Task 1:

Aim: To check whether a given integer number is even or odd and return

Algorithm:

1 Start

2 Read an integer n

3 Check if n % 2 == 0

- If true, return 1 (Even)
- Else, return 0 (Odd)

4 Print the returned value

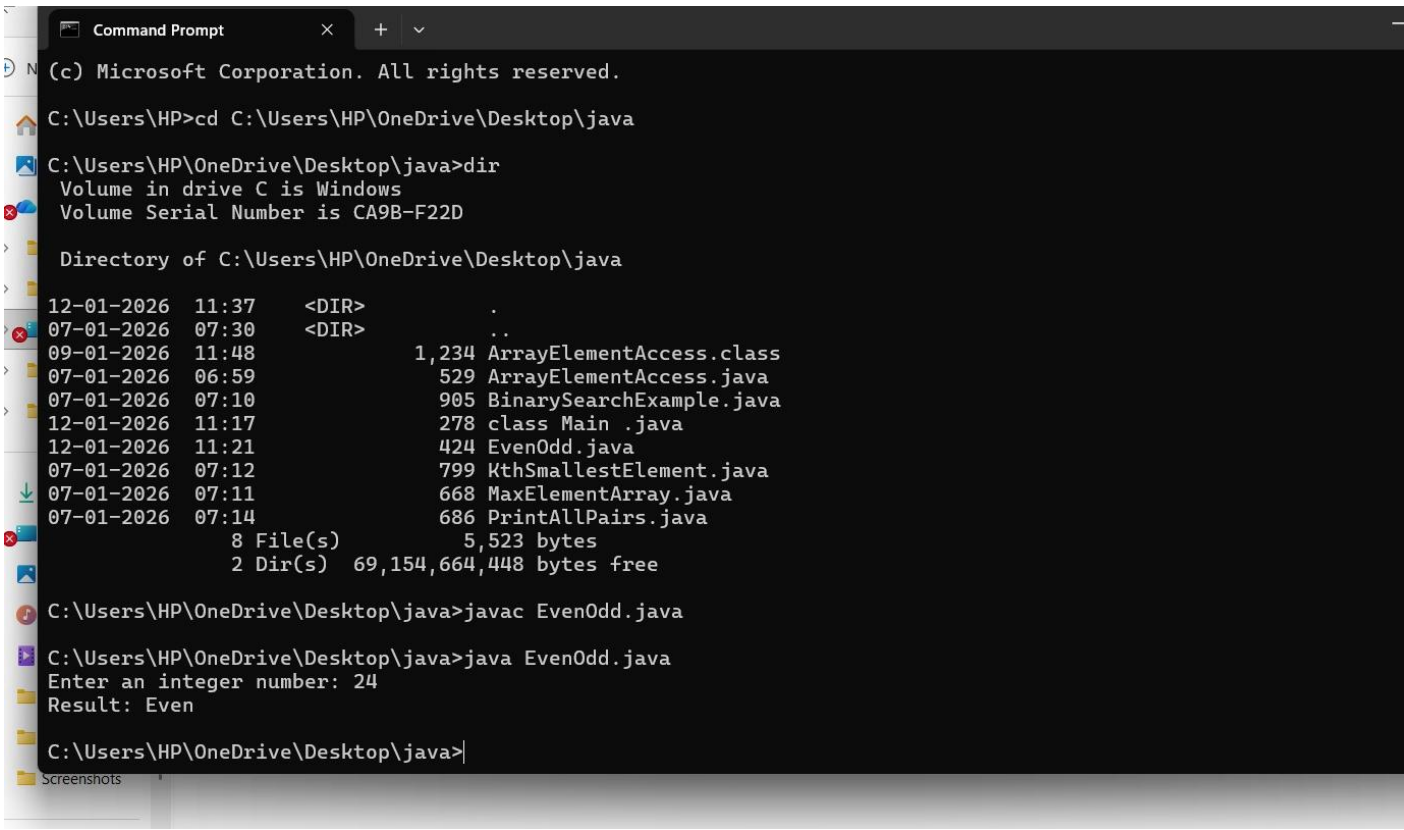
5 Stop

Program:

```
import java.util.*;
public class Main {
   static int isEven(int n) {
      if (n % 2 == 0)
         return 1;
      else
 return 0;
   }

            public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      System.out.print(isEven(n));
      sc.close();
   }
}
```

Output:



Result: Thus finding the Even or odd is shown sucssesfully

Task2:

Aim: To access and print the element present at a given index in an array**.**

Algorithm:

1 Start

2 Read the size of the array n

3 Read n elements into the array

4 Read the index value index

5 Check whether index is valid

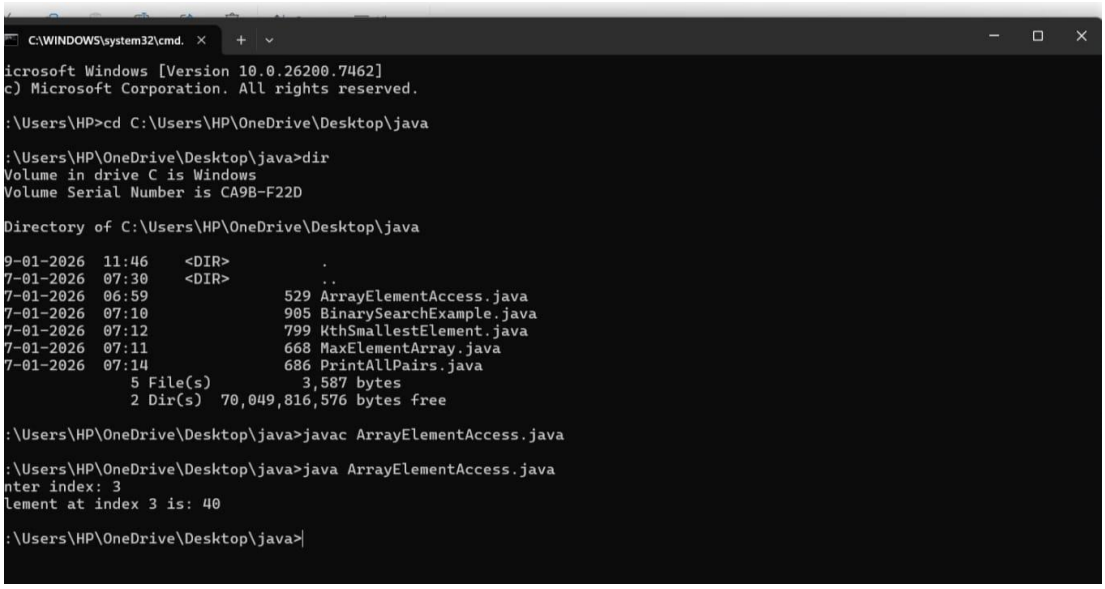
- If index ≥ 0 and index < n, print the element at that index
- Else, print an error message (invalid index)

6 Stop

Program:

```
import java.util.Scanner;
public class ArrayIndexAccess {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
  int[] arr = {10, 20, 30, 40, 50};
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!");
        }
sc.close();
    }
```

Output:



Result: Thus finding the element at index was shown sucessfully

Task3:

Aim: To search for a given element in a sorted array using Binary Search technique**.**

Algorithm:

1 Start

2 Read the number of elements n

3 Read the n elements of the sorted array

4 Read the element to be searched key

5 Initialize low = 0 and high = n − 1

6 Repeat while low ≤ high

1. Calculate mid = (low + high) / 2
2. If array[mid] == key, print the position and stop
3. If key < array[mid], set high = mid − 1
4. Else, set low = mid + 1

7 If the element is not found, print “Element not found”

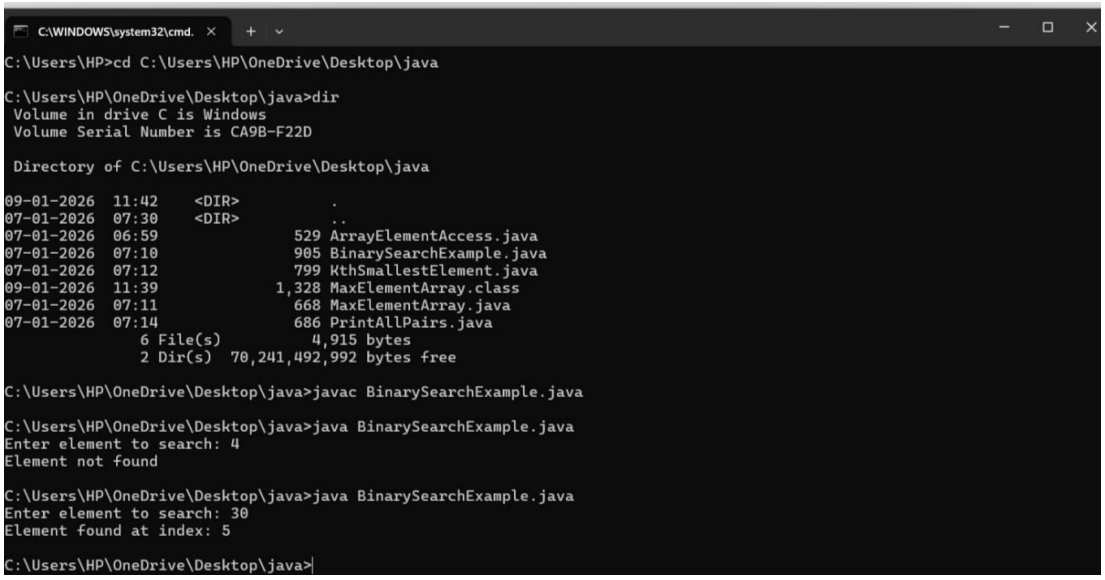
8 Stop

Program:

```
import java.util.Scanner;
public class BinarySearchExample {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
int[] arr = {10, 20, 30, 40, 50, 60, 70};
 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");
        }
 sc.close();
    }
}
```

Output:



Result: Thus finding the element in sorted array using binary search is shown successfully

Task4:

Aim: To find the maximum element in an array of n integers.

Algorithm:

1 Start

2 Read the number of elements n

3 Read n integers into the array

4 Assume the first element of the array as maximum

- max = array[0]

5 Compare max with each remaining element of the array

- If any element is greater than max, update max

6 After completing the loop, max contains the largest element

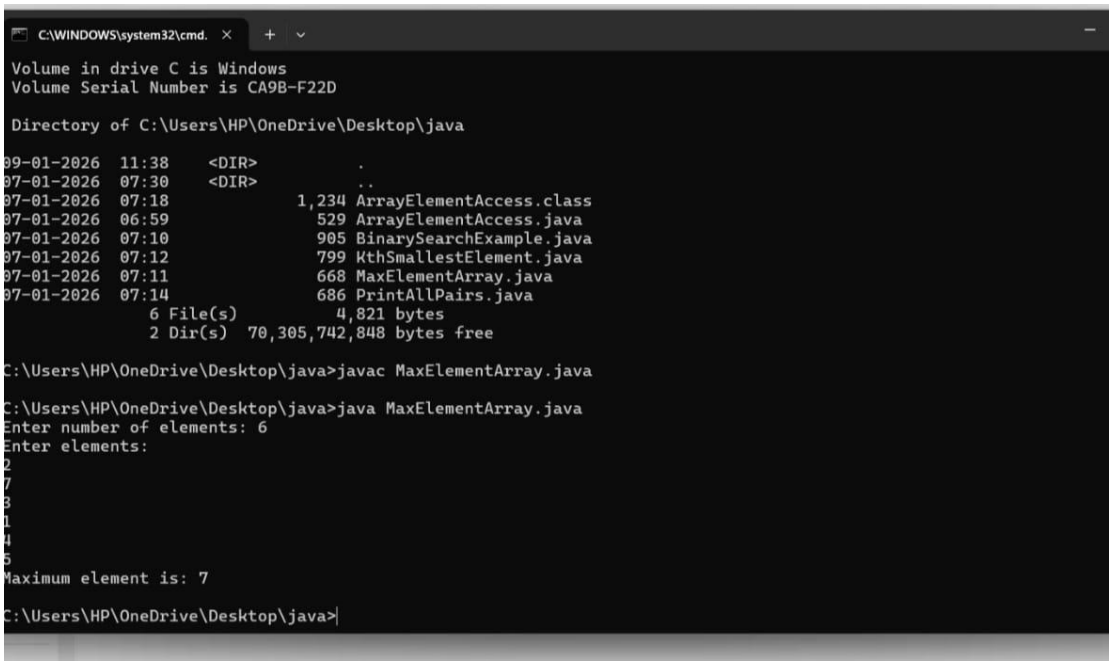
7 Print the value of max

8 Stop

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 array elements:");
      for (int i = 0; i < n; i++) {
         arr[i] = sc.nextInt();
      }
int max = arr[0];   // assume first element is maximum
 for (int i = 1; i < n; i++) {
         if (arr[i] > max) {
```

Output:



Result:Thus finding the max element was shown successfully

Task5:

Aim: To find the Kth smallest element in a given array of integers**.**

Algorithm:

1 Start

2 Read the number of elements n

3 Read n integers into the array

4 Read the positive integer k

5 Sort the array in **ascending order**

6 The element at index k − 1 is the **Kth smallest element**

7 Print the Kth smallest element

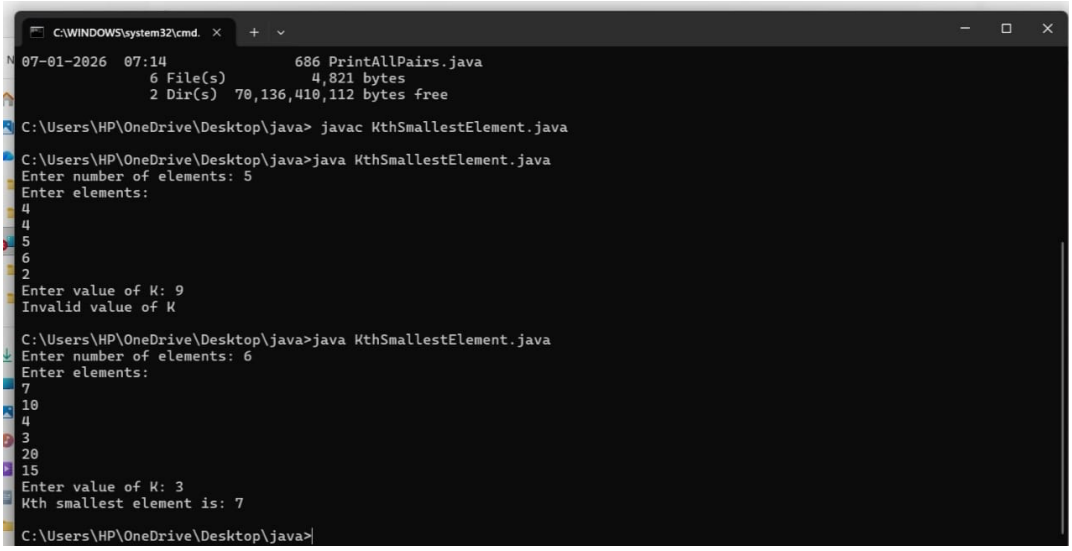
8 Stop

Program:

```
import java.util.Arrays;
import java.util.Scanner;
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 array elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
```

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

Output:



Result: Thus Finding the kth smallest element was shown successfully

Task6:

Aim: To print all possible pairs of elements from an array of size n.

Algorithm:

1 Start

2 Read the number of elements n

3 Read n elements into the array

4 Use two nested loops to generate pairs

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

5 Print the pair (array[i], array[j])

6 Repeat until all possible pairs are printed

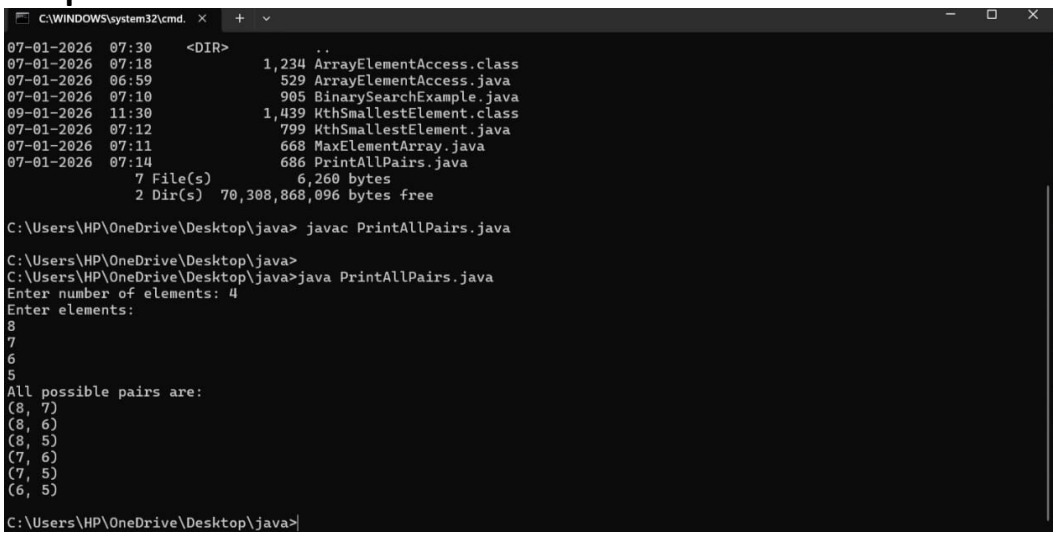
7 Stop

Program:

```
import java.util.Scanner;
public class PrintAllPairs {
   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 array 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] + ")");
            }
        }
sc.close();
    }
}
```

Output:



Result:Thus Finding the All pairs of elements was shown Sucessfully

Task7:

Aim: To calculate the sum of even digits or odd digits of a given number based on the user's choice.

Algorithm:

1 Start

2 Read an integer input1 and a string input2

3 Initialize sum = 0

4 Repeat while input1 > 0to

1. Extract the last digit using digit = input1 % 10
2. If input2 is "even" and digit is even, add digit to sum
3. If input2 is "odd" and digit is odd, add digit to sum
4. Remove the last digit using input1 = input1 / 10

5 Return sum

6 Stop

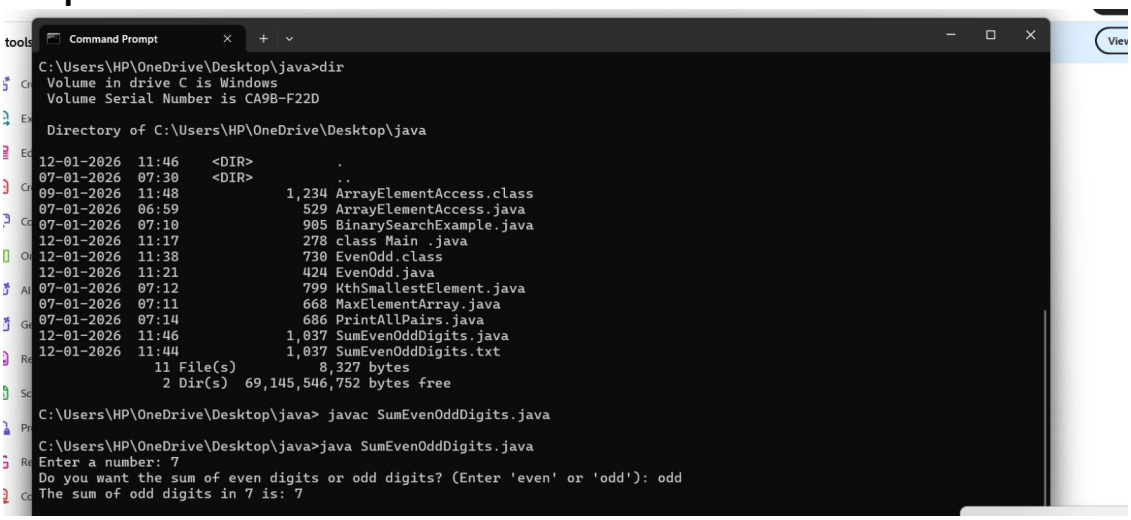
Program:

```
class UserMainCode
{
   public int evenodddigitSum(int input1, String input2)
   {
      int sum = 0;
while (input1 > 0)
      {
         int digit = input1 % 10;

         if (input2.equalsIgnoreCase("even") && digit % 2 == 0)
         {
            sum += digit;
```

```
        }
        else if (input2.equalsIgnoreCase("odd") && digit % 2 != 0)
        {
            sum += digit;
        }
input1 = input1 / 10;
      }
return sum;
   }
}
```

Output:



Result:Thus to calculate sum of even digits was shown sucessfully

Task8:

Aim: To find the N^{th} Fibonacci number using an iterative approach**.**

Algorithm:

1. Start
2. Read the integer value input1 (position of Fibonacci series)
3. If input1 is equal to 1, return 0
4. If input1 is equal to 2, return 1
5. Initialize two variables:
    - a = 0 (first Fibonacci number)
    - b = 1 (second Fibonacci number)
6. Repeat the following steps from i = 3 to input1:

   Compute c = a + b

   Assign a = b

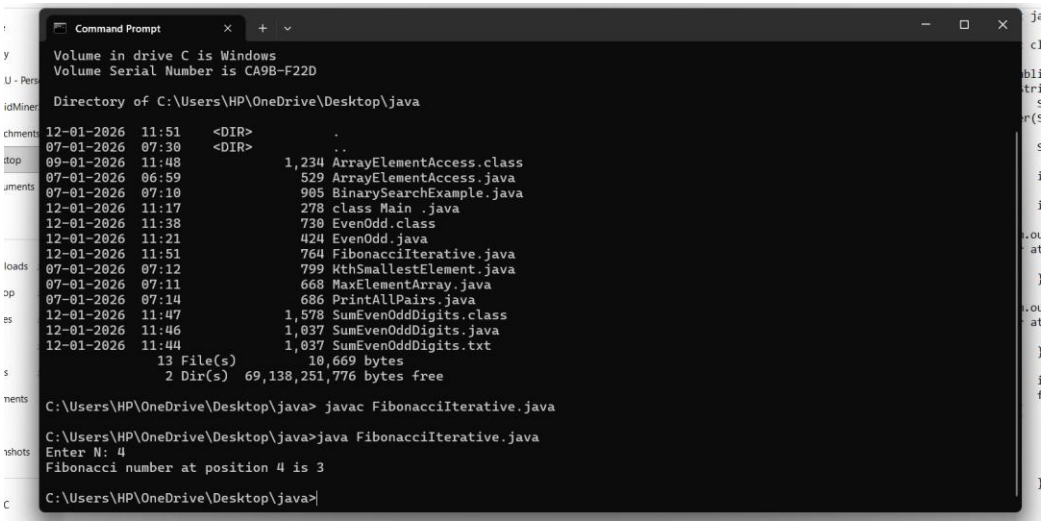
   Assign b = c
7. After the loop ends, return the value of b as the N^{th} Fibonacci number
8. Stop

Program:

```
class UserMainCode
{
   public int nthFibonacci(int input1)
   {
      if (input1 == 1)
         return 0;
      if (input1 == 2)
         return 1;
```

```
        int a = 0, b = 1, c;
        for (int i = 3; i <= input1; i++)
        {
            c = a + b;
            a = b;
            b = c;
        }
        return b;
    }
}
```

Output:



Result:Thus finding the Nth Fibonaci was shown sucessfully

Task9:

AIM: To check whether a given number is a palindrome number and return

Algorithm:

1. Start
2. Read the integer value input1
3. Store the value of input1 in a variable original
4. Initialize reverse = 0
5. Repeat the following steps while input1 > 0:
    1. Extract the last digit using digit = input1 % 10
    2. Form the reversed number using reverse = reverse * 10 + digit
    3. Remove the last digit using input1 = input1 / 10
6. Compare original with reverse
    - If both are equal, return 2 (palindrome)
    - Otherwise, return 1 (not palindrome)
7. Stop

Program:

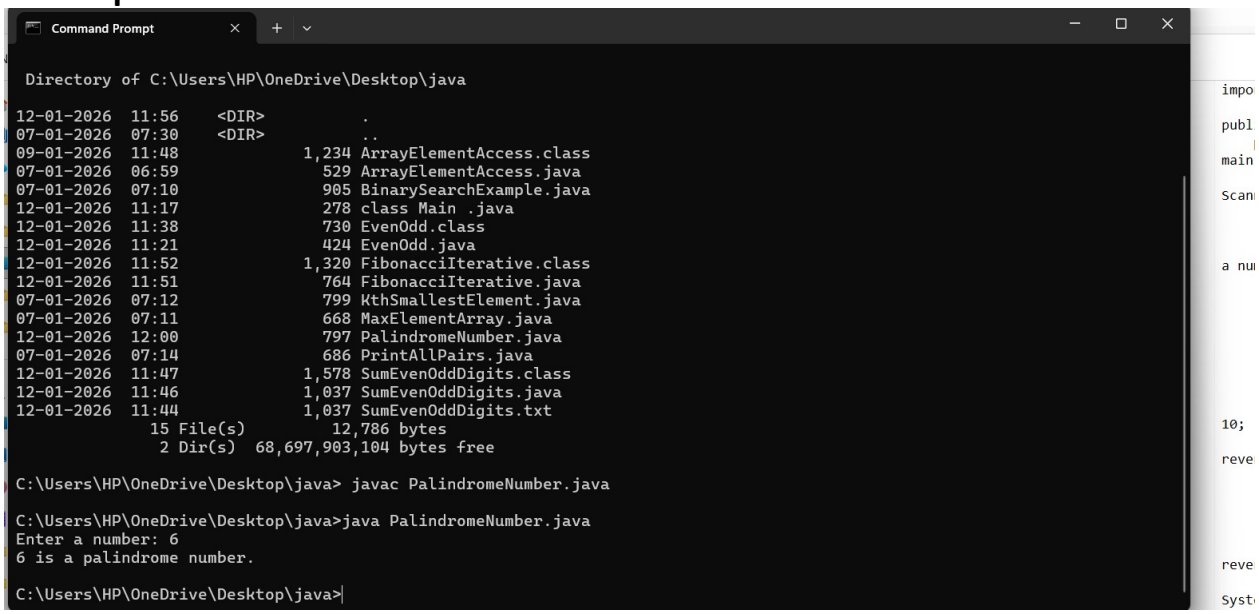
```
class UserMainCode
{
   public int isPalinNum(int input1)
   {
      int original = input1;
      int reverse = 0;

      while (input1 > 0)
      {
         int digit = input1 % 10;
```

```
            reverse = reverse * 10 + digit;
            input1 = input1 / 10;
        }

        if (original == reverse)
            return 2;  // palindrome
        else
            return 1;   // not palindrome
    }
}
```

Output:



Result:Thus checking the palindrome was shown sucessfully

Task10:

AIM: To find the sum of the last digits of two given integers**.**

Algorithm:

1. Start
2. Read two integer values input1 and input2
3. Convert both numbers to their absolute values to handle negative inputs
4. Find the last digit of the first number using last1 = input1 % 10
5. Find the last digit of the second number using last2 = input2 % 10
6. Calculate the sum of the last digits
    - sum = last1 + last2
7. Return the value of sum
8. Stop

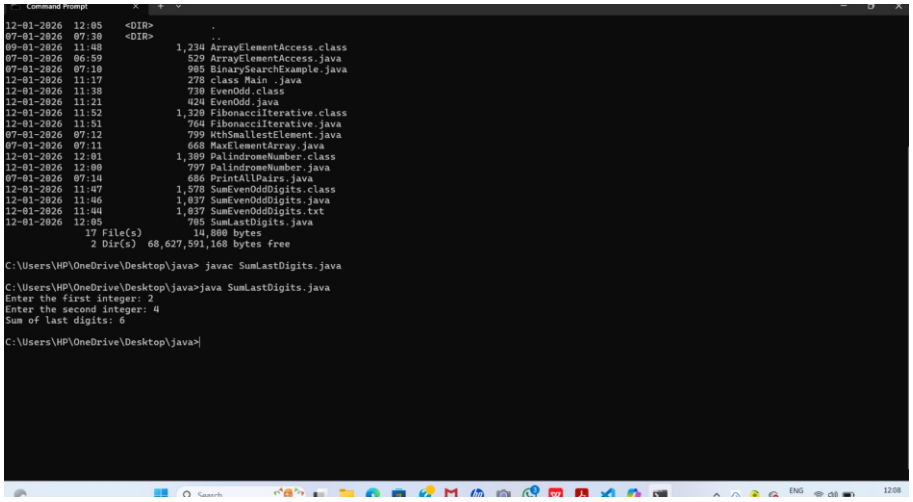
Program:

```
class UserMainCode
{
   public int addLastDigits(int input1, int input2)
   {
      input1 = Math.abs(input1);
      input2 = Math.abs(input2);

      int last1 = input1 % 10;
      int last2 = input2 % 10;

      return last1 + last2;
   }
}
```

OUTPUT:



Result: Thus sum of the last two digits was shown sucessfully