

1. Find common elements between two arrays

```
import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;

public class CommonElementsSet {
    public static void main(String[] args) {
        Integer[] array1 = {4, 2, 3, 1, 6};
        Integer[] array2 = {6, 7, 9, 8, 4};

        Set<Integer> set1 = new HashSet<>(Arrays.asList(array1));
        Set<Integer> set2 = new HashSet<>(Arrays.asList(array2));

        set1.retainAll(set2); // set1 now contains only common elements
        System.out.println("Common Elements: " + set1);
    }
}
```

Common Elements: [4, 6]

2. Find first and last element of ArrayList

```
import java.util.ArrayList;

public class ArrayListElements {
    public static void main(String[] args) {
        ArrayList<String> fruits = new ArrayList<>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Cherry");

        if (!fruits.isEmpty()) {
            String firstFruit = fruits.get(0);
            String lastFruit = fruits.get(fruits.size() - 1);

            System.out.println("First fruit: " + firstFruit);
            System.out.println("Last fruit: " + lastFruit);
        } else {
            System.out.println("The ArrayList is empty.");
        }
    }
}
```

First element: Apple
Last element: Elderberry

3. Sort an array without using in-built method

```
import java.util.Arrays; // Import the Arrays class

public class SortArrayBuiltIn {
    public static void main(String[] args) {
        int[] numbers = {5, 2, 8, 1, 9, 3}; // Declare and initialize an integer array

        System.out.println("Original array: " + Arrays.toString(numbers));

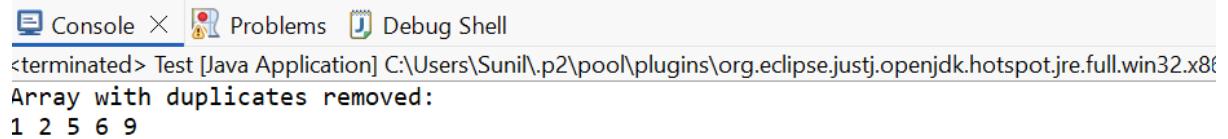
        Arrays.sort(numbers); // Sort the array in ascending order

        System.out.println("Sorted array (ascending): " + Arrays.toString(numbers));
    }
}
```

```
Original array: [5, 2, 8, 1, 9, 3]
Sorted array (ascending): [1, 2, 3, 5, 8, 9]
```

4. Remove duplicates from an Array

```
/ 
8 public class Test {
9     public static void main(String[] args) {
10         int[] array = { 5, 2, 9, 1, 6, 2, 5 };
11         int[] uniqueArray = removeDuplicates(array);
12         System.out.println("Array with duplicates removed:");
13         for (int num : uniqueArray)
14         {
15             System.out.print(num + " ");
16         }
17     }
18
19     public static int[] removeDuplicates(int[] array)
20     {
21         Set<Integer> set = new HashSet<>();
22         for (int num : array) {
23             set.add(num);
24         }
25         int[] result = new int[set.size()];
26         int i = 0;
27         for (int num : set) {
28             result[i++] = num;
29         }
30         return result;
31     }
32 }
33 }
```



The screenshot shows the Eclipse IDE interface. The code editor displays the Java code for removing duplicates from an array. Below the editor, the 'Console' view is active, showing the output of the program's execution. The output text is:
<terminated> Test [Java Application] C:\Users\Sunil\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86
Array with duplicates removed:
1 2 5 6 9

5. Remove duplicates from an ArrayList

The screenshot shows the Eclipse IDE interface with a Java code editor and a terminal window. The code demonstrates two ways to remove duplicates from an ArrayList of strings. The first method uses a HashSet to store unique elements, resulting in an unordered list. The second method uses a LinkedHashSet to preserve the insertion order of unique elements. Both methods then print the original list and the resulting list without duplicates.

```
10 public class Test {
11     public static void main(String[] args) {
12         ArrayList<String> originalList = new ArrayList<>();
13         originalList.add("Apple");
14         originalList.add("Banana");
15         originalList.add("Apple");
16         originalList.add("Orange");
17         originalList.add("Banana");
18
19         System.out.println("Original List: " + originalList);
20
21         // Using HashSet (order not guaranteed)
22         HashSet<String> uniqueElementsSet = new HashSet<>(originalList);
23         ArrayList<String> listWithoutDuplicates = new ArrayList<>(uniqueElementsSet);
24         System.out.println("List without duplicates (HashSet): " + listWithoutDuplicates);
25
26         // Using LinkedHashSet (preserves insertion order)
27         LinkedHashSet<String> uniqueElementsLinkedSet = new LinkedHashSet<>(originalList);
28         ArrayList<String> listWithoutDuplicatesOrdered = new ArrayList<>(uniqueElementsLinkedSet);
29         System.out.println("List without duplicates (LinkedHashSet, ordered): " + listWithoutDuplicatesOrdered);
30     }
31 }
```

Console X Problems Debug Shell
<terminated> Test Java Application C:\Users\Sunil\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (18-Oct-2025, 5:36:31 pr
Original List: [Apple, Banana, Apple, Orange, Banana]
List without duplicates (HashSet): [Apple, Orange, Banana]
List without duplicates (LinkedHashSet, ordered): [Apple, Banana, Orange]

6. Find the missing number in an Array

The screenshot shows the Eclipse IDE interface with a Java code editor and a terminal window. The code finds the missing number in a sequence of integers. It calculates the expected sum of the first n numbers and subtracts the actual sum obtained by summing the elements of the array. The result is the missing number.

```
10 public class Test {
11     public static void main(String[] args) {
12         int[] arr = { 1, 2, 4, 5, 6 };
13         int n = arr.length + 1; // n is the total count of numbers in the complete sequence
14         int expectedSum = n * (n + 1) / 2;
15         int actualSum = 0;
16
17         for (int num : arr) {
18             actualSum += num;
19         }
20         int l = expectedSum - actualSum;
21         System.out.println("Missing number in arr1: " + l); // Output: 3
22
23     }
24 }
```

Console X Problems Debug Shell
<terminated> Test Java Application C:\Users\Sunil\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0
Missing number in arr1: 3

7. Find the largest and smallest element in an Array

```
public class ArrayMinMax {  
  
    public static void main(String[] args) {  
        int[] arr = {12, 13, 1, 10, 34, 10};  
  
        // Initialize largest and smallest with the first element of the array  
        int largest = arr[0];  
        int smallest = arr[0];  
  
        // Iterate through the array starting from the second element  
        for (int i = 1; i < arr.length; i++) {  
            if (arr[i] < smallest) {  
                smallest = arr[i]; // Update smallest if a smaller element is found  
            }  
            if (arr[i] > largest) {  
                largest = arr[i]; // Update largest if a larger element is found  
            }  
        }  
  
        System.out.println("Smallest element: " + smallest);  
        System.out.println("Largest element: " + largest);  
    }  
}
```

```
Smallest element: 1  
Largest element: 34
```

8. Search element in an Array



The screenshot shows the Eclipse IDE interface with a Java code editor and a terminal window.

Java Code:

```
1 package test;  
2  
3 import java.util.ArrayList;  
4  
5 public class Test {  
6     public static void main(String[] args) {  
7         int[] array = {5, 2, 9, 1, 6, 3};  
8         int target = 6;  
9         for (int i = 0; i < array.length; i++) {  
10             if (array[i] == target) {  
11                 if (i != -1) {  
12                     System.out.println("Element " + target + " found at index: " + i);  
13                 } else {  
14                     System.out.println("Element " + target + " not found in the array.");  
15                 }  
16             }  
17         }  
18     }  
19 }  
20
```

Eclipse IDE Status Bar:

- Console X
- Problems
- Debug Shell

Terminal Output:

```
<terminated> Test [Java Application] C:\Users\Sunil\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin  
Element 6 found at index: 4
```

9. Array consists of integers and special characters, sum only integers

The screenshot shows the Eclipse IDE interface. The code editor displays a Java class named 'Test' with a main method. The main method initializes an array of strings containing integers and special characters, then iterates through the array, attempting to parse each element into an integer and adding it to a sum if successful. Non-integer elements are ignored. The output window shows the result: 'Sum of integers in the array: 26'.

```
9
10 public class Test {
11     public static void main(String[] args) {
12         String[] array = { "5", "2", "9", "a", "1", "6", "#", "3" };
13         int sum = 0;
14         for (String element : array) {
15             try {
16                 int num = Integer.parseInt(element);
17                 sum += num;
18             } catch (NumberFormatException e) {
19                 // Ignore non-integer elements
20             }
21         }
22         System.out.println("Sum of integers in the array: " + sum);
23     }
24 }
25
```

Console X Problems Debug Shell
<terminated> Test [Java Application] C:\Users\Sunil\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21
Sum of integers in the array: 26

10. Find Minimum and Maximum from an Array

The screenshot shows the Eclipse IDE interface. The code editor displays a Java class named 'Test' with a main method. The main method initializes an array of integers and then iterates through it to find the minimum and maximum values. The output window shows the results: 'Minimum value in the array is: 10' and 'Maximum value in the array is: 89'.

```
10 public class Test {
11     public static void main(String[] args) {
12         int[] arr = { 10, 34, 67, 55, 89, 76, 14 };
13
14         // Initialize min and max with the first element of the array
15         int min = arr[0];
16         int max = arr[0];
17
18         // Iterate through the array starting from the second element (index 1)
19         for (int i = 1; i < arr.length; i++) {
20             if (arr[i] < min) {
21                 min = arr[i]; // Update min if current element is smaller
22             }
23             if (arr[i] > max) {
24                 max = arr[i]; // Update max if current element is larger
25             }
26         }
27
28         System.out.println("Minimum value in the array is: " + min);
29         System.out.println("Maximum value in the array is: " + max);
30     }
31 }
```

Console X Problems Debug Shell
<terminated> Test [Java Application] C:\Users\Sunil\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v202
Minimum value in the array is: 10
Maximum value in the array is: 89

11. Java program to count Odd and Even number from given array

```
10 public class Test {
11     public static void main(String[] args) {
12         // Declare and initialize an array of integers
13         int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
14
15         // Initialize counters for even and odd numbers
16         int evenCount = 0;
17         int oddCount = 0;
18
19         System.out.println("Original Array: " + Arrays.toString(numbers));
20
21         // Iterate through the array to count even and odd numbers
22         for (int number : numbers) { // Enhanced for loop for cleaner iteration
23             if (number % 2 == 0) {
24                 // If the number is divisible by 2, it's even
25                 evenCount++;
26             } else {
27                 // Otherwise, it's odd
28                 oddCount++;
29             }
30         }
31         // Print the counts of even and odd numbers
32         System.out.println("Number of even elements in the array: " + evenCount);
33         System.out.println("Number of odd elements in the array: " + oddCount);
34     }
35 }
```

Console X Problems Debug Shell
<terminated> Test [Java Application] C:\Users\Sunil\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7
Original Array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Number of even elements in the array: 5
Number of odd elements in the array: 5

12. Java program – input array was given [1,1,2,2,3,4,5,5,6,6]

```
13 public class Test {
14     public static void main(String[] args) {
15         int[] array = { 1, 1, 2, 2, 3, 4, 5, 5, 6, 6 };
16         List<Integer> result = findNonRepeatedElements(array);
17         System.out.println("Non-repeated elements: " + result);
18     }
19
20     public static List<Integer> findNonRepeatedElements(int[] array) {
21         // Step 1: Count occurrences of each element using a HashMap
22         Map<Integer, Integer> countMap = new HashMap<>();
23         for (int num : array) {
24             countMap.put(num, countMap.getOrDefault(num, 0) + 1);
25         }
26         // Step 2: Identify elements with count equal to 1 (non repeated)
27         List<Integer> nonRepeatedElements = new ArrayList<>();
28         for (Map.Entry<Integer, Integer> entry : countMap.entrySet()) {
29             if (entry.getValue() == 1) {
30                 nonRepeatedElements.add(entry.getKey());
31             }
32         }
33         return nonRepeatedElements;
34     }
35 }
```

Console X Problems Debug Shell
<terminated> Test [Java Application] C:\Users\Sunil\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe
Non-repeated elements: [3, 4]