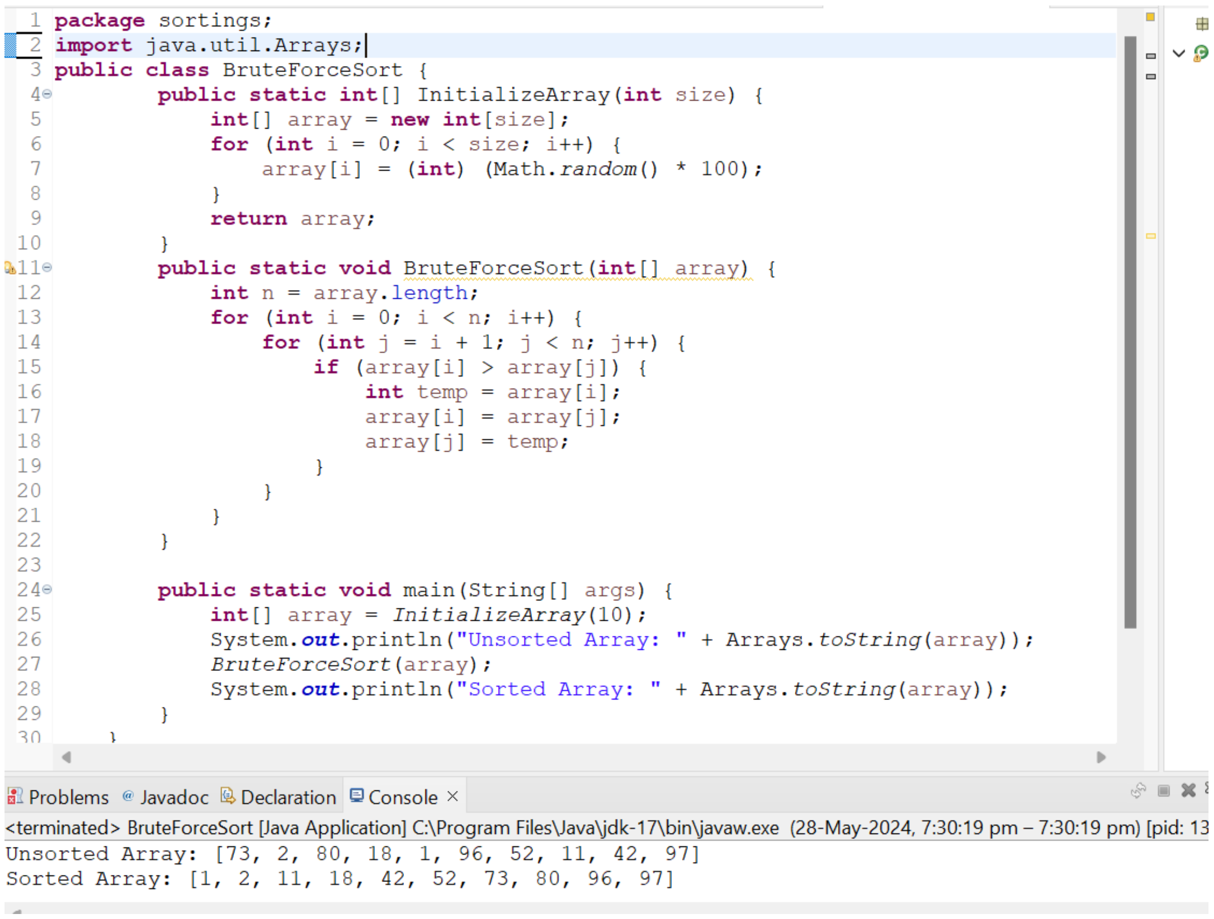
Manjula Nannuri

Day-4_Assignment

Brute Force Sorting:

Brute Force Sorting, also known as Simple Sort, involves comparing each element with every other element in the array and swapping them if they are out of order. This approach is typically inefficient with a time complexity of O(n^2), making it less ideal for large datasets but useful for educational purposes or very small datasets.

Implementing BruteForceSort and InitializeArray in Java



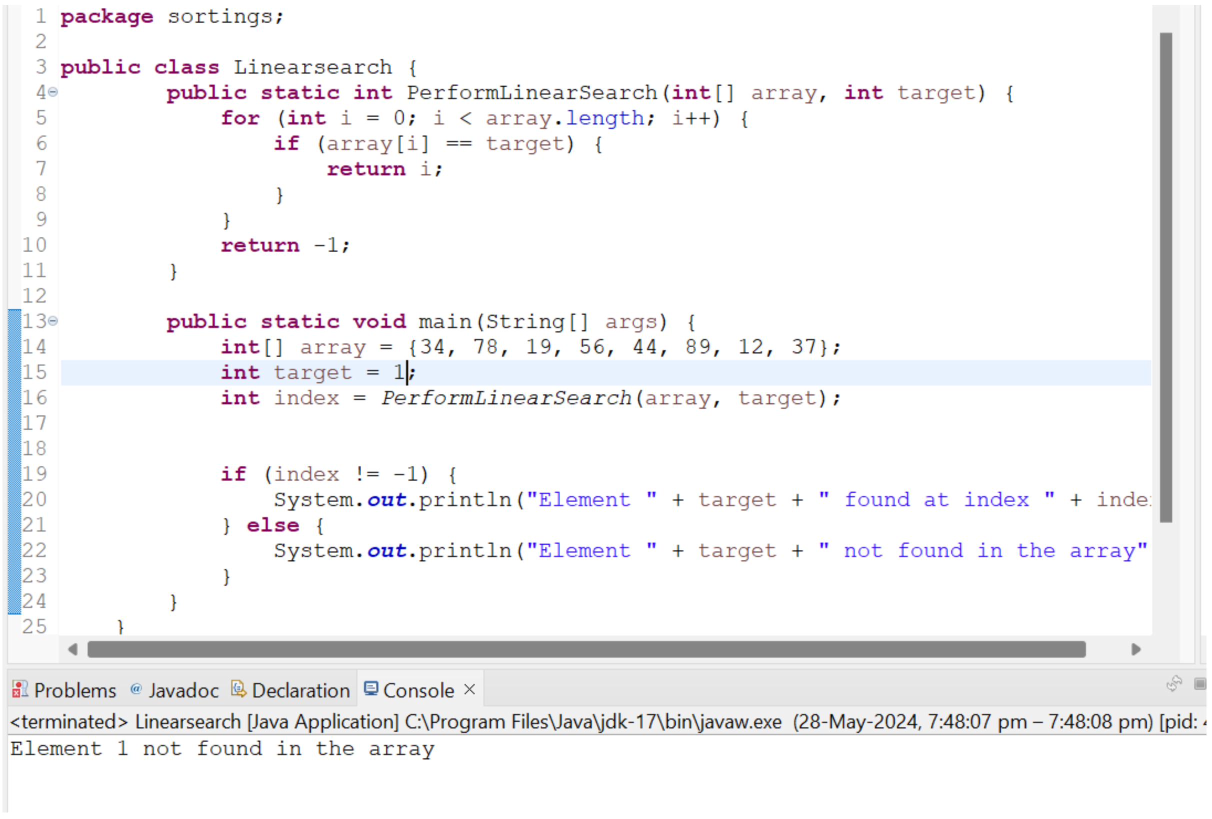
```java
package sortings;
import java.util.Arrays;
public class BruteForceSort {
        public static int[] InitializeArray(int size) {
            int[] array = new int[size];
            for (int i = 0; i < size; i++) {
                array[i] = (int) (Math.random() * 100);
            }
            return array;
        }
        public static void BruteForceSort(int[] array) {
            int n = array.length;
            for (int i = 0; i < n; i++) {
                for (int j = i + 1; j < n; j++) {
                    if (array[i] > array[j]) {
                        int temp = array[i];
                        array[i] = array[j];
                        array[j] = temp;
                    }
                }
            }
        }

        public static void main(String[] args) {
            int[] array = InitializeArray(10);
            System.out.println("Unsorted Array: " + Arrays.toString(array));
            BruteForceSort(array);
            System.out.println("Sorted Array: " + Arrays.toString(array));
        }
    }
```

```
<terminated> BruteForceSort [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe  (28-May-2024, 7:30:19 pm – 7:30:19 pm) [pid: 13
Unsorted Array: [73, 2, 80, 18, 1, 96, 52, 11, 42, 97]
Sorted Array: [1, 2, 11, 18, 42, 52, 73, 80, 96, 97]
```

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.

Sol:
Definition of Linear Search

Linear search, also known as sequential search, is a straightforward searching algorithm. It works by iterating through each element in the array and comparing it with the target value. If the target value is found, it returns the index of the element. If the target value is not found after checking all elements, it returns -1. The time complexity of linear search is O(n), where n is the number of elements in the array



```java
package sortings;

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

	public static void main(String[] args) {
		int[] array = {34, 78, 19, 56, 44, 89, 12, 37};
		int target = 1;
		int index = PerformLinearSearch(array, target);


		if (index != -1) {
			System.out.println("Element " + target + " found at index " + inde
		} else {
			System.out.println("Element " + target + " not found in the array"
		}
	}
}
```

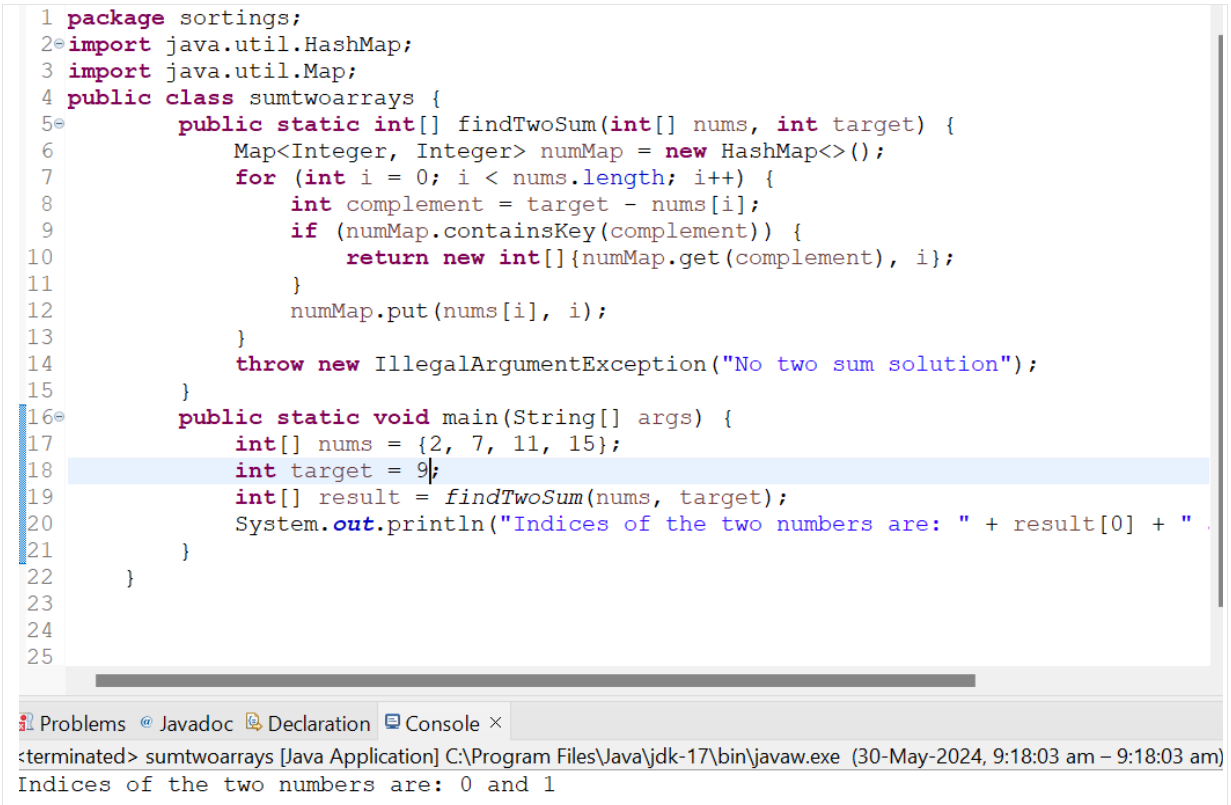
Problems Javadoc Declaration Console

```
<terminated> Linearsearch [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe  (28-May-2024, 7:48:07 pm – 7:48:08 pm) [pid:
Element 1 not found in the array
```

PerformLinearSearch:

- This function takes an array and a target value as inputs.
- It uses a for loop to iterate through each element in the array.
- For each element, it checks if the current element is equal to the target value.
- If the target value is found, it returns the index of the element.
- If the loop completes without finding the target value, it returns -1.

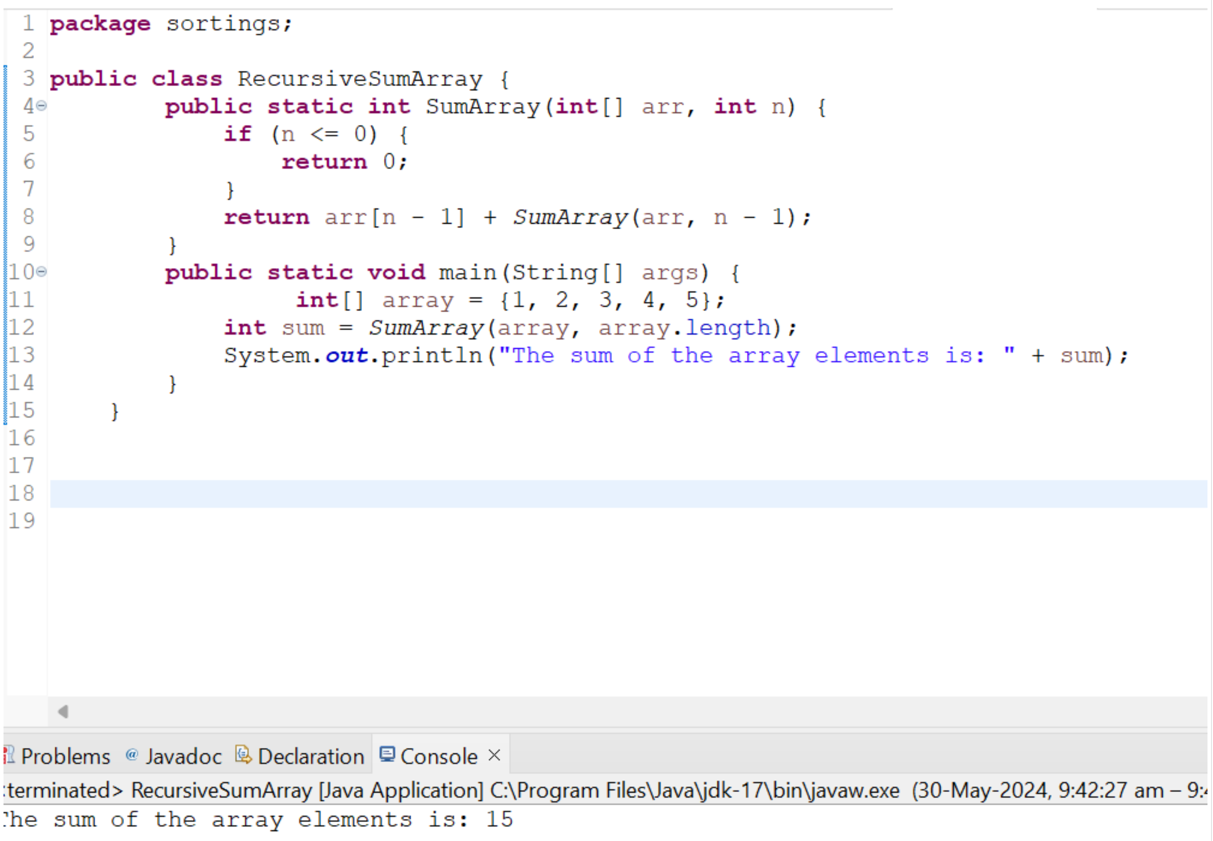
a) Given an array of integers, write a program that finds if there are two numbers that add up to a specific target. You may assume that each input would have exactly one solution, and you may not use the same element twice. Optimize the solution for time complexity.



```java
package sortings;
import java.util.HashMap;
import java.util.Map;
public class sumtwoarrays {
        public static int[] findTwoSum(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);
            }
            throw new IllegalArgumentException("No two sum solution");
        }
        public static void main(String[] args) {
            int[] nums = {2, 7, 11, 15};
            int target = 9;
            int[] result = findTwoSum(nums, target);
            System.out.println("Indices of the two numbers are: " + result[0] + "
        }
    }
```

```
<terminated> sumtwoarrays [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (30-May-2024, 9:18:03 am – 9:18:03 am)
Indices of the two numbers are: 0 and 1
```

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



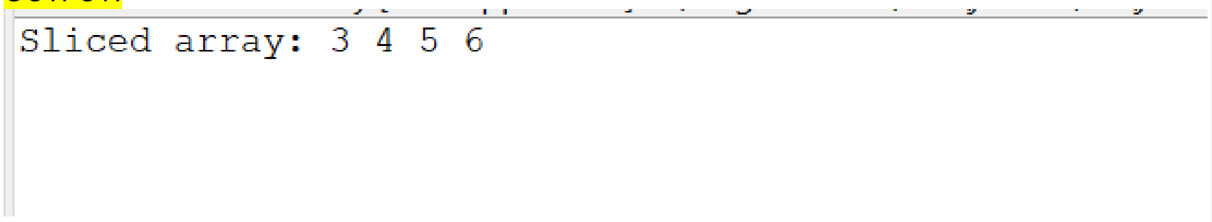
```java
package sortings;

public class RecursiveSumArray {
        public static int SumArray(int[] arr, int n) {
            if (n <= 0) {
                return 0;
            }
            return arr[n - 1] + SumArray(arr, n - 1);
        }
        public static void main(String[] args) {
              int[] array = {1, 2, 3, 4, 5};
            int sum = SumArray(array, array.length);
            System.out.println("The sum of the array elements is: " + sum);
        }
    }
```

```
terminated> RecursiveSumArray [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (30-May-2024, 9:42:27 am – 9:
he sum of the array elements is: 15
```

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.

```
package sortings;

public class SliceArray {

public static int[] SliceArray(int[] arr, int start, int end) {

if (start < 0 || end >= arr.length || start > end) {

throw new IllegalArgumentException("Invalid start or end index");

}

int size = end - start + 1;

int[] slicedArray = new int[size];

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

slicedArray[i] = arr[start + i];

}

return slicedArray;

}

public static void main(String[] args) {

int[] array = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int start = 2;

int end = 5;

int[] result = SliceArray(array, start, end);

System.out.print("Sliced array: ");

for (int num : result) {

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

}
}

}
```

OUTPUT:

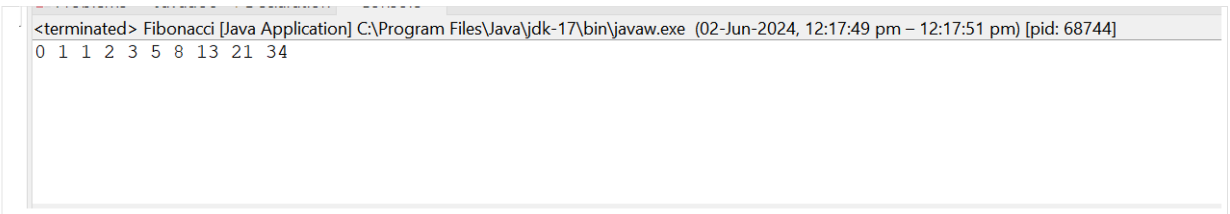


```
Sliced array: 3 4 5 6
```

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

```
package sortings;

class Fibonacci{

public static void main(String args[])

{

int n1=0,n2=1,n3,i,count=10;

System.out.print(n1+" "+n2);

for(i=2;i<count;++i)

{

n3=n1+n2;

System.out.print(" "+n3);

n1=n2;

n2=n3;

}

}

}
```

OUTPUT:



```
<terminated> Fibonacci [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe  (02-Jun-2024, 12:17:49 pm – 12:17:51 pm) [pid: 68744]
0 1 1 2 3 5 8 13 21 34
```