**Data structures Algorithm**

**Assignment1:**

Patterns:

import java.util.Scanner;

public class pattern1{

public static void main(String[] args) {

try (Scanner scanner = new Scanner(System.in)) {

int n = scanner.nextInt();

for(int i = 1; i<=n; i++)

{

for(int j=1;j<=i;j++)

{

System.out.print("\*");

}

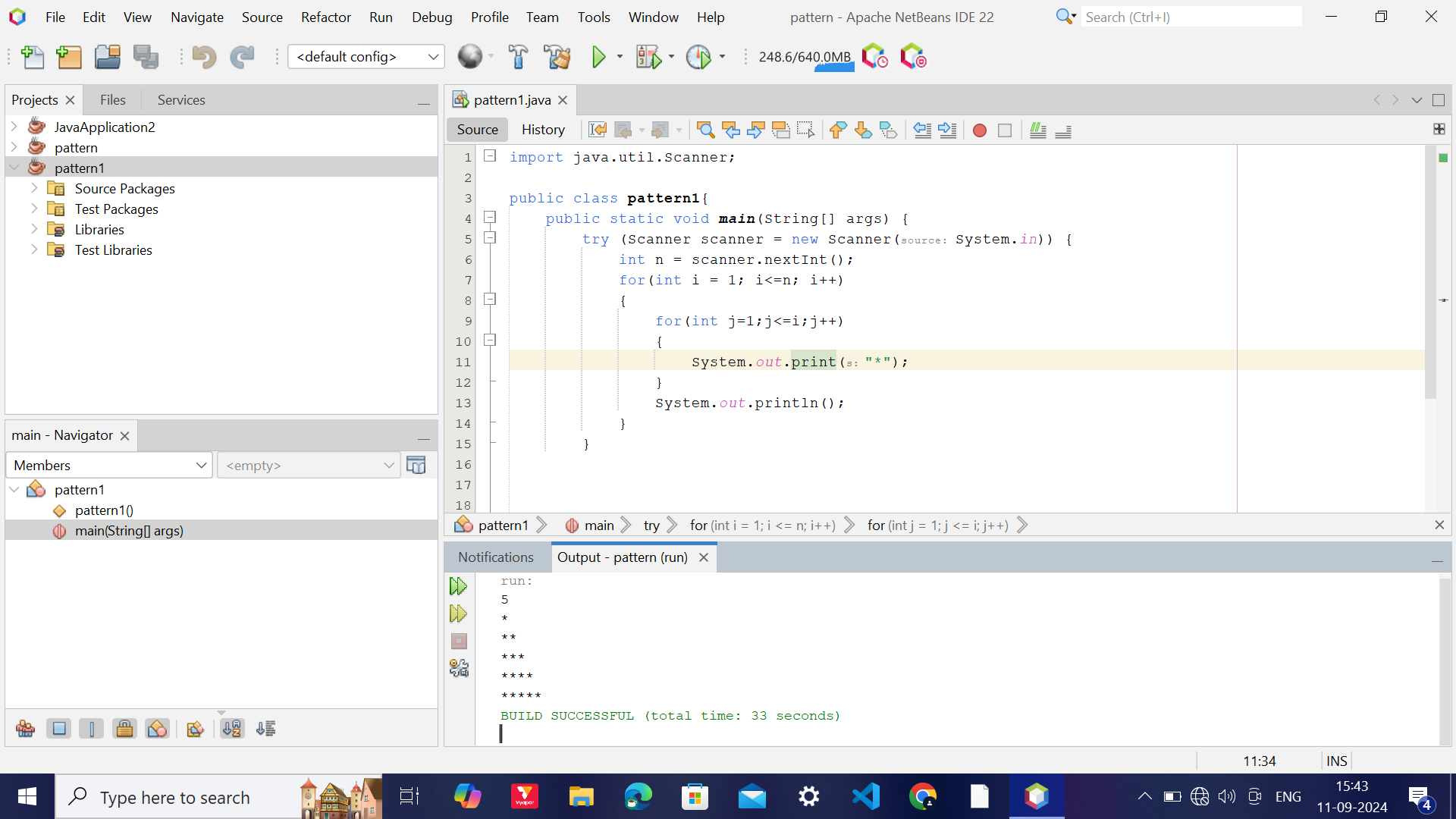
System.out.println();

}

}

}

}



import java.util.Scanner;

public class pattern2{

public static void main(String[] args) {

try (Scanner scanner = new Scanner(System.in)) {

int n = scanner.nextInt();

for(int i = 1; i<=n; i++)

{

for(int j=n;j>=i;j--)

{

System.out.print("\*");

}

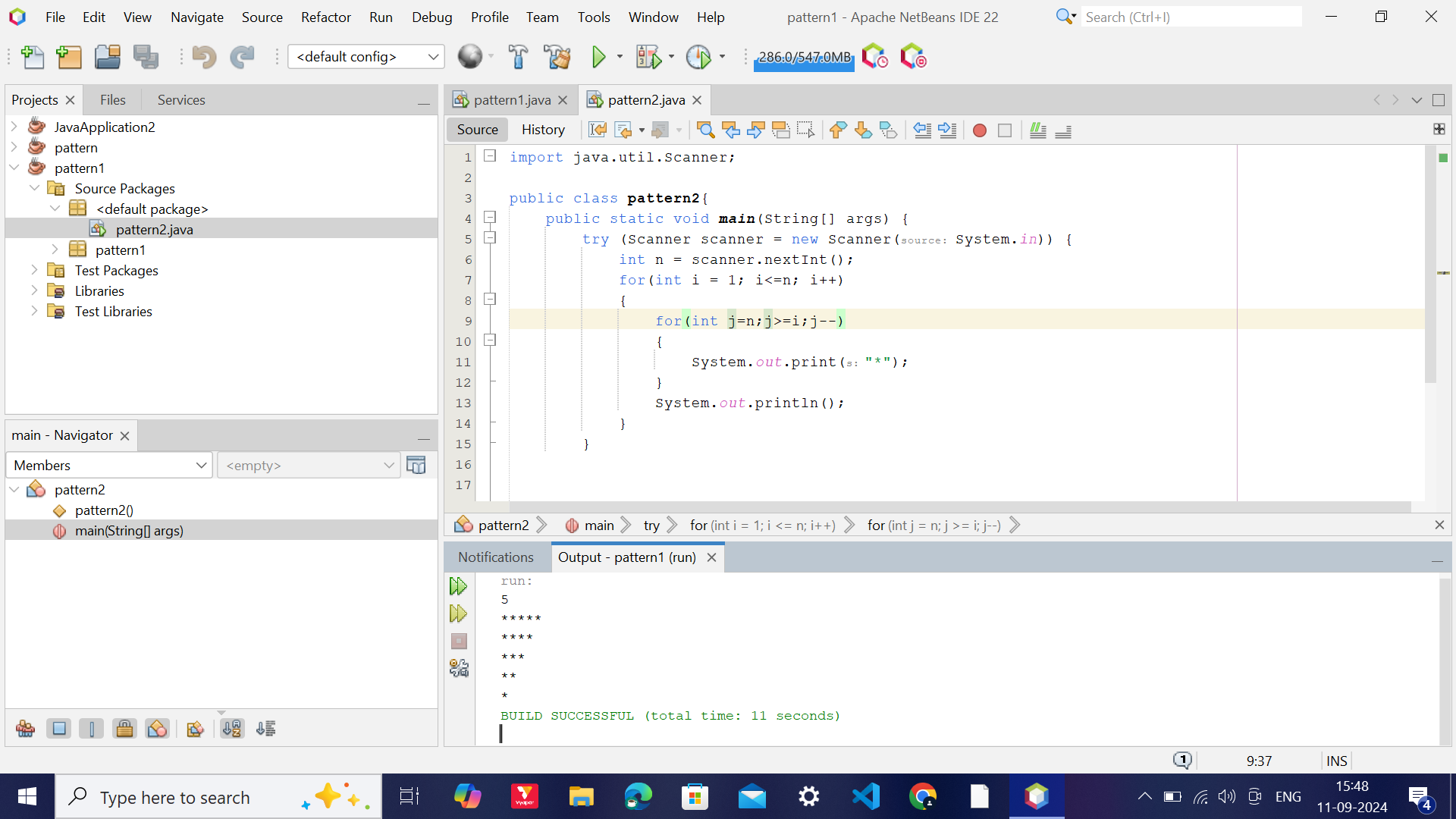
System.out.println();

}

}

}

}



import java.util.Scanner;

public class pattern3{

public static void main(String[] args) {

try (Scanner scanner = new Scanner(System.in)) {

int n = scanner.nextInt();

for(int i = 0; i<n; i++)

{

for(int j=0;j<i;j++)

{

System.out.print(j+1+" ");

}

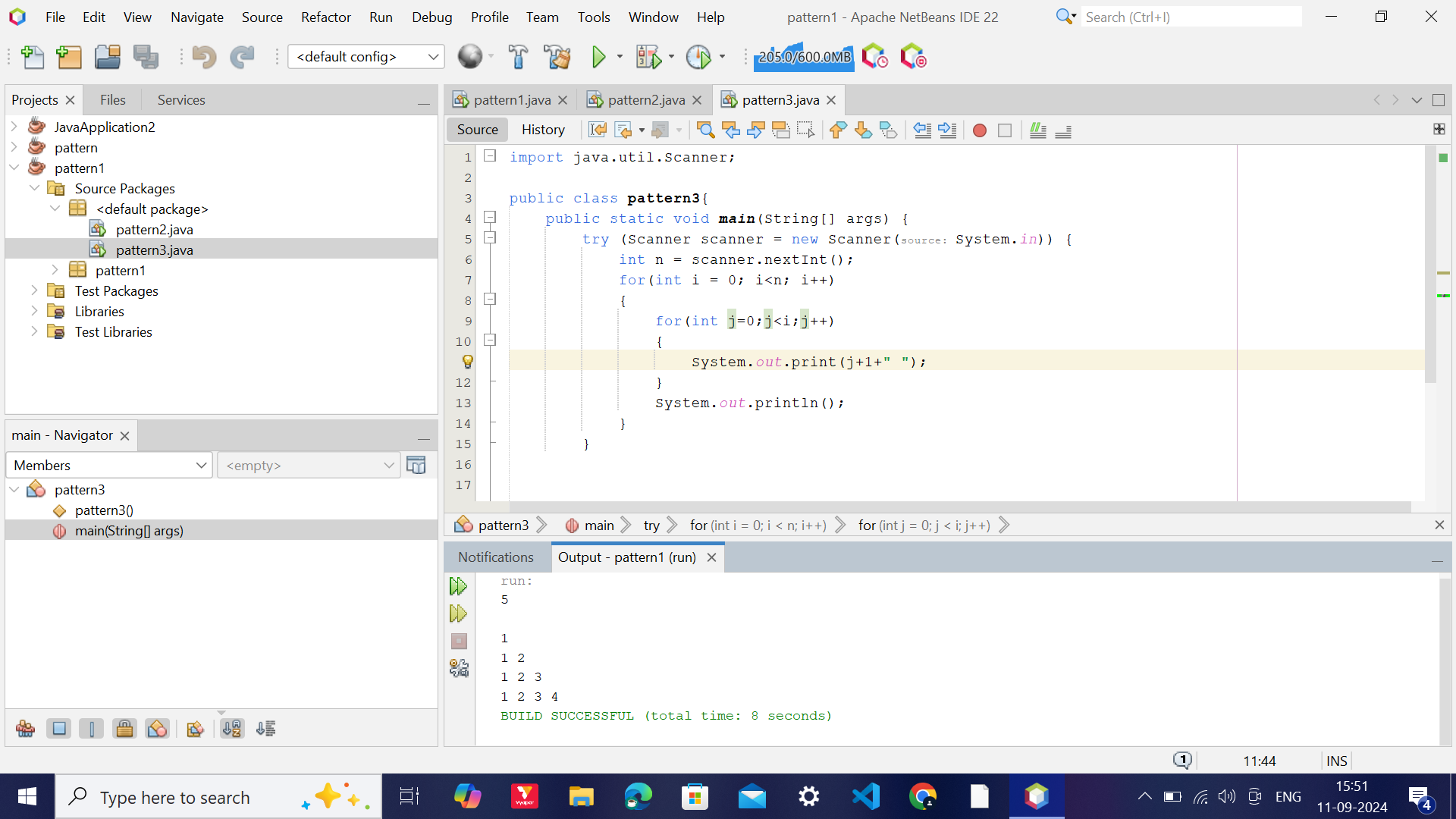
System.out.println();

}

}

}

}



import java.util.Scanner;

public class pattern4{

public static void main(String[] args) {

try (Scanner scanner = new Scanner(System.in)) {

int n = scanner.nextInt();

for(int i = n; i>0; i--)

{

for(int j=0;j<n-i;j++)

{

System.out.print(" ");

}

for(int j=0;j<i;j++){

System.out.print("\*");

}

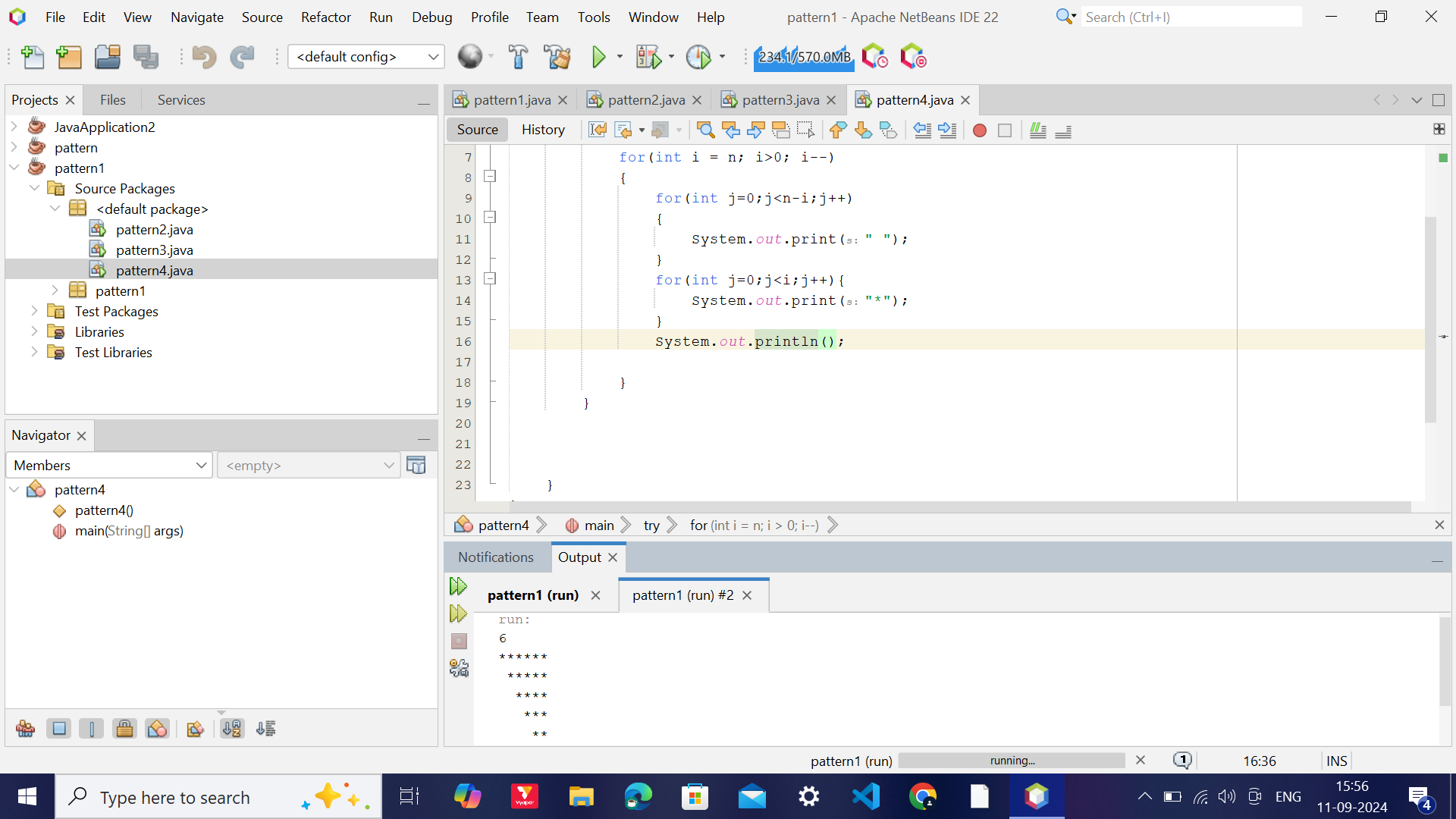
System.out.println();

}

}

}

}



Climbing Stairs:

class Solution {

    public int climbStairs(int n) {

        int arr[] = new int[n+1];

        arr[0]= 1;

        arr[1]=1;

        for(int i=2;i<=n;i++)

        {

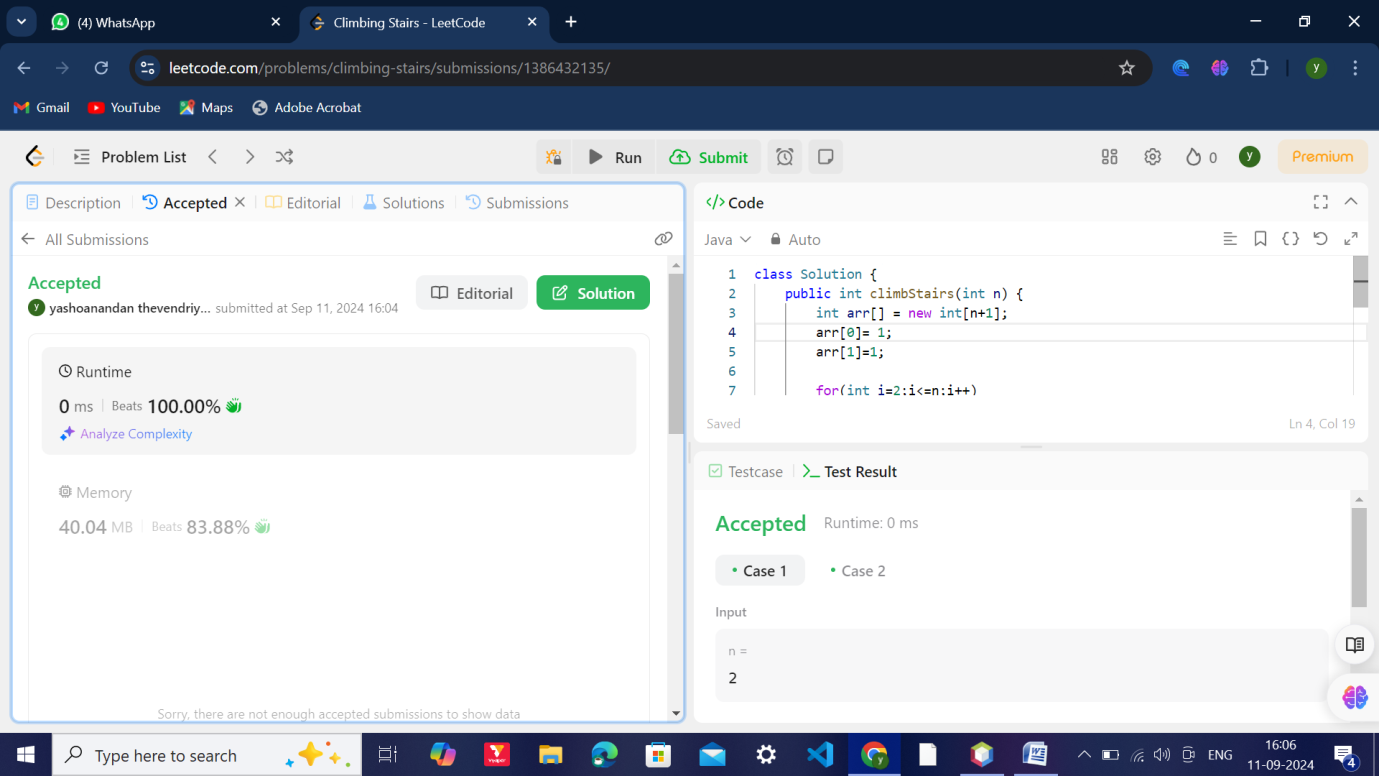
            arr[i] = arr[i-1] + arr[i-2];

        }

        return arr[n];

    }

}



Majority Element:

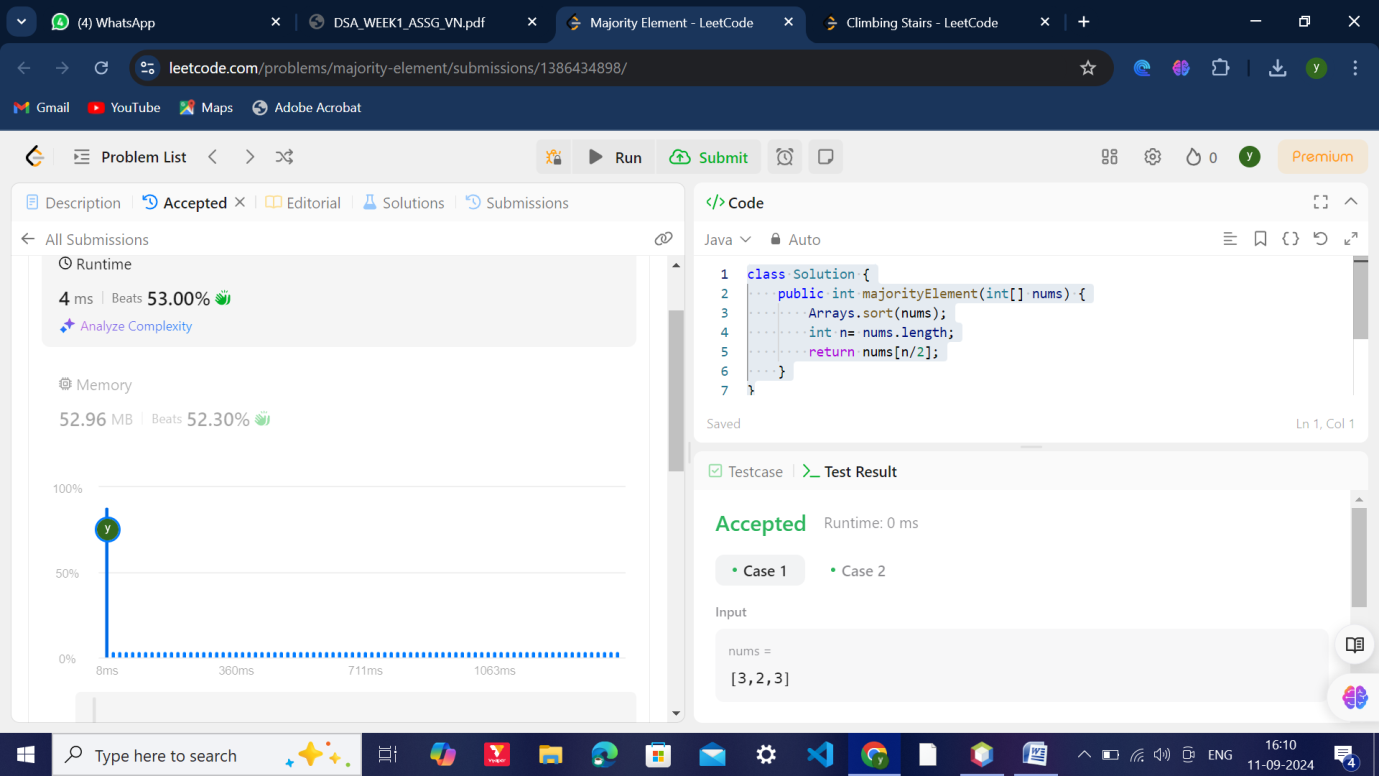
class Solution {

    public int majorityElement(int[] nums) {

        Arrays.sort(nums);

        int n= nums.length;

        return nums[n/2];

    }}

Move Zeroes:

class Solution {

    public void moveZeroes(int[] nums) {

       int flag=0,i,numsSize= nums.length;

       for(i=0;i<numsSize;i++) {

        if(nums[i]!=0)

        {

            nums[flag++]=nums[i];

        }

       }

       for(i=flag;i<numsSize;i++)

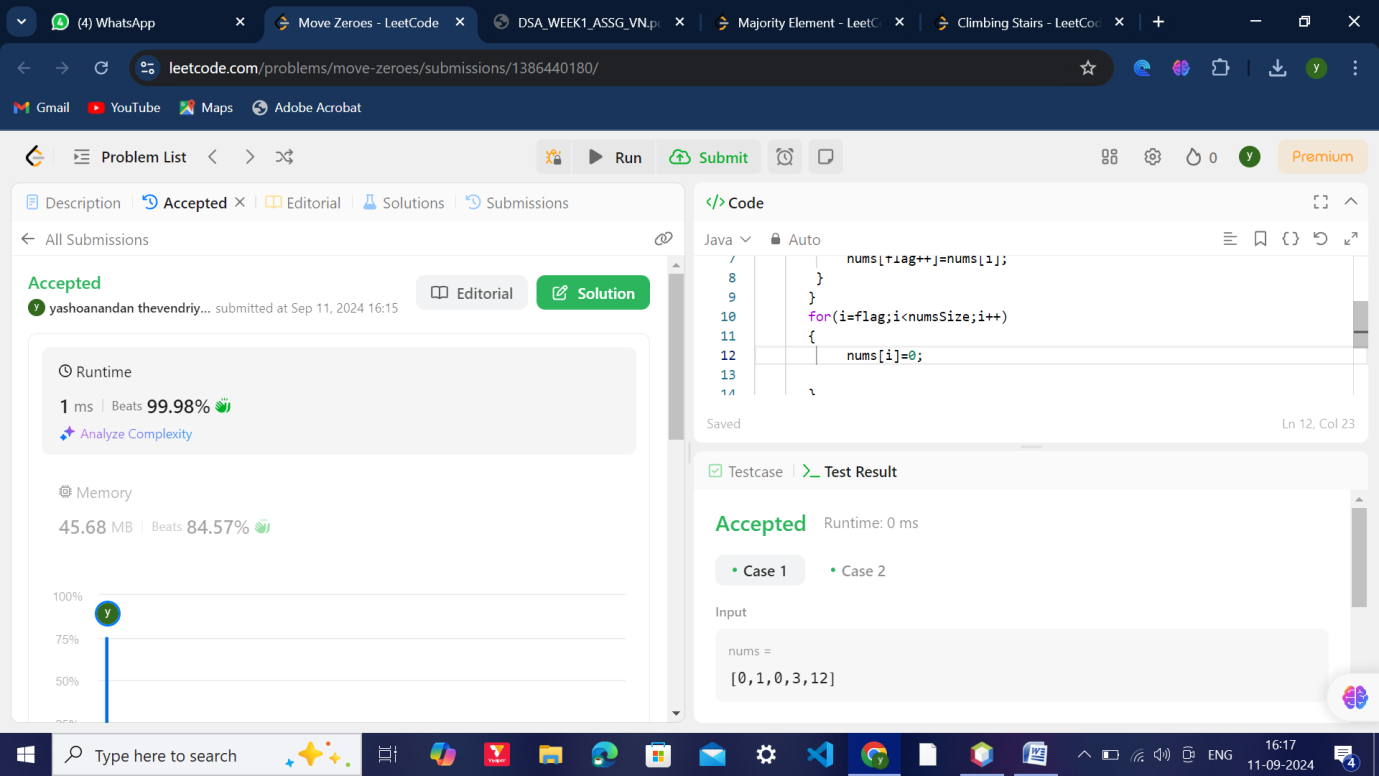
       {

            nums[i]=0;

       }

    }

}



Jump Game:

class Solution {

    public int jump(int[] nums) {

        int n = nums.length;

        int jumps = 0, currentEnd = 0, farthest = 0;

        for (int i = 0; i < n - 1; i++) {

            farthest = Math.max(farthest, i + nums[i]);

            if (i == currentEnd) {

                jumps++;           // Make another jump

                currentEnd = farthest;  // Set the end of this jump to farthest position

                if (currentEnd >= n - 1) {

                    break;

                }

            }

        }

        return jumps;

    }

    public static void main(String[] args) {

        Solution sol = new Solution();

        int[] nums1 = {2, 3, 1, 1, 4};

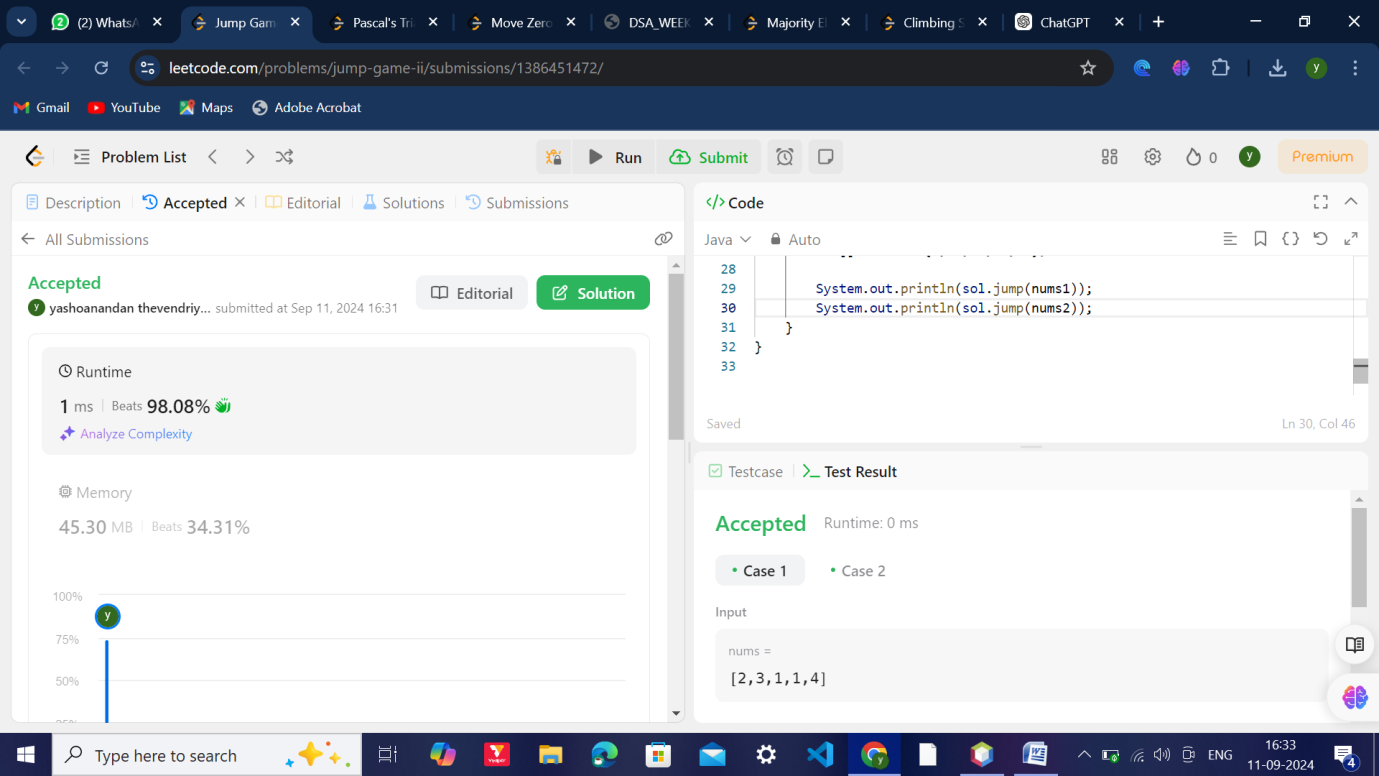
        int[] nums2 = {2, 3, 0, 1, 4};

        System.out.println(sol.jump(nums1));

        System.out.println(sol.jump(nums2));

    }

}



Palindrome:

class Solution {

    public boolean isPalindrome(int x) {

        int ans=0,rem,n=x;

        if(x>=0)

        {

            while(x>0)

            {

                rem= x%10;

                ans= (ans\*10)+rem;

                x/=10;

            }

            return (ans == n)?true:false;

        }

        return false;

    }

}

