|  |
| --- |
| **1.Odd\_OR\_Even** |
| **public class OddEven {**  **static boolean  normal(int n){**  **return n%2==0;**  **}**  **static boolean  recursion(int n){**  **if (n==0)  return true;**  **if(n==1) return false;**  **return recursion(n-2);**  **}**  **public static void main(String[] args) {**  **int n=50;**  **System.out.println(normal(n));**  **System.out.println(recursion(n));**  **}**  **}** |
| **2.Prime** |
| **public class Prime {**  **static boolean normal(int n){**  **if (n<=1) return false;**  **for(int i=2;i<=Math.sqrt(n);i++){**  **if (n%i==0) {**  **return false;**  **}**  **}**  **return  true;**  **}**  **static boolean recursion(int n,int i){**  **if (n<=1) return false;**  **if(i\*i>n) return true;**  **if(n%i==0) return false;**  **return recursion(n,i+1);**  **}**  **public static void main(String[] args) {**  **int n=0;**  **System.out.println(normal(n));**  **System.out.println(recursion(n,2));**  **}**  **}** |
| **3.palindrome** |
| **public class Palindrome {**  **public static void main(String[] args) {**  **int num=1221;**  **int og=num;**  **int rev=0;**  **while (num!=0) {**  **rev=rev\*10+num%10;**  **num/=10;}**  **System.out.println(og==rev);**  **}**  **}** |
| **4.Factorial** |
| **public class Factorial {**  **public static void main(String[] args) {**  **int n=20;**  **long fact=1;**  **for (int i=1;i<=n;i++) {**  **fact\*=i;**  **}**  **System.out.println(fact);**  **}**  **}** |
| **5.Amstrong** |
| **public class Amstrong {**  **public static void main(String[] args) {**  **int num=9474;**  **int og=num;**  **int sum=0;**  **int digits=String.valueOf(num).length();**  **while (num!=0) {**  **int digit=num%10;**  **sum+=Math.pow(digit, digits);**  **num/=10;**  **}**  **System.out.println("Amstrong of "+og+" is "+(sum==og));**  **}**  **}** |
| **6.Swap with temp without temp** |
| **public class Swap {**  **static void withTemp(int a ,int b){**  **int temp=a;**  **a=b;**  **b=temp;**  **System.out.println(a+" , "+b);**  **}**  **static void notemp(int a ,int b){**  **a=a+b;**  **b=a-b;**  **a=a-b;**  **System.out.println(a+" , "+b);**  **}**  **public static void main(String[] args) {**  **int a=105;**  **int b=125;**  **withTemp(a, b);**  **notemp(a, b);**  **}**  **}** |
| **7.largestOfThree** |
| **public class FindLargestThree {**  **public static void main(String[] args) {**  **int a=50,b=120,c=100;**  **int largest=a;**  **if (b>largest)largest=b;**  **if(c>largest)largest=c;**  **//largest=(a>b)?(b>c?a:c):(b>c?b:c);**  **System.out.println(largest);**  **}**  **}** |
| **8.bubbleSort** |
| **import java.util.Arrays;**  **public class BubbleSort {**  **static void Sort(int[]a){**  **boolean swapped=true;**  **for(int i =0;i<a.length-1;i++){**  **swapped=false;**  **for(int j=0;j<a.length-i-1;j++){**  **if(a[j]>a[j+1]){**  **int temp=a[j+1];**  **a[j+1]=a[j];**  **a[j]=temp;**  **swapped=true;**  **}**  **}**  **if (!swapped)  break;**  **}**  **}**  **public static void main(String[] args) {**  **int[]a={20,30,175,0,4,5,7,90};**  **Sort(a);**  **System.out.println(Arrays.toString(a));**  **}**  **}** |
| **9.InsertionSort** |
| **import java.util.Arrays;**  **public class InsertionSort {**  **static void sort(int a[]){**  **for(int i=1;i<a.length;i++){**  **int key=a[i];**  **int j=i-1;**  **while(j>=0&&a[j]<key){**  **a[j+1]=a[j];**  **j--;**  **}**  **a[j+1]=key;**  **}**  **}**  **public static void main(String[] args) {**  **int[]a={20,30,175,0,4,5,7,90};**  **sort(a);**  **System.out.println(Arrays.toString(a));**  **}**  **}** |
| **10.selectionSort** |
| **import java.util.Arrays;**  **public class SelectionSort {**  **static void sort(int a[]){**  **for (int i=0;i<a.length-1;i++) {**  **int mid=i;**  **for (int j = i+1; j <a.length; j++) {**  **if (a[mid]>a[j]) {**  **mid=j;**  **}**  **}**  **int temp=a[mid];**  **a[mid]=a[i];**  **a[i]=temp;**  **}**  **}**  **public static void main(String[] args) {**  **int[]a={20,30,175,0,4,5,7,90};**  **sort(a);**  **System.out.println(Arrays.toString(a));**  **}**  **}** |
| **11.MergeSort** |
| **import java.util.Arrays;**  **public class MergeSort {**  **static void sort(int[] a, int left, int right) {**  **if (left<right) {**  **int mid=(left+right)/2;**  **sort(a, left, mid);**  **sort(a, mid+1, right);**  **merge(a,left,mid,right);**  **}**  **}**  **static void merge(int[] a, int left, int mid, int right) {**  **int n1=mid-left+1;**  **int n2=right-mid;**  **int l[]=new int[n1];**  **int r[]=new int[n2];**  **System.arraycopy(a,left, l, 0, n1);**  **System.arraycopy(a,mid+1, r, 0, n2);**  **int i=0,j=0,k=left;**  **while (i<n1&&j<n2) {**  **a[k++]=(l[i]<=r[j])?l[i++]:r[j++];**  **}**  **while(i<n1) a[k++]=l[i++];**  **while(j<n2) a[k++]=r[j++];**  **}**  **public static void main(String[] args) {**  **int[]a={20,30,175,0,4,5,7,90};**  **int left=0,right=a.length-1;**  **sort(a,left,right);**  **System.out.println(Arrays.toString(a));**  **}**  **}** |
| **12.QuickSort** |
| **import java.util.Arrays;**  **public class QuickSort {**  **static void quickSort(int a[],int low ,int high){**  **if (low<high) {**  **int pivotindex=partision(a,low,high);**  **quickSort(a, low, pivotindex-1);**  **quickSort(a,pivotindex+1,high);**  **}}**  **static int partision(int a[],int low,int high){**  **int key=a[high];**  **int i=low-1;**  **for (int j=low;j<high;j++) {**  **if (a[j]<key) {**  **i++;**  **int temp=a[i];**  **a[i]=a[j];**  **a[j]=temp;**  **}**  **}**  **int temp=a[i+1];**  **a[i+1]=a[high];**  **a[high]=temp;**  **return i+1;**  **}**  **public static void main(String[] args) {**  **int[]a={20,30,175,0,4,5,7,90};**  **int low=0;**  **int high=a.length-1;**  **quickSort(a, low, high);**  **System.out.println(Arrays.toString(a));**  **}**  **}** |
| **13. linearSearch** |
| **public class LinearSearch {**  **public static void main(String[] args) {**  **int[]a={20,30,175,0,4,5,7,90};**  **int k=784;**  **boolean isPresent=false;**  **for(int i=0;i<a.length;i++){**  **if (a[i]==k) {**  **isPresent=true ;**  **break;**  **}**  **}**  **System.out.println(isPresent);**  **}**  **}** |
| **14.Binary Search** |
| **import java.util.Arrays;**  **public class BinarySearch {**  **public static void main(String[] args) {**  **int[]a={20,30,175,0,4,5,7,90};**  **Arrays.sort(a);**  **int target=90;**  **int start=0;**  **int end=a.length-1;**  **boolean irruka=false;**  **while (start<=end) {**  **int mid=(start+end)/2;**  **if (a[mid]==target) {**  **irruka=true;**  **break;**  **}else if(a[mid]<target) start=mid+1;**  **else end=mid-1;**  **}**  **System.out.println(irruka);**  **}**  **}** |
| **15.MatrixAddition** |
| **public class MatrixAddition {**  **public static void main(String[] args) {**  **int a[][]={{1,2,4},{3,4,5},{2,5,6}};**  **int b[][]={{1,2,4},{3,4,5},{2,5,6}};**  **int row=a.length;**  **int col=a[0].length;**  **int res[][]=new int[row][col];**  **for (int i=0;i<row;i++) {**  **for (int j=0;j<col;j++) {**  **res[i][j]=a[i][j]+b[i][j];**  **}**  **}**  **for (int col1[] : res) {**  **for (int row1 : col1) {**  **System.out.print(row1+" ");**  **}**  **System.out.println();}**  **}}** |
| **16.transpose the matrix** |
| **public class TransposeMatrix {**  **public static void main(String[] args) {**  **int a[][]={{1,2,4},{3,4,5},{2,5,6}};**  **int row=a.length;**  **int col=a[0].length;**  **int res[][]=new int[col][row];**  **for(int i=0;i<row;i++){**  **for(int j=0;j<col;j++){**  **res[j][i]=a[i][j];**  **}**  **}**  **for (int col1[] : res) {**  **for (int row1 : col1) {**  **System.out.print(row1+" ");**  **}**  **System.out.println();}**  **}**  **}** |
| **17. matrix multiplication** |
| **public class MatrixMultiplication {**  **public static void main(String[] args) {**  **int a[][]={{1,2,3},{4,5,6},{7,8,9}};**  **int b[][]={{1,2,3},{4,5,6},{7,8,9}};**  **int row=a.length,col1=a[0].length,col2=b[0].length;**  **int res[][]=new int[row][col2];**  **for(int i=0;i<row;i++){**  **for(int j=0;j<col2;j++){**  **for(int k=0;k<col1;k++){**  **res[i][j]+=a[i][k]\*b[k][j];**  **}**  **}**  **}**  **for (int colp[] : res) {**  **for (int row1 : colp) {**  **System.out.print(row1+" ");**  **}**  **System.out.println();}**  **}**  **}** |
| **18. Rotate90Matrix** |
| **public class Rotate90Matrix {**  **public static void main(String[] args) {**  **int a[][]={{1,2,3},{4,5,6},{7,8,9}};**  **int n=a.length;**  **for(int i=0;i<n;i++){**  **for(int j=i+1;j<n;j++){**  **int temp=a[i][j];**  **a[i][j]=a[j][i];**  **a[j][i]=temp;**  **}**  **}**  **for(int i=0;i<n;i++){**  **for(int j=0,k=n-1;k>j;j++,k--){**  **int temp=a[i][j];**  **a[i][j]=a[i][k];**  **a[i][k]=temp;**  **}**  **}**  **for (int[] copy: a) {**  **System.out.println(Arrays.toString(copy));**  **}**  **}}** |
| **19.ReverseMatrix** |
| **import java.util.Arrays;**  **public class ReverseMatrix {**  **public static void main(String[] args) {**  **int a[][]={{1,2,3},{4,5,6},{7,8,9}};**  **int n=a.length;**  **for(int i=0;i<n;i++){**  **for(int j=0,k=n-1;k>j;j++,k--){**  **int temp=a[i][j];**  **a[i][j]=a[i][k];**  **a[i][k]=temp;**  **}**  **}**  **for (int[]row : a) {**  **System.out.println(Arrays.toString(row));**  **}**  **}**  **}** |
| **20.FindMinMaxArray** |
| **public class FindMinMaxArray {**  **public static void main(String[] args) {**  **int []a={0,1,5,89,5,6,2};**  **int max=a[0];**  **int min=a[0];**  **for (int i : a) {**  **if (max<i)max=i;**  **if (min>i)min=i;**  **}**  **System.out.println(max);**  **System.out.println(min);**  **}**  **}** |
| **21.ReverseArray** |
| **import java.util.Arrays;**  **public class ReverseArray {**  **public static void main(String[] args) {**  **int []a={0,1,5,89,5,6,2};**  **int start=0;**  **int end =a.length-1;**  **while(start < end){**  **int temp=a[start];**  **a[start++]=a[end];**  **a[end--]=temp;**  **}**  **System.out.println(Arrays.toString(a));**  **}**  **}** |
| **22.FindNthMinNthMax** |
| **public class FindNthMinNthMax {**  **public static void main(String[] args) {**  **int []a={0,1,5,89,5,6,2};**  **for (int i=0;i<a.length-1;i++) {**  **for(int j=0;j<a.length-1;j++){**  **if (a[j]>a[j+1]) {**  **int temp=a[j+1];**  **a[j+1]=a[j];**  **a[j]=temp;**  **}**  **}**  **}**  **int n=8;**  **if(n<=(a.length))**  **{**  **System.out.println("the second min is " + (a[n-1]));**  **System.out.println("the second max is " + (a[a.length-(n)]));**  **}**  **else System.out.println("enter the valid size");**  **}**  **}** |
| **23.FindArraySortedOrNot** |
| **public class FindArraySortedOrNot {**  **public static void main(String[] args) {**  **int []a={0,1,2,3,5,6,7};**  **boolean isSorted=true;**  **for(int j=0;j<a.length-1;j++){**  **if (a[j]>a[j+1]) {**  **isSorted=false;**  **break;**  **}**  **}**  **System.out.println((isSorted)?"ama da its sorted":"illa d its not sorted");**  **}**  **}** |
| **24.FindDuplicateUsingHashmap** |
| **import java.util.HashMap;**  **import java.util.Map;**  **public class FindDuplicateUsingHashmap {**  **public static void main(String[] args) {**  **int []a={1, 2, 3, 4, 5};**  **boolean irrukan=false;**  **HashMap<Integer,Integer> dup=new HashMap<>();**  **for (int num : a) {**  **dup.put(num,dup.getOrDefault(num,0)+1);**  **}**  **for (Map.Entry<Integer,Integer> entry : dup.entrySet() ){**  **if (entry.getValue() > 1) {**  **System.out.println("the duplicate "+entry.getKey()+" the occurance are "+entry.getValue());**  **irrukan=true;}**  **}**  **if (!irrukan) {**  **System.out.println("no duplicates");**  **}**  **}}** |
| **25.RotateArrayByN** |
| **import java.util.Arrays;**  **public class RotateArrayByN {**  **public static void main(String[] args) {**  **int arr[]={1,2,3,4,5,6};**  **int n=6;**  **int k=n%arr.length;**  **int start=0;**  **int end=arr.length-1;**  **swap(arr,start,end);**  **swap(arr,start,k-1);**  **swap(arr,k,end);**  **System.out.println(Arrays.toString(arr));**  **}**  **static void swap(int[]arr,int start,int end){**  **while (start<end) {**  **int temp=arr[start];**  **arr[start++]=arr[end];**  **arr[end--]=temp;**  **}**  **}**  **}** |
| **26.findProductOfArray** |
| **import java.util.Arrays;**  **public class FindProductOfArray {**  **public static void main(String[] args) {**  **int arr[]={1,2,0,4};**  **int count=0;**  **int total=1;**  **for(int i=0;i<arr.length;i++){**  **if (arr[i]==0) count++;**  **else total\*=arr[i];**  **}**  **for(int i=0;i<arr.length;i++){**  **if (count>1) {**  **arr[i]=0;**  **}**  **else if(count==1){**  **arr[i]=(arr[i]==0)?total:0;**  **}**  **else arr[i]=total/arr[i];**  **}**  **System.out.println(Arrays.toString(arr));}**  **}** |
| **27.FindPairWithSum** |
| **import java.util.HashSet;**  **public class FindPairWithSum {**  **public static void main(String[] args) {**  **int arr[] = {1, 5, 2, 3, 0, 4};**  **int compliment=0;**  **int target=5;**  **HashSet <Integer> set=new HashSet<>();**  **for(int num:arr){**  **compliment=target-num;**  **if (set.contains(compliment)) {**  **System.out.println("here it is ( " +num+" , "+compliment+" );");**  **}**  **set.add(num);**  **}**  **}**  **}** |
| **28. findMissingNum1ToN** |
| **public class findMissingNum1ToN {**  **public static void main(String[] args) {**  **int a[] = {1,2,3,4,6,7,8,9};**  **int n=a.length;**  **int ExpectedNum=0;**  **int Acutalsum=(n+1)\*(n+2)/2;**  **for(int num:a){**  **ExpectedNum +=num ;**  **}**  **System.out.println("the missing number"+" "+(Acutalsum-ExpectedNum));**  **}**  **}** |
| **29 .FindSubArrayWithN** |
| **import java.util.ArrayList;**  **import java.util.List;**  **public class FindSubArrayWithN {**  **public static void main(String[] args) {**  **int a []={1,2,3,1,2,2};**  **int target=4;**  **List<List<Integer>> subArrays= new ArrayList();**  **for (int i=0;i<a.length;i++) {**  **int sum=0;**  **List<Integer> tempList=new ArrayList();**  **for (int j=i;j<a.length;j++) {**  **sum+=a[j];**  **tempList.add(a[j]);**  **if (sum==target){**  **subArrays.add(new ArrayList<>(tempList));**  **break;}**  **}**  **}if (subArrays.isEmpty()) {**  **System.out.println("there is no possible to give sub array with this");**    **}else{**  **System.out.println(subArrays);**  **}**  **}**  **}** |
| **30.MergeTwoArrayAsSorted** |
| **import java.util.Arrays;**  **public class MergeTwoArrayAsSorted {**  **public static void main(String[] args) {**  **int a[]={1,5,6,1,2,7,2};**  **int b[]={9,7,5,3,6,1,2};**  **Arrays.sort(a);**  **Arrays.sort(b);**  **int res[]=new int[a.length+b.length];**  **int i=0,j=0,k=0;**  **while (i<a.length&&j<b.length) {**  **res[k++]=(a[i]<=b[j])?a[i++]:b[j++];**  **}**  **while(i<a.length) res[k++]=a[i++];**  **while(j<b.length) res[k++]=b[j++];**  **System.out.println(Arrays.toString(res));**  **}**  **}** |
| 1. **Square Fill Pattern** |
| **public class SquareFillPattern {**  **public static void main(String[] args) {**  **int n=5;**  **for (int i=1;i<=n;i++) {**  **for(int j=1;j<=n;j++){**  **System.out.print("\*");**  **}**  **System.out.println();**    **}**  **}**  **}** |
| 1. **Left Half Pyramid Pattern** |
| **public class leftHalfPyramidPattern {**  **public static void main(String[] args) {**  **int n=5;**  **for(int i=1;i<=n;i++){**  **for(int j=1;j<=i;j++){**  **System.out.print("\*");**  **}**  **System.out.println();**  **}**  **}**  **}** |
| **33. Reverse left Half Pyramid Pattern** |
| **public class ReverseleftHalfPyrmid {**  **public static void main(String[] args) {**  **int n=5;**  **for(int i=n;i>=1;i--){**  **for(int j=1;j<=i;j++){**  **System.out.print("\*");**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Number increasing pyramid pattern** |
| **public class NumberIncreasingpyrmid {**  **public static void main(String[] args) {**  **int n=5;**  **for(int i=1;i<=n;i++){**  **for(int j=1;j<=i;j++){**  **System.out.print(j+" ");**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Number-increasing reverse Pyramid Pattern** |
| **public class NumberReversePyrmid {**  **public static void main(String[] args) {**  **int n=5;**  **for(int i=1;i<=n;i++){**  **for(int j=1;j<=n-i+1;j++){**  **System.out.print(j);**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Right Pascal’s Triangle** |
| **public class RightPascalTriangle {**  **public static void main(String[] args) {**  **int n = 5;**  **for (int i = 1; i <= n; i++) {**  **for (int j = 1; j <= i; j++) {**  **System.out.print("\*");**  **}**  **System.out.println();**  **}**  **for (int i = n - 1; i >= 1; i--) {**  **for (int j = 1; j <= i; j++) {**  **System.out.print("\*");**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **K Pattern** |
| **import java.util.Scanner;**  **public class K\_pattern {**  **public static void main(String[] args) {**  **Scanner sc = new Scanner(System.in);**  **System.out.print("Enter the input: ");**  **int n = sc.nextInt();**  **for(int i=1;i<=n;i++){**  **for(int j=1;j<=n-i+1;j++){**  **System.out.print("\*");**  **}**  **System.out.println();**  **}**  **for(int i=2;i<=n;i++){**  **for(int j=1;j<=i;j++){**  **System.out.print("\*");**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Zero-One Triangle Pattern** |
| **public class ZeroNOnePattern {**  **public static void main(String[] args) {**  **int rows = 5;**  **for (int i = 1; i <= rows; i++) {**  **for (int j = 1; j <= i; j++) {**  **System.out.print(j % 2 == 0 ? 1 : 0);**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Square hallow pattern** |
| **public class SquareHallowPattern {**  **public static void main(String[] args) {**  **int n=5;**  **for(int i=1;i<=n;i++){**  **for(int j=1;j<=n;j++){**  **if (i==1||i==n||j==1||j==n ){**  **System.out.print("\*");**  **}else{**  **System.out.print(" ");**  **}**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Number-changing Pyramid Pattern** |
| **public class NumberChangingPattern {**  **public static void main(String[] args) {**  **int n=5;**  **int num=0;**  **for(int i=1;i<=n;i++){**  **for(int j=1;j<=i;j++){**  **System.out.print(++num+" ");**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **right Half Pyramid Pattern** |
| **public class RightHalfPyrmid {**  **public static void main(String[] args) {**  **int n=5;**  **for(int i=1;i<=n;i++){**  **for(int j=1;j<=n-i;j++){**  **System.out.print(" ");**  **}**  **for(int j=1;j<=i;j++){**  **System.out.print("\*");**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Reverse right Half Pyramid Pattern** |
| **public class ReverseRightHalfPyramid {**  **public static void main(String[] args) {**  **int n = 5;**  **for (int i = n; i >= 1; i--) {  // Rows**  **for (int j = 1; j <= n - i; j++) {  // Spaces**  **System.out.print(" ");**  **}**  **for (int j = 1; j <= i; j++) {  // Stars**  **System.out.print("\*");**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Triangle Star Pattern** |
| **public class TriangleStarPattern {**  **public static void main(String[] args) {**  **int n = 5;**  **for (int i = 1; i <= n; i++) {  // Rows**  **for (int j = 1; j <= n - i; j++) {  // Spaces**  **System.out.print(" ");**  **}**  **for (int j = 1; j <= 2 \* i - 1; j++) {  // Stars**  **System.out.print("\*");**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Number triangle pattern** |
| **public class TriangleStarPattern {**  **public static void main(String[] args) {**  **int n = 5;**  **for (int i = 1; i <= n; i++) {**  **for (int j = 1; j <= n - i; j++) {**  **System.out.print(" ");**  **}**  **for (int j = 1; j <= 2 \* i - 1; j++) {**  **System.out.print(i);**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Reverse number Triangle Pattern** |
| **import java.util.Scanner;**  **public class ReverseNumberTriangle{**  **public static void main(String[] args) {**  **Scanner sc = new Scanner(System.in);**  **System.out.print("Enter the input: ");**  **int n = sc.nextInt();**  **for(int i=1;i<=n;i++){**  **for(int j=1;j<i;j++){**  **System.out.print(" ");**  **}**  **for(int j=i;j<=n;j++){**  **System.out.print(j + " ");**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Rhombus Pattern** |
| **import java.util.Scanner;**  **public class Rhombus {**  **public static void main(String[] args) {**  **Scanner sc = new Scanner(System.in);**  **System.out.print("Enter the input: ");**  **int n = sc.nextInt();**  **for (int i = 1; i <= n; i++) {**  **for(int j=1;j<=n-i;j++){**  **System.out.print(" ");**  **}**  **for(int j=1;j<=n;j++){**  **System.out.print("\*");**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Palindrome Triangle Pattern** |
| **import java.util.Scanner;**  **public class PalindromeTrianglePattern {**  **public static void main(String[] args) {**  **Scanner sc = new Scanner(System.in);**  **System.out.print("Enter the input: ");**  **int n = sc.nextInt();**  **for (int i = 1; i <= n; i++) {**  **for (int j = i; j < n; j++) {**  **System.out.print("  ");**  **}**  **for (int j = i; j >= 1; j--) {**  **System.out.print(j + " ");**  **}**  **for (int j = 2; j <= i; j++) {**  **System.out.print(j + " ");**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Diamond Star Pattern** |
| **public class DiamondPyramid {**  **public static void main(String[] args) {**  **int n = 5;**  **for (int i = 1; i <= n; i++) {**  **for (int j = 1; j <= n - i; j++) {**  **System.out.print(" ");**  **}**  **for (int j = 1; j <= 2 \* i - 1; j++) {**  **System.out.print("\*");**  **}**  **System.out.println();**  **}**  **for (int i = n - 1; i >= 1; i--) {**  **for (int j = 1; j <= n - i; j++) {**  **System.out.print(" ");**  **}**  **for (int j = 1; j <= 2 \* i - 1; j++) {**  **System.out.print("\*");**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Butterfly Star Pattern** |
| **import java.util.Scanner;**  **public class ButterflyPattern {**  **public static void main(String[] args) {**  **Scanner sc = new Scanner(System.in);**  **System.out.print("Enter the input: ");**  **int n = sc.nextInt();**  **/\***  **i=1, spaces = 8, n = 5, star = 1 , s = (n-i)\*2**  **i=2, spaces = 6, n = 5 ,star = 2**  **i=3, spaces = 4, n = 5 ,star = 3**  **i=4, spaces = 2, n = 5 ,star = 4**  **i=5, spaces = 0, n = 5 ,star = 5**  **\*/**  **//Upper pattern**  **//outer loop for rows**  **for(int i=1;i<=n;i++){**  **//print left stars**  **for(int j=1;j<=i;j++){**  **System.out.print("\*");**  **}**  **//Print spaces**  **for(int j=1;j<=(n-i)\*2;j++){**  **System.out.print(" ");**  **}**  **//Print right stars**  **for(int j=1;j<=i;j++){**  **System.out.print("\*");**  **}**  **System.out.println();**  **}**  **/\***  **i=1, spaces = 0, n = 5, star = 5 , s = 2\*i-2**  **i=2, spaces = 2, n = 5 ,star = 4 , s = 2\*i-2**  **i=3, spaces = 4, n = 5 ,star = 3**  **i=4, spaces = 6, n = 5 ,star = 2**  **i=5, spaces = 8, n = 5 ,star = 1**  **\*/**  **//Lower pattern**  **//outer loop for rows**  **for(int i=1;i<=n;i++){**  **//print left stars**  **for(int j=1;j<=n-i+1;j++){**  **System.out.print("\*");**  **}**  **//Print spaces**  **for(int j=1;j<=2\*i-2;j++){**  **System.out.print(" ");**  **}**  **//Print right stars**  **for(int j=1;j<=n-i+1;j++){**  **System.out.print("\*");**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Mirror Image Triangle Pattern** |
| **import java.util.Scanner;**  **public class mirrorTriangle {**  **public static void main(String[] args) {**  **Scanner sc = new Scanner(System.in);**  **System.out.print("Enter the input: ");**  **int n = sc.nextInt();**    **for(int i=1;i<=n;i++){**  **//print spaces**  **for(int j=1;j<i;j++){**  **System.out.print(" ");**  **}**  **//print numbers**  **for(int j=i;j<=n;j++){**  **System.out.print(j+" ");**  **}**  **System.out.println();**  **}**  **for(int i=1;i<n;i++){**  **//print spaces**  **for(int j=1;j<=n-i-1;j++){**  **System.out.print(" ");**  **}**  **//print numbers**  **for(int j=n-i;j<=n;j++){**  **System.out.print(j + " ");**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Hollow Triangle Pattern** |
| **import java.util.Scanner;**  **public class HallowTrianglePattern {**  **public static void main(String[] args) {**  **Scanner sc = new Scanner(System.in);**  **System.out.print("Enter the input: ");**  **int n = sc.nextInt();**  **for(int i=1;i<=n;i++){  //runs n times**  **for(int j=1;j<=n-i;j++){**  **System.out.print(" ");**  **}**  **for(int j=1;j<=2\*i-1;j++){**  **if(j==1 || i==n || j==2\*i-1){**  **System.out.print("\*");**  **}else{**  **System.out.print(" ");**  **}**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Hollow Reverse Triangle Pattern** |
| **import java.util.\*;**  **public class reverseHallowTriangle {**  **public static void main(String[] args) {**  **Scanner sc = new Scanner(System.in);**  **System.out.print("Enter the input: ");**  **int n = sc.nextInt();**  **for(int i=1;i<=n;i++){**  **for(int j=1;j<i;j++){  //1<3, 2<3**  **System.out.print(" ");**  **}**  **for(int j=1;j<=2\*(n-i+1)-1;j++){**  **if(j==1 || i==1 || j == 2\*(n-i+1)-1){**  **System.out.print("\*");**  **}else {**  **System.out.print(" ");**  **}**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Hollow Diamond Pyramid** |
| **import java.util.Scanner;**  **public class HallowPrimidDIamond {**  **public static void main(String[] args) {**  **//Square hallow pattern**  **Scanner sc = new Scanner(System.in);**  **System.out.print("Enter the input: ");**  **int n = sc.nextInt();**    **for(int i=1;i<=n;i++){**  **//Print leading spaces**  **for(int j=1;j<=n-i;j++){**  **System.out.print(" ");**  **}**  **//Print stars and hallow spaces**  **for(int j=1;j<=2\*i-1;j++){**  **if(j==1 || j == 2\*i-1){**  **System.out.print("\*");**  **}else{**  **System.out.print(" ");**  **}**  **}**  **System.out.println();**  **}**    **for(int i=1;i<n;i++){ //1,2,3,4**  **//Print leading spaces**  **for(int j=1;j<=i;j++){**  **System.out.print(" ");**  **}**  **//Print stars and hallow spaces in triangle**  **for(int j=1;j<=(n-i)\*2-1;j++){**  **if(j==1 || j == (n-i)\*2-1){**  **System.out.print("\*");**  **}else {**  **System.out.print(" ");**  **}**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **Hollow Hourglass Pattern** |
| **import java.util.Scanner;**  **public class HallowHourPattern {**  **public static void main(String[] args) {**  **Scanner sc = new Scanner(System.in);**  **System.out.print("Enter the input: ");**  **int n = sc.nextInt();**  **for(int i=1;i<=n;i++){**  **for(int j=1;j<i;j++){**  **System.out.print(" ");**  **}**  **for(int j=1;j<=n-i+1;j++){**  **if(j==1 || i==1 || j == n-i+1){**  **System.out.print("\*" + " ");**  **}else{**  **System.out.print("  "); //double space**  **}**  **}**  **System.out.println();**  **}**    **for(int i=1;i<n;i++){**  **for(int j=1;j<=n-i-1;j++){**  **System.out.print(" ");**  **}**  **for(int j=1;j<=i+1;j++){**  **if(j==1 || i == n-1 || j==i+1){**  **System.out.print("\*"+" ");**  **}else{**  **System.out.print("  ");**  **}**  **}**  **System.out.println();**  **}**  **}**  **}**  **}**  **}** |
| 1. **X - Star Pattern** |
| **import java.util.Scanner;**  **public class Xpattern {**  **public static void main(String[] args) {**    **Scanner sc = new Scanner(System.in);**  **System.out.print("Enter the input: ");**  **int n = sc.nextInt();**  **for(int i=1;i<=n;i++){**  **for(int j=1;j<=i-1;j++){**  **System.out.print(" ");**  **}**  **for(int j=1;j<=n-i+1;j++){**  **if(j==1 || j==n-i+1){**  **System.out.print("\* ");**  **}else {**  **System.out.print("  ");**  **}**  **}**  **System.out.println();**  **}**  **for(int i=1;i<n;i++){**  **for(int j=1;j<=n-i-1;j++){**  **System.out.print(" ");**  **}**  **for(int j=1;j<=i+1;j++){**  **if(j==1 || j==i+1){**  **System.out.print("\* ");**  **}else{**  **System.out.print("  ");**  **}**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **pascal triangle** |
| **public class PascalTriangle {**  **public static void main(String[] args) {**  **int n = 5;**  **for (int i = 0; i < n; i++) {**  **for (int j = 0; j < n - i; j++) {**  **System.out.print(" ");**  **}**  **int number = 1;**  **for (int j = 0; j <= i; j++) {**  **System.out.print(number + " ");**  **number = number \* (i - j) / (j + 1);**  **}**  **System.out.println();**  **}**  **}**  **}** |
| 1. **CountFirstLetter** |
| **public class CountFirstLetter {**  **public static void main(String[] args) {**  **String str="vijay";**  **int count=0;**  **char c=str.charAt(0);**  **for (int i=0;i<str.length();i++) {**  **if (c==str.charAt(i)) count++;**  **}**  **System.out.println("the first char "+c+" present in " +count+" times");**  **}**  **}** |
| **58. CountEverythingString** |
| **public class CountEverythingString {**  **public static void main(String[] args) {**  **String str="yara antha paiyan nanthan antha paiyan";**  **int charCount=0,wordCount=0,vowelsCount=0,consonentCount=0;**  **String[] words=str.split("\\s+");**  **wordCount=words.length;**  **str=str.replaceAll("\\s", "");**  **charCount=str.length();**  **for(int i=0;i<str.length();i++){**  **char c=str.charAt(i);**  **if ("aeiouAEIOU".indexOf(c)!=-1) {**  **vowelsCount++;**    **}}**  **consonentCount=charCount-vowelsCount;**  **System.out.println("the number of character is "+charCount);**  **System.out.println("the number of Words is "+wordCount);**  **System.out.println("the number of vowels is "+vowelsCount);**  **System.out.println("the number of consonent is "+consonentCount);**  **}**  **}** |
| **59.SeprateString** |
| **public class SeprateString {**  **public static void main(String[] args) {**  **String str="yara antha l--su n@n th@a an&ha l00su ";**  **str=str.replaceAll("[^a-zA-z\s]", "");**  **System.out.println(str);**  **}**  **}** |
| **60. CheckPalindrome** |
| **public class CheckPalindrome {**  **public static void main(String[] args) {**  **String str="vijiv";**  **String og=str;**  **String rev="";**  **for ( int i=str.length()-1;i>=0;i--) {**  **rev+=str.charAt(i);**  **}**  **System.out.println(rev.equals(og));**  **}**  **}** |
| **61.UpperSmallerChange** |
| **import java.util.Arrays;**  **public class UpperSmallerChange {**  **public static void main(String[] args) {**  **String str="yara intha paiyan";**  **char[] c=str.toCharArray();**  **System.out.println(Arrays.toString(c));**  **for(int i=0;i<str.length();i++){**  **if (i%2==0&&Character.isLowerCase(c[i])) {**  **c[i]= Character.toUpperCase(c[i]);**  **}**  **str=new String(c);**  **}**  **System.out.println(str);**  **}}** |
| **62.FirstNonRepeatChar** |
| **public class FirstNonRepeatChar {**  **public static void main(String[] args) {**  **String str="programming";**  **char c=' ';**  **boolean dup=false;**  **for(int i =0;i<str.length();i++){**  **c=str.charAt(i);**  **dup=false;**  **for(int j=0;j<str.length();j++){**  **if(i!=j&&c==str.charAt(j)){**  **dup=true;**  **}**  **}**  **if (!dup) {**  **System.out.println("the 1st non repeat character  "+str.charAt(i));**  **break;**  **}        }**  **}**  **}** |
| **63.CheckDup** |
| **public class CheckDup {**  **public static void main(String[] args) {**  **String str="no duplicate";**  **boolean dup=false;**  **for(int i=0;i<str.length();i++){**  **char c=str.charAt(i);**  **for(int j=i+1;j<str.length();j++){**  **if(i!=j&&c==str.charAt(j)) {**  **dup=true;**  **break;**  **}**  **}**  **if (dup)    break;**  **}**  **if (dup) System.out.println("yes duplicate are there");**  **else System.out.println("no duplicates is here");**    **}**  **}** |
| **64. RemoveAdjacentn\_1** |
| **public class RemoveAdjacentn\_1 {**  **public static void main(String[] args) {**  **String str="programminnngg";**  **StringBuilder res=new StringBuilder();**  **StringBuilder copy=new StringBuilder();**  **int i=0;**  **while (i<str.length()) {**  **char c=str.charAt(i);**  **int count=1;**  **if (i+1<str.length()&&c==str.charAt(i+1)) {**  **count++;**  **i++;**  **}**  **if (count>1) {**  **copy.append(String.valueOf(c).repeat(count-1));**  **res.append(copy);**  **copy.setLength(0);**  **}else{**  **res.append(c);**  **}**  **i++;**  **}**  **System.out.println(res);**  **}**  **}** |
| **65.ReverseEachWordS** |
| **import java.util.Arrays;**  **public class ReverseEachWordS {**  **public static void main(String[] args) {**  **String str="vijayakumar nan than da";**  **StringBuilder result=new StringBuilder();**  **String[]words=str.split("\\s+");**  **for (String word : words) {**  **for(int i=word.length()-1;i>=0;i--){**  **result.append(word.charAt(i));**  **}**  **result.append(" ");**  **}**  **System.out.println(result);**  **System.out.println(Arrays.toString(words));**  **}**  **}** |
| **66.AnagramCheck** |
| **public class AnagramCheck {**  **public static void main(String[] args) {**    **String str1="vijay";**  **String str2="vijay";**  **int arr[]=new int[128];**  **if (str1.length()!=str2.length()){**  **System.out.println("the given string not anagram");**  **return; }**  **for (char c : str1.toCharArray()) {**  **arr[c]++;**  **}**  **for (char c : str2.toCharArray()) {**  **arr[c]--;**  **}**  **boolean isAnagram=true;**  **for (int i : arr) {**  **if (i!=0) {**  **isAnagram=false;**  **break;**  **}**  **}**  **if (isAnagram) {**  **System.out.println("the given string is an anagram");**  **}else{**  **System.out.println("the given string not an anagram");**  **}**  **}**  **}** |
| **67. RemoveSpecialChar** |
| **public class RemoveSpecialChar {**  **public static void main(String[] args) {**  **String str="vnj^ 808( &@&#@knBH!#H)";**  **str=str.replaceAll("[^a-zA-Z0-9\s]", "");**  **System.out.println(str);**  **}**  **}** |
| **68. CheckFrequency** |
| **public class CheckFrequency {**  **public static void main(String[] args) {**  **String str="vijayakuumar";**  **int []charCount=new int[128];**  **for (char c : str.toCharArray()) {**  **charCount[c]++;**  **}**  **for (int i=0;i<charCount.length;i++) {**  **if (charCount[i]>0) {**  **System.out.println("the frequency of "+(char)i+" is present "+charCount[i]+" times" );**  **}}**  **}**  **}** |
| **69 FindSubstring** |
| **import java.util.ArrayList;**  **public class FindSubstring {**  **public static void main(String[] args) {**  **String str="vijay";**  **ArrayList<String> subStrings=new ArrayList<>();**  **for(int i=0;i<str.length();i++){**  **for(int j=i+1;j<=str.length();j++){**  **subStrings.add(str.substring(i,j));**  **}}**  **System.out.println(subStrings);}**  **}** |
| **70.SumOfDigits(with and without recurision)** |
| **public class SumOfDigits {**  **public static void main(String[] args) {**  **int num=4587;**  **System.out.println (recursion(num));**  **System.out.println(normal(num));**  **}**  **static int recursion(int num){**  **if (num==0) return 0;**  **return (num%10) + recursion(num/10);**  **}**  **static int normal(int num){**  **int digit=0;**  **while(num!=0){**  **digit+=num%10;**  **num/=10;**  **}**  **return digit;**  **}**  **}** |
| **71.FindGcdLcm** |
| **public class FindGcdLcm {**  **public static void main(String[] args) {**    **int a =50,y=a;**  **int b=54,x=b;**  **while (b!=0) {**  **int temp=b;**  **b=a%b;**  **a=temp;**  **}**  **int gcd=a;**  **int lcm=(x\*y)/gcd;**  **System.out.println(gcd);**  **System.err.println(lcm);**  **}**  **}** |
| **72. PrimeUntilN** |
| **public class PrimeUntilN {**  **public static void main(String[] args) {**  **int times=5542;**  **for (int i=0;i<=times;i++){**  **if(isPrime(i)){**  **System.out.println("the prime number between "+times+" = "+i);**  **}**  **}**  **}**  **static boolean isPrime(int num){**  **if (num<=1)return false;**  **for(int i=2;i<=Math.sqrt(num);i++){**  **if (num%i==0) {**  **return false;**  **}**  **}**  **return  true;**  **}**  **}** |
| **73. ToBinary** |
| **public class ToBinary {**  **public static void main(String[] args) {**  **int decimal=50;**  **if (decimal<=0) {**  **System.out.println(decimal);**  **return;**  **}**  **StringBuilder result=new StringBuilder();**  **while (decimal>0) {**  **result.insert(0,decimal%2);**  **decimal/=2;**  **}**  **System.out.println(result);**  **}**  **}** |
| **74.ToDecimal** |
| **public class ToDecimal {**  **public static void main(String[] args) {**    **String binary="0000000011";**  **if (!binary.matches("[10]+")) {**  **System.out.println("invalid number");**  **return;**  **}**  **int decimal=0;**  **for (int i=0;i<binary.length();i++) {**  **decimal=decimal\*2+(binary.charAt(i)-'0');**  **}**  **System.out.println(decimal);**  **}**  **}** |
| **75. SumOfNatural** |
| **public class SumOfNatural {**  **public static void main(String[] args) {**  **int n=5;**  **System.out.println(loop(n));**  **System.out.println(formula(n));**  **}**  **static int loop(int n){**  **int sum=0;**  **for(int i=1;i<=n;i++){**  **sum+=i;**  **}**  **return sum;**  **}**  **static int formula(int n){**  **return  n\*(n+1)/2;**  **}**  **}** |
| **76. EvenAndOddToN** |
| **public class EvenAndOddToN {**  **public static void main(String[] args) {**  **int n = 5;**  **System.out.println("Sum of Even Numbers using Loop: " + sumUsingLoopEven(n));**  **System.out.println("Sum of odd Numbers using Loop: " + sumUsingLoopOdd(n));**  **System.out.println("Sum of Even Numbers using Formula: " + sumUsingFormulaEven(n));**  **System.out.println("Sum of odd Numbers using Formula: " + sumUsingFormulaOdd(n));**  **}**  **static int sumUsingLoopEven(int n) {**  **int sum = 0;**  **for (int i = 1; i <= n; i++) {**  **if (i % 2 == 0) {**  **sum += i;**  **}**  **}**  **return sum;**  **}**  **static int sumUsingLoopOdd(int n) {**  **int sum=0;**  **for (int i=1;i<=n;i++) {**  **if (i%2!=0) {**  **sum+=i;**  **}**  **}**  **return sum;**  **}**  **static int sumUsingFormulaEven(int n) {**  **int m = n / 2;**  **return m \* (m + 1);**  **}**  **static int sumUsingFormulaOdd(int n) {**  **int m=(n+1)/2;**  **return m\*m;**  **}**  **}** |
| **77.SquareRoot1toN** |
| **public class SquareRoot1toN {**  **public static void main(String[] args) {**  **int n=50;**  **usingLoop(n);**  **usingFormula(n);**  **}**  **static void usingLoop(int n){**  **int sum=0;**  **for(int i=0;i<=n;i++){**  **sum+=i\*i;**  **}**  **System.out.println(sum);**  **}**  **static void usingFormula(int n){**  **int sum=(n\*(n+1)\*(2\*n+1))/6;**  **System.out.println(sum);**  **}**  **}** |
| **78. SumOfEvenOddSqrt** |
| **public class SumOfEvenOddSqrt {**  **public static void main(String[] args) {**  **int n=5;**  **SqrtEvenOddLoop(n);**  **}**  **static void SqrtEvenOddLoop(int n){**  **long evenSqrt=0;**  **long oddSqrt=0;**  **for(int i=1;i<=n;i++){**  **if (i%2==0) {**  **evenSqrt+=i\*i;**  **}else{**  **oddSqrt+=i\*i;**  **}**  **}**  **System.out.println(evenSqrt);**  **System.out.println(oddSqrt);**  **}**  **}** |
| **79. Find\_N\_Perfect** |
| **public class Find\_N\_Perfect {**  **public static void main(String[] args) {**  **int n=496;**  **int sum=0;**  **for(int i=1;i<=n/2;i++){**  **if (n%i==0) {**  **sum+=i;**  **}}**  **if (sum==n) {**  **System.out.println("its perfect number");**  **}else{**  **System.out.println("its not an perfect");**  **}**  **}**  **}** |
| **80. CountDigitsInNum** |
| **public class CountDigitsInNum {**  **public static void main(String[] args) {**  **int num=123456789;**  **int count=0;**  **while (num!=0) {**  **num/=10;**  **count++;**  **}**  **System.out.println(count);**  **}**  **}** |
| **81.sum of prime upto n** |
| **public class SumOfPrimeUptoN {**  **public static void main(String[] args) {**  **int n=50;**  **int sum=0;**  **for(int i=2;i<=n;i++){**  **if (isPrime(i))**  **{**  **sum+=i;**  **}**  **}**  **System.out.println(sum);**  **}**  **static boolean isPrime(int n){**  **if (n<2) return false;**  **for (int i=2;i<=Math.sqrt(n);i++){**  **if (n%i==0) {**  **return false;**  **}**  **}**  **return true;**  **}**  **}** |
| **82.PowerOfNum** |
| **public class PowerOfNum {**  **public static void main(String[] args) {**  **int base=3;**  **int exponent=3;**  **int result=1;**  **for(int i=1;i<=exponent;i++){**  **result\*=base;**  **}**  **System.out.println(result);**  **}**  **}** |
| **83. AmstrongNumInRange** |
| **public class AmstrongNumInRange {**  **public static void main(String[] args) {**  **int start =100;**  **int end=170;**  **if (start<end) {**  **for (int i=start ;i<=end;i++) {**  **if (isAmstrong(i)) {**  **System.out.println(i+" this number is amstrong");**  **}**  **}**  **}else{**  **System.out.println("the range is not correct");**  **}**  **}**  **static boolean isAmstrong(int n){**  **int digit=String.valueOf(n).length();**  **int sum=0;**  **int og=n;**  **while (n!=0) {**  **int f=n%10;**  **sum+=(int)Math.pow(f, digit);**  **n/=10;**  **}**  **return (sum==og);**  **}**  **}** |
| **84. FindSmallestLargestinDigit** |
| **public class FindSmallestLargestinDigit {**  **public static void main(String[] args) {**  **int num=2306;**  **int smallest=9;**  **int largest=0;**  **while(num!=0){**  **int digit=num%10;**  **if (digit>largest)largest=digit;**  **if (digit<smallest)smallest=digit;**  **num/=10;**  **}**  **System.out.println("the largest number is :: "+largest);**  **System.out.println("the smallest number is :: "+smallest);**  **}**  **}** |
| **85. PrintNumbersInword** |
| **public class PrintNumbersInword {**  **public static void main(String[] args) {**  **String[]numbers={"zero","one","two","three","four","five","six","seven","eight","nine"};**  **int num=120;**  **if (num==0) {**  **System.out.println("zero");**  **return;**  **};**  **String numStr = String.valueOf(num);**  **for (char digit : numStr.toCharArray()) {**  **System.out.print(numbers[digit - '0'] + " ");**  **}**    **}**    **}** |
| **86.CheckAmicablePairs** |
| **public class Amicable {**  **public static void main(String[] args) {**  **int a=220;**  **int b=284;**  **System.out.println(isAmicable(a, b));**  **}**  **static boolean isAmicable(int a,int b){**  **return (sumOfDivisor(a)==b&&sumOfDivisor(b)==a);**  **}**  **static int sumOfDivisor(int num){**  **int sum=0;**  **for(int i=1;i<=num/2;i++){**  **if (num%i==0) {**  **sum+=i;**  **}**  **}**  **return sum;**  **}**  **}** |
| **87. SumOfDigitUntilOne** |
| **public class SumOfDigitUntilOne {**  **public static void main(String[] args) {**  **int n=541;**  **if(n<0){**  **System.out.println(n);**  **}**  **System.out.println(1+(n-1)%9);**  **}**  **}** |
| **88. SumOfPrimeFactor** |
| **public class SumOfPrimeFactor {**  **public static void main(String[] args) {**  **int n=45;**  **int sum=0;**  **while (n%2==0) {**  **sum+=2;**  **n/=2;**  **}**  **for (int i=3;i\*i<=n;i+=2) {**  **while (n % i ==0) {**  **sum+=i;**  **n/=i;**  **}**  **}**  **if (n > 2) {**  **sum+=n;**  **}**  **System.out.println(sum);**  **}**  **}** |
| **89.CollatzConjecture** |
| **public class CollatzConjecture {**  **public static void main(String[] args) {**  **int num=5;**  **if (num>1) {**  **while (num!=1) {**  **System.out.print(num+" --> ");**  **if (num%2==0) {**  **num/=2;**  **}else{**  **num=3\*num+1;**  **}**  **}**    **}**  **}**  **}** |
| **90.XylemPhloem** |
| **import java.util.Scanner;**  **public class XylemPhloem {**  **public static void main(String[] args) {**  **Scanner sc = new Scanner(System.in);**  **System.out.print("Enter a number: ");**  **int num = sc.nextInt();**  **sc.close();**  **int firstDigit = num % 10;**  **int lastDigit = 0;**  **int middleSum = 0;**  **num /= 10; // Remove last digit**  **while (num > 9) {**  **middleSum += num % 10;**  **num /= 10;**  **}**  **lastDigit = num;**  **int boundarySum = firstDigit + lastDigit;**  **if (boundarySum > middleSum) {**  **System.out.println("Xylem Number");**  **} else {**  **System.out.println("Phloem Number");**  **}**  **}**  **}** |
| **91. Encapsulation** |
| **import java.util.Scanner;**  **public class Encapsulation {**  **private int stdId;**  **private int stdMark;**  **public void setId(int id) {**  **this.stdId = id;**  **}**  **public void setMark(int mark) {**  **this.stdMark = mark;**  **}**  **public boolean result() {**  **return stdMark >= 50 && stdMark < 100;**  **}**  **public static void main(String[] args) {**  **Encapsulation en = new Encapsulation();**  **en.setId(5);**  **en.setMark(50);**  **System.out.println("The ID is: " + en.stdId);**  **if (en.result())**  **System.out.println("Pass");**  **else**  **System.out.println("Fail");**  **}**  **}** |
| **92.Single Inheritance** |
| **class DadSI {**  **static void dance(){**  **System.out.println("i can dance");**  **}}**  **class son extends DadSI{**  **static void sing(){**  **System.out.println("i can singe");**  **}**  **}**  **public class SingleInheritance{**  **public static void main(String[] args) {**  **son s=new son();**  **s.dance();**  **s.sing();**  **}**  **}** |
| **92.muti-level inheritance** |
| **class GrandFather {**  **static void dance(){**  **System.out.println("i can dance");**  **}**  **}**  **class Dad extends GrandFather {**  **static void sing(){**  **System.out.println("i can sing");**  **}**  **}**  **class son extends Dad{**  **static void act(){**  **System.out.println("i can act");**  **}**    **}**  **public class MultiLevelIn {**  **public static void main(String[] args) {**  **son s=new son();**  **s.dance();**  **s.sing();**  **s.act();**  **}**  **}** |
| **93. Hirarchical** |
| **class dad{**  **static void dance(){**  **System.out.println("i can dance");**  **}**  **}**  **class son extends dad{**  **static void sing(){**  **System.out.println("i can sing");**  **}**  **}**  **class grandSon extends  son{**  **static void joke(){**  **System.out.println("i can joke");**  **} }**  **class grandDaugther extends son{**  **static void cook(){**  **System.out.println("i can cook");**  **}**  **}**  **public class Hirarchical {**  **public static void main(String[] args) {**  **grandSon gS=new grandSon();**  **grandDaugther gD=new grandDaugther();**  **gS.sing();**  **gS.dance();**  **gS.joke();**  **gD.dance();**  **}**  **}** |
| **94. MultipleIn** |
| **interface Dad {**  **void dance();**  **}**  **interface Mom {**  **void sing();**    **}**  **class son implements Mom,Dad{**  **public  void dance(){**  **System.out.println("i can dance");**  **}**  **public void sing(){**  **System.out.println("i can sing");**  **}**  **public void study(){**  **System.out.println("i can study");**  **}**  **}**  **public class MultipleIn {**  **public static void main(String[] args) {**    **}**  **}** |
| **95. HybridIn** |
| **interface GrandFather {**  **void act();**  **}**  **interface grandMother{**  **void cook();**    **}**  **class mom implements grandMother,GrandFather {**  **public void cook(){**  **System.out.println("i can cook");**  **}**    **public void act(){**  **System.out.println("i can act");**  **}**  **static void sing(){**  **System.out.println("i can sing");**  **}**  **}**  **class  daugther extends mom {**  **void dance(){**  **System.out.println("i can dance");**  **}**  **}**  **public class HybridIn {**  **public static void main(String[] args) {**  **daugther d=new daugther();**  **d.cook();**  **d.act();**  **d.dance();**  **d.sing();**  **}**  **}** |
| **96. methodOverloading** |
| **public class methodOverloading {**  **void m1(){**  **System.out.println("i don't have anything");**  **}**  **void m1(int a){**  **System.out.println("i have int a "+a);**  **}**  **void m1(int a ,int b){**  **System.out.println("i have in a "+a+" and int b"+b);**  **}**  **public static void main(String[] args) {**  **methodOverloading mO=new methodOverloading();**  **mO.m1();**  **mO.m1(0);**  **mO.m1(1, 2);**  **}**  **}** |
| **97. ConstructorOverloading** |
| **public class ConstructorOverloading {**  **ConstructorOverloading(int a, int b) {**  **System.out.println("int a = " + a + ", int b = " + b);**  **}**  **ConstructorOverloading(int a, String b) {**  **System.out.println("The id is " + a + ", The name is " + b);**  **}**  **public static void main(String[] args) {**  **ConstructorOverloading co = new ConstructorOverloading(21, "vijay");**  **ConstructorOverloading co1 = new ConstructorOverloading(50, 58);**  **}**  **}** |
| **98.** **MethodOverriding** |
| **class Dad {**  **public void cook() {**  **System.out.println("I know cooking in Indian style");**  **}**  **}**  **class Son extends Dad {**  **public void cook() {**  **System.out.println("I know cooking in Chinese style");**  **}**  **}**  **public class MethodOverriding {**  **public static void main(String[] args) {**  **Dad d = new Dad();**  **Son s = new Son();**  **d.cook();**  **s.cook();**  **}**  **}** |
| **99. Abstraction** |
| **abstract class StudentManagement {**  **abstract void id();**  **abstract void name();**  **abstract void marks();**  **}**  **class StudentXII extends StudentManagement {**  **int id;**  **String name;**  **int mark;**  **public StudentXII(int id, String name, int mark) {**  **this.id = id;**  **this.name = name;**  **this.mark = mark;**  **}**  **public void id() {**  **System.out.println("The ID is " + id);**  **}**  **public void name() {**  **System.out.println("The Name is " + name);**  **}**  **public void marks() {**  **System.out.println("The Marks are " + mark);**  **}**  **}**  **public class Abstraction {**  **public static void main(String[] args) {**  **StudentXII student = new StudentXII(6, "Vijay", 500);**  **student.id();**  **student.name();**  **student.marks();**  **}**  **}** |
| **100. ArrayList** |
| **import java.util.\*;**  **public class ArrayListExample {**  **public static void main(String[] args) {**  **// Creating an ArrayList**  **ArrayList<String> list = new ArrayList<>();**    **// 1. add() - Adding elements**  **list.add("Apple");**  **list.add("Banana");**  **list.add("Cherry");**  **System.out.println("ArrayList: " + list);**    **// 2. add(index, element) - Adding at specific index**  **list.add(1, "Blueberry");**  **System.out.println("After adding at index 1: " + list);**    **// 3. get() - Get element at index**  **System.out.println("Element at index 2: " + list.get(2));**    **// 4. set() - Update element at index**  **list.set(1, "Blackberry");**  **System.out.println("After setting index 1: " + list);**    **// 5. remove(index) - Remove element by index**  **list.remove(2);**  **System.out.println("After removing index 2: " + list);**    **// 6. remove(object) - Remove by value**  **list.remove("Apple");**  **System.out.println("After removing 'Apple': " + list);**    **// 7. contains() - Check if element exists**  **System.out.println("Contains 'Banana'?: " + list.contains("Banana"));**    **// 8. size() - Get the size of ArrayList**  **System.out.println("Size of list: " + list.size());**    **// 9. isEmpty() - Check if list is empty**  **System.out.println("Is list empty?: " + list.isEmpty());**    **// 10. indexOf() - Get index of element**  **System.out.println("Index of 'Banana': " + list.indexOf("Banana"));**    **// 11. forEach() - Iterate using forEach**  **System.out.print("Iterating: ");**  **list.forEach(e -> System.out.print(e + " "));**  **System.out.println();**    **// 12. sort() - Sorting elements**  **Collections.sort(list);**  **System.out.println("After sorting: " + list);**    **// 13. clear() - Remove all elements**  **list.clear();**  **System.out.println("After clearing list: " + list);**    **// 14. addAll() - Adding a collection**  **list.addAll(Arrays.asList("Mango", "Orange", "Grapes"));**  **System.out.println("After adding multiple elements: " + list);**    **// 15. clone() - Clone the list**  **ArrayList<String> clonedList = (ArrayList<String>) list.clone();**  **System.out.println("Cloned list: " + clonedList);**  **}**  **}** |
| **101. LinkedList** |
| **import java.util.\*;**  **public class LinkedListExample {**  **public static void main(String[] args) {**  **// Creating a LinkedList**  **LinkedList<String> list = new LinkedList<>();**    **// 1. add() - Adding elements**  **list.add("Apple");**  **list.add("Banana");**  **list.add("Cherry");**  **System.out.println("LinkedList: " + list);**    **// 2. addFirst() - Add element at the beginning**  **list.addFirst("Mango");**  **System.out.println("After adding first: " + list);**    **// 3. addLast() - Add element at the end**  **list.addLast("Grapes");**  **System.out.println("After adding last: " + list);**    **// 4. get() - Get element at index**  **System.out.println("Element at index 2: " + list.get(2));**    **// 5. set() - Update element at index**  **list.set(1, "Blackberry");**  **System.out.println("After setting index 1: " + list);**    **// 6. remove(index) - Remove element by index**  **list.remove(2);**  **System.out.println("After removing index 2: " + list);**    **// 7. remove(object) - Remove by value**  **list.remove("Apple");**  **System.out.println("After removing 'Apple': " + list);**    **// 8. removeFirst() - Remove first element**  **list.removeFirst();**  **System.out.println("After removing first: " + list);**    **// 9. removeLast() - Remove last element**  **list.removeLast();**  **System.out.println("After removing last: " + list);**    **// 10. contains() - Check if element exists**  **System.out.println("Contains 'Banana'?: " + list.contains("Banana"));**    **// 11. size() - Get the size of LinkedList**  **System.out.println("Size of list: " + list.size());**    **// 12. isEmpty() - Check if list is empty**  **System.out.println("Is list empty?: " + list.isEmpty());**    **// 13. indexOf() - Get index of element**  **System.out.println("Index of 'Banana': " + list.indexOf("Banana"));**    **// 14. forEach() - Iterate using forEach**  **System.out.print("Iterating: ");**  **list.forEach(e -> System.out.print(e + " "));**  **System.out.println();**    **// 15. sort() - Sorting elements**  **Collections.sort(list);**  **System.out.println("After sorting: " + list);**    **// 16. clear() - Remove all elements**  **list.clear();**  **System.out.println("After clearing list: " + list);**    **// 17. addAll() - Adding a collection**  **list.addAll(Arrays.asList("Mango", "Orange", "Grapes"));**  **System.out.println("After adding multiple elements: " + list);**    **// 18. clone() - Clone the list**  **LinkedList<String> clonedList = (LinkedList<String>) list.clone();**  **System.out.println("Cloned list: " + clonedList);**  **}**  **}** |
| **102. VectorExample** |
| **import java.util.\*;**  **public class VectorExample {**  **public static void main(String[] args) {**  **// Creating a Vector**  **Vector<String> list = new Vector<>();**    **// 1. add() - Adding elements**  **list.add("Apple");**  **list.add("Banana");**  **list.add("Cherry");**  **System.out.println("Vector: " + list);**    **// 2. add(index, element) - Adding at specific index**  **list.add(1, "Blueberry");**  **System.out.println("After adding at index 1: " + list);**    **// 3. get() - Get element at index**  **System.out.println("Element at index 2: " + list.get(2));**    **// 4. set() - Update element at index**  **list.set(1, "Blackberry");**  **System.out.println("After setting index 1: " + list);**    **// 5. remove(index) - Remove element by index**  **list.remove(2);**  **System.out.println("After removing index 2: " + list);**    **// 6. remove(object) - Remove by value**  **list.remove("Apple");**  **System.out.println("After removing 'Apple': " + list);**    **// 7. contains() - Check if element exists**  **System.out.println("Contains 'Banana'?: " + list.contains("Banana"));**    **// 8. size() - Get the size of Vector**  **System.out.println("Size of list: " + list.size());**    **// 9. isEmpty() - Check if list is empty**  **System.out.println("Is list empty?: " + list.isEmpty());**    **// 10. indexOf() - Get index of element**  **System.out.println("Index of 'Banana': " + list.indexOf("Banana"));**    **// 11. forEach() - Iterate using forEach**  **System.out.print("Iterating: ");**  **list.forEach(e -> System.out.print(e + " "));**  **System.out.println();**    **// 12. sort() - Sorting elements**  **Collections.sort(list);**  **System.out.println("After sorting: " + list);**    **// 13. clear() - Remove all elements**  **list.clear();**  **System.out.println("After clearing list: " + list);**    **// 14. addAll() - Adding a collection**  **list.addAll(Arrays.asList("Mango", "Orange", "Grapes"));**  **System.out.println("After adding multiple elements: " + list);**    **// 15. clone() - Clone the Vector**  **Vector<String> clonedList = (Vector<String>) list.clone();**  **System.out.println("Cloned list: " + clonedList);**  **}**  **}** |
| **103. StackExample** |
| **import java.util.\*;**  **public class StackExample {**  **public static void main(String[] args) {**  **// Creating a Stack**  **Stack<String> stack = new Stack<>();**    **// 1. push() - Adding elements**  **stack.push("Apple");**  **stack.push("Banana");**  **stack.push("Cherry");**  **System.out.println("Stack: " + stack);**    **// 2. peek() - Get the top element**  **System.out.println("Top element: " + stack.peek());**    **// 3. pop() - Remove and return top element**  **System.out.println("Popped element: " + stack.pop());**  **System.out.println("Stack after pop: " + stack);**    **// 4. search() - Get position of an element**  **System.out.println("Position of 'Apple': " + stack.search("Apple"));**    **// 5. isEmpty() - Check if stack is empty**  **System.out.println("Is stack empty?: " + stack.isEmpty());**    **// 6. size() - Get the size of Stack**  **System.out.println("Size of stack: " + stack.size());**    **// 7. contains() - Check if element exists**  **System.out.println("Contains 'Banana'?: " + stack.contains("Banana"));**    **// 8. forEach() - Iterate using forEach**  **System.out.print("Iterating: ");**  **stack.forEach(e -> System.out.print(e + " "));**  **System.out.println();**    **// 9. clear() - Remove all elements**  **stack.clear();**  **System.out.println("After clearing stack: " + stack);**    **// 10. addAll() - Adding multiple elements**  **stack.addAll(Arrays.asList("Mango", "Orange", "Grapes"));**  **System.out.println("After adding multiple elements: " + stack);**    **// 11. clone() - Clone the stack**  **Stack<String> clonedStack = (Stack<String>) stack.clone();**  **System.out.println("Cloned stack: " + clonedStack);**  **}**  **}** |
| **104. QueueExample** |
| **import java.util.\*;**  **public class QueueExample {**  **public static void main(String[] args) {**  **// Creating a Queue (LinkedList implementation)**  **Queue<String> queue = new LinkedList<>();**    **// 1. add() - Adding elements**  **queue.add("Apple");**  **queue.add("Banana");**  **queue.add("Cherry");**  **System.out.println("Queue: " + queue);**    **// 2. peek() - Get the head element**  **System.out.println("Head element: " + queue.peek());**    **// 3. poll() - Remove and return head element**  **System.out.println("Polled element: " + queue.poll());**  **System.out.println("Queue after poll: " + queue);**    **// 4. remove() - Remove head element**  **System.out.println("Removed element: " + queue.remove());**  **System.out.println("Queue after remove: " + queue);**    **// 5. contains() - Check if element exists**  **System.out.println("Contains 'Banana'?: " + queue.contains("Banana"));**    **// 6. size() - Get the size of Queue**  **System.out.println("Size of queue: " + queue.size());**    **// 7. isEmpty() - Check if queue is empty**  **System.out.println("Is queue empty?: " + queue.isEmpty());**    **// 8. forEach() - Iterate using forEach**  **System.out.print("Iterating: ");**  **queue.forEach(e -> System.out.print(e + " "));**  **System.out.println();**    **// 9. clear() - Remove all elements**  **queue.clear();**  **System.out.println("After clearing queue: " + queue);**    **// 10. addAll() - Adding multiple elements**  **queue.addAll(Arrays.asList("Mango", "Orange", "Grapes"));**  **System.out.println("After adding multiple elements: " + queue);**  **}**  **}** |
| **105. HashSetExample** |
| **import java.util.\*;**  **public class HashSetExample {**  **public static void main(String[] args) {**  **// Creating a HashSet**  **HashSet<String> set = new HashSet<>();**    **// 1. add() - Adding elements**  **set.add("Apple");**  **set.add("Banana");**  **set.add("Cherry");**  **System.out.println("HashSet: " + set);**    **// 2. add() - Adding duplicate element (will not be added)**  **set.add("Apple");**  **System.out.println("After adding duplicate: " + set);**    **// 3. contains() - Check if element exists**  **System.out.println("Contains 'Banana'?: " + set.contains("Banana"));**    **// 4. remove() - Remove an element**  **set.remove("Cherry");**  **System.out.println("After removing 'Cherry': " + set);**    **// 5. size() - Get the size of HashSet**  **System.out.println("Size of set: " + set.size());**    **// 6. isEmpty() - Check if set is empty**  **System.out.println("Is set empty?: " + set.isEmpty());**    **// 7. forEach() - Iterate using forEach**  **System.out.print("Iterating: ");**  **set.forEach(e -> System.out.print(e + " "));**  **System.out.println();**    **// 8. clear() - Remove all elements**  **set.clear();**  **System.out.println("After clearing set: " + set);**    **// 9. addAll() - Adding multiple elements**  **set.addAll(Arrays.asList("Mango", "Orange", "Grapes"));**  **System.out.println("After adding multiple elements: " + set);**  **}**  **}** |
| **106. LinkedHashSetExample** |
| **import java.util.\*;**  **public class LinkedHashSetExample {**  **public static void main(String[] args) {**  **// Creating a LinkedHashSet**  **LinkedHashSet<String> set = new LinkedHashSet<>();**    **// 1. add() - Adding elements**  **set.add("Apple");**  **set.add("Banana");**  **set.add("Cherry");**  **System.out.println("LinkedHashSet: " + set);**    **// 2. add() - Adding duplicate element (will not be added)**  **set.add("Apple");**  **System.out.println("After adding duplicate: " + set);**    **// 3. contains() - Check if element exists**  **System.out.println("Contains 'Banana'?: " + set.contains("Banana"));**    **// 4. remove() - Remove an element**  **set.remove("Cherry");**  **System.out.println("After removing 'Cherry': " + set);**    **// 5. size() - Get the size of LinkedHashSet**  **System.out.println("Size of set: " + set.size());**    **// 6. isEmpty() - Check if set is empty**  **System.out.println("Is set empty?: " + set.isEmpty());**    **// 7. forEach() - Iterate using forEach**  **System.out.print("Iterating: ");**  **set.forEach(e -> System.out.print(e + " "));**  **System.out.println();**    **// 8. clear() - Remove all elements**  **set.clear();**  **System.out.println("After clearing set: " + set);**    **// 9. addAll() - Adding multiple elements**  **set.addAll(Arrays.asList("Mango", "Orange", "Grapes"));**  **System.out.println("After adding multiple elements: " + set);**  **}**  **}** |
| **107.TreeSetExample** |
| **import java.util.\*;**  **public class TreeSetExample {**  **public static void main(String[] args) {**  **// Creating a TreeSet**  **TreeSet<String> set = new TreeSet<>();**    **// 1. add() - Adding elements (sorted order)**  **set.add("Apple");**  **set.add("Banana");**  **set.add("Cherry");**  **System.out.println("TreeSet: " + set);**    **// 2. add() - Adding duplicate element (will not be added)**  **set.add("Apple");**  **System.out.println("After adding duplicate: " + set);**    **// 3. contains() - Check if element exists**  **System.out.println("Contains 'Banana'?: " + set.contains("Banana"));**    **// 4. remove() - Remove an element**  **set.remove("Cherry");**  **System.out.println("After removing 'Cherry': " + set);**    **// 5. size() - Get the size of TreeSet**  **System.out.println("Size of set: " + set.size());**    **// 6. isEmpty() - Check if set is empty**  **System.out.println("Is set empty?: " + set.isEmpty());**    **// 7. first() - Get first element**  **System.out.println("First element: " + set.first());**    **// 8. last() - Get last element**  **System.out.println("Last element: " + set.last());**    **// 9. forEach() - Iterate using forEach**  **System.out.print("Iterating: ");**  **set.forEach(e -> System.out.print(e + " "));**  **System.out.println();**    **// 10. clear() - Remove all elements**  **set.clear();**  **System.out.println("After clearing set: " + set);**    **// 11. addAll() - Adding multiple elements**  **set.addAll(Arrays.asList("Mango", "Orange", "Grapes"));**  **System.out.println("After adding multiple elements: " + set);**  **}**  **}** |
| **108. HashMapExample** |
| **import java.util.\*;**  **public class HashMapExample {**  **public static void main(String[] args) {**  **// Creating a HashMap**  **HashMap<Integer, String> map = new HashMap<>();**    **// 1. put() - Adding elements**  **map.put(1, "Apple");**  **map.put(2, "Banana");**  **map.put(3, "Cherry");**  **System.out.println("HashMap: " + map);**    **// 2. put() - Adding duplicate key (updates value)**  **map.put(1, "Avocado");**  **System.out.println("After updating key 1: " + map);**    **// 3. get() - Get value by key**  **System.out.println("Value of key 2: " + map.get(2));**    **// 4. getOrDefault() - Get value with default if key is absent**  **System.out.println("Value of key 4 (default): " + map.getOrDefault(4, "Not Found"));**    **// 5. containsKey() - Check if key exists**  **System.out.println("Contains key 3?: " + map.containsKey(3));**    **// 6. containsValue() - Check if value exists**  **System.out.println("Contains value 'Banana'?: " + map.containsValue("Banana"));**    **// 7. remove() - Remove an element by key**  **map.remove(3);**  **System.out.println("After removing key 3: " + map);**    **// 8. size() - Get the size of HashMap**  **System.out.println("Size of map: " + map.size());**    **// 9. isEmpty() - Check if map is empty**  **System.out.println("Is map empty?: " + map.isEmpty());**    **// 10. keySet() - Get all keys**  **System.out.println("Keys: " + map.keySet());**    **// 11. values() - Get all values**  **System.out.println("Values: " + map.values());**    **// 12. entrySet() - Get all key-value pairs**  **System.out.println("Entries: " + map.entrySet());**    **// 13. forEach() - Iterate using forEach**  **System.out.println("Iterating:");**  **map.forEach((key, value) -> System.out.println(key + " -> " + value));**    **// 14. clear() - Remove all elements**  **map.clear();**  **System.out.println("After clearing map: " + map);**    **// 15. putAll() - Adding multiple elements**  **map.putAll(Map.of(4, "Mango", 5, "Orange", 6, "Grapes"));**  **System.out.println("After adding multiple elements: " + map);**  **}**  **}** |
| **109. LinkedHashMapExample** |
| **import java.util.\*;**  **public class LinkedHashMapExample {**  **public static void main(String[] args) {**  **// Creating a LinkedHashMap**  **LinkedHashMap<Integer, String> map = new LinkedHashMap<>();**    **// 1. put() - Adding elements**  **map.put(1, "Apple");**  **map.put(2, "Banana");**  **map.put(3, "Cherry");**  **System.out.println("LinkedHashMap: " + map);**    **// 2. put() - Adding duplicate key (updates value)**  **map.put(1, "Avocado");**  **System.out.println("After updating key 1: " + map);**    **// 3. get() - Get value by key**  **System.out.println("Value of key 2: " + map.get(2));**    **// 4. getOrDefault() - Get value with default if key is absent**  **System.out.println("Value of key 4 (default): " + map.getOrDefault(4, "Not Found"));**    **// 5. containsKey() - Check if key exists**  **System.out.println("Contains key 3?: " + map.containsKey(3));**    **// 6. containsValue() - Check if value exists**  **System.out.println("Contains value 'Banana'?: " + map.containsValue("Banana"));**    **// 7. remove() - Remove an element by key**  **map.remove(3);**  **System.out.println("After removing key 3: " + map);**    **// 8. size() - Get the size of LinkedHashMap**  **System.out.println("Size of map: " + map.size());**    **// 9. isEmpty() - Check if map is empty**  **System.out.println("Is map empty?: " + map.isEmpty());**    **// 10. keySet() - Get all keys**  **System.out.println("Keys: " + map.keySet());**    **// 11. values() - Get all values**  **System.out.println("Values: " + map.values());**    **// 12. entrySet() - Get all key-value pairs**  **System.out.println("Entries: " + map.entrySet());**    **// 13. forEach() - Iterate using forEach**  **System.out.println("Iterating:");**  **map.forEach((key, value) -> System.out.println(key + " -> " + value));**    **// 14. clear() - Remove all elements**  **map.clear();**  **System.out.println("After clearing map: " + map);**    **// 15. putAll() - Adding multiple elements**  **map.putAll(Map.of(4, "Mango", 5, "Orange", 6, "Grapes"));**  **System.out.println("After adding multiple elements: " + map);**  **}**  **}** |
| **110. TreeMapExample** |
| **import java.util.\*;**  **public class TreeMapExample {**  **public static void main(String[] args) {**  **// Creating a TreeMap**  **TreeMap<Integer, String> map = new TreeMap<>();**    **// 1. put() - Adding elements (sorted order by key)**  **map.put(3, "Cherry");**  **map.put(1, "Apple");**  **map.put(2, "Banana");**  **System.out.println("TreeMap: " + map);**    **// 2. put() - Adding duplicate key (updates value)**  **map.put(1, "Avocado");**  **System.out.println("After updating key 1: " + map);**    **// 3. get() - Get value by key**  **System.out.println("Value of key 2: " + map.get(2));**    **// 4. getOrDefault() - Get value with default if key is absent**  **System.out.println("Value of key 4 (default): " + map.getOrDefault(4, "Not Found"));**    **// 5. containsKey() - Check if key exists**  **System.out.println("Contains key 3?: " + map.containsKey(3));**    **// 6. containsValue() - Check if value exists**  **System.out.println("Contains value 'Banana'?: " + map.containsValue("Banana"));**    **// 7. remove() - Remove an element by key**  **map.remove(3);**  **System.out.println("After removing key 3: " + map);**    **// 8. size() - Get the size of TreeMap**  **System.out.println("Size of map: " + map.size());**    **// 9. isEmpty() - Check if map is empty**  **System.out.println("Is map empty?: " + map.isEmpty());**    **// 10. keySet() - Get all keys**  **System.out.println("Keys: " + map.keySet());**    **// 11. values() - Get all values**  **System.out.println("Values: " + map.values());**    **// 12. entrySet() - Get all key-value pairs**  **System.out.println("Entries: " + map.entrySet());**    **// 13. firstKey() - Get the first (lowest) key**  **System.out.println("First key: " + map.firstKey());**    **// 14. lastKey() - Get the last (highest) key**  **System.out.println("Last key: " + map.lastKey());**    **// 15. forEach() - Iterate using forEach**  **System.out.println("Iterating:");**  **map.forEach((key, value) -> System.out.println(key + " -> " + value));**    **// 16. clear() - Remove all elements**  **map.clear();**  **System.out.println("After clearing map: " + map);**    **// 17. putAll() - Adding multiple elements**  **map.putAll(Map.of(4, "Mango", 5, "Orange", 6, "Grapes"));**  **System.out.println("After adding multiple elements: " + map);**  **}**  **}** |