AIM

To write a Java program to access and display an element at a given index of an array entered by the user, with proper index validation.

ALGORITHM

1. Start
2. Declare an array
3. Read the index from the user
4. Check if the index is valid ($0 \leq$ index $<$ length of array)
5. If valid, print the element at that index
6. Else, print “Invalid index”
7. Stop

PROCEDURE

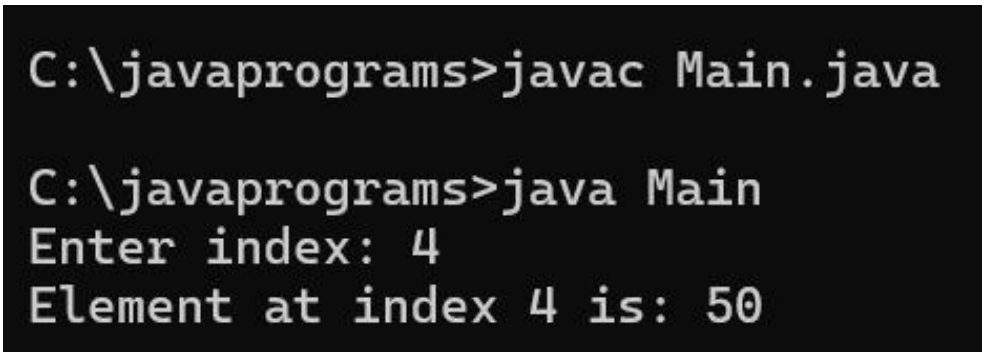
1. Initialize the array with elements
2. Take input for the index
3. Use condition to validate index
4. Access the element using arr[index]
5. Display the result

PROGRAM

```
import java.util.Scanner;



public class Main {

    public static void main(String[] args) {

        int[] arr = {10, 20, 30, 40, 50};



        Scanner sc = new Scanner(System.in);

        System.out.print("Enter index: ");

        int index = sc.nextInt();
```

```
        if (index >= 0 && index < arr.length) {
            System.out.println("Element at index " + index + " is: " + arr[index]);
        } else {
            System.out.println("Invalid index!");
        }
    }
}
```

OUTPUT



RESULT

The program successfully accesses and prints the element at the specified index when a valid index is provide.

AIM

To write a Java program to search for a given element in a sorted array using the Binary Search technique.

ALGORITHM

1.Start
   a. Declare a sorted array
   b. Input the search element (key)
   c. Set low = 0 and high = n - 1
2. While low ≤ high:
3. Find mid = (low + high) / 2
4. If arr[mid] == key, print index and stop
5. If arr[mid] < key, set low = mid + 1
6. Else set high = mid - 1
7. If not found, print “Element not found”
8. Stop

PROCEDURE

1. Input the sorted array
2. Input the element to search
3. Apply binary search logic
4. Compare mid element with key
5. Repeat until found or range becomes zero
6. Display the result

PROGRAM

```
import java.util.Scanner;


public class BinarySearchExample {
   public static void main(String[] args) {
      int[] arr = {10, 20, 30, 40, 50, 60, 70};
```

```
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter element to search: ");
        int key = sc.nextInt();

        int low = 0;
        int high = arr.length - 1;
        boolean found = false;

        while (low <= high) {
            int mid = (low + high) / 2;

            if (arr[mid] == key) {
                System.out.println("Element found at index: " + mid);
                found = true;
                break;
            } else if (arr[mid] < key) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }

        if (!found) {
            System.out.println("Element not found.");
        }
    }
}
```

OUTPUT

```
C:\javaprograms>javac BinarySearchExample.java

C:\javaprograms>java BinarySearchExample
Enter element to search: 50
Element found at index: 4
```

RESULT

The program efficiently searches for a given element in a sorted array using the Binary Search method and displays the index if the element is found.

AIM

To write a Java program to find the Kth smallest element in an array of integers.

ALGORITHM

1. Start
2. Read n (number of elements)
3. Input array elements
4. Input K
5. Sort the array
6. Print element at index K-1
7. Stop

PROCEDURE

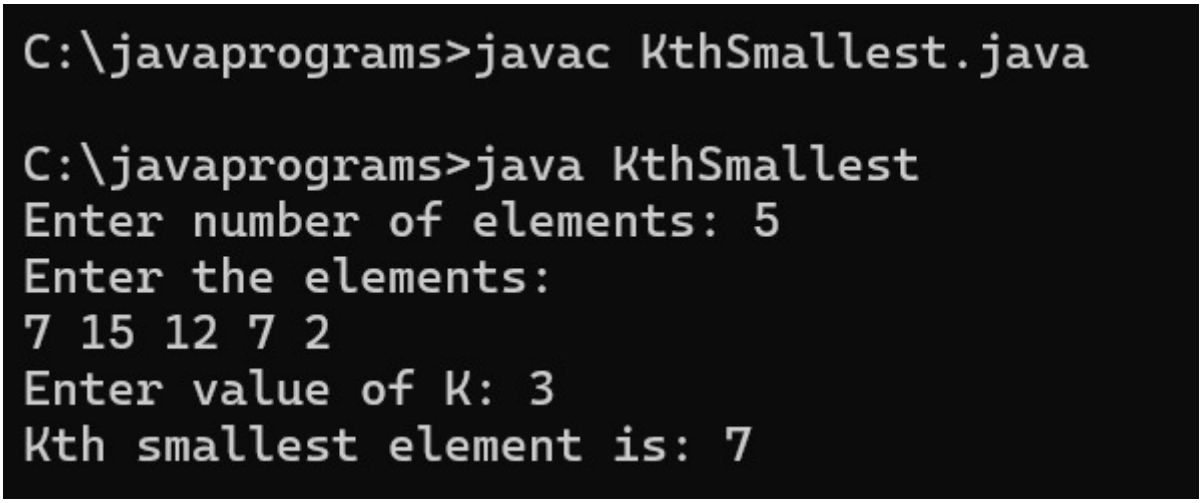
1. Input array size
2. Input elements
3. Input value of K
4. Sort the array in ascending order
5. Access element at position K-1
6. Display the result

PROGRAM

```
import java.util.Scanner;

import java.util.Arrays;



public class KthSmallest {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);



        System.out.print("Enter number of elements: ");
```

```
        int n = sc.nextInt();

        int[] arr = new int[n];

        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        System.out.print("Enter value of K: ");
        int k = sc.nextInt();

        if (k > 0 && k <= n) {
            Arrays.sort(arr);
            System.out.println("Kth smallest element is: " + arr[k - 1]);
        } else {
            System.out.println("Invalid value of K!");
        }
    }
}
```

OUTPUT



RESULT

The program successfully determines the Kth smallest element in the given array.If the entered value of K is invalid, the program displays an appropriate error message.

AIM

To write a Java program to find and display the maximum element in a given array of integers.

ALGORITHM

1. Start
2. Read n (number of elements)
3. Input array elements
4. Set max = first element
5. Traverse the array
6. If current element > max, update max
7. After traversal, print max
8. Stop

PROCEDURE

1. Input size of array
2. Input elements
3. Assume first element as maximum
4. Compare with all other elements
5. Update max when larger value is found
6. Print the maximum value

PROGRAM

```
import java.util.Scanner;

public class MaxElementArray {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter number of elements: ");
      int n = sc.nextInt();

      int[] arr = new int[n];
```

```
        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        int max = arr[0];

        for (int i = 1; i < n; i++) {
            if (arr[i] > max) {
                max = arr[i];
            }
        }

        System.out.println("Maximum element is: " + max);
    }
}
```

OUTPUT

```
C:\javaprograms>javac MaxElementArray.java

C:\javaprograms>java MaxElementArray
Enter number of elements: 3
Enter the elements:
10 45 85
Maximum element is: 85
```

RESULT

The program successfully identifies and prints the maximum element present in the given array of integers.

AIM

To write a Java program to print all possible unique pairs of elements from a given array.

ALGORITHM

1.Start

2.Read n (number of elements)

3.Input array elements

4.Use two loops:

- Outer loop from i = 0 to n-1
- Inner loop from j = i+1 to n-1

5.Print pair (arr[i], arr[j])

6.Stop

PROCEDURE

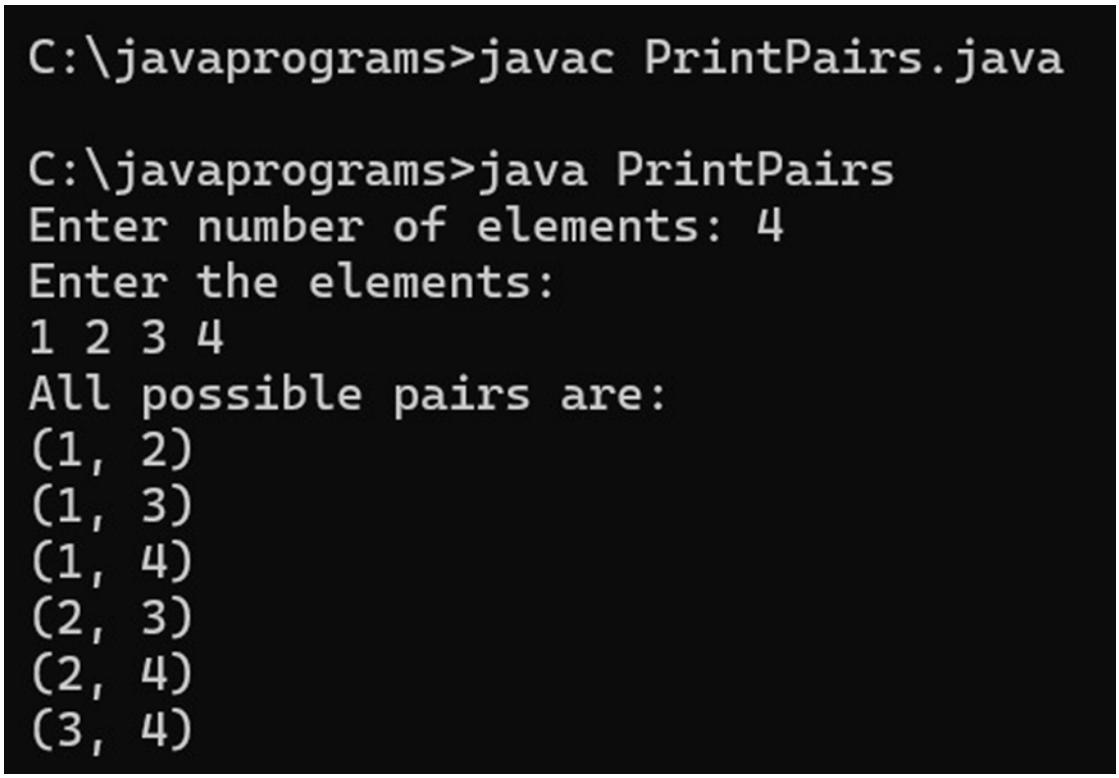
1. Input array size
2. Input elements
3. Use nested loops
4. Form pairs using two different indices
5. Print each pair
6. Continue until all pairs are printed
7. Display each unique pair
8. Close the Scanner object

PROGRAM

```
import java.util.Scanner;



public class PrintPairs {

    public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of elements: ");
        int n = sc.nextInt();

        int[] arr = new int[n];

        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        System.out.println("All possible pairs are:");

        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
                System.out.println("(" + arr[i] + ", " + arr[j] + ")");
            }
        }
    }
}
```

OUTPUT



RESULT

The program successfully **prints all possible unique pairs of elements** present in the given array.