Task : 1

Aim :

To Access and print the element at a given index in an array using java.

Algorithms :

Start

1. Input the array (list of elements).
2. Input the index (position of the element you want to access).
3. Check validity of index:

   If index < 0 OR index >= array length → print "Invalid index".

   Else → continue.
4. Access the element at the given index.
5. Print the element.
6. End.

Program :

```
public class AccessArrayElement {
   public static void main(String[] args) {
      int[] numbers = {10, 20, 30, 40, 50};
      int index = 2;
     if (index >= 0 && index < numbers.length) {
         System.out.println("Element at index " + index + " is: " + numbers[index]);
      } else {
    System.out.println("Invalid index! Please choose between 0 and " + (numbers.length - 1));
      }
   }
}
```

Out put :

Result :

Thus the implementation of Access and print the element at a given index in an array using java was executed successfully.

Task : 2

Aim :

To Search for a given element in a sorted array using Binary Search.

Algorithms :

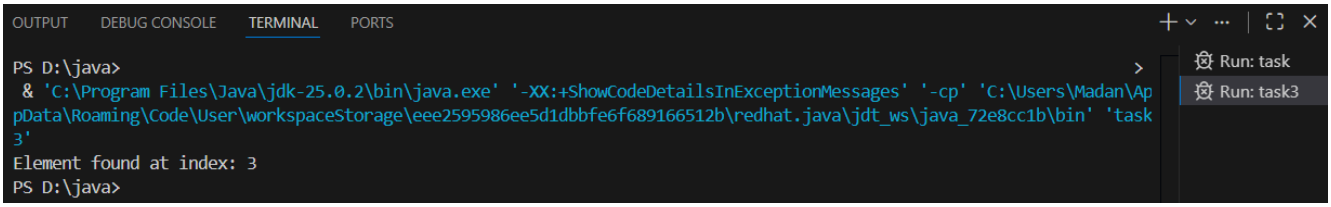
1. Start
2. Input: a sorted array A and the element key to search
3. Initialize low = 0 and high = n - 1 (where n is the array size)
4. Repeat while low <= high:
    a. Calculate mid = (low + high) / 2
    b. If A[mid] == key → return mid (element found)
    c. Else if A[mid] < key → set low = mid + 1 (search right half)
    d. Else → set high = mid - 1 (search left half)
5. If loop ends without finding → return -1 (element not found)
6. End.

Program :

```
public class BinarySearchShort {
  public BinarySearchShort() {
  }
  public static int binarySearch(int[] var0, int var1) {
    int var2 = 0;
    int var3 = var0.length - 1;
    while(var2 <= var3) {
      int var4 = (var2 + var3) / 2;
      if (var0[var4] == var1) {
        return var4;
      }
```

```
        if (var0[var4] < var1) {
            var2 = var4 + 1;
        } else {
            var3 = var4 - 1;
        }
    }
    return -1;
  }
  public static void main(String[] var0) {
    int[] var1 = new int[]{10, 20, 30, 40, 50};
    byte var2 = 40;
    int var3 = binarySearch(var1, var2);
    System.out.println(var3 != -1 ? "Found at index " + var3 : "Not found");
  }
}
```

Output :



Result :

Thus the implementation of Search for a given element in a sorted array using Binary Search was executed successfully.

Task 3:

Aim :

To Find the maximum element in an array of n integers using java.

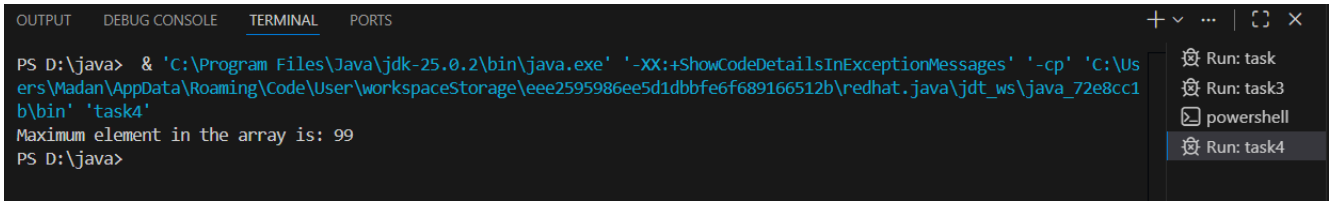
Algorithm :

1. Start
2. Input: an array A of size n
3. Initialize: set max = A[0] (first element as the current maximum)
4. Repeat for each element from index 1 to n-1:
    a. If A[i] > max → update max = A[i]
5. After loop ends, max holds the largest element in the array
6. Output: print max
7. End.

Program :

```
public class MaxElementInArray{
   public static void main(String[]args){
      int[]arr={12,45,67,23,89,34};
      int n=arr.length;
      int max=arr[0];
      for(int i=1;i<n;i++){
         if (arr[i]>max){
           max = arr[i];
         }
      }
      System.out.println("Maximum element in the array is:"+max);
   }
```

Output :



Result :

Thus the implementation Find the maximum element in an array of n integers using java was executed successfully.

Task 4 :

Aim :

To Given an array of integers and a positive integer K, write a program to find:The Kth smallest element using java.

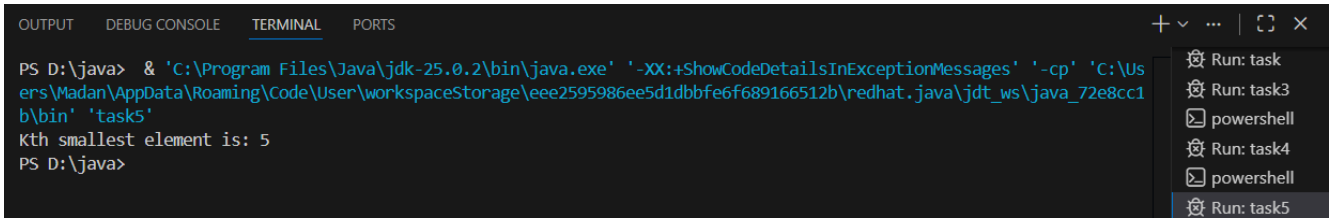
Algorithm :

1. Start
2. Input:
    a. An array A of size n
    b. A positive integer K (where $1 \leq K \leq n$)
3. Sort the array A in ascending order
4. Access the element at position K-1 (since arrays are usually 0-indexed)
5. Output: print the element as the Kth smallest
6. End.

Program :

```
import java.util.Arrays;
public class KthSmallestElement{
   public static void main(String[]args){
      int[] arr={12,3,5,7,19};
      int k = 2;
      Arrays.sort(arr);
      int KthSmallest = arr[k - 1];
      System.out.println("The"+ k +"th smallest element is :"+ KthSmallest);
   }
}
```

Output :



Result :

Thus the implementation of Given an array of integers and a positive integer K, write a program to find:The Kth smallest element using java was executed successfully.

Task 5 :

Aim :

To Print all possible pairs of elements from an array of size n using java.

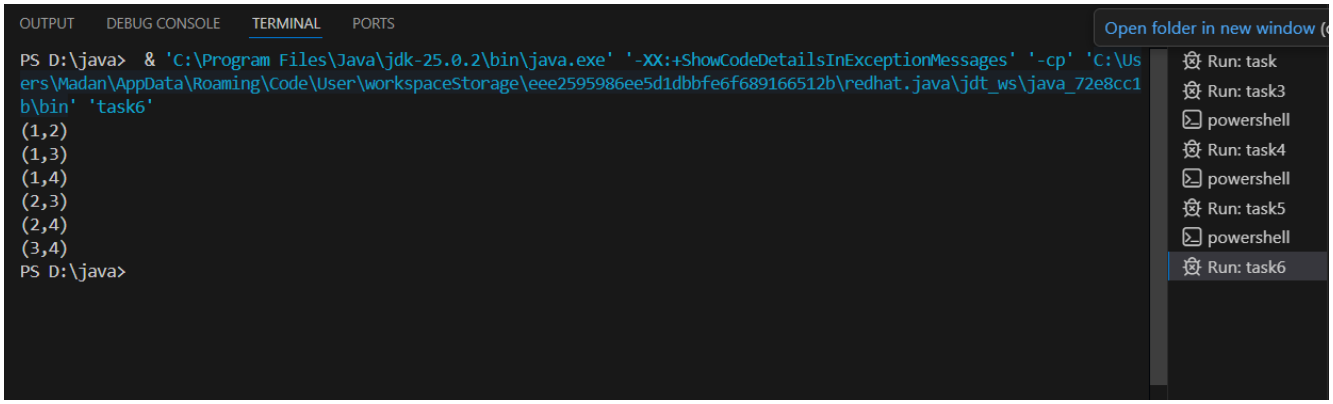
Algorithm :

1. Start
2. Input: an array A of size n
3. Repeat for each element A[i] from index 0 to n-1:
    a. For each element A[j] from index i+1 to n-1:
    b. Print the pair (A[i], A[j])
4. End.

Program :

```
public class cv {
   public static void main(String[] args) {
      int[] arr = {1, 2, 3, 4}; // Example array
      int n = arr.length;
      for (int i = 0; i < n; i++) {
         for (int j = i + 1; j < n; j++) {
            System.out.println("(" + arr[i] + ", " + arr[j] + ")");
         }
      }
   }
}
```

Output :



Result :

Thus the implementation Print all possible pairs of elements from an array of size n using java was executed successfully.

Task 6 :

Aim :

To perform the program that calculates the sum of digits of a given integer, with an option to compute either the sum of even digits or the sum of odd digits based on user choice.

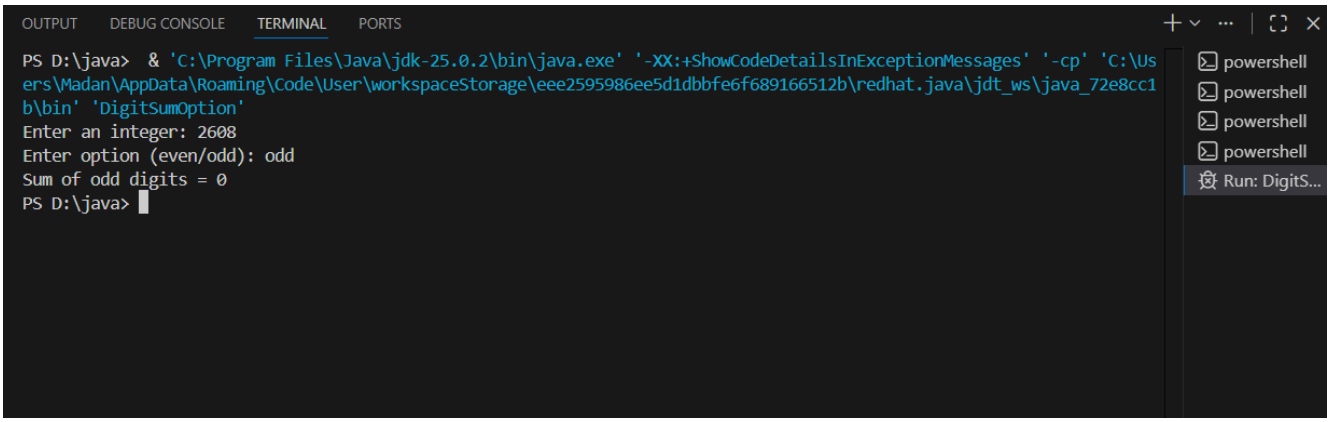
Algorithm :

1. Start
2. Input:
    a. An integer num
    b. An option opt (either "even" or "odd")
3. Initialize: sum = 0
4. Set a temporary variable temp = num
5. Repeat while temp > 0:
    a. Extract the last digit → digit = temp % 10
    b. If opt = "even" and digit % 2 == 0 → add digit to sum
    c. Else if opt = "odd" and digit % 2 != 0 → add digit to sum
    d. Remove the last digit → temp = temp / 10
6. After loop ends, sum contains the required digit sum
7. Output: print "Sum of even/odd digits = sum"
8. End

Program :

```
 import java.util.Scanner;
public class DigitSumOption {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        int num = sc.nextInt();
        System.out.print("Enter option (even/odd): ");
        String opt = sc.next().toLowerCase();
        int sum = 0;
        int temp = num;
        while (temp > 0) {
            int digit = temp % 10;
            if (opt.equals("even") && digit % 2 == 0) {
                sum += digit;
            } else if (opt.equals("odd") && digit % 2 != 0) {
                sum += digit;
            }
            temp /= 10;
        }
        System.out.println("Sum of " + opt + " digits = " + sum);
        sc.close();
    }
}
```

Output :



Result :

Thus the implementation calculates the sum of digits of a given integer, with an option to compute either the sum of even digits or the sum of odd digits based on user choice was executed successfully.

Task 7:

Aim :

To implement the Nth Fibonacci using java.

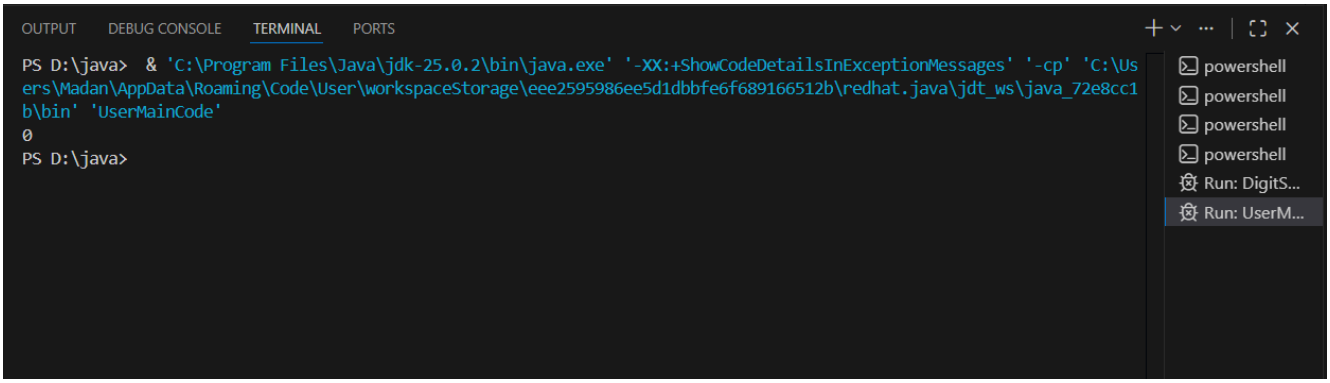
Algorithm :

1. Start
2. Input: an integer n (the position of the Fibonacci number to find)
3. Check base cases:
    a. If n == 1 → return 0
    b. If n == 2 → return 1
4. Initialize variables:
    a. a = 0 (first Fibonacci number)
    b. b = 1 (second Fibonacci number)
    c. c = 0 (temporary variable for next Fibonacci number)
5. Repeat from i = 3 to n:
    a. Compute c = a + b
    b. Update a = b
    c. Update b = c
6. After loop ends, b holds the Nth Fibonacci number
7. Return b
8. END.

Program :

```
public class UserMainCode {
    public static long nthFibonacci(int input1) {
        if(input1 == 1) return 0;
        if(input1 == 2) return 1;
        long a = 0, b = 1, c = 0;
        for(int i = 3; i <= input1; i++) {
            c = a + b;
            a = b;
            b = c;
        }
        return b;
    }
    public static void main(String[] args) {
        int n = 1;
        System.out.println(nthFibonacci(n));
    }
}
```

Output :



Result :

Thus the implementation Nth Fibonacci using java was executed successfully.

Task 8 :

Aim :

To implement the isPalindrome Number using java.

Algorithm :

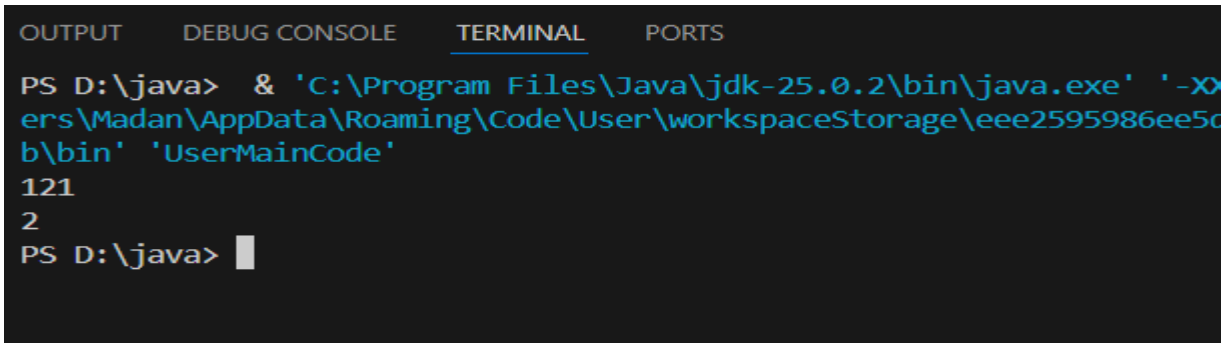
1. Start
2. Input: an integer n
3. Store original number → original = n
4. Initialize → reverse = 0
5. Repeat while n > 0:
    a. Extract last digit → digit = n % 10
    b. Update reverse → reverse = reverse * 10 + digit
    c. Remove last digit → n = n / 10
6. After loop ends:
    a. If original == reverse → return 2 (number is palindrome)
    b. Else → return 1 (number is not palindrome)
7. Output: print result
8. End.

Program :

```
import java.util.*;
```

```
class UserMainCode
{
    public int isPalinNum(int input1){
        int original = input1;
        int reverse = 0;
        while (input1 > 0) {
            int digit = input1 % 10;
            reverse = reverse * 10 + digit;
            input1 = input1 / 10;
        }
        if (original == reverse)
            return 2;
        else
            return 1;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        UserMainCode obj = new UserMainCode();
        int result = obj.isPalinNum(n);
        System.out.println(result);
    }
}
```

Output :



Result :

Thus the implementation isPalindrome Number using java was executed successfully.

Task 9 :

Aim :

To implement Sum of last digit of two given numbers using java .

Algorithm :

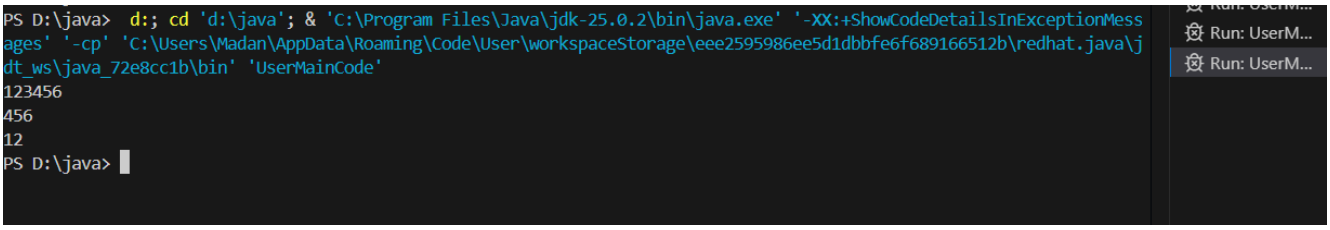
1. Start
2. Read two integers input1 and input2
3. If input1 is negative, convert it to positive
4. If input2 is negative, convert it to positive
5. Find the last digit of input1 using input1 % 10
6. Find the last digit of input2 using input2 % 10
7. Add both last digits
8. Return the sum
9. Stop

Program :

```
import java.util.*;

class UserMainCode

{

   public int addLastDigits(int input1, int input2){

      if(input1 < 0){

         input1 = -input1;

      }

      if(input2 < 0){

         input2 = -input2;

      }

      int lastdigit1 = input1 % 10;

      int lastdigit2 = input2 % 10;

      return lastdigit1 + lastdigit2

}

   public static void main(String[] args){

      Scanner sc = new Scanner(System.in);

      int a = sc.nextInt();

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

```
        UserMainCode obj = new UserMainCode();
        System.out.println(obj.addLastDigits(a,  b));
    }
}
```

Output :



Result :

Thus the implementation Sum of last digit of two given numbers using java was executed successfully.