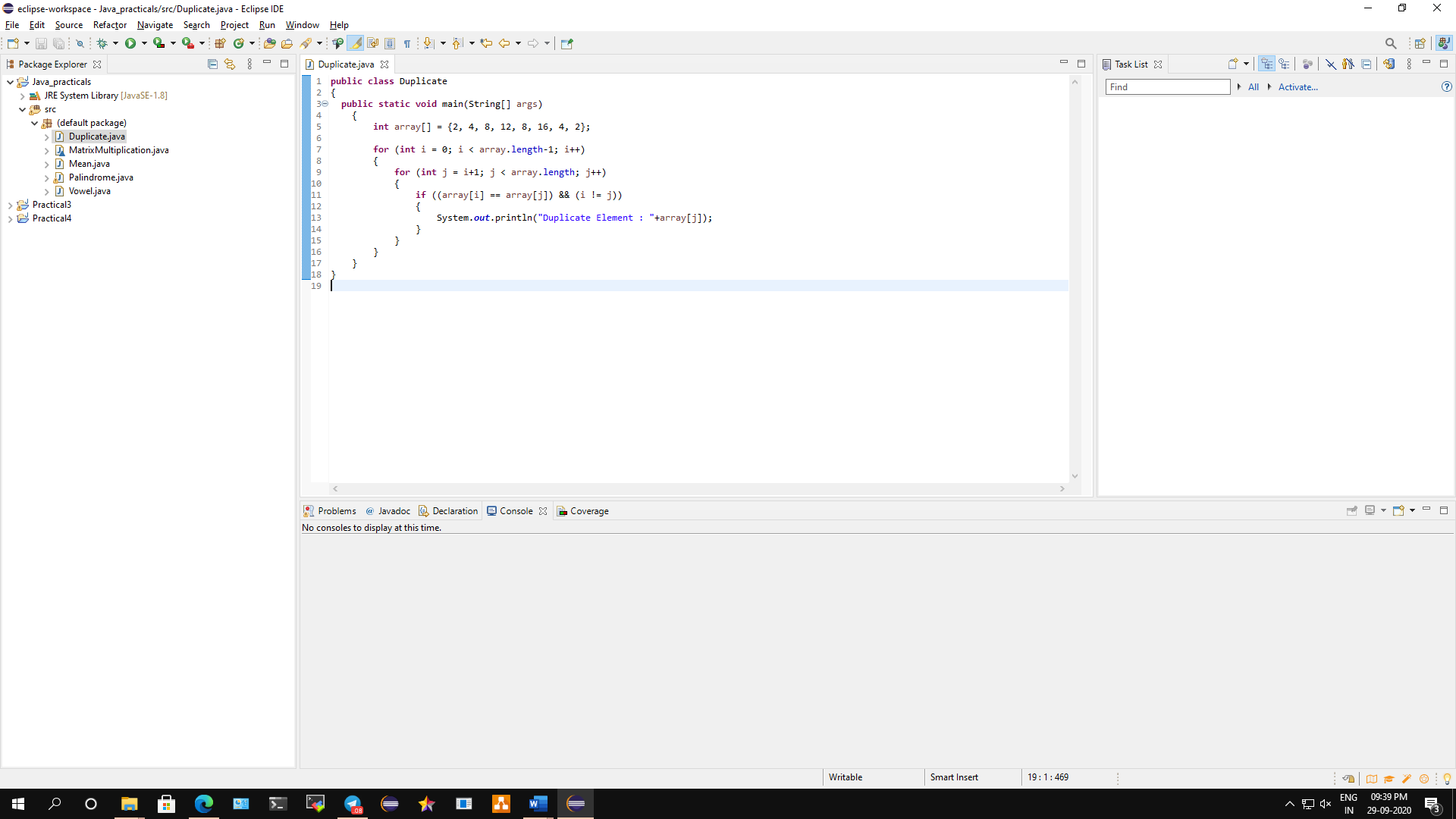
**Practical No 2**

**A) Program to check if the array has any duplicate element**

**Aim: Write a program to check if the array has any duplicate element in java.**

**Description:**

An array is a collection of similar type of elements which has contiguous memory location. Java array is an object which contains elements of a similar data type. Additionally, the elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array. Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on. Unlike C/C++, we can get the length of the array using the length member. In C/C++, we need to use the sizeof operator. In Java, array is an object of a dynamically generated class. Java array inherits the Object class, and implements the Serializable as well as Cloneable interfaces. We can store primitive values or objects in an array in Java. Like C/C++, we can also create single dimensional or multidimensional arrays in Java. We have defined the class name which is class Duplicate. A class is a blueprint for the object, it represents a set of properties or methods that are common to all objects of one type. Now we have defined the main method to accept an array of string arguments. We have 2 loops in one set of arrays, the first loop i.e. i will loop from start to end traversal. Then second loop i.e. j which is an inner loop which will take the value i to compare with another loop to find the duplicate array. Once we have found the duplicate array it will stop compiling and get the output.



**Conclusion: We have written a program to check if the array has any duplicate element in java.**

**Code:**

**public** **class** Duplicate

{

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

{

**int** array[] = {2, 4, 8, 12, 8, 16, 4, 2};

**for** (**int** i = 0; i < array.length-1; i++)

{

**for** (**int** j = i+1; j < array.length; j++)

{

**if** ((array[i] == array[j]) && (i != j))

{

System.***out***.println("Duplicate Element: "+array[j]);

}

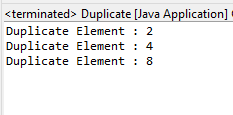
}

}

}

}

**Output:**

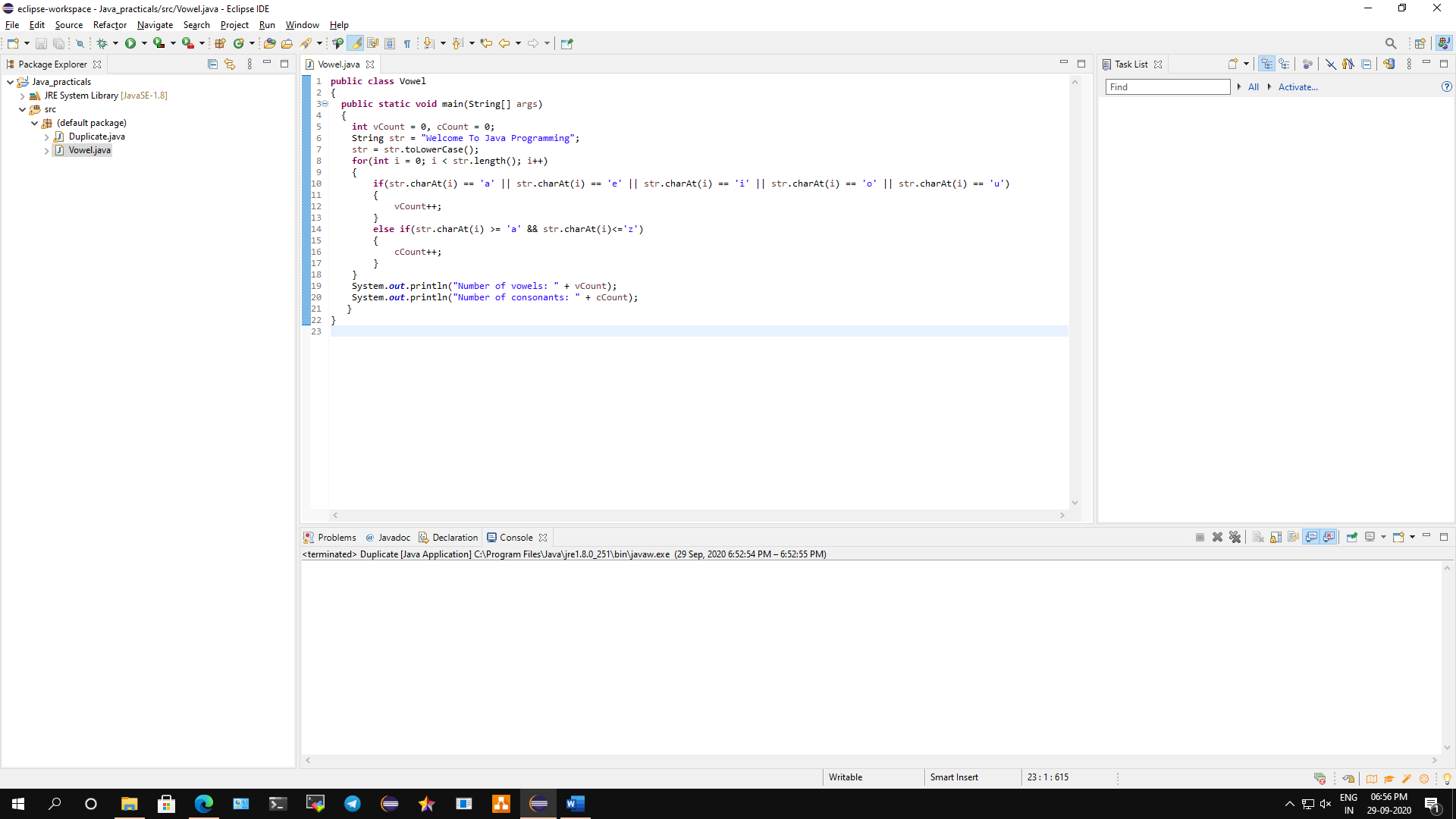


**B) Program to count the no of Vowel and Consonant in the given string.**

**Aim: Write a java program to count the no of Vowel and Consonant in the given string.**

**Description:**

We have defined the class name which is class Vowel. A class is a blueprint for the object, it represents a set of properties or methods that are common to all objects of one type. Our task is to count the total number of vowels and consonants present in the given string. We know that, the characters a, e, i, o, u is known as vowels in the English alphabet and any other character than vowels are known as consonant. Now we need to convert every upper-case character in the string to lower-case so that the comparisons can be done with the lower-case vowels only not upper-case vowels, i.e. (A, E, I, O, U). Then, we have to traverse the string using a for or while loop and match each character with all the vowels, i.e., a, e, i, o, u. If the match is found, increase the value of count by 1 otherwise continue with the normal flow of the program. Once the code is compiled, we will get the number of vowels and consonants in the output.



**Conclusion: We have written a program to count the no of Vowel and Consonant in the given string.**

**Code:**

**public** **class** Vowel

{

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

{

**int** vCount = 0, cCount = 0;

String str = "Welcome To Java Programming";

str = str.toLowerCase();

**for**(**int** i = 0; i < str.length(); i++)

{

**if**(str.charAt(i) == 'a' || str.charAt(i) == 'e' || str.charAt(i) == 'i' || str.charAt(i) == 'o' || str.charAt(i) == 'u')

{

vCount++;

}

**else** **if**(str.charAt(i) >= 'a' && str.charAt(i)<='z')

{

cCount++;

}

}

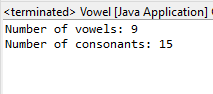
System.***out***.println("Number of vowels: " + vCount);

System.***out***.println("Number of consonants: " + cCount);

}

}

**Output:**

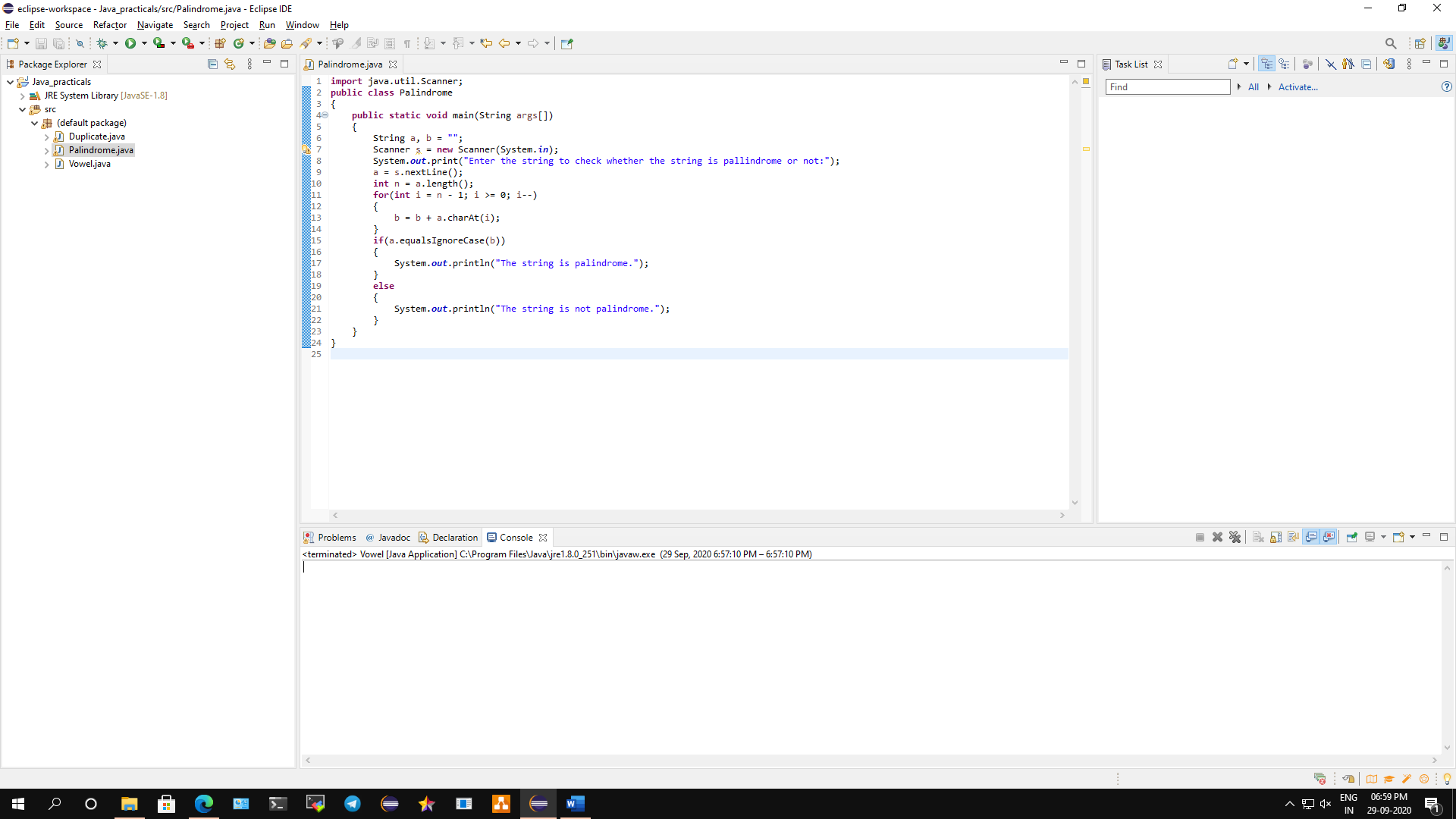


**C) Program to check whether the string is Palindrome or not.**

**Aim: Write a java program to check whether the string is Palindrome or not.**

**Description:**

A palindrome is a word, [number](https://en.wikipedia.org/wiki/Palindromic_number), phrase, or other sequence of [characters](https://en.wikipedia.org/wiki/Character_(symbol)) which reads the same backward as forward, such as madam, racecar. There are also numeric palindromes, including date/time stamps using short digits 11/11/11 11:11 and long digits 02/02/2020. Sentence-length palindromes ignore capitalization, punctuation, and word boundaries. We first imported the Scanner utils to load scanner class then, defined the class name which is class Palindrome. A class is a blueprint for the object, it represents a set of properties or methods that are common to all objects of one type. Now we have defined the main method to accept an array of string arguments, we have also defined the scanner class to check if the given number is palindrome or not if the given number is palindrome then the output will execute as number is palindrome otherwise it will show number is not palindrome. Palindrome works on the following algorithm Get the number/strings to check for palindrome. Hold the number/string in temporary variable. Reverse the number/string. Compare the temporary number/string with reversed number/string. If both number/string are same, print "palindrome". Else print "not a palindrome"



**Conclusion: We have implemented a program to implement a program to check whether the string is Palindrome or not.**

**Code:**

**import** java.util.Scanner;

**public** **class** Palindrome

{

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

{

String a, b = "";

Scanner s = **new** Scanner(System.***in***);

System.***out***.print("Enter the string to check whether the string is pallindrome or not:");

a = s.nextLine();

**int** n = a.length();

**for**(**int** i = n - 1; i >= 0; i--)

{

b = b + a.charAt(i);

}

**if**(a.equalsIgnoreCase(b))

{

System.***out***.println("The string is palindrome.");

}

**else**

{

System.***out***.println("The string is not palindrome.");

}

}

}

**Output:**



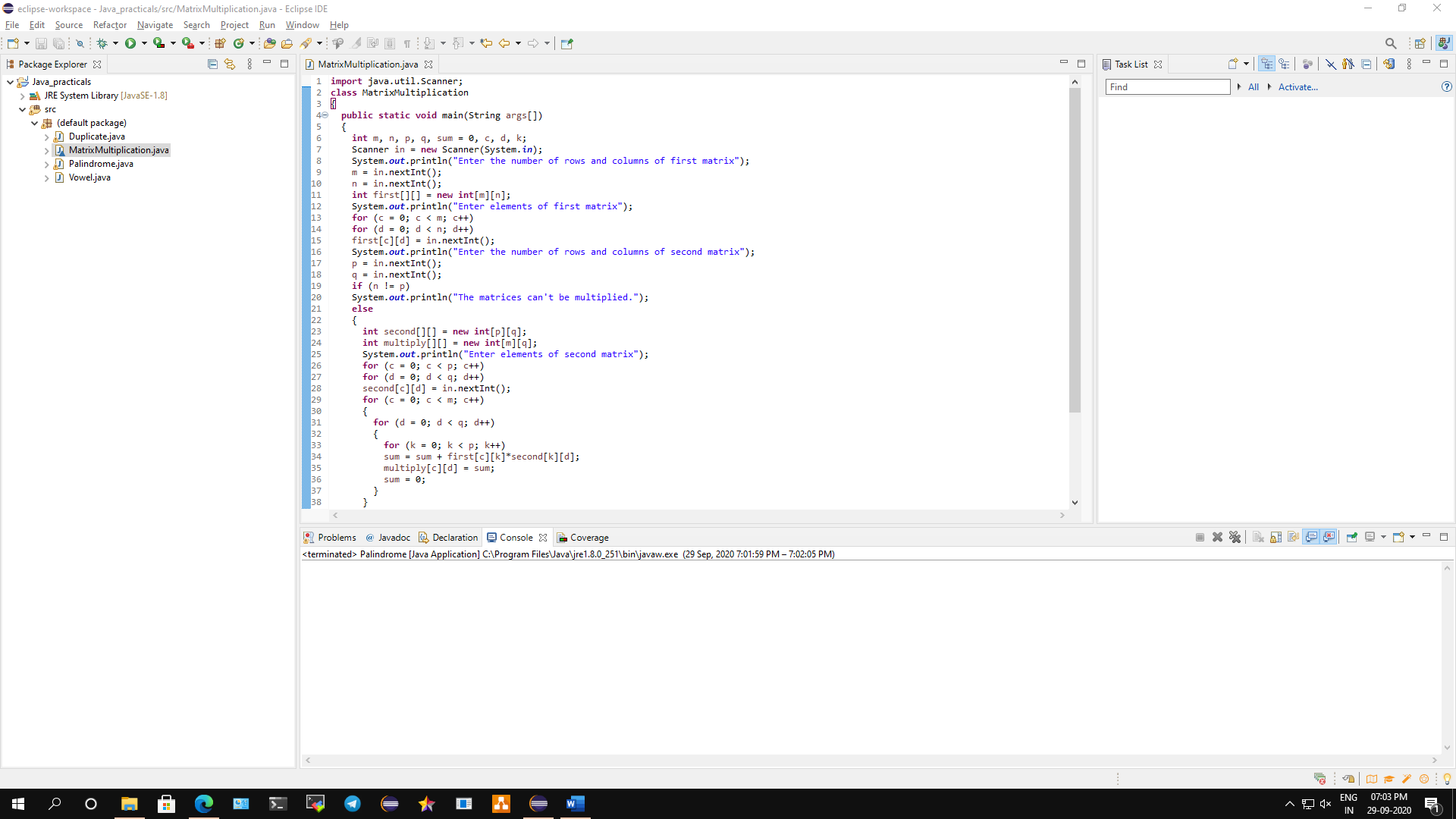


**D) Program to perform Matrix multiplication.**

**Aim: Write a java program to perform Matrix multiplication.**

**Description:**

Matrix multiplication is a binary operation that produces a matrix from two matrices. For matrix multiplication, the number of columns in the first matrix must be equal to the number of rows in the second matrix. The resulting matrix, known as the matrix product, has the number of rows of the first and the number of columns of the second matrix. The product of matrices A and B is then denoted simply as AB. Matrix multiplication was first described by the French mathematician Jacques Philippe Marie Binet in 1812, to represent the composition of linear maps that are represented by matrices. Matrix multiplication is thus a basic tool of linear algebra, and as such has numerous applications in many areas of mathematics, as well as in applied mathematics, statistics, physics, economics, and engineering. Computing matrix products is a central operation in all computational applications of linear algebra. We have imported Scanner utils class to load scanner then we have defined the public class which is MatrixMultiplication, then we use int function for input operation of matrix. For matrix multiplication to happen the column of the first matrix should be equal to the row of the second matrix. Once the output is obtained we have successfully compiled the matrix multiplication in java.



**Conclusion: We have performed a java program on Matrix multiplication.**

**Code:**

**import** java.util.Scanner;

**class** MatrixMultiplication

{

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

{

**int** m, n, p, q, sum = 0, c, d, k;

Scanner in = **new** Scanner(System.***in***);

System.***out***.println("Enter the number of rows and columns of first matrix");

m = in.nextInt();

n = in.nextInt();

**int** first[][] = **new** **int**[m][n];

System.***out***.println("Enter elements of first matrix");

**for** (c = 0; c < m; c++)

**for** (d = 0; d < n; d++)

first[c][d] = in.nextInt();

System.***out***.println("Enter the number of rows and columns of second matrix");

p = in.nextInt();

q = in.nextInt();

**if** (n != p)

System.***out***.println("The matrices can't be multiplied.");

**else**

{

**int** second[][] = **new** **int**[p][q];

**int** multiply[][] = **new** **int**[m][q];

System.***out***.println("Enter elements of second matrix");

**for** (c = 0; c < p; c++)

**for** (d = 0; d < q; d++)

second[c][d] = in.nextInt();

**for** (c = 0; c < m; c++)

{

**for** (d = 0; d < q; d++)

{

**for** (k = 0; k < p; k++)

sum = sum + first[c][k]\*second[k][d];

multiply[c][d] = sum;

sum = 0;

}

}

System.***out***.println("Matrix Multiplication are:");

**for** (c = 0; c < m; c++)

{

**for** (d = 0; d < q; d++)

System.***out***.print(multiply[c][d]+"\t");

System.***out***.print("\n");

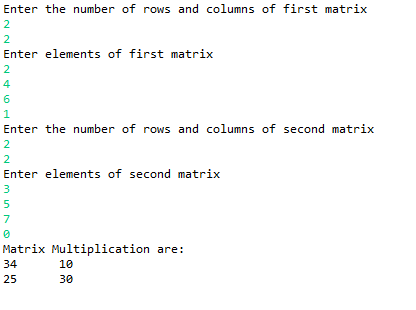
}

}

}

}

**Output:**

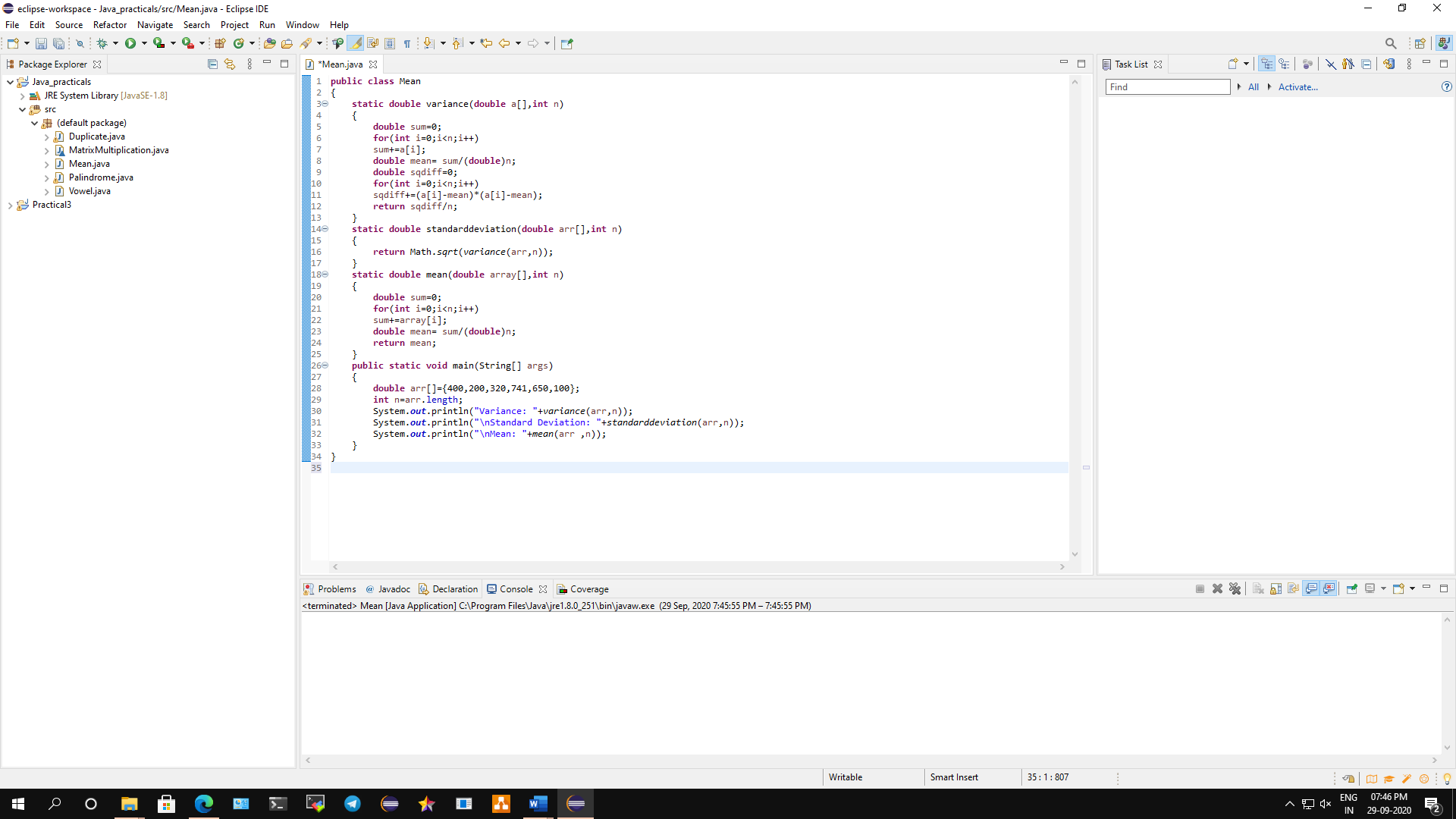


**E) Program to perform Mean, variance and deviation of array.**

**Aim: Write a java program to perform Mean, variance and deviation of array.**

**Description:**

The Arithmetic Mean is the average of the numbers: a calculated "central" value of a set of numbers. It is calculated by adding up all the numbers, then divide by how many numbers there are. The Variance is defined as the average of the squared differences from the Mean. We can calculate the variance by working out the [Mean](https://www.mathsisfun.com/mean.html), Then for each number: subtract the Mean and square the result, Then work out the average of those squared differences. The Standard Deviation is a measure of how spread out numbers are. It is defined as σ symbol. It can be calculated by using the square root of the Variance. In Java, a static method is a method that belongs to a class rather than an instance of a class. The method is accessible to every instance of a class, but methods defined in an instance are only able to be accessed by that member of a class. To calculate the mean, we have defined “+*mean*(arr ,n)” function. To calculate variance we have defined “+*variance*(arr,n)” and to find deviation we have defined “+*standarddeviation*(arr,n)” functions this will helps us to find all the values of mean, variance and deviation and the output will be generated.



**Conclusion: We have performed a java program on Mean, variance and deviation of array.**

**Code:**

**public** **class** Mean

{

**static** **double** variance(**double** a[],**int** n)

{

**double** sum=0;

**for**(**int** i=0;i<n;i++)

sum+=a[i];

**double** mean= sum/(**double**)n;

**double** sqdiff=0;

**for**(**int** i=0;i<n;i++)

sqdiff+=(a[i]-mean)\*(a[i]-mean);

**return** sqdiff/n;

}

**static** **double** standarddeviation(**double** arr[],**int** n)

{

**return** Math.*sqrt*(*variance*(arr,n));

}

**static** **double** mean(**double** array[],**int** n)

{

**double** sum=0;

**for**(**int** i=0;i<n;i++)

sum+=array[i];

**double** mean= sum/(**double**)n;

**return** mean;

}

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

{

**double** arr[]={400,200,320,741,650,100};

**int** n=arr.length;

System.***out***.println("Variance: "+*variance*(arr,n));

System.***out***.println("\nStandard Deviation: "+*standarddeviation*(arr,n));

System.***out***.println("\nMean: "+*mean*(arr ,n));

}

}

**Output:**

