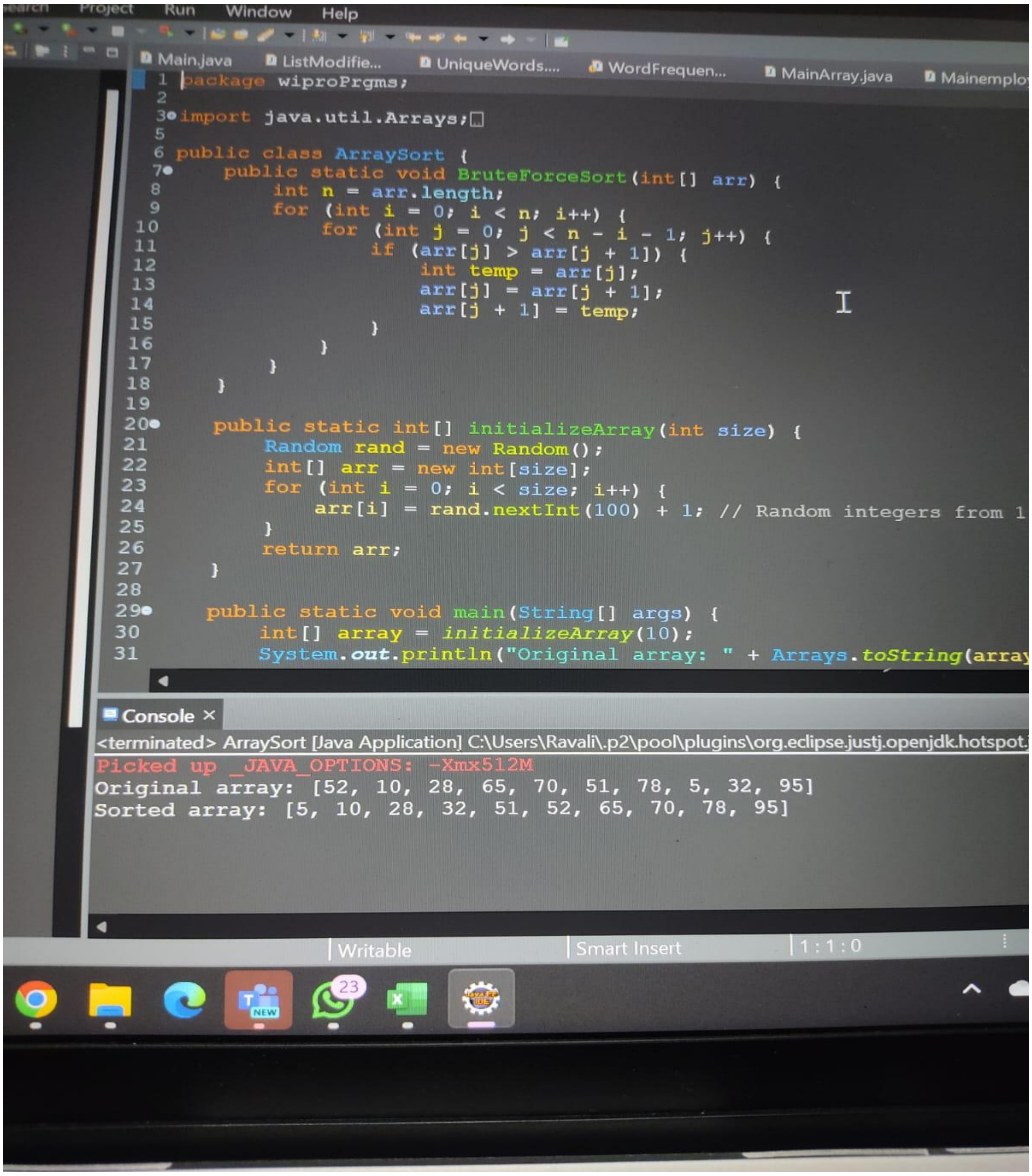
NAME :- Jangili Ravali
EMAIL :- jangiliravali9@gmail.com

Day 4 :

Task 1: Array Sorting and Searching

a) Implement a function called BruteForceSort that sorts an array using the brute force approach. Use this function to sort an array created with InitializeArray.

the equivalent implementation in Java:



```java
package wiproPrgms;

import java.util.Arrays;

public class ArraySort {
    public static void BruteForceSort(int[] arr) {
        int n = arr.length;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }

    public static int[] initializeArray(int size) {
        Random rand = new Random();
        int[] arr = new int[size];
        for (int i = 0; i < size; i++) {
            arr[i] = rand.nextInt(100) + 1; // Random integers from 1
        }
        return arr;
    }

    public static void main(String[] args) {
        int[] array = initializeArray(10);
        System.out.println("Original array: " + Arrays.toString(array
```

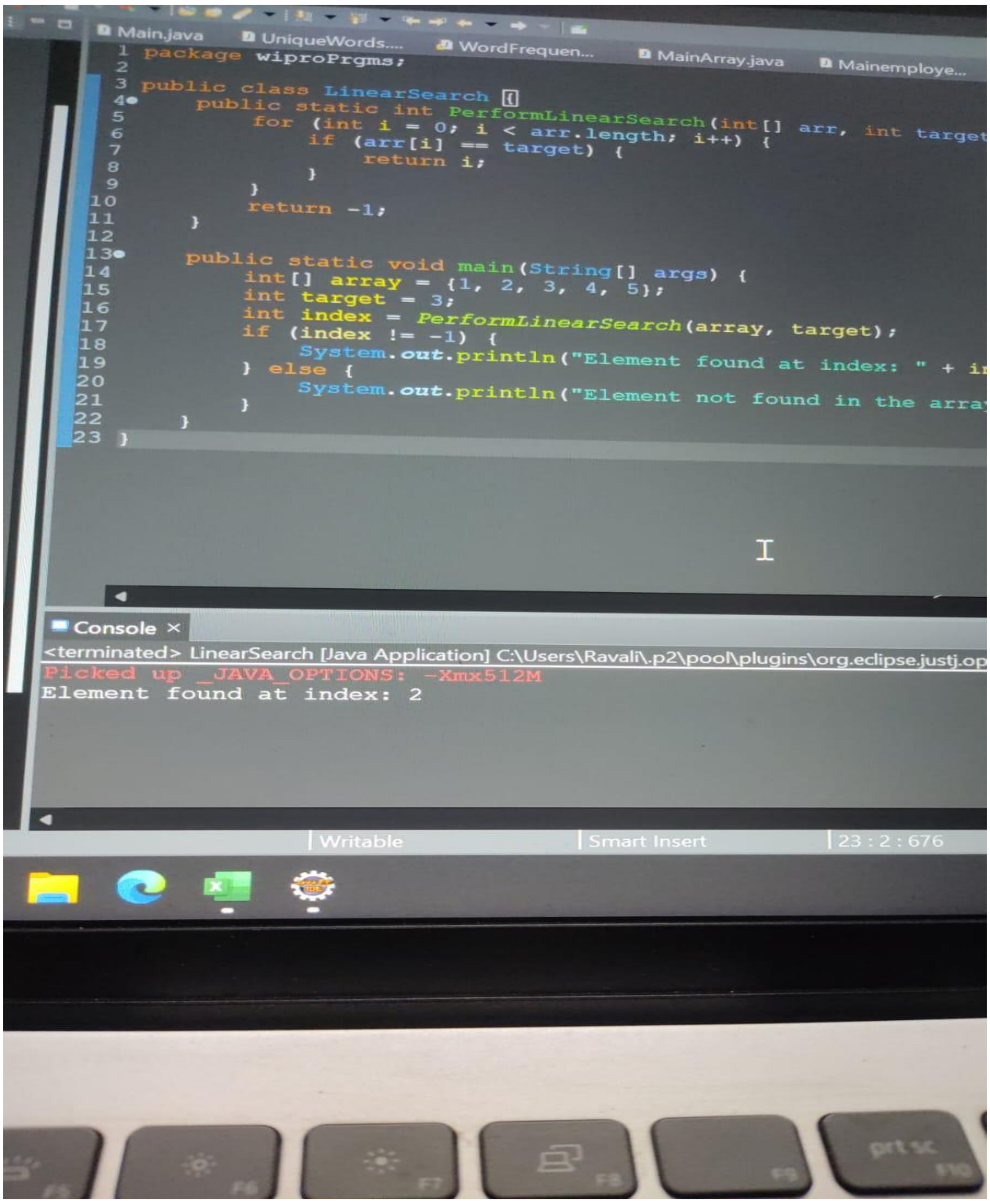
```
<terminated> ArraySort [Java Application] C:\Users\Ravali\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.j
Picked up _JAVA_OPTIONS: -Xmx512M
Original array: [52, 10, 28, 65, 70, 51, 78, 5, 32, 95]
Sorted array: [5, 10, 28, 32, 51, 52, 65, 70, 78, 95]
```

NAME :- Jangili Ravali
EMAIL :- jangiliravali9@gmail.com

---

b) Write a function named PerformLinearSearch that searches for a specific element in an array and returns the index of the element if found or -1 if not found.

The Java implementation of the PerformLinearSearch function:



```java
package wiproPrgms;

public class LinearSearch {
    public static int PerformLinearSearch(int[] arr, int target
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == target) {
                return i;
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        int[] array = {1, 2, 3, 4, 5};
        int target = 3;
        int index = PerformLinearSearch(array, target);
        if (index != -1) {
            System.out.println("Element found at index: " + i
        } else {
            System.out.println("Element not found in the arra
        }
    }
}
```

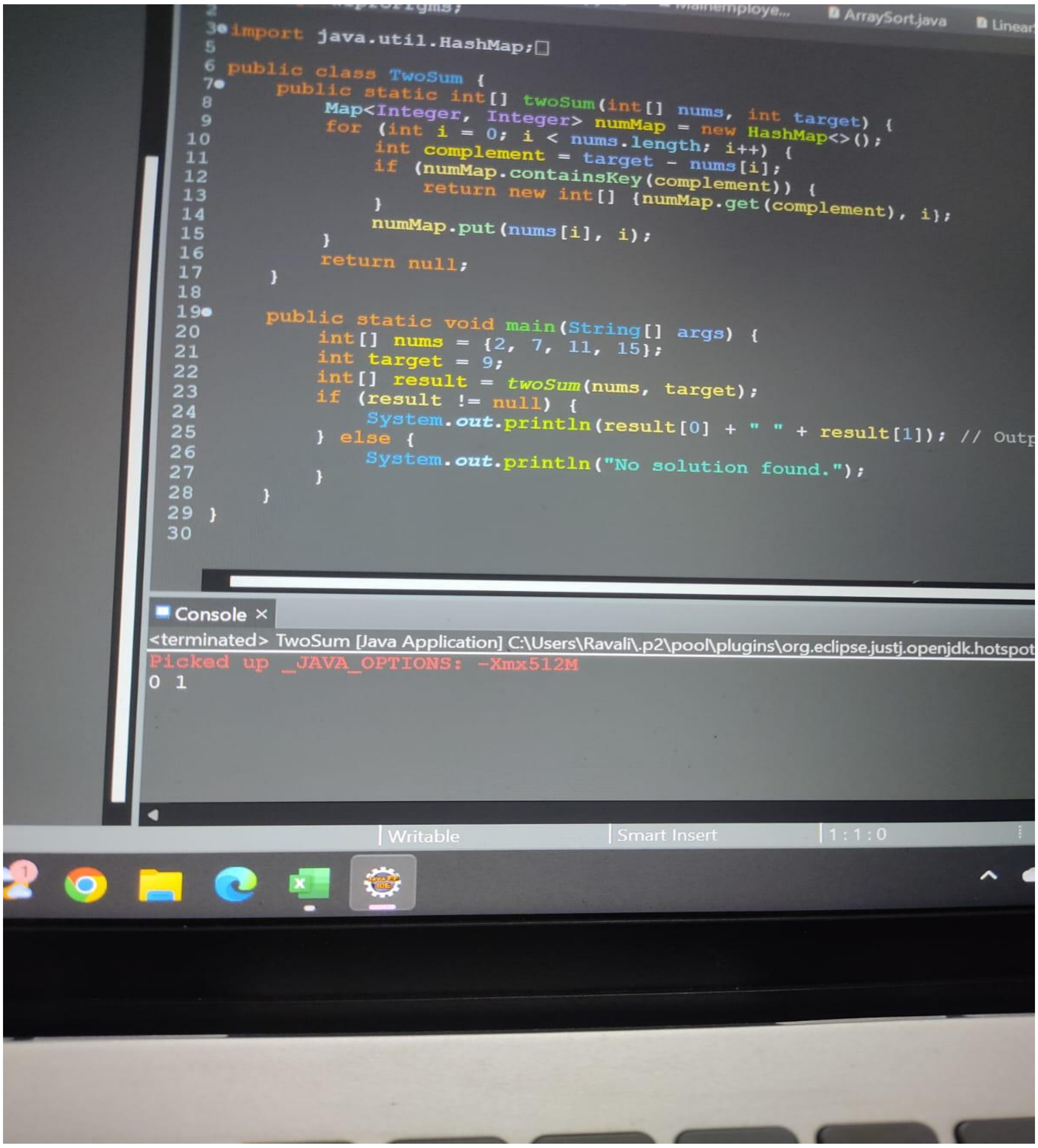
```
<terminated> LinearSearch [Java Application] C:\Users\Ravali\.p2\pool\plugins\org.eclipse.justj.op
Picked up _JAVA_OPTIONS: -Xmx512M
Element found at index: 2
```

You can call the PerformLinearSearch function passing the integer array and the target element, and it will return the index of the element if found, or -1 if not found.

NAME :- Jangili Ravali
EMAIL :- jangiliravali9@gmail.com

Task 2: Two-Sum Problem

a) Given an array of integers, write a program that finds if there are two numbers that add up to a specific target. You may assume that each input would have exactly one solution, and you may not use the same element twice. Optimize the solution for time complexity.



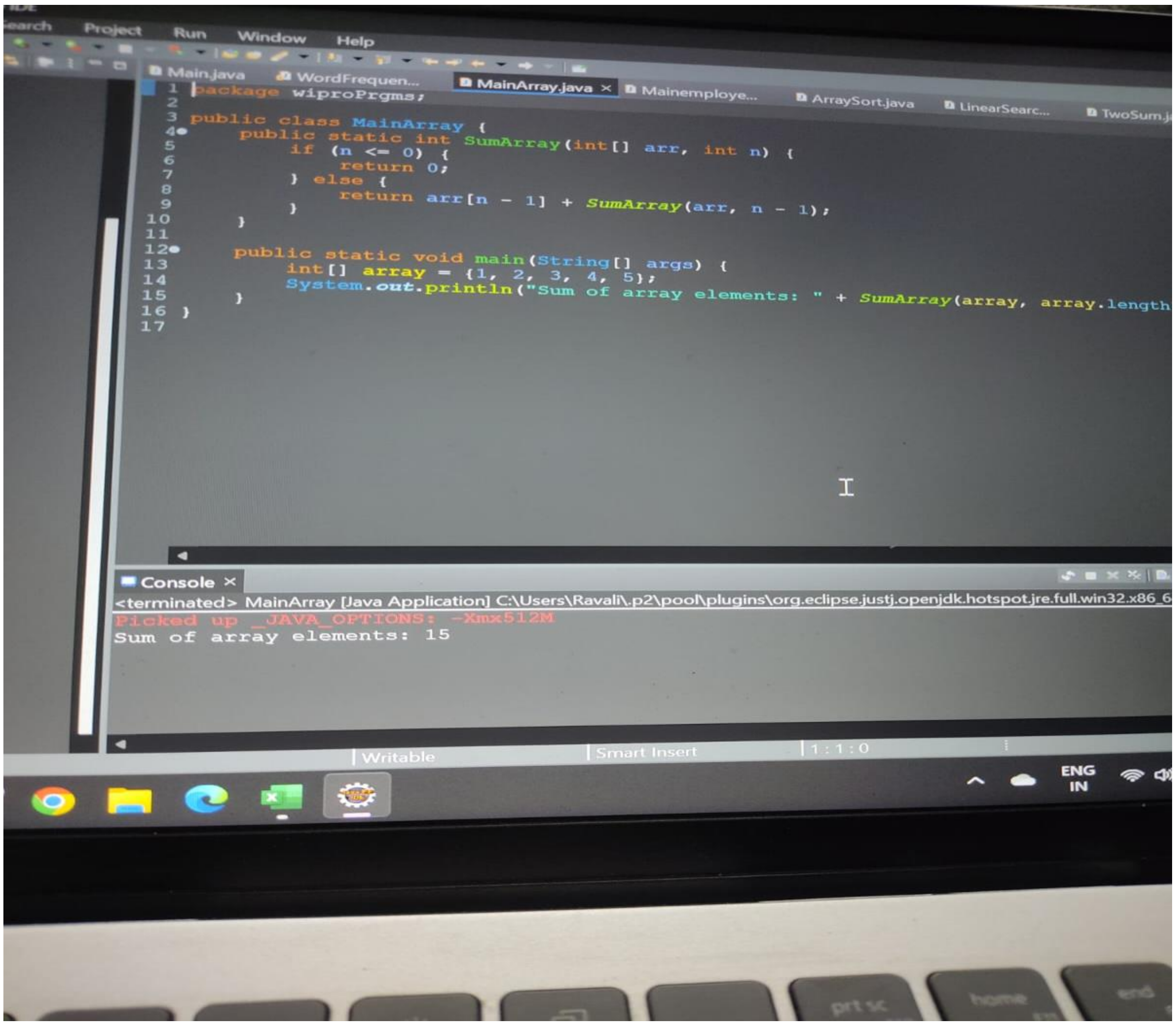
```java
import java.util.HashMap;

public class TwoSum {
    public static int[] twoSum(int[] nums, int target) {
        Map<Integer, Integer> numMap = new HashMap<>();
        for (int i = 0; i < nums.length; i++) {
            int complement = target - nums[i];
            if (numMap.containsKey(complement)) {
                return new int[] {numMap.get(complement), i};
            }
            numMap.put(nums[i], i);
        }
        return null;
    }

    public static void main(String[] args) {
        int[] nums = {2, 7, 11, 15};
        int target = 9;
        int[] result = twoSum(nums, target);
        if (result != null) {
            System.out.println(result[0] + " " + result[1]); // Outp
        } else {
            System.out.println("No solution found.");
        }
    }
}
```

```
<terminated> TwoSum [Java Application] C:\Users\Ravali\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot
Picked up _JAVA_OPTIONS: -Xmx512M
0 1
```

NAME :- Jangili Ravali
EMAIL :- jangiliravali9@gmail.com

Task 3: Understanding Functions through Arrays

a) Write a recursive function named SumArray that calculates and returns the sum of elements in an array, demonstarte with example.



```java
package wiproPrgms;

public class MainArray {
    public static int SumArray(int[] arr, int n) {
        if (n <= 0) {
            return 0;
        } else {
            return arr[n - 1] + SumArray(arr, n - 1);
        }
    }

    public static void main(String[] args) {
        int[] array = {1, 2, 3, 4, 5};
        System.out.println("Sum of array elements: " + SumArray(array, array.length
    }
}
```

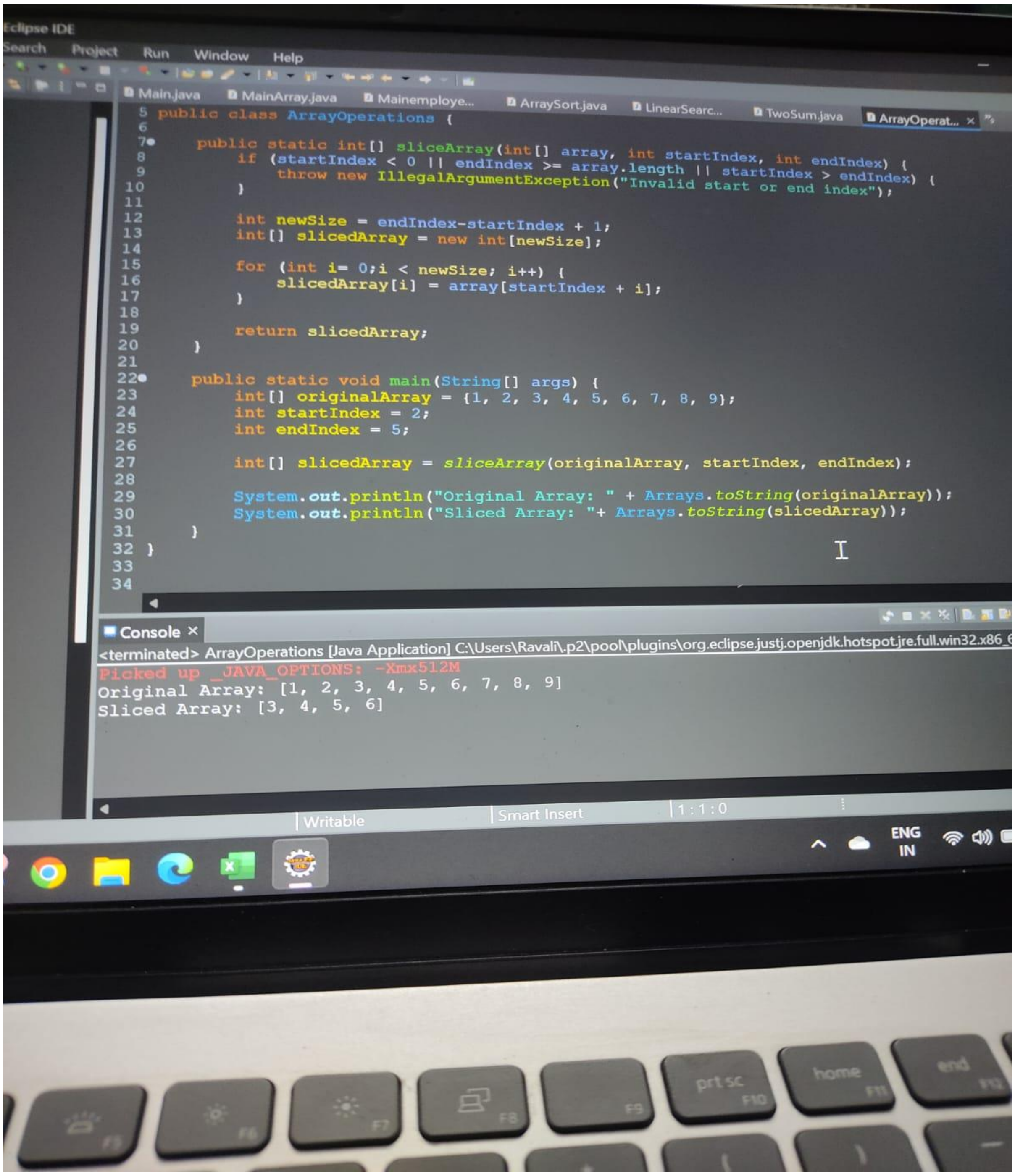
```
<terminated> MainArray [Java Application] C:\Users\Ravali\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_6
Picked up _JAVA_OPTIONS: -Xmx512M
Sum of array elements: 15
```

Java code defines a class with a recursive method SumArray that calculates the sum of elements in an array. The method takes the array and its length as parameters. In the main method, an example array is created and passed to the SumArray method.

Task 4: Advanced Array Operations

a) Implement a method SliceArray that takes an array, a starting index, and an end index, then returns a new array containing the elements from the start to the end index.

NAME :- Jangili Ravali
EMAIL :- jangiliravali9@gmail.com



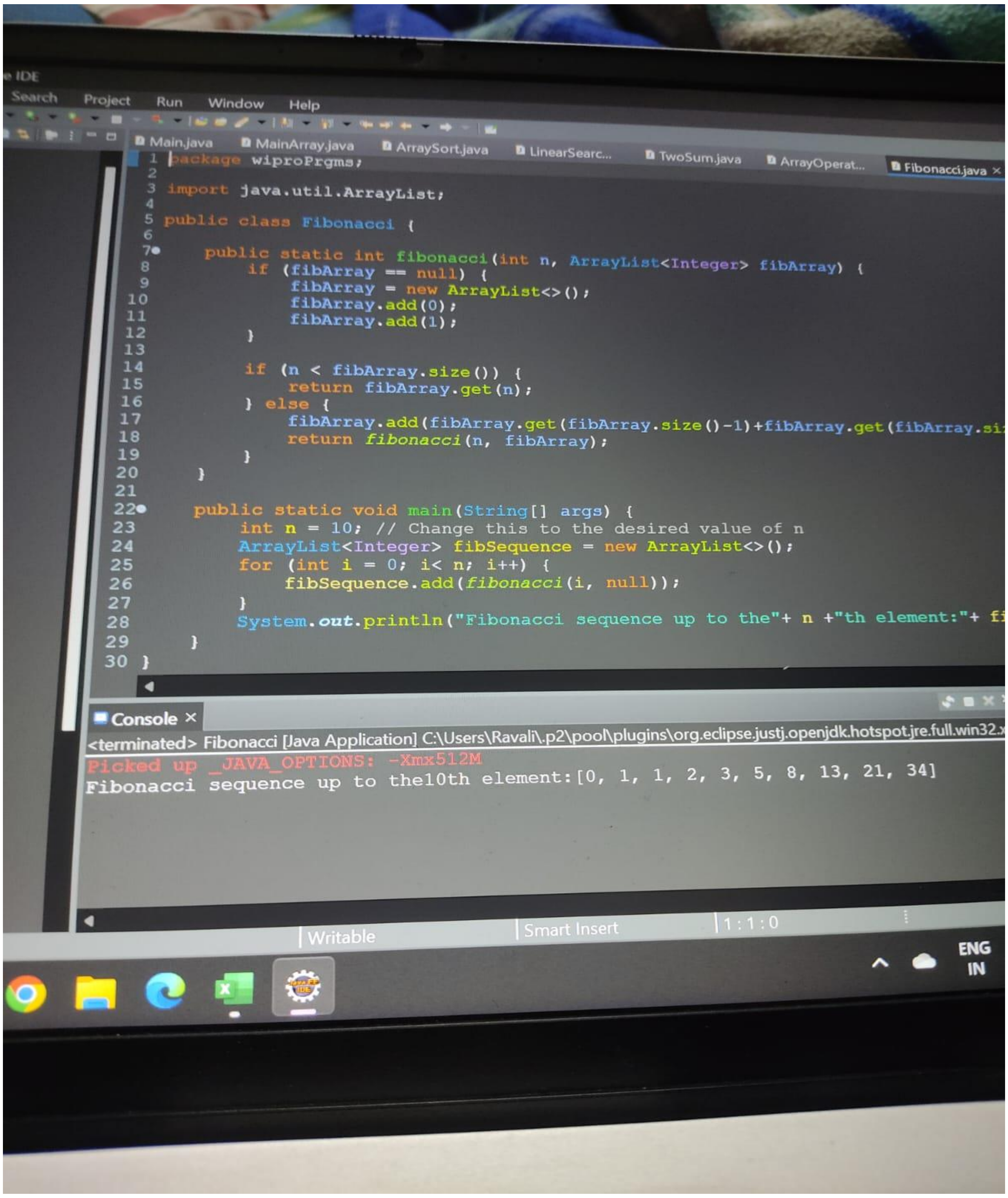
```java
public class ArrayOperations {

    public static int[] sliceArray(int[] array, int startIndex, int endIndex) {
        if (startIndex < 0 || endIndex >= array.length || startIndex > endIndex) {
            throw new IllegalArgumentException("Invalid start or end index");
        }

        int newSize = endIndex-startIndex + 1;
        int[] slicedArray = new int[newSize];

        for (int i= 0;i < newSize; i++) {
            slicedArray[i] = array[startIndex + i];
        }

        return slicedArray;
    }

    public static void main(String[] args) {
        int[] originalArray = {1, 2, 3, 4, 5, 6, 7, 8, 9};
        int startIndex = 2;
        int endIndex = 5;

        int[] slicedArray = sliceArray(originalArray, startIndex, endIndex);

        System.out.println("Original Array: " + Arrays.toString(originalArray));
        System.out.println("Sliced Array: "+ Arrays.toString(slicedArray));
    }
}
```

```
<terminated> ArrayOperations [Java Application] C:\Users\Ravali\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_6
Picked up _JAVA_OPTIONS: -Xmx512M
Original Array: [1, 2, 3, 4, 5, 6, 7, 8, 9]
Sliced Array: [3, 4, 5, 6]
```

This code defines a method sliceArray that takes an array, a starting index, and an end index, then returns a new array containing the elements from the start to the end index. The method first checks if the input indices are valid, then creates a new array of appropriate size and copies the elements from the original array within the specified range. Finally, it returns the sliced array.

NAME :- Jangili Ravali
EMAIL :- jangiliravali9@gmail.com

b) Create a recursive function to find the nth element of a Fibonacci sequence and store the first n elements in an array.



```java
package wiproPrgms;

import java.util.ArrayList;

public class Fibonacci {

    public static int fibonacci(int n, ArrayList<Integer> fibArray) {
        if (fibArray == null) {
            fibArray = new ArrayList<>();
            fibArray.add(0);
            fibArray.add(1);
        }

        if (n < fibArray.size()) {
            return fibArray.get(n);
        } else {
            fibArray.add(fibArray.get(fibArray.size()-1)+fibArray.get(fibArray.si
            return fibonacci(n, fibArray);
        }
    }

    public static void main(String[] args) {
        int n = 10; // Change this to the desired value of n
        ArrayList<Integer> fibSequence = new ArrayList<>();
        for (int i = 0; i< n; i++) {
            fibSequence.add(fibonacci(i, null));
        }
        System.out.println("Fibonacci sequence up to the"+ n +"th element:"+ fi
    }
}
```

```
<terminated> Fibonacci [Java Application] C:\Users\Ravali\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x
Picked up _JAVA_OPTIONS: -Xmx512M
Fibonacci sequence up to the10th element:[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

This Java code defines a Fibonacci class with a fibonacci method that calculates the nth Fibonacci number recursively and stores the first n elements in an ArrayList. The main method demonstrates how to use this method to generate and print the Fibonacci sequence up to the nth element.

NAME :- **Jangili Ravali**
EMAIL :- **jangiliravali9@gmail.com**