// Java program to illustrate creating an array

// of integers,  puts some values in the array,

// and prints each value to standard output.

class GFG

{

    public static void main (String[] args)

    {

      // declares an Array of integers.

      int[] arr;

      // allocating memory for 5 integers.

      arr = new int[5];

      // initialize the first elements of the array

      arr[0] = 10;

      // initialize the second elements of the array

      arr[1] = 20;

      //so on...

      arr[2] = 30;

      arr[3] = 40;

      arr[4] = 50;

      // accessing the elements of the specified array

      for (int i = 0; i < arr.length; i++)

         System.out.println("Element at index " + i +

                                      " : "+ arr[i]);

    }

}

// Java program to illustrate creating an array of

// objects

class Student

{

    public int roll\_no;

    public String name;

    Student(int roll\_no, String name)

    {

        this.roll\_no = roll\_no;

        this.name = name;

    }

}

// Elements of array are objects of a class Student.

public class GFG

{

    public static void main (String[] args)

    {

        // declares an Array of integers.

        Student[] arr;

        // allocating memory for 5 objects of type Student.

        arr = new Student[5];

        // initialize the first elements of the array

        arr[0] = new Student(1,"Anil");

        // initialize the second elements of the array

        arr[1] = new Student(2,"Raman");

        // so on...

        arr[2] = new Student(3,"Abhishek");

        arr[3] = new Student(4,"Ananya");

        arr[4] = new Student(5,"Manan");

        // accessing the elements of the specified array

        for (int i = 0; i < arr.length; i++)

            System.out.println("Element at " + i + " : " +

                        arr[i].roll\_no +" "+ arr[i].name);

    }

}

**What happens if we try to access element outside the array size?**

Compiler throws **ArrayIndexOutOfBoundsException** to indicate that array has been accessed with an illegal index. The index is either negative or greater than or equal to size of array.

|  |
| --- |
| class GFG  {      public static void main (String[] args)      {          int[] arr = new int[2];          arr[0] = 10;          arr[1] = 20;            for (int i = 0; i <= arr.length; i++)              System.out.println(arr[i]);      }  } |

Runtime error

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 2

at GFG.main(File.java:12)

**Multidimensional Arrays**

Multidimensional arrays are **arrays of arrays** with each element of the array holding the reference of other array. These are also known as [Jagged Arrays](https://www.geeksforgeeks.org/jagged-array-in-java/). A multidimensional array is created by appending one set of square brackets ([]) per dimension. Examples:

int[][] intArray = new int[10][20]; //a 2D array or matrix

int[][][] intArray = new int[10][20][10]; //a 3D array

|  |
| --- |
| class multiDimensional  {      public static void main(String args[])      {          // declaring and initializing 2D array          int arr[][] = { {2,7,9},{3,6,1},{7,4,2} };            // printing 2D array          for (int i=0; i< 3 ; i++)          {              for (int j=0; j < 3 ; j++)                  System.out.print(arr[i][j] + " ");                System.out.println();          }      }  } |

**Passing Arrays to Methods**

Like variables, we can also pass arrays to methods.For example, below program pass array to method sum for calculating sum of array’s values.

|  |
| --- |
| // Java program to demonstrate  // passing of array to method    class Test  {      // Driver method      public static void main(String args[])      {          int arr[] = {3, 1, 2, 5, 4};            // passing array to method m1          sum(arr);        }        public static void sum(int[] arr)      {          // getting sum of array values          int sum = 0;            for (int i = 0; i < arr.length; i++)              sum+=arr[i];            System.out.println("sum of array values : " + sum);      }  } |

Output :

sum of array values : 15

**Returning Arrays from Methods**

As usual, a method can also return an array. For example, below program returns an array from method *m1*.

|  |
| --- |
| // Java program to demonstrate  // return of array from method    class Test  {      // Driver method      public static void main(String args[])      {          int arr[] = m1();            for (int i = 0; i < arr.length; i++)              System.out.print(arr[i]+" ");        }        public static int[] m1()      {          // returning  array          return new int[]{1,2,3};      }  } |

Java Program to Find the Largest Two Numbers in a Given Array

This is a Java Program to Find the Largest Two Numbers in a Given Array.

Enter size of array and then enter all the elements of that array. Now we first sort the array in decreasing order using double for loops and hence get the first two elements as output.

Here is the source code of the Java Program to Find the Largest Two Numbers in a Given Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

1. **import** java.util.Scanner;
2. **public** **class** Largest\_Numbers
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** n, l1, l2, temp;
7. Scanner s = **new** Scanner(System.in);
8. System.out.print("Enter no. of elements you want in array:");
9. n = s.nextInt();
10. **if**(n > 1)
11. {
12. **int** a[] = **new** **int**[n];
13. System.out.println("Enter all the elements:");
14. **for**(**int** i = 0; i < n ; i++)
15. {
16. a[i] = s.nextInt();
17. }
18. **for**(**int** i = 0; i < n; i++)
19. {
20. **for**(**int** j = i + 1; j < n; j++)
21. {
22. **if**(a[i] < a[j])
23. {
24. temp = a[i];
25. a[i] = a[j];
26. a[j] = temp;
27. }
28. }
29. }
30. System.out.println("Largest two numbers are:"+a[0]+" and "+a[1]);
31. }
32. **else**
33. {
34. System.out.println("Enter number greater than 1");
35. }
36. }
37. }

Output:

$ javac Largest\_Numbers.java

$ java Largest\_Numbers

Enter no. of elements you want in array:8

Enter all the elements:

5

3

7

9

2

3

9

10

Largest two numbers are:10 and 9

Java Program to Find the Second Largest & Smallest Elements in an Array

This is a Java Program to Find the Second Largest & Smallest Elements in an Array.

Enter size of array and then enter all the elements of that array. Now with the help of for loop and temp variable we sort the array in ascending order. Hence we get the first and second last element as output.

Here is the source code of the Java Program to Find the Second Largest & Smallest Elements in an Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

1. **import** java.util.Scanner;
2. **public** **class** SecondLargest\_Smallest
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** n, temp;
7. Scanner s = **new** Scanner(System.in);
8. System.out.print("Enter no. of elements you want in array(Minimum 2):");
9. n = s.nextInt();
10. **int** a[] = **new** **int**[n];
11. System.out.println("Enter all the elements:");
12. **for** (**int** i = 0; i < n; i++)
13. {
14. a[i] = s.nextInt();
15. }
16. **for** (**int** i = 0; i < n; i++)
17. {
18. **for** (**int** j = i + 1; j < n; j++)
19. {
20. **if** (a[i] > a[j])
21. {
22. temp = a[i];
23. a[i] = a[j];
24. a[j] = temp;
25. }
26. }
27. }
28. System.out.println("Second Largest:"+a[n-2]);
29. System.out.println("Smallest:"+a[0]);
30. }
31. }

Output:

$ javac SecondLargest\_Smallest.java

$ java SecondLargest\_Smallest

Enter no. of elements you want in array(Minimum 2):8

Enter all the elements:

2

5

1

7

8

6

9

3

Second Largest:8

Smallest:1

Java Program to Find the Largest Number in an Array

This is a Java Program to Find the Largest Number in an Array.

Enter the elements of array as input. By comparing elements of array with each other we get the largest number of the array.

Here is the source code of the Java Program to Find the Largest Number in an Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

1. **import** java.util.Scanner;
2. **public** **class** Largest\_Number
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** n, max;
7. Scanner s = **new** Scanner(System.in);
8. System.out.print("Enter number of elements in the array:");
9. n = s.nextInt();
10. **int** a[] = **new** **int**[n];
11. System.out.println("Enter elements of array:");
12. **for**(**int** i = 0; i < n; i++)
13. {
14. a[i] = s.nextInt();
15. }
16. max = a[0];
17. **for**(**int** i = 0; i < n; i++)
18. {
19. **if**(max < a[i])
20. {
21. max = a[i];
22. }
23. }
24. System.out.println("Maximum value:"+max);
25. }
26. }

Output:

$ javac Largest\_Number.java

$ java Largest\_Number

Enter number of elements in the array:5

Enter elements of array:

4

2

3

6

1

Maximum value:6

Java Program to Put Even & Odd Elements of an Array in 2 Separate Arrays

This is a Java Program to Put Even & Odd Elements of an Array in 2 Separate Arrays.

Enter size of array and then enter all the elements of that array. Now with the help of for loop and if condition we check whether its odd or not and hence print it accordingly.

Here is the source code of the Java Program to Put Even & Odd Elements of an Array in 2 Separate Arrays. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

1. **import** java.util.Scanner;
2. **public** **class** Even\_Odd
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** n, j = 0, k = 0;
7. Scanner s = **new** Scanner(System.in);
8. System.out.print("Enter no. of elements you want in array:");
9. n = s.nextInt();
10. **int** a[] = **new** **int**[n];
11. **int** odd[] = **new** **int**[n];
12. **int** even[] = **new** **int**[n];
13. System.out.println("Enter all the elements:");
14. **for**(**int** i = 0; i < n; i++)
15. {
16. a[i] = s.nextInt();
17. }
18. **for**(**int** i = 0; i < n; i++)
19. {
20. **if**(a[i] % 2 != 0)
21. {
22. odd[j] = a[i];
23. j++;
24. }
25. **else**
26. {
27. even[k] = a[i];
28. k++;
29. }
30. }
31. System.out.print("Odd:");
32. **if**(j > 1)
33. {
34. **for**(**int** i = 0;i < (j-1); i++)
35. {
36. System.out.print(odd[i]+",");
37. }
38. System.out.print(odd[j-1]);
39. }
40. **else**
41. {
42. System.out.println("No number");
43. }
44. System.out.println("");
45. System.out.print("Even:");
46. **if**(k > 1)
47. {
48. **for**(**int** i = 0; i < (k-1); i++)
49. {
50. System.out.print(even[i]+",");
51. }
52. System.out.print(even[k-1]);
53. }
54. **else**
55. {
56. System.out.println("No number");
57. }
58. }
59. }

Output:

$ javac Even\_Odd.java

$ java Even\_Odd

Enter no. of elements you want in array:8

Enter all the elements:

1

2

3

4

5

6

7

8

Odd:1,3,5,7

Even:2,4,6,8

Java Program to Insert an Element in a Specified Position in a Given Array

This is a Java Program to Insert an Element in a Specified Position in a Given Array.

Enter size of array and then enter all the elements of that array. Now enter the element you want to insert and position where you want to insert that element. We shift all the elements in the array from that location by one and hence insert the element easily.

Here is the source code of the Java Program to Insert an Element in a Specified Position in a Given Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

1. **import** java.util.Scanner;
2. **public** **class** Insert\_Array
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** n, pos, x;
7. Scanner s = **new** Scanner(System.in);
8. System.out.print("Enter no. of elements you want in array:");
9. n = s.nextInt();
10. **int** a[] = **new** **int**[n+1];
11. System.out.println("Enter all the elements:");
12. **for**(**int** i = 0; i < n; i++)
13. {
14. a[i] = s.nextInt();
15. }
16. System.out.print("Enter the position where you want to insert element:");
17. pos = s.nextInt();
18. System.out.print("Enter the element you want to insert:");
19. x = s.nextInt();
20. **for**(**int** i = (n-1); i >= (pos-1); i--)
21. {
22. a[i+1] = a[i];
23. }
24. a[pos-1] = x;
25. System.out.print("After inserting:");
26. **for**(**int** i = 0; i < n; i++)
27. {
28. System.out.print(a[i]+",");
29. }
30. System.out.print(a[n]);
31. }
32. }

Output:

$ javac Insert\_Array.java

$ java Insert\_Array

Enter no. of elements you want in array:6

Enter all the elements:

2

4

6

9

4

5

Enter the position where you want to insert element:3

Enter the element you want to insert:7

After inserting:2,4,7,6,9,4,5

Java Program to Delete the Specified Integer from an Array

This is a Java Program to Delete the Specified Integer from an Array.

Enter size of array and then enter all the elements of that array. Now enter the element you want to delete. We first find the location of that element and then shift the positions of all the elements after the element to be removed by one.

Here is the source code of the Java Program to Delete the Specified Integer from an Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

1. **import** java.util.Scanner;
2. **public** **class** Delete
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** n, x, flag = 1, loc = 0;
7. Scanner s = **new** Scanner(System.in);
8. System.out.print("Enter no. of elements you want in array:");
9. n = s.nextInt();
10. **int** a[] = **new** **int**[n];
11. System.out.println("Enter all the elements:");
12. **for** (**int** i = 0; i < n; i++)
13. {
14. a[i] = s.nextInt();
15. }
16. System.out.print("Enter the element you want to delete:");
17. x = s.nextInt();
18. **for** (**int** i = 0; i < n; i++)
19. {
20. **if**(a[i] == x)
21. {
22. flag =1;
23. loc = i;
24. **break**;
25. }
26. **else**
27. {
28. flag = 0;
29. }
30. }
31. **if**(flag == 1)
32. {
33. **for**(**int** i = loc+1; i < n; i++)
34. {
35. a[i-1] = a[i];
36. }
37. System.out.print("After Deleting:");
38. **for** (**int** i = 0; i < n-2; i++)
39. {
40. System.out.print(a[i]+",");
41. }
42. System.out.print(a[n-2]);
43. }
44. **else**
45. {
46. System.out.println("Element not found");
47. }
48. }
49. }

Output:

$ javac Delete.java

$ java Delete

Enter no. of elements you want in array:5

Enter all the elements:

3

5

8

1

4

Enter the element you want to delete:5

After Deleting:3,8,1,4

Java Program to Sort the Array in an Ascending Order

This is a Java Program to Sort the Array in an Ascending Order.

Enter size of array and then enter all the elements of that array. Now with the help of for loop and temp variable we sort the array in ascending order.

Here is the source code of the Java Program to Sort the Array in an Ascending Order. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

1. **import** java.util.Scanner;
2. **public** **class** Ascending \_Order
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** n, temp;
7. Scanner s = **new** Scanner(System.in);
8. System.out.print("Enter no. of elements you want in array:");
9. n = s.nextInt();
10. **int** a[] = **new** **int**[n];
11. System.out.println("Enter all the elements:");
12. **for** (**int** i = 0; i < n; i++)
13. {
14. a[i] = s.nextInt();
15. }
16. **for** (**int** i = 0; i < n; i++)
17. {
18. **for** (**int** j = i + 1; j < n; j++)
19. {
20. **if** (a[i] > a[j])
21. {
22. temp = a[i];
23. a[i] = a[j];
24. a[j] = temp;
25. }
26. }
27. }
28. System.out.print("Ascending Order:");
29. **for** (**int** i = 0; i < n - 1; i++)
30. {
31. System.out.print(a[i] + ",");
32. }
33. System.out.print(a[n - 1]);
34. }
35. }

Output:

$ javac Ascending \_Order.java

$ java Ascending \_Order

Enter no. of elements you want in array:5

Enter all the elements:

4

3

2

6

1

Ascending Order:1,2,3,4,6

Java Program to Sort the Array in Descending Order

This is a Java Program to Sort the Array in Descending Order.

Enter size of array and then enter all the elements of that array. Now with the help of for loop and temp variable we sort the array in descending order.

Here is the source code of the Java Program to Sort the Array in Descending Order. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

1. **import** java.util.Scanner;
2. **public** **class** Descending\_Order
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** n, temp;
7. Scanner s = **new** Scanner(System.in);
8. System.out.print("Enter no. of elements you want in array:");
9. n = s.nextInt();
10. **int** a[] = **new** **int**[n];
11. System.out.println("Enter all the elements:");
12. **for** (**int** i = 0; i < n; i++)
13. {
14. a[i] = s.nextInt();
15. }
16. **for** (**int** i = 0; i < n; i++)
17. {
18. **for** (**int** j = i + 1; j < n; j++)
19. {
20. **if** (a[i] < a[j])
21. {
22. temp = a[i];
23. a[i] = a[j];
24. a[j] = temp;
25. }
26. }
27. }
28. System.out.print("Descending Order:");
29. **for** (**int** i = 0; i < n - 1; i++)
30. {
31. System.out.print(a[i] + ",");
32. }
33. System.out.print(a[n - 1]);
34. }
35. }

Output:

$ javac Descending\_Order.java

$ java Descending\_Order

Enter no. of elements you want in array:5

Enter all the elements:

2

3

5

1

4

Descending Order:5,4,3,2,1

Java Program to Sort Names in an Alphabetical Order

This is a Java Program to Sort Names in an Alphabetical Order.

Enter size of array and then enter all the names in that array. Now with the help of compareTo operator we can easily sort names in Alphabetical Order.

Here is the source code of the Java Program to Sort Names in an Alphabetical Order. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

1. **import** java.util.Scanner;
2. **public** **class** Alphabetical\_Order
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** n;
7. String temp;
8. Scanner s = **new** Scanner(System.in);
9. System.out.print("Enter number of names you want to enter:");
10. n = s.nextInt();
11. String names[] = **new** String[n];
12. Scanner s1 = **new** Scanner(System.in);
13. System.out.println("Enter all the names:");
14. **for**(**int** i = 0; i < n; i++)
15. {
16. names[i] = s1.nextLine();
17. }
18. **for** (**int** i = 0; i < n; i++)
19. {
20. **for** (**int** j = i + 1; j < n; j++)
21. {
22. **if** (names[i].compareTo(names[j])>0)
23. {
24. temp = names[i];
25. names[i] = names[j];
26. names[j] = temp;
27. }
28. }
29. }
30. System.out.print("Names in Sorted Order:");
31. **for** (**int** i = 0; i < n - 1; i++)
32. {
33. System.out.print(names[i] + ",");
34. }
35. System.out.print(names[n - 1]);
36. }
37. }

Output:

$ javac Alphabetical\_Order.java

$ java Alphabetical\_Order

Enter number of names you want to enter:5

Enter all the names:

bryan

adam

rock

chris

scott

Names in Sorted Order:adam,bryan,chris,rock,scott

Java Program to Split an Array from Specified Position

This is a Java Program to Split an Array from Specified Position.

Enter size of array and then enter all the elements of that array. Now enter the position from where you want to split. We first copy the elements from first position to that given position in secnd array and remaining elements in third array.

Here is the source code of the Java Program to Split an Array from Specified Position. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

1. **import** java.util.Scanner;
2. **public** **class** Split
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** n, x, flag = 1, loc = 0, k = 0,j = 0;
7. Scanner s = **new** Scanner(System.in);
8. System.out.print("Enter no. of elements you want in array:");
9. n = s.nextInt();
10. **int** a[] = **new** **int**[n];
11. **int** b[] = **new** **int**[n];
12. **int** c[] = **new** **int**[n];
13. System.out.println("Enter all the elements:");
14. **for** (**int** i = 0; i < n; i++)
15. {
16. a[i] = s.nextInt();
17. }
18. System.out.print("Enter the position from where you want to split:");
19. loc = s.nextInt();
20. **for**(**int** i = 0; i < loc; i++)
21. {
22. b[k] = a[i];
23. k++;
24. }
25. **for**(**int** i = loc; i < n; i++)
26. {
27. c[j] = a[i];
28. j++;
29. }
30. System.out.print("First array:");
31. **for**(**int** i = 0;i < k; i++)
32. {
33. System.out.print(b[i]+" ");
34. }
35. System.out.println("");
36. System.out.print("Second array:");
37. **for**(**int** i = 0; i < j; i++)
38. {
39. System.out.print(c[i]+" ");
40. }
41. }
42. }

Output:

$ javac Split.java

$ java Split

Enter no. of elements you want in array:8

Enter all the elements:

2

3

4

7

1

9

11

6

Enter the position from where you want to split:4

First array:2 3 4 7

Second array:1 9 11 6

Java Program to Calculate Sum & Average of an Array

This is a Java Program to Calculate Sum & Average of an Array.

Enter size of array and then enter all the elements of that array. Now using for loop we calculate sum of elements of array and hence we divide it by number of elements in array to get average.

Here is the source code of the Java Program to Calculate Sum & Average of an Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

1. **import** java.util.Scanner;
2. **public** **class** Sum\_Average
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** n, sum = 0;
7. **float** average;
8. Scanner s = **new** Scanner(System.in);
9. System.out.print("Enter no. of elements you want in array:");
10. n = s.nextInt();
11. **int** a[] = **new** **int**[n];
12. System.out.println("Enter all the elements:");
13. **for**(**int** i = 0; i < n ; i++)
14. {
15. a[i] = s.nextInt();
16. sum = sum + a[i];
17. }
18. System.out.println("Sum:"+sum);
19. average = (**float**)sum / n;
20. System.out.println("Average:"+average);
21. }
22. }

Output:

$ javac Sum\_Average.java

$ java Sum\_Average

Enter no. of elements you want in array:5

Enter all the elements:

4

7

6

9

3

Sum:29

Average:5.8

Java Program to Identify Missing Numbers in a Given Array

This is a Java Program to Identify Missing Numbers in a Given Array.

We made a method as getMissingNo in which we pass the array and length of the array as arguments and using this method we calculate the missing number and hence we get the desired output.

Here is the source code of the Java Program to Identify Missing Numbers in a Given Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

1. **public** **class** Missing
2. {
3. **static** **int** getMissingNo (**int** a[], **int** n)
4. {
5. **int** i, total;
6. total = (n + 1) \* (n + 2) / 2;
7. **for** ( i = 0; i < n; i++)
8. total -= a[i];
9. **return** total;
10. }
11. **public** **static** **void** main(String... s)
12. {
13. **int** a[ ] = {1, 2, 4, 5, 6};
14. **int** miss = getMissingNo(a, 5);
15. System.out.println("The number missing is :"+miss);
16. }
17. }

Output:

$ javac Missing.java

$ java Missing

The number missing is :3

Java Program to Count the Number of Occurrence of an Element in an Array

This is a Java Program to Count the Number of Occurrence of an Element in an Array.

Enter size of array and then enter all the elements of that array. Now enter the element of which you want to count occurrences. With the help of for loop now we can easily calculate number of occurrences of the given element.

Here is the source code of the Java Program to Count the Number of Occurrence of an Element in an Array. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

1. **import** java.util.Scanner;
2. **public** **class** Count\_Occurrence
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** n, x, count = 0, i = 0;
7. Scanner s = **new** Scanner(System.in);
8. System.out.print("Enter no. of elements you want in array:");
9. n = s.nextInt();
10. **int** a[] = **new** **int**[n];
11. System.out.println("Enter all the elements:");
12. **for**(i = 0; i < n; i++)
13. {
14. a[i] = s.nextInt();
15. }
16. System.out.print("Enter the element of which you want to count number of occurrences:");
17. x = s.nextInt();
18. **for**(i = 0; i < n; i++)
19. {
20. **if**(a[i] == x)
21. {
22. count++;
23. }
24. }
25. System.out.println("Number of Occurrence of the Element:"+count);
26. }
27. }

Output:

$ javac Count\_Occurrence.java

$ java Count\_Occurrence

Enter no. of elements you want in array:5

Enter all the elements:

2

3

3

4

3

Enter the element of which you want to count number of occurrences:3

Number of Occurrence of the Element:3

Java Program to Find Union & Intersection of 2 Arrays

This is the Java Program to Find the Union and Intersection of 2 Arrays.

**Problem Description**

Given two arrays of integers, find and print the union and intersection of the arrays.

Example:  
Array: [1,2,3,4,5]

Array1: [5,3,6,7,9]

Output:  
Union = [1,2,3,4,5,6,7,9]

Intersection = [3,5]

**Problem Solution**

**Create two sets called union and intersection**. To find the union, add the first array’s elements to the set union. Now, add all the second array elements if they are not present in the set union. To find the intersection, find the common elements and add them to the set.

**Program/Source Code**

Here is the source code of the Java Program to Find the Union and Intersection of 2 Arrays. The program is successfully compiled and tested using IDE IntelliJ Idea in Windows 7. The program output is also shown below.

2. *// Java Program to Find the Union and Intersection of 2 Arrays.*
4. **import** java.io.BufferedReader;
5. **import** java.io.IOException;
6. **import** java.io.InputStreamReader;
7. **import** java.util.Arrays;
8. **import** java.util.HashSet;
9. **import** java.util.Set;
11. **public** **class** UnionAndIntersection {
12. *// Function to find and display the union and intersection*
13. **static** **void** displayUnionAndIntersection(**int**[] arrayOne,**int**[] arrayTwo){
14. Set<Integer> obj = **new** HashSet<>();
15. **int** i,j;
16. **for**(i=0; i<arrayOne.length; i++){
17. obj.add(arrayOne[i]);
18. }
19. **for**(j=0; j<arrayTwo.length; j++){
20. obj.add(arrayTwo[j]);
21. }
22. System.out.println("The union of both the arrays is");
23. **for**(Integer I: obj){
24. System.out.print(I + " ");
25. }
26. System.out.println();
27. obj.clear();
28. System.out.println("The intersection of both the arrays is");
29. **for**(i=0; i<arrayOne.length; i++){
30. obj.add(arrayOne[i]);
31. }
32. **for**(j=0; j<arrayTwo.length; j++){
33. **if**(obj.contains(arrayTwo[j]))
34. System.out.print(arrayTwo[j] + " ");
35. }
36. }
37. *// Function to read the input*
38. **public** **static** **void** main(String[] args) {
39. BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.in));
40. **int** m,n;
41. System.out.println("Enter the size of the two arrays");
42. **try** {
43. n = Integer.parseInt(br.readLine());
44. m = Integer.parseInt(br.readLine());
45. }
46. **catch** (IOException e)
47. {
48. System.out.println("Invalid input");
49. **return**;
50. }
51. **int**[] arrayOne = **new** **int**[n];
52. **int**[] arrayTwo = **new** **int**[m];
53. System.out.println("Enter the first array elements");
54. **int** i,j;
55. **for**(i=0; i<arrayOne.length; i++){
56. **try** {
57. arrayOne[i] = Integer.parseInt(br.readLine());
58. }
59. **catch** (IOException e)
60. {
61. System.out.println("Invalid array element. Enter it again");
62. i--;
63. }
64. }
65. System.out.println("Enter the second array elements");
66. **for**(i=0; i<arrayTwo.length; i++){
67. **try** {
68. arrayTwo[i] = Integer.parseInt(br.readLine());
69. }
70. **catch** (IOException e)
71. {
72. System.out.println("Invalid array element. Enter it again");
73. i--;
74. }
75. }
76. displayUnionAndIntersection(arrayOne,arrayTwo);
77. }
78. }

**Program Explanation**

1. In function displayUnionAndIntersection(), a hash set object is created, which follows all the properties of a set.  
2. The loops for(j=0; j<arrayTwo.length; j++) and for(i=0; i<arrayOne.length; i++) adds the elements of the two arrays to the HashSet. An element if already present in the HashSet is not added automatically.  
3. The loop for(Integer I: obj) is a for-each loop and it displays the union of the two arrays. Next, the HashSet is cleared.  
4. Now the elements of the arrayOne are again added to the HashSet.  
5. The last loop for(j=0; j<arrayTwo.length; j++) displays the intersection of the two arrays.

**Time Complexity: O(n + m)** where n is the number of elements in the first array and m is the number of elements in the second array.

**Runtime Test Cases**

Case 1 (Simple Test Case):

Enter the size of the two arrays

5

5

Enter the first array elements

1

2

3

4

5

Enter the second array elements

5

3

6

7

9

The union of both the arrays is

1 2 3 4 5 6 7 9

The intersection of both the arrays is

5 3

Case 2 (Simple Test Case - entirely different arrays):

Enter the size of the two arrays

4

4

Enter the first array elements

1

2

3

4

Enter the second array elements

6

7

8

9

The union of both the arrays is

1 2 3 4 6 7 8 9

The intersection of both the arrays is

Case 3 (Simple Test Case - entirely same arrays):

Enter the size of the two arrays

4

4

Enter the first array elements

1

2

3

4

Enter the second array elements

1

2

3

4

The union of both the arrays is

1 2 3 4

The intersection of both the arrays is

1 2 3 4

Java Program to Accept the Marks of a Student into a 1D Array and find Total Marks and Percentage

This is a Java Program to Accept the Marks of a Student into a 1D Array and find Total Marks and Percentage. A one-dimensional array is, essentially, a list of like-typed variables.

Enter the number of subjects and then enter marks if students in all those subjects. Now we us for loop to calculate total marks of the student and hence divide it by the total number of subjects to get percentage marks.

Here is the source code of the Java Program to Accept the Marks of a Student into a 1D Array and find Total Marks and Percentage. The Java program is successfully compiled and run on a Windows system. The program output is also shown below.

1. **import** java.util.Scanner;
2. **public** **class** Student\_Marks
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** n, total = 0, percentage;
7. Scanner s = **new** Scanner(System.in);
8. System.out.print("Enter no. of subject:");
9. n = s.nextInt();
10. **int** marks[] = **new** **int**[n];
11. System.out.println("Enter marks out of 100:");
12. **for**(**int** i = 0; i < n; i++)
13. {
14. marks[i] = s.nextInt();
15. total = total + marks[i];
16. }
17. percentage = total / n;
18. System.out.println("Sum:"+total);
19. System.out.println("Percentage:"+percentage);
20. }
21. }

Output:

$ javac Student\_Marks.java

$ java Student\_Marks

Enter no. of subject:5

Enter marks out of 100:

86

89

91

82

78

Sum:426

Percentage:85