(int num) {

int sum = num;

while (sum > 9) {

int tempSum = 0;

// Sum up digits of the current number

while (sum > 0) {

tempSum += sum % 10;

sum /= 10;

}

sum = tempSum;

}

return sum;

}

}

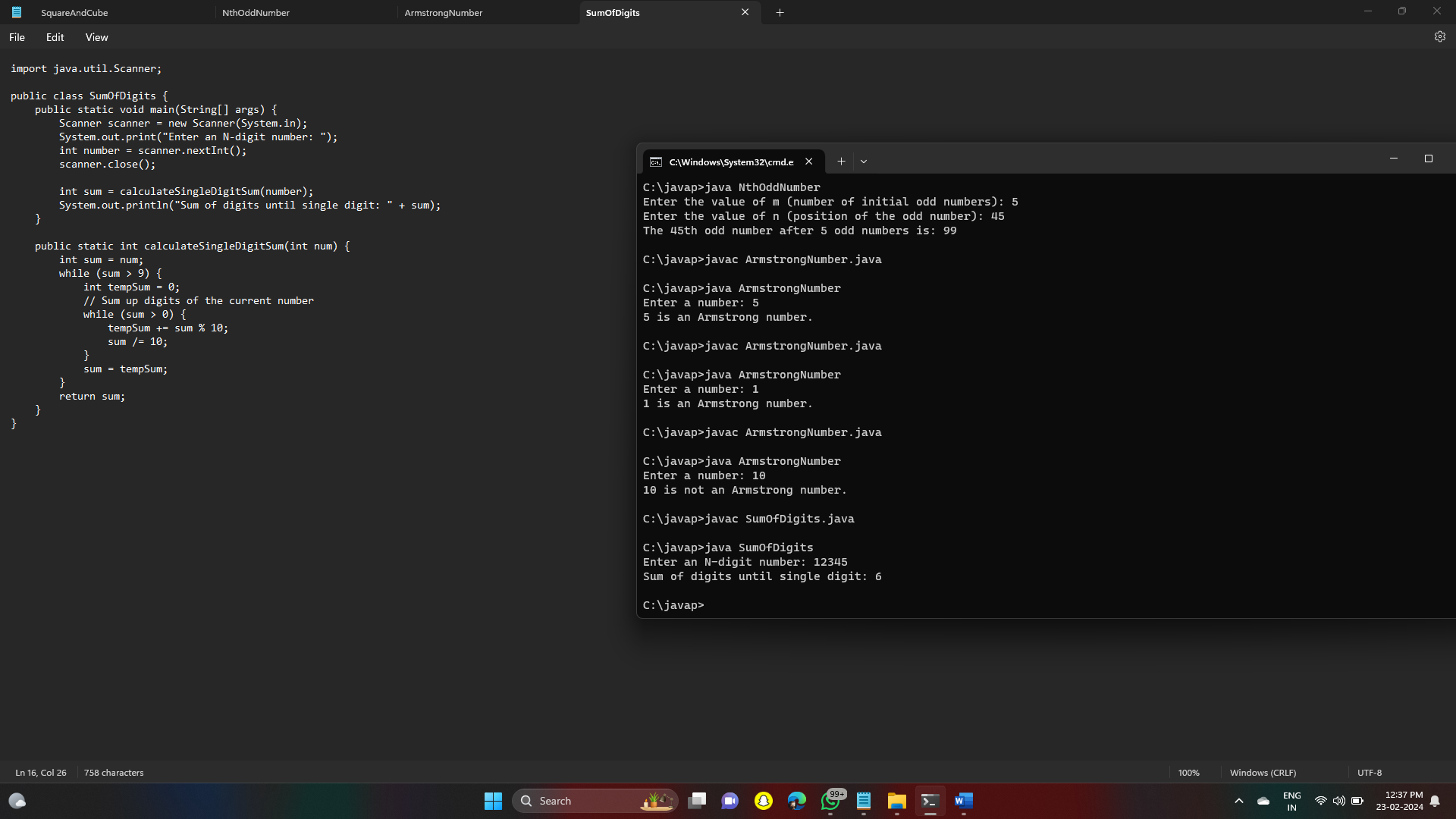
OUTPUT:

C:\javap>javac SumOfDigits.java

C:\javap>java SumOfDigits

Enter an N-digit number: 12345

Sum of digits until single digit: 6



45. Write a program to find the square root of a perfect square number(print both the positive and negative values)

CODE:

import java.util.Scanner;

public class PerfectSquareRoot {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a perfect square number: ");

int number = scanner.nextInt();

scanner.close();

double squareRootPositive = Math.sqrt(number);

double squareRootNegative = -Math.sqrt(number);

if (number < 0 || Math.floor(squareRootPositive) != squareRootPositive) {

System.out.println("The number entered is not a perfect square.");

} else {

System.out.println("Square root (positive value): " + squareRootPositive);

System.out.println("Square root (negative value): " + squareRootNegative);

}

}

}

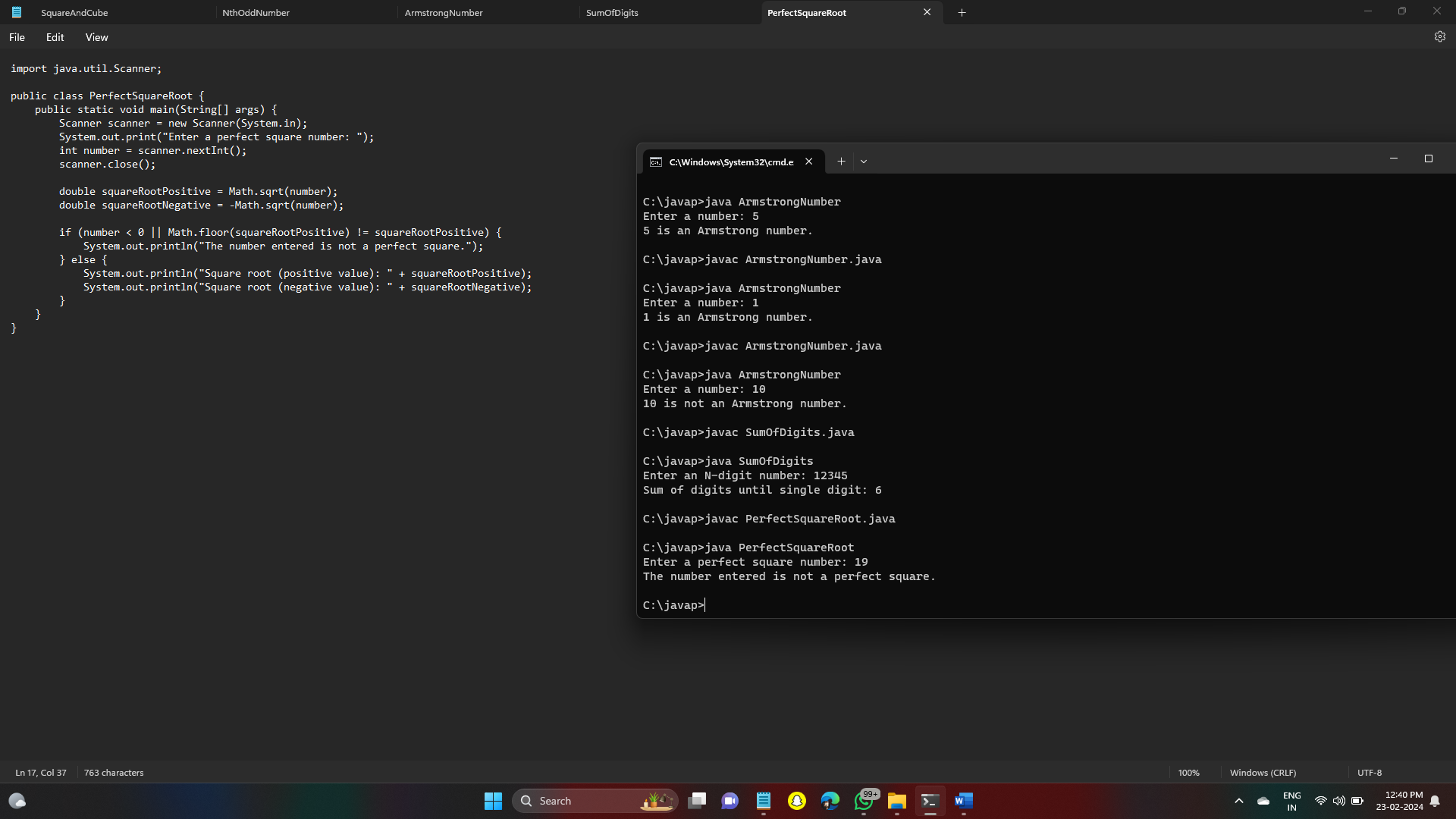
OUTPUT:

C:\javap>javac PerfectSquareRoot.java

C:\javap>java PerfectSquareRoot

Enter a perfect square number: 19

The number entered is not a perfect square.



46. Write a program to given an integer n, return true if it is a power of three. Otherwise, return false.

CODE:

import java.util.Scanner;

public class PowerOfThree {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter an integer: ");

int n = scanner.nextInt();

scanner.close();

boolean isPowerOfThree = checkPowerOfThree(n);

if(isPowerOfThree)

System.out.println(n + " is a power of three.");

else

System.out.println(n + " is not a power of three.");

}

public static boolean checkPowerOfThree(int n) {

if (n <= 0) return false;

while (n % 3 == 0) {

n /= 3;

}

return n == 1;

}

}

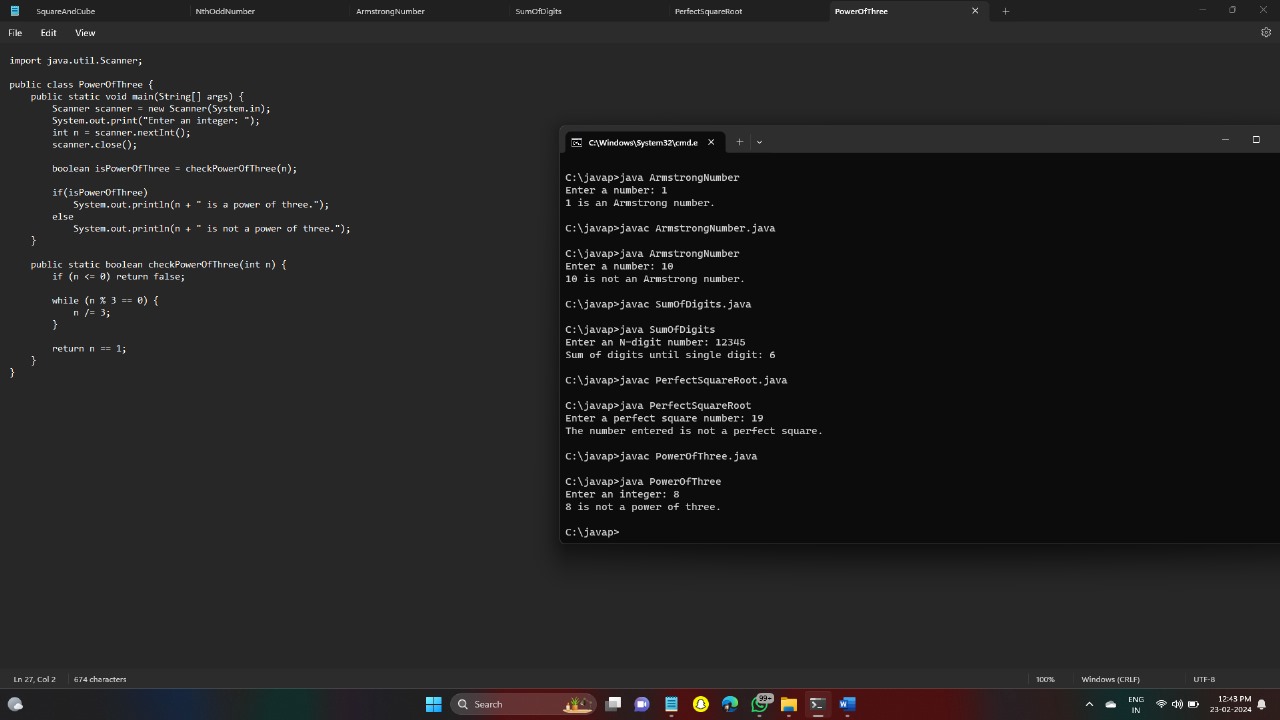
OUTPUT:

C:\javap>javac PowerOfThree.java

C:\javap>java PowerOfThree

Enter an integer: 8

8 is not a power of three.



47. Write a program to given a string paragraph and a string array of the banned words banned, return the most frequent word that is not banned. It is guaranteed there is at least one word that is not banned, and that the answer is unique.

CODE:

import java.util.\*;

public class MostFrequentWord {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the paragraph:");

String paragraph = scanner.nextLine();

System.out.println("Enter the banned words (comma-separated):");

String bannedString = scanner.nextLine();

String[] banned = bannedString.split(",");

scanner.close();

String result = findMostFrequentWord(paragraph, banned);

System.out.println("The most frequent word not banned is: " + result);

}

public static String findMostFrequentWord(String paragraph, String[] banned) {

Set<String> bannedSet = new HashSet<>(Arrays.asList(banned));

Map<String, Integer> wordFreq = new HashMap<>();

String[] words = paragraph.toLowerCase().split("\\W+");

for (String word : words) {

if (!bannedSet.contains(word)) {

wordFreq.put(word, wordFreq.getOrDefault(word, 0) + 1);

}

}

String mostFrequentWord = "";

int maxFrequency = 0;

for (Map.Entry<String, Integer> entry : wordFreq.entrySet()) {

if (entry.getValue() > maxFrequency) {

maxFrequency = entry.getValue();

mostFrequentWord = entry.getKey();

}

}

return mostFrequentWord;

}

}

OUTPUT:

C:\javap>javac MostFrequentWord.java

C:\javap>java MostFrequentWord

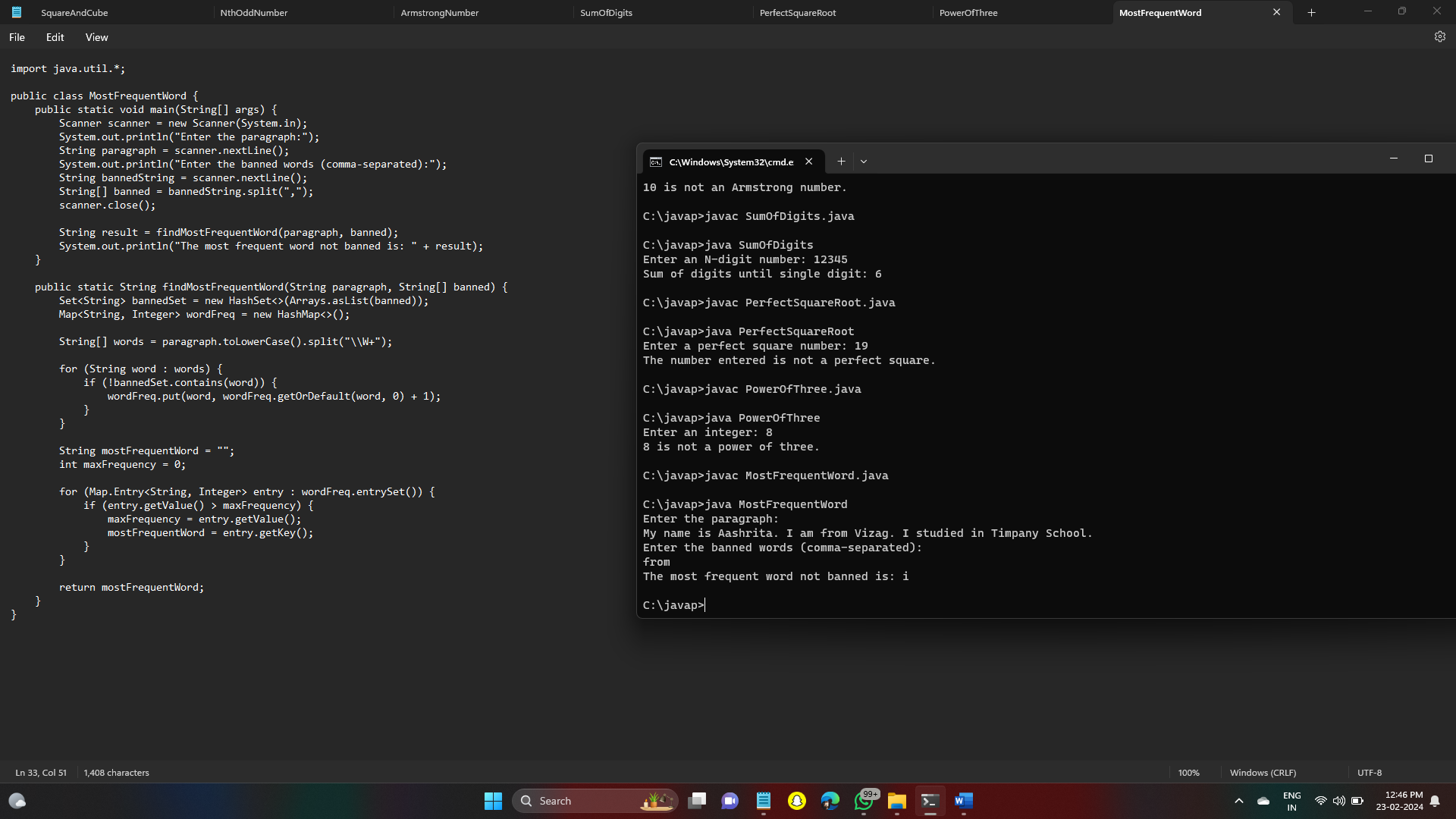
Enter the paragraph:

My name is Aashrita. I am from Vizag. I studied in Timpany School.

Enter the banned words (comma-separated):

from

The most frequent word not banned is: i



48. Write a program to given a fixed-length integer array arr, duplicate each occurrence of zero, shifting the remaining elements to the right.

CODE:

import java.util.Scanner;

public class DuplicateZeros {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the length of the array: ");

int length = scanner.nextInt();

int[] arr = new int[length];

System.out.println("Enter the elements of the array:");

for (int i = 0; i < length; i++) {

arr[i] = scanner.nextInt();

}

scanner.close();

duplicateZeros(arr);

System.out.println("Array after duplicating zeros:");

for (int num : arr) {

System.out.print(num + " ");

}

}

public static void duplicateZeros(int[] arr) {

int possibleDups = 0;

int length = arr.length - 1;

// Count the number of zeros to be duplicated

for (int i = 0; i <= length - possibleDups; i++) {

if (arr[i] == 0) {

if (i == length - possibleDups) {

// For cases where there are no more space left to duplicate zeros

arr[length] = 0;

length--;

break;

}

possibleDups++;

}

}

// Start from the last element and move elements to their correct positions

for (int i = length - possibleDups; i >= 0; i--) {

if (arr[i] == 0) {

arr[i + possibleDups] = 0;

possibleDups--;

arr[i + possibleDups] = 0;

} else {

arr[i + possibleDups] = arr[i];

}

}

}

}

OUTPUT:

C:\javap>javac DuplicateZeros.java

C:\javap>java DuplicateZeros

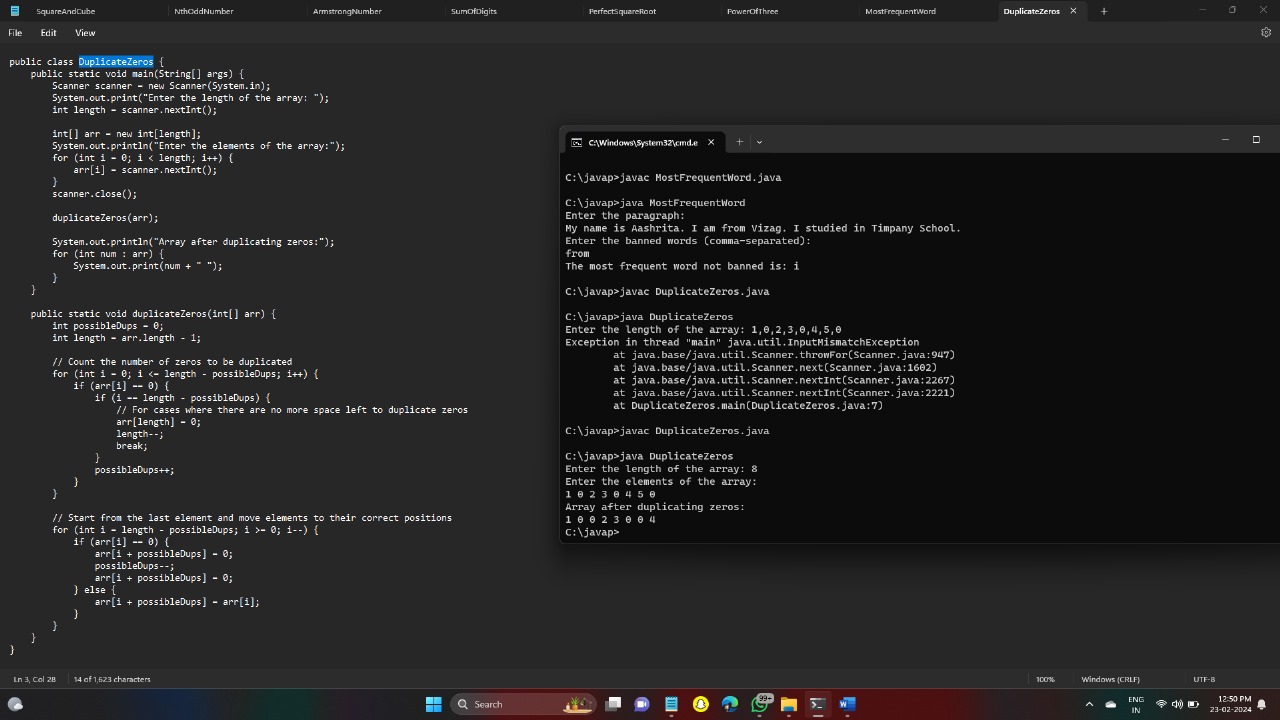
Enter the length of the array: 8

Enter the elements of the array:

1 0 2 3 0 4 5 0

Array after duplicating zeros:

1 0 0 2 3 0 0 4



49. Write a program to given an array nums containing n distinct numbers in the range [0, n], return the only number in the range that is missing from the array.

CODE:

import java.util.Scanner;

public class MissingNumber {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the length of the array: ");

int length = scanner.nextInt();

int[] nums = new int[length];

System.out.println("Enter the elements of the array:");

for (int i = 0; i < length; i++) {

nums[i] = scanner.nextInt();

}

scanner.close();

int missingNumber = findMissingNumber(nums);

System.out.println("The missing number in the array is: " + missingNumber);

}

public static int findMissingNumber(int[] nums) {

int n = nums.length;

int totalSum = (n \* (n + 1)) / 2;

int arraySum = 0;

// Calculate the sum of all elements in the array

for (int num : nums) {

arraySum += num;

}

// The missing number is the difference between the sum of all natural numbers up to n

// and the sum of elements in the array

return totalSum - arraySum;

}

}

OUTPUT:

C:\javap>javac MissingNumber.java

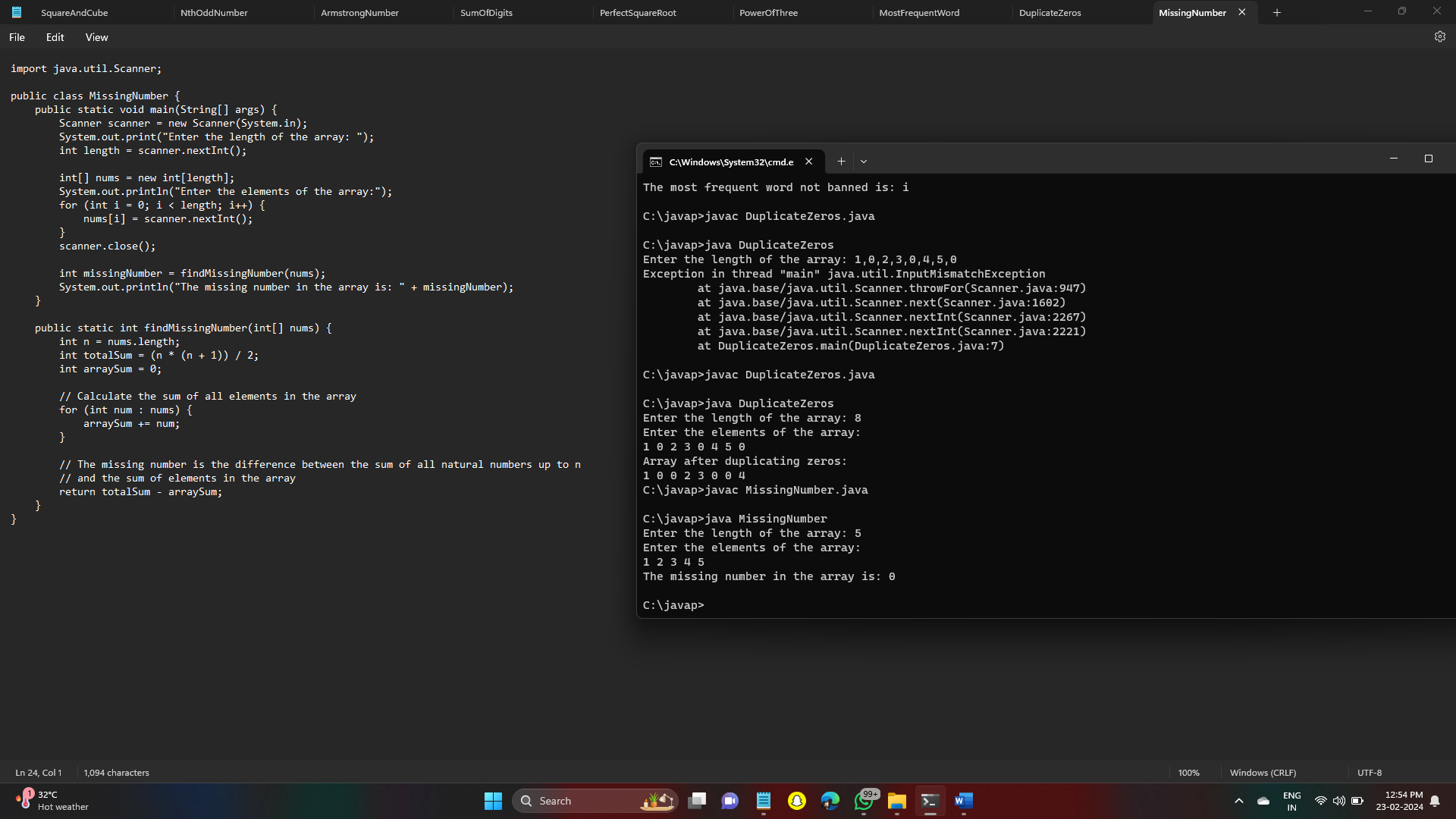
C:\javap>java MissingNumber

Enter the length of the array: 5

Enter the elements of the array:

1 2 3 4 5

The missing number in the array is: 0



50. Write a program to given an integer array nums, find the subarray with the largest sum, and return its sum.

CODE:

import java.util.Scanner;

public class MaximumSubarraySum {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the length of the array: ");

int length = scanner.nextInt();

int[] nums = new int[length];

System.out.println("Enter the elements of the array:");

for (int i = 0; i < length; i++) {

nums[i] = scanner.nextInt();

}

scanner.close();

int maxSum = findMaximumSubarraySum(nums);

System.out.println("The sum of the largest subarray is: " + maxSum);

}

public static int findMaximumSubarraySum(int[] nums) {

int maxSum = nums[0];

int currentSum = nums[0];

for (int i = 1; i < nums.length; i++) {

// Choose between extending the current subarray or starting a new subarray

currentSum = Math.max(nums[i], currentSum + nums[i]);

// Update the maximum sum if the current subarray sum is greater

maxSum = Math.max(maxSum, currentSum);

}

return maxSum;

}

}

OUTPUT:

C:\javap>javac MaximumSubarraySum.java

C:\javap>java MaximumSubarraySum

Enter the length of the array: 7

Enter the elements of the array:

1 2 3 4 -7 -8 -9

The sum of the largest subarray is: 10

