Q-1)  Write the same programme to sort the integers 8, 4, 3,5,6 and the alphabetical string C, O, I, P, U, in ascending order. Show the resulting output.

**Program**:

**package** Home;

**import** java.util.Arrays;

**public** **class** SortLists {

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

// Initialize an array of integers

**int**[] integerArray = {8, 4, 3, 5, 6};

// Initialize an array of strings

String[] stringArray = {"C", "O", "I", "P", "U"};

// Sort the array of integers

Arrays.*sort*(integerArray);

// The Arrays.sort() method sorts the specified array into ascending numerical order

// Sort the array of strings

Arrays.*sort*(stringArray);

// The Arrays.sort() method sorts the specified array of strings into ascending lexicographical order

// Print the sorted arrays

System.***out***.println("Sorted list of integers: " + Arrays.*toString*(integerArray));

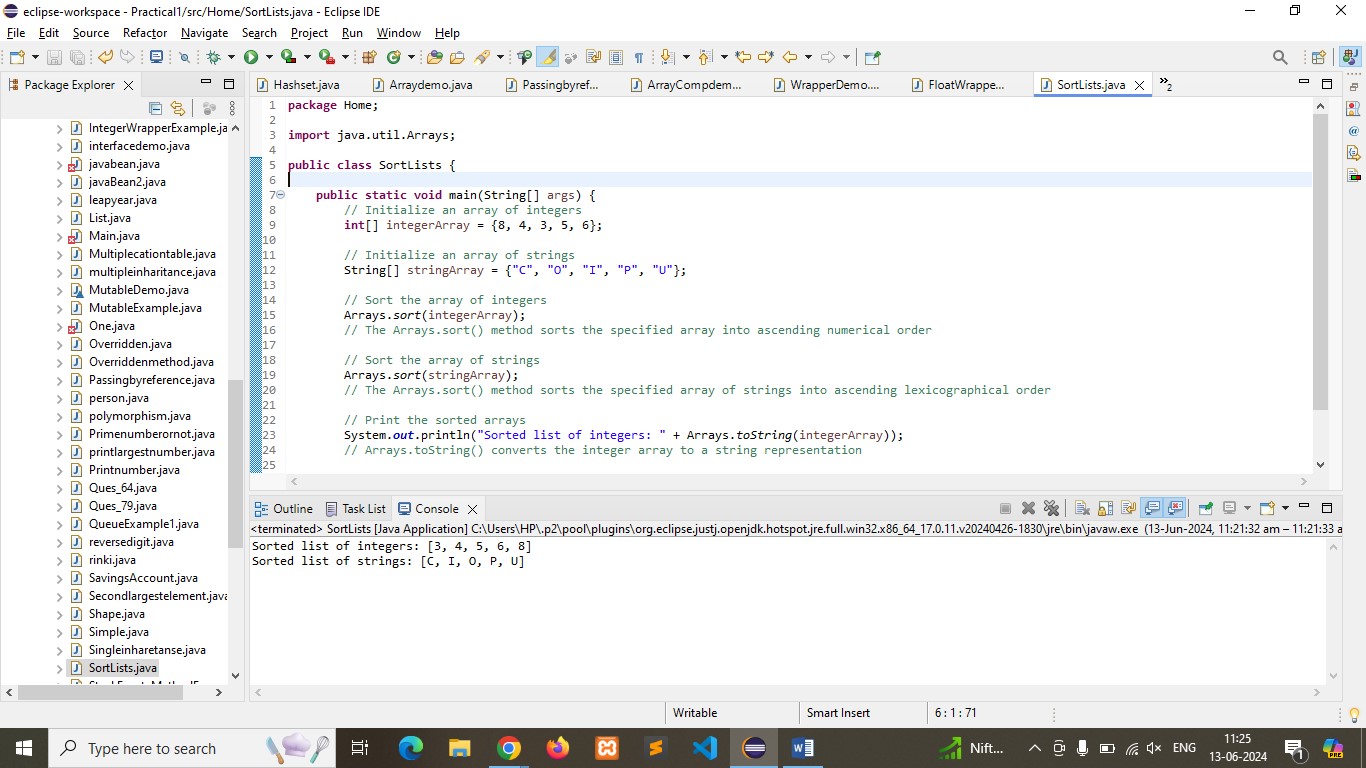
// Arrays.toString() converts the integer array to a string representation

System.***out***.println("Sorted list of strings: " + Arrays.*toString*(stringArray));

// Arrays.toString() converts the string array to a string representation

}

}

**Output**: 

Q-2)  Write a Java program to implement the bubble sort algorithm to sort an array of integers in ascending order.

**Program:**

**package** Home;

**public** **class** BubbleSortExample {

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

// Initialize an array of integers to be sorted

**int**[] array = {8, 4, 3, 5, 6};

// Print the original array

System.***out***.println("Original array:");

*printArray*(array);

// Call the bubble sort method to sort the array

*bubbleSort*(array);

// Print the sorted array

System.***out***.println("Sorted array:");

*printArray*(array);

}

// Method to implement bubble sort

**public** **static** **void** bubbleSort(**int**[] array) {

**int** n = array.length;

**boolean** swapped;

// Outer loop to traverse through all elements

**for** (**int** i = 0; i < n - 1; i++) {

swapped = **false**; // Initialize swapped as false

// Inner loop for comparing adjacent elements

**for** (**int** j = 0; j < n - 1 - i; j++) {

// Swap if the current element is greater than the next element

**if** (array[j] > array[j + 1]) {

**int** temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

swapped = **true**; // Set swapped to true

}

}

// If no elements were swapped, the array is already sorted

**if** (!swapped) **break**;

}

}

// Method to print an array

**public** **static** **void** printArray(**int**[] array) {

**for** (**int** element : array) {

System.***out***.print(element + " ");

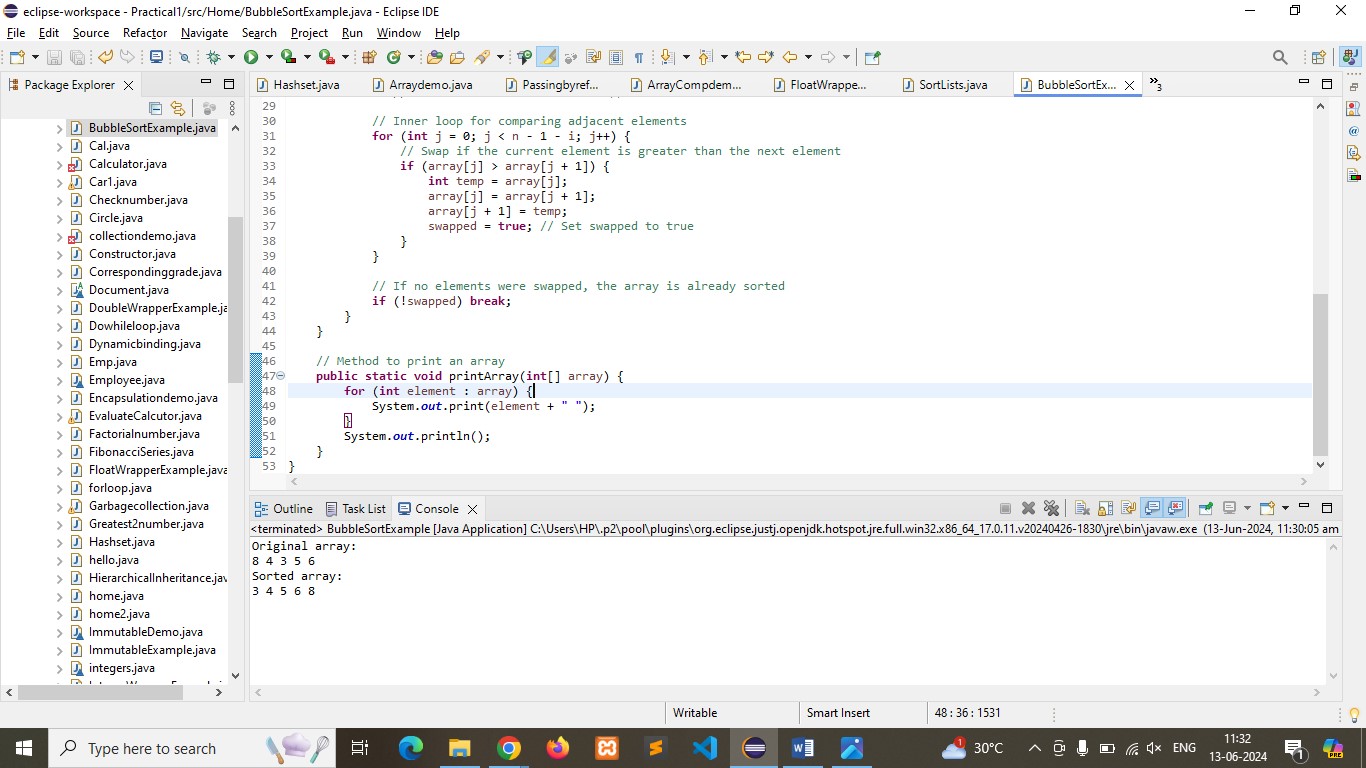
}

System.***out***.println();

}

}

**Output**:



Q-3) Write a program to input an array 10 elements and print the cube of prime numbers in it.

**Program:**

**package** Home;

**import** java.util.Scanner;

**public** **class** CubeOfPrimes {

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

// Create a scanner object for input

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

// Initialize an array to hold 10 elements

**int**[] array = **new** **int**[10];

// Input 10 elements from the user

System.***out***.println("Enter 10 integers:");

**for** (**int** i = 0; i < 10; i++) {

array[i] = scanner.nextInt();

}

// Print the cubes of prime numbers in the array

System.***out***.println("Cubes of prime numbers in the array:");

**for** (**int** i = 0; i < 10; i++) {

**if** (*isPrime*(array[i])) {

**int** cube = array[i] \* array[i] \* array[i];

System.***out***.println("Cube of " + array[i] + " is " + cube);

}

}

// Close the scanner

scanner.close();

}

// Method to check if a number is prime

**public** **static** **boolean** isPrime(**int** num) {

// Numbers less than 2 are not prime

**if** (num < 2) {

**return** **false**;

}

// Check for factors from 2 to the square root of the number

**for** (**int** i = 2; i <= Math.*sqrt*(num); i++) {

**if** (num % i == 0) {

**return** **false**; // Not a prime number

}

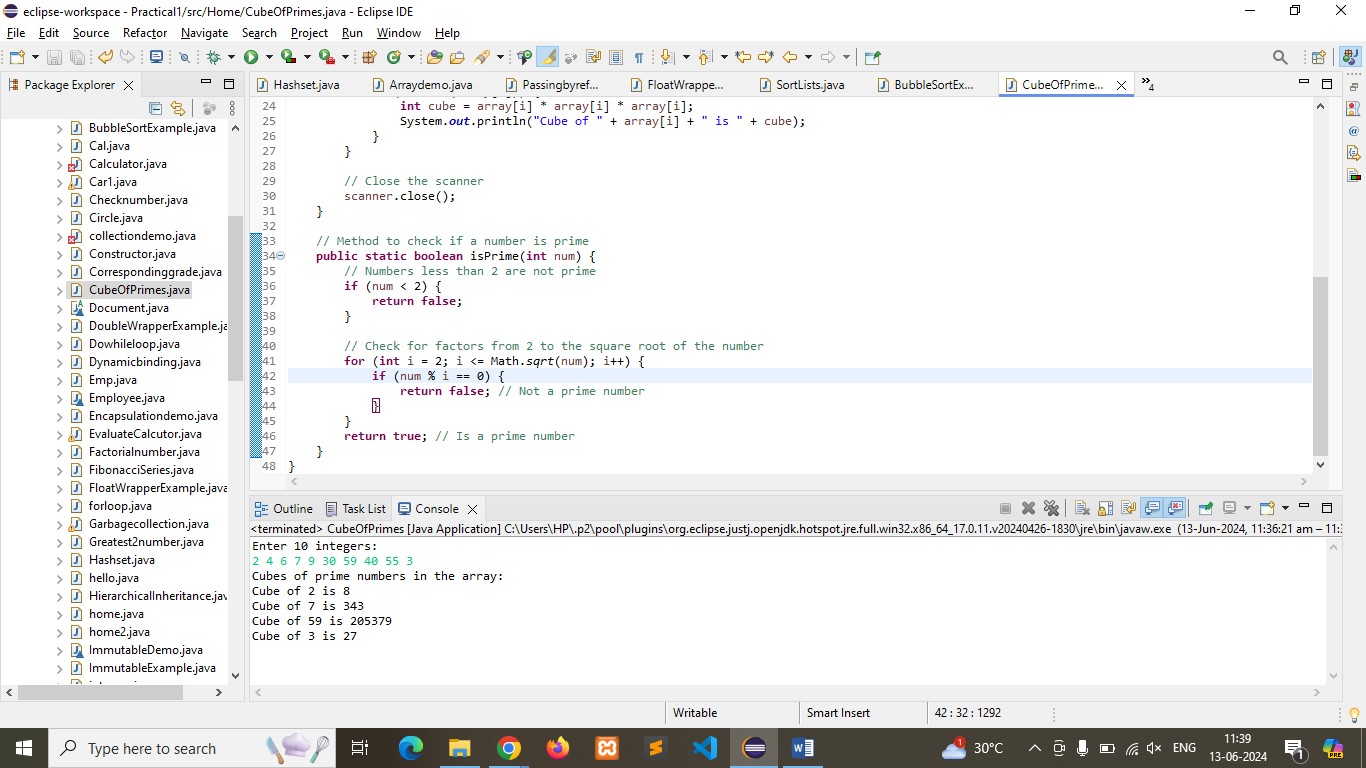
}

**return** **true**; // Is a prime number

}

}

**Output:**



Q-4)   Write a java program to implement integer wrapper class methods.

(Any 5 methods)

**Program:**

**package** Home;

**public** **class** IntegerWrapper {

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

// 1. Integer.parseInt(String s) - Converts a string to an int

String numberStr = "123";

**int** number = Integer.*parseInt*(numberStr);

System.***out***.println("1. Integer.parseInt(\"123\") = " + number);

// 2. Integer.valueOf(String s) - Returns an Integer object holding the value of the specified String

Integer integerValue = Integer.*valueOf*(numberStr);

System.***out***.println("2. Integer.valueOf(\"123\") = " + integerValue);

// 3. Integer.toString(int i) - Returns a String object representing the specified int

String stringRepresentation = Integer.*toString*(number);

System.***out***.println("3. Integer.toString(123) = " + stringRepresentation);

// 4. Integer.compare(int x, int y) - Compares two int values numerically

**int** comparisonResult = Integer.*compare*(10, 20);

System.***out***.println("4. Integer.compare(10, 20) = " + comparisonResult);

// 5. Integer.MAX\_VALUE - A constant holding the maximum value an int can have

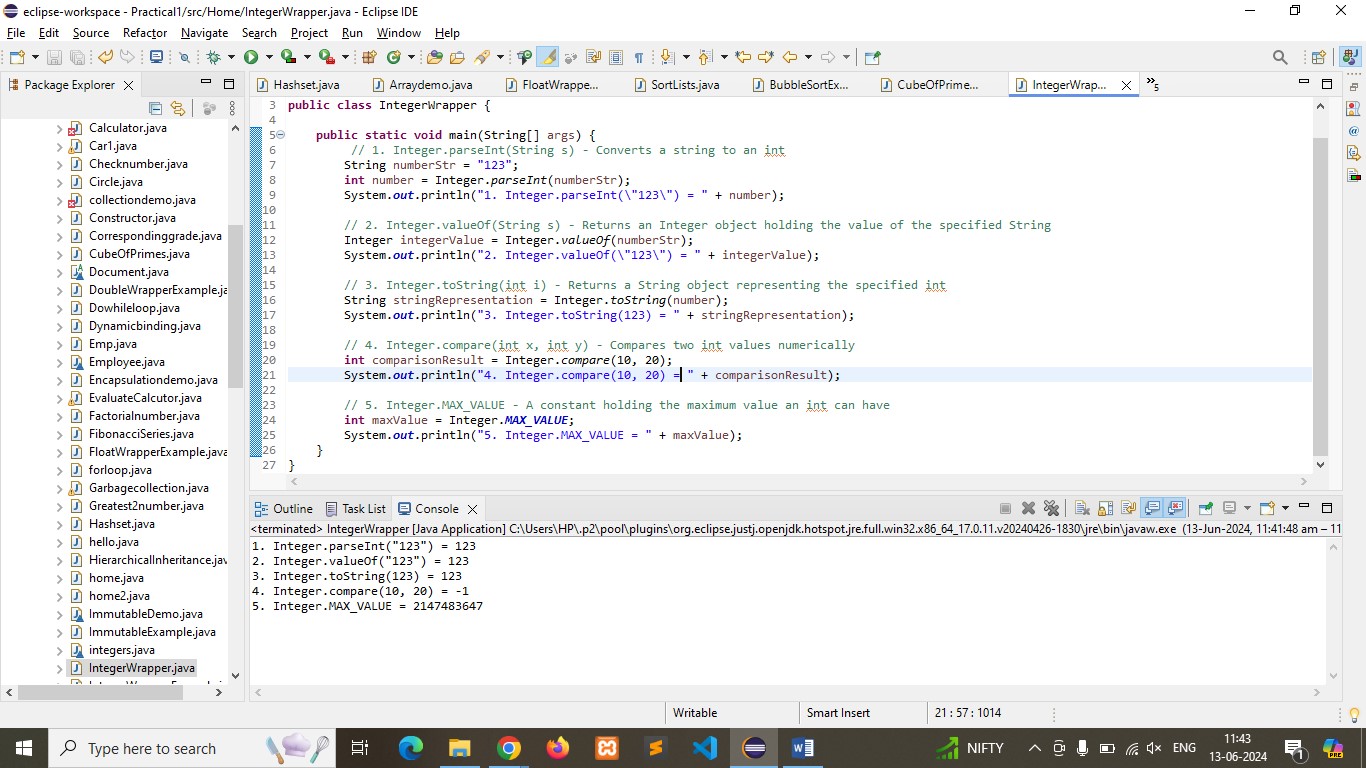
**int** maxValue = Integer.***MAX\_VALUE***;

System.***out***.println("5. Integer.MAX\_VALUE = " + maxValue);

}

}

**Output:**



Q-5)  Write a java program to implement double wrapper class methods. (Any 5 methods)

**Program:**

**package** Home;

**public** **class** DoubleWrapper {

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

// 1. Double.parseDouble(String s) - Converts a string to a double

String doubleStr = "123.45";

**double** number = Double.*parseDouble*(doubleStr);

System.***out***.println("1. Double.parseDouble(\"123.45\") = " + number);

// 2. Double.valueOf(String s) - Returns a Double object holding the value of the specified String

Double doubleValue = Double.*valueOf*(doubleStr);

System.***out***.println("2. Double.valueOf(\"123.45\") = " + doubleValue);

// 3. Double.toString(double d) - Returns a String object representing the specified double

String stringRepresentation = Double.*toString*(number);

System.***out***.println("3. Double.toString(123.45) = " + stringRepresentation);

// 4. Double.compare(double d1, double d2) - Compares two double values numerically

**int** comparisonResult = Double.*compare*(10.5, 20.5);

System.***out***.println("4. Double.compare(10.5, 20.5) = " + comparisonResult);

// 5. Double.MAX\_VALUE - A constant holding the maximum value a double can have

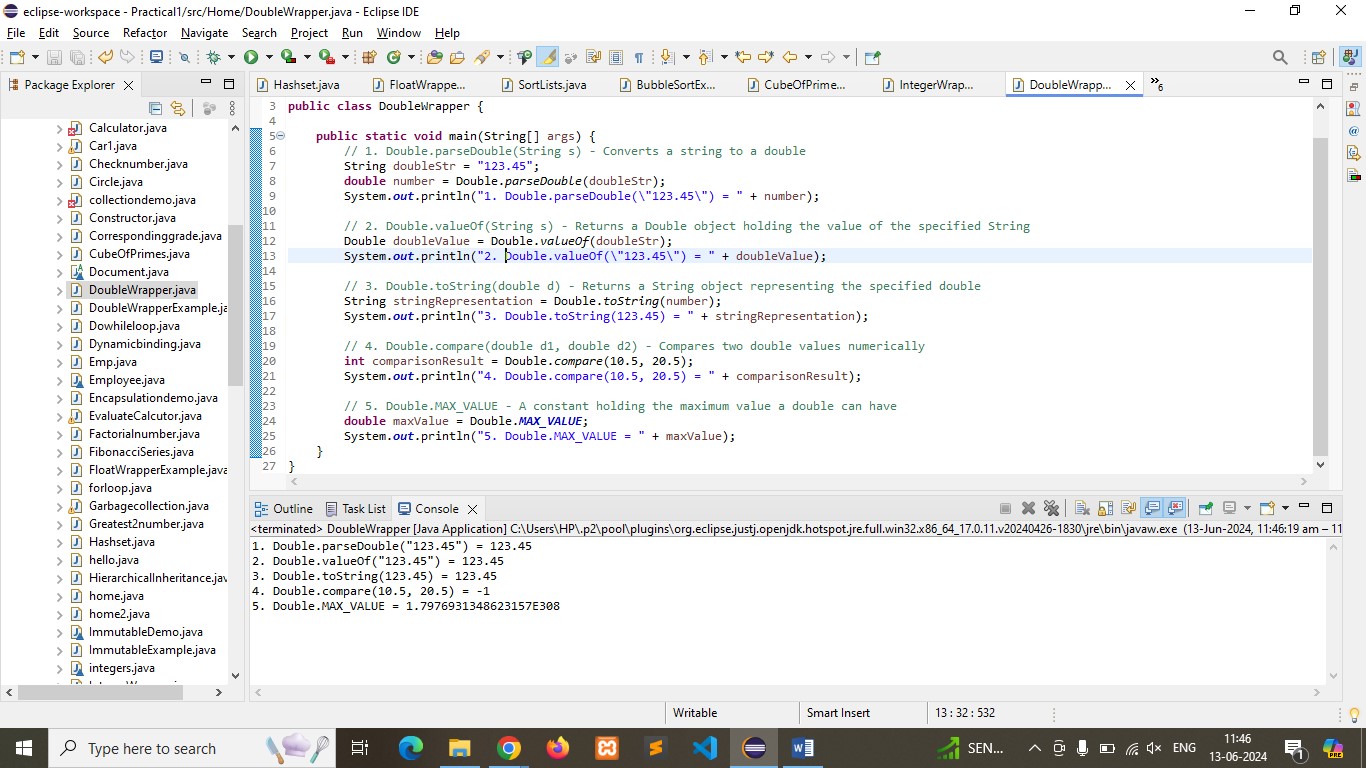
**double** maxValue = Double.***MAX\_VALUE***;

System.***out***.println("5. Double.MAX\_VALUE = " + maxValue);

}

}

**Output:**



Q-6)   Write a java program to implement float wrapper class methods. (Any 5 methods)

**Program:**

**package** Home;

**public** **class** FloatWrapper {

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

// 1. Float.parseFloat(String s) - Converts a string to a float

String floatStr = "123.45";

**float** number = Float.*parseFloat*(floatStr);

System.***out***.println("1. Float.parseFloat(\"123.45\") = " + number);

// 2. Float.valueOf(String s) - Returns a Float object holding the value of the specified String

Float floatValue = Float.*valueOf*(floatStr);

System.***out***.println("2. Float.valueOf(\"123.45\") = " + floatValue);

// 3. Float.toString(float f) - Returns a String object representing the specified float

String stringRepresentation = Float.*toString*(number);

System.***out***.println("3. Float.toString(123.45f) = " + stringRepresentation);

// 4. Float.compare(float f1, float f2) - Compares two float values numerically

**int** comparisonResult = Float.*compare*(10.5f, 20.5f);

System.***out***.println("4. Float.compare(10.5f, 20.5f) = " + comparisonResult);

// 5. Float.MAX\_VALUE - A constant holding the maximum value a float can have

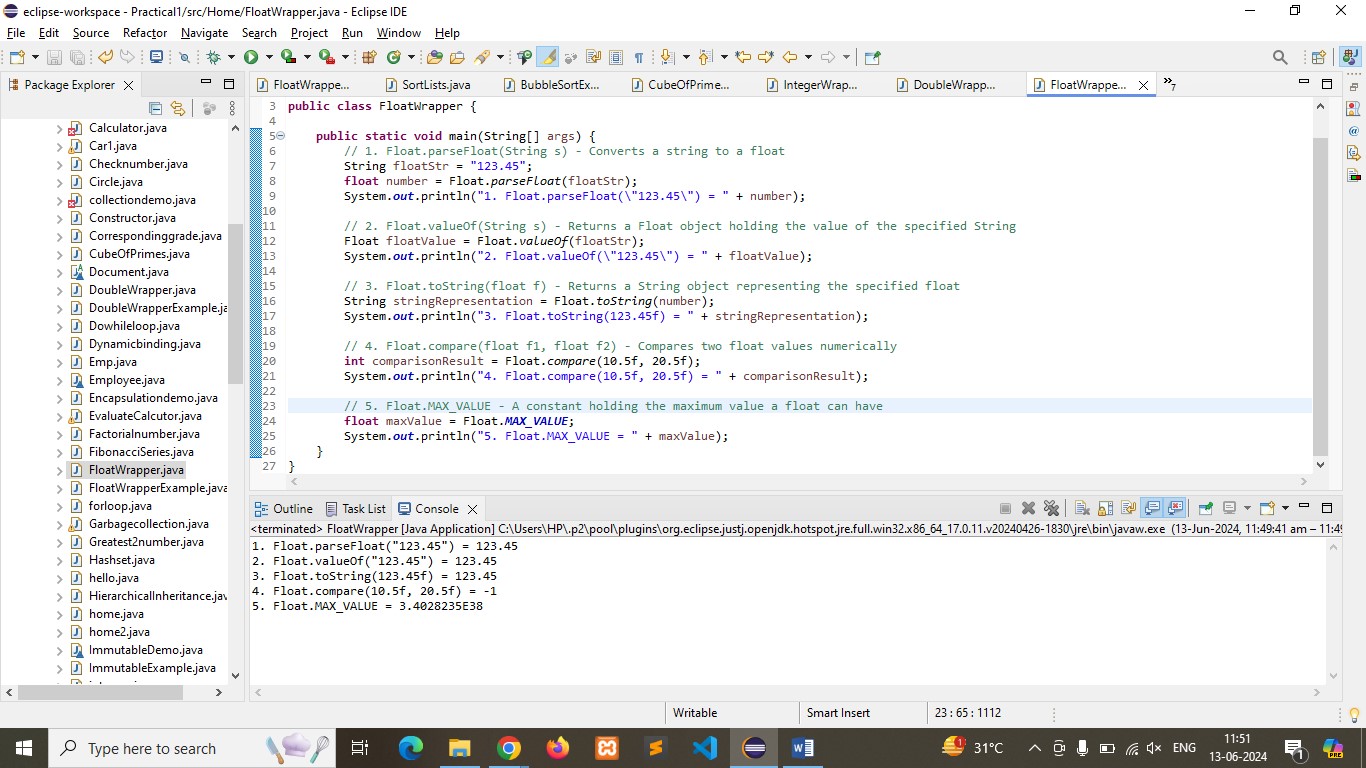
**float** maxValue = Float.***MAX\_VALUE***;

System.***out***.println("5. Float.MAX\_VALUE = " + maxValue);

}

}

**Output:**



Q-7)    Write a Java program to validate email addresses using regular expressions. The email should have the format username@domain.com where username and domain can contain alphanumeric characters, dots, and hyphens.

**Program:**

**package** Home;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**import** java.util.Scanner;

**public** **class** EmailValidator {

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

// Create a scanner object for input

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

// Regular expression pattern for validating email addresses

String emailRegex = "^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,6}$";

Pattern pattern = Pattern.*compile*(emailRegex);

// Input email address from the user

System.***out***.println("Enter an email address to validate:");

String email = scanner.nextLine();

// Match the input email against the pattern

Matcher matcher = pattern.matcher(email);

// Validate the email and print the result

**if** (matcher.matches()) {

System.***out***.println("The email address " + email + " is valid.");

} **else** {

System.***out***.println("The email address " + email + " is invalid.");

}

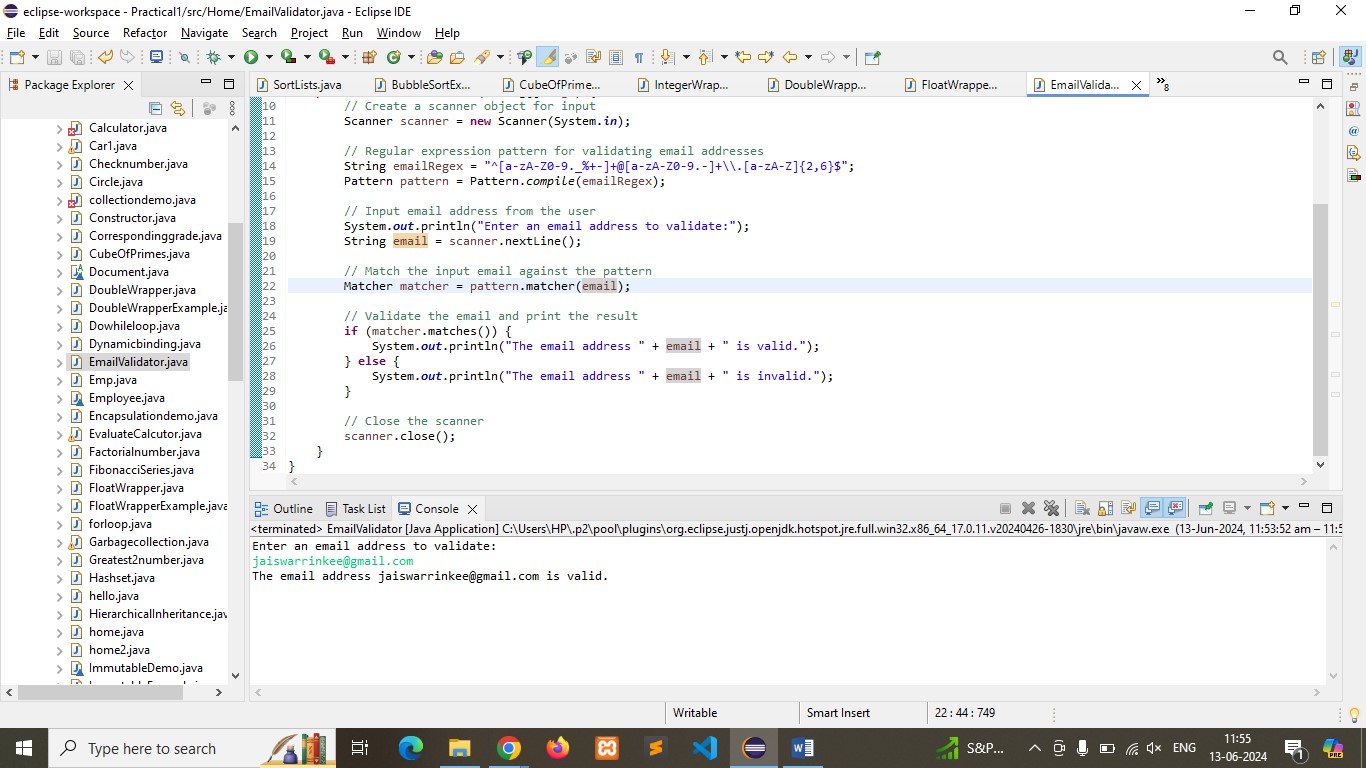
// Close the scanner

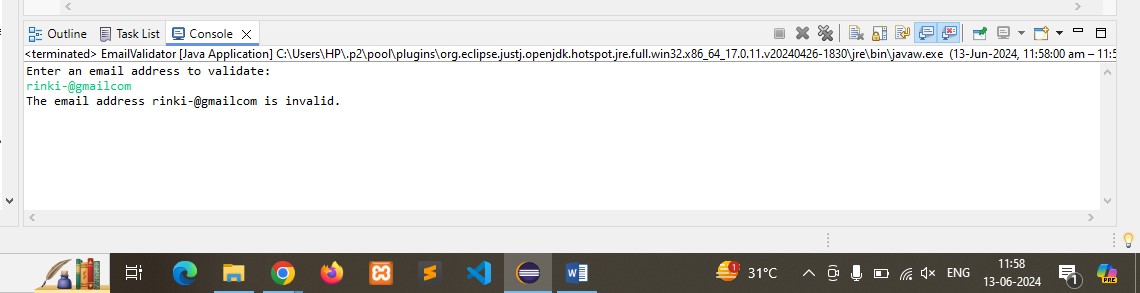
scanner.close();

}

}

**Output:**





Q-8) Create a Java program to validate phone numbers. The format should be (xxx) xxx-xxxx where x is a digit.

Program:

**package** Home;

**import** java.util.Scanner;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** PhoneNumberValidator {

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

// Create a scanner object for input

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

// Regular expression pattern for validating phone numbers

String phoneRegex = "^\\(\\d{3}\\) \\d{3}-\\d{4}$";

Pattern pattern = Pattern.*compile*(phoneRegex);

// Input phone number from the user

System.***out***.println("Enter a phone number to validate (format: (xxx) xxx-xxxx):");

String phoneNumber = scanner.nextLine();

// Match the input phone number against the pattern

Matcher matcher = pattern.matcher(phoneNumber);

// Validate the phone number and print the result

**if** (matcher.matches()) {

System.***out***.println("The phone number " + phoneNumber + " is valid.");

} **else** {

System.***out***.println("The phone number " + phoneNumber + " is invalid.");

}

// Close the scanner

scanner.close();

}

}

Output:

