Lesson:

Java

Problems on Array-1

PW SKILLS

Pre-Requisites

- Arrays in Java

List of Concepts Involved

- Basic problem-solving in Arrays

Topic: Taking input in an array

Pattern: Target sum (Brute Force Approach)
In this approach, we try to sum the elements of the array to a particular target value. In the brute force approach, we use the nested for loop and try to match the value at every point.

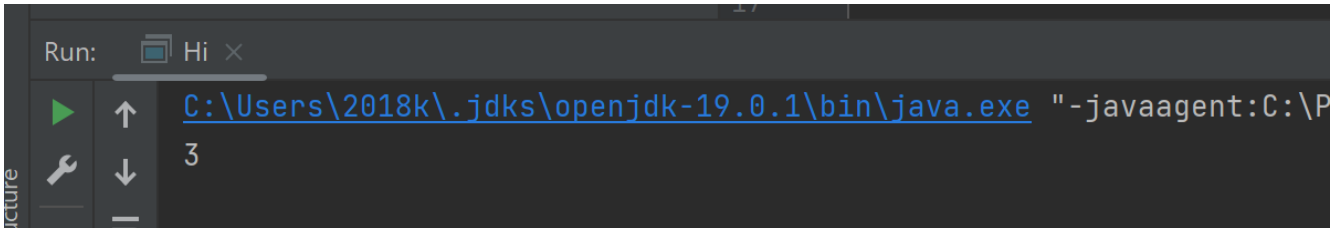
Look at the problems discussed below to have a better understanding.

Problem 1: Find the total number of pairs in the array whose sum is equal to the given value x.

Important Note : The number of loops depends on the number of elements we have to take in the array, for example, for pair sum, we use 2 nested for loops. And for triple sum, we use triple nested for loop.

Code:

```
import java.util.*;
import java.lang.*;
  public class Target {
    public static int pairSum(int a[], int x) {
      int count = 0;
      int n = a.length;
      for(int i=0;i<n;i++)
        for(int j=i+1;j<n;j++)
          if(a[i]+a[j]==x)
            count++;
      return count;
    }
    public static void main(String[] args){
      int a[]={4,7,3,21,6,5,5};
      System.out.println(pairSum(a,10));
    }
}
```



Explanation of approach: Traverse through the array using a nested for loop, and just add the sum of the elements at each iteration, then compare that sum with target value 'x' and increment the count if found equal. Return the value of count in the end.

Problem 2: Count the number of triplets whose sum is equal to the given value x.

Code:

```
import java.util.*;
import java.lang.*;
  public class Target {
    public static int findTriplet(int[] a, int x) {
      int n = a.length;
      int count = 0;
      for(int i=0;i<n;i++){
        for(int j=i+1;j<n;j++){
          for(int k=j+1;k<n;k++){
            if(a[i]+a[j]+a[k]==x){
              count++;
            }
          }
        }
    }
    return count;
  }
  public static void main(String[] args){
    int a[]={4,7,3,21,6,5,5};
    System.out.println(findTriplet(a,16));
  }
}
```

Explanation of approach: Traverse through the array using a triple nested for loop, and just add the sum of the elements at each iteration, then compare that sum with target value 'x' and increment the count if found equal. Return the value of count in the end.

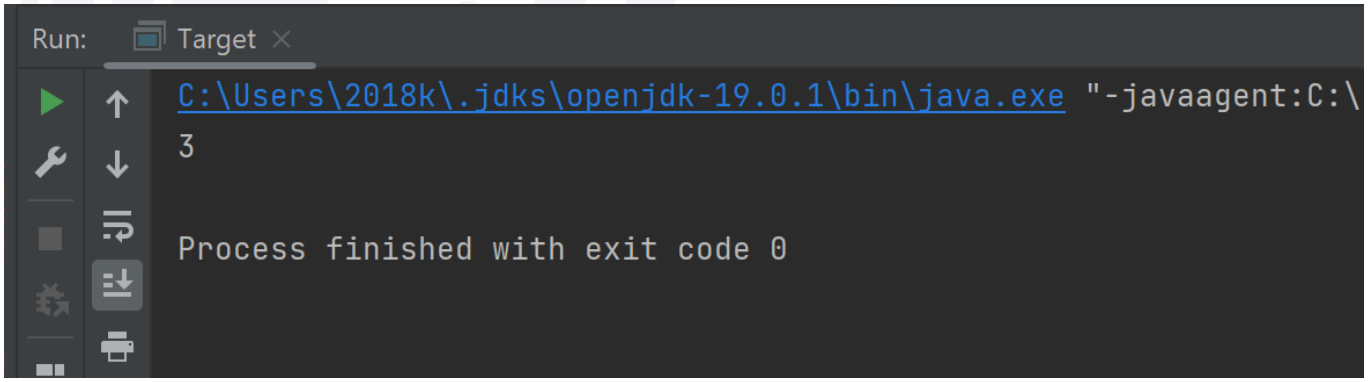
Pattern: Array Manipulation Problems

Here, we will discuss array manipulation problems where we perform some operations on the array in order to solve the problem.

Problem 3: Find the unique number in a given array where all the elements are repeated twice with one value being unique.

Code:

```
import java.util.*;
import java.lang.*;
  public class Target {
    public static int findUnique(int[] a){
      int n = a.length;
      for(int i=0;i<n;i++){
        for(int j=i+1;j<n;j++){
          if(a[i]==a[j]){
            a[i] = a[j] = -1;
          }
        }
      }
      for(int i=0;i<n;i++)
        if(a[i]>0)
          return a[i];
      return 1;
  }
  public static void main(String[] args){
    int a[]={21,7,3,21,5,7,5};
    System.out.println(findUnique(a));
  }
}
```

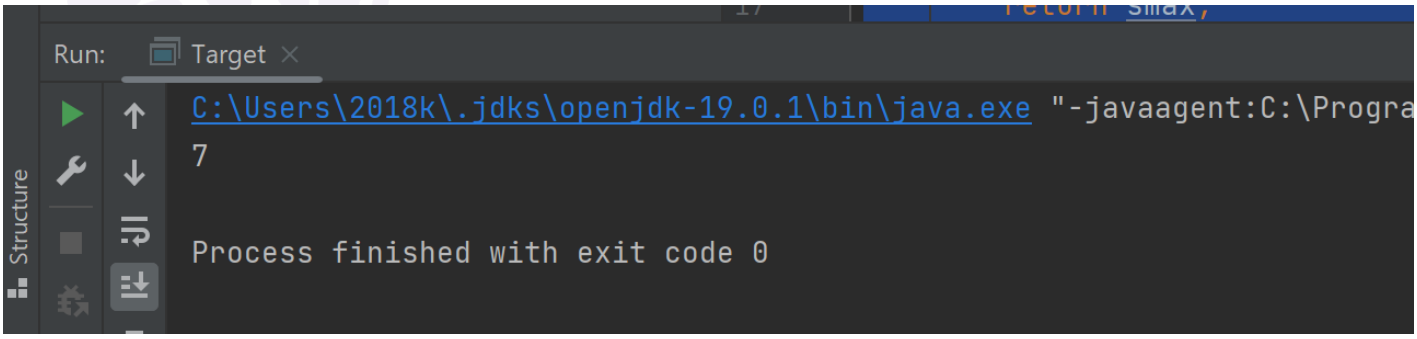


Explanation of approach: Go through the array and in the first traversal, we take the current element and try to find if it exists again in the array using a nested for loop. If it exists again in the array, just mark it as -1. In the end, use a single traversal to find if any element is positive, that means it has no duplicate and hence it is our answer.

Problem 4: Find the second-largest value in the given array.

Code:

```
import java.util.*;
  import java.lang.*;
  public class Target {
  public static int secondLargestValue(int[] a) {
    int n = a.length;
    int max = Integer.MIN_VALUE;
    int smax = Integer.MIN_VALUE;
    for(int i=0;i<n;i++){
      if(max<a[i])
        max = a[i];
    }
    for(int i=0;i<n;i++){
      if(a[i]≠max)
      {
        if(a[i]>smax)
          smax = a[i];
      }
    }
    return smax;
  }
  public static void main(String[] args){
    int a[]={21,7,3,21,5,7,5};
    System.out.println(secondLargestValue(a));
  }
}
```



Explanation of approach: First find the max using a single traversal, which can be found by comparing it with every element while traversing through the array. In the second traversal, check at every point if the current element is not the max element, it can be compared to find the second max element.

Problem 5: Given an array 'a' consisting of integers. Return the first value that is repeating in this array. If no value is being repeated, return -1.

Input :
n = 7
a[] = {1, 5, 3, 4, 3, 5, 6}
Output: 2
Explanation: 5 is the first value that is being repeated so return the first index of 5 that is 2.

Code:

```
import java.util.*;
import java.lang.*;
public class Main {
  public static int firstRepeated(int[] a, int n) {
    for(int i = 0; i < n; i++){
      int curr = a[i];
      for(int j = i+1; j < n; j++) {
        if(a[j] == curr) return i+1;
      }
    }
    return -1;
  }
  public static void main(String[] args) {
    int n;
    Scanner sc = new Scanner(System.in);
    n = sc.nextInt();
    int[] a = new int[n];
    for(int i = 0; i < n; i++) {
      a[i] = sc.nextInt();
    }
    int ans = firstRepeated(a, n);
    System.out.println(ans);
  }
}
```

Explanation of approach: For every element in the array, iterate again through the whole array forming a nested loop and check if it is being repeated. Return the first index of the element being repeated.

That is all for this class! Let us meet in the next one!
Till then, happy learning!

Upcoming Class Teasers:

- Problem Solving in Arrays