Package: ArrayExample

File: ArrayToListExample.java

```
 package ArrayExample;
import java.util.Arrays;
import java.util.List;
//Array is data structure and it store homogenious data.
public class ArrayToListExample {
  public static void main(String[] args) {
         String[] array = {"RAVI", "KIRAN", "ATUL", "CHAVAN"};
         // Convert array to List
         List<String> list = Arrays.asList(array);
         System.out.println("Converted List: " + list);
         System.out.println("original Array: " + Arrays.toString(array));
     }
}
```

File: BubbleSort.java

```
package ArrayExample;
import java.util.Arrays;
//Bubble Sort And find out Min and Max Element
public class BubbleSort {
 public static void main(String[] args) {
  int [] arr = {-3,3,7,2,8};
  System.out.println("Original Array :"+Arrays.toString(arr));

  int []sorted =bubbleSort(arr);
  System.out.println("Bubble Soted Array :"+Arrays.toString(sorted));

  //find out Min and Max
  System.out.println("Min Element from Array :"+sorted[0]);
  System.out.println("max Element from Array :"+sorted[sorted.length-1]);

 }
 private static int[] bubbleSort(int[] arr) {

  for (int i = 0; i < arr.length; i++) {
   for (int j = i+1; j < arr.length; j++) {
    if(arr[i] > arr[j]) {
     int temp = arr[i];
     arr[i] = arr[j];
     arr[j] = temp;
    }
   }
  }
  return arr;
 }
}
```

File: CheckSortedArray.java

```
package ArrayExample;
public class CheckSortedArray {
 public static void main(String[] args) {
//  int[] array = {10,22,5,23,9,4,-7};
  int[] array = {10,22,45,67,89,97};

  boolean flag=isSorted(array);
  if(flag) {
   System.out.println("Array is Sorted");
  }else {
   System.out.println("Array is not Sorted");
  }
 }

 public static boolean isSorted(int[] array) {
        for (int i = 0; i < array.length - 1; i++) {
            if (array[i] > array[i + 1]) { // If previous element is greater than next, not
sorted
                return false;
            }
        }
        return true;
    }
}
```

File: ConvertArrayListToArray.java

```
package ArrayExample;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
//To Convert ArrayList to Array--toArray()
//To Covert Array to ArrayList
public class ConvertArrayListToArray {
 public static void main(String[] args) {
  List<String> arrayList = new
ArrayList<>(Arrays.asList("Cricket","Football","Baseball","Hockey"));

  String[] array = arrayList.toArray(new String[0]);

  for (String item : array) {
            System.out.println(item);
        }

  //Approch 1st To Covert Array to ArrayList
  List<String> newList1 = new ArrayList<>();
  for (String arr : array) {
   newList1.add(arr);
  }
  System.out.println("New ArrayList :"+newList1);

  //Approch 2nd
  List<String> newList2 = new ArrayList<>(Arrays.asList(array));
  System.out.println("New ArrayList :"+newList2);
 }
}
```

File: ConvertArrayToArrayList.java

```
package ArrayExample;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
public class ConvertArrayToArrayList {
 public static void main(String[] args) {
  String city [] = {"Pune","Mumbai","Dehli","Chennai"};

  //convert Array to ArrayList
  List<String> list = Arrays.asList(city);
  System.out.println(list);

  //now Convert ArrayList to Array
   ArrayList<String> fruitList = new ArrayList<>();
   fruitList.add("Orange");
   fruitList.add("Apple");
   fruitList.add("Banana");

   String fruitArray[] = new String[fruitList.size()];
   fruitArray = fruitList.toArray(fruitArray);

   for (String str : fruitArray) {
           System.out.print(str + " ");
   }
 }
}
```

File: FindPositiveElementAverage.java

```
package ArrayExample;
import java.math.BigDecimal;
import java.util.ArrayList;
// Find Out Averae of Positive Number
public class FindPositiveElementAverage {
 public static void main(String[] args) {

  ArrayList<Integer> aList = new ArrayList<>();
  aList.add(-6);
  aList.add(2);
  aList.add(-56);
  aList.add(8);
  aList.add(-5);
  aList.add(25);
  calculateAvgForPositiveNumber(aList);
 }
 private static void calculateAvgForPositiveNumber(ArrayList<Integer> aList) {
  int count =0;
//  int sum = 0;
   BigDecimal sum = BigDecimal.ZERO;

  for (Integer element : aList) {
   if(element > 0) {
    count++;
//    sum+=element;
    sum = sum.add(BigDecimal.valueOf(element));
   }
  }
//  int average = sum/count;
  BigDecimal average = sum.divide(BigDecimal.valueOf(count), 2, BigDecimal.ROUND_HALF_UP);
  System.out.println("Average of Positve Number is "+average);
 }
}
```

File: LeftRotationOfArray.java

```
package ArrayExample;
import java.util.Arrays;
public class LeftRotationOfArray {
 public static void main(String[] args) {
  String []array= {"RAVI","KIRAN","ATUL","CHAVAN"};//{"KIRAN", "ATUL", "CHAVAN", "RAVI"}
  System.out.println("Original Array"+Arrays.toString(array));
  leftRotationArray(array);
 }
 private static void leftRotationArray(String[] array) {
  String temp=array[0];

  for(int i = 0; i < array.length-1; i++) {
   array[i] = array[i+1];
  }
  array[array.length-1]=temp;
  System.out.println("left rotation of Array"+Arrays.toString(array));
 }
}
```

File: MaxAndMinElement.java

```
package ArrayExample;
import java.util.Arrays;
public class MaxAndMinElement {
 public static void main(String[] args) {
  int[]arr= {1,-3,5,7,2};
  System.out.println("original array:-"+Arrays.toString(arr));

  findoutMinAndMaxElement(arr);
 }
 private static void findoutMinAndMaxElement(int[] arr) {
  for (int i = 0; i < arr.length; i++) {
            for(int j=i+1; j<arr.length; j++) {
             if(arr[i]>arr[j]) {
              int temp=arr[i];
              arr[i]=arr[j];
              arr[j]=temp;
             }
            }
        }
  System.out.println("Array is sort by Asc order:"+Arrays.toString(arr));
  System.out.println("Max Element From Array is:"+arr[arr.length - 1]);
  System.out.println("Min Element From Array is:"+arr[0]);
 }
}
```

File: MergeTwoArray.java

```
package ArrayExample;
import java.util.Arrays;
public class MergeTwoArray {
 public static void main(String[] args) {
  int[] arr1 = {1,2,5,7,9};
  int[] arr2 = {3,-9,7,3,5,8};
  System.out.println("original array:-"+Arrays.toString(arr1));
  System.out.println("original array:-"+Arrays.toString(arr2));

  //without inbuilt method
  mergeArray(arr1,arr2);

  //with inbuilt method
  int[] mergedArray = mergeArrays(arr1, arr2);
  System.out.println("Merged array: " + Arrays.toString(mergedArray));
 }
 private static void mergeArray(int[] arr1, int[] arr2) {
  int[] arr3 = new int[arr1.length + arr2.length];
  // Copy elements from arr1 to arr3
        for (int i = 0; i < arr1.length; i++) {
            arr3[i] = arr1[i];
        }
        // Copy elements from arr2 to arr3
        for (int j = 0; j < arr2.length; j++) {
            arr3[arr1.length + j] = arr2[j]; // Start inserting at arr1.length
        }
  System.out.println("merge array:-"+Arrays.toString(arr3));
 }
 public static int[] mergeArrays(int[] arr1, int[] arr2) {
  int[] mergedArray = Arrays.copyOf(arr1, arr1.length + arr2.length);
  // Copy second array elements
  System.arraycopy(arr2, 0, mergedArray, arr1.length, arr2.length);
  return mergedArray;
 }

}
```

File: PairOfSum.java

```
package ArrayExample;
//To find pairs of numbers whose sum is 10 from a given array:
//int[] arr = {1, 4, 7, 5, 3, 2, 6};
public class PairOfSum {
 public static void main(String[] args) {
  int arr[] = {1, 4, 7, 5, 3, 2, 6};
  for(int i=0; i<arr.length; i++) {
   for(int j=i+1; j<arr.length; j++) {
    if(arr[i]+arr[j] == 10) {
     System.out.println(arr[i] +" + "+arr[j]+ " = 10");
    }
   }
  }


 }
}
```

File: RemoveElementFromArray.java

```
package ArrayExample;
import java.util.Arrays;
import java.util.Scanner;
public class RemoveElementFromArray {
 public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Array Size:");
        int size = sc.nextInt();
        System.out.println("Enter Array Elements:");
        int[] arr = new int[size];
        for (int i = 0; i < size; i++) {
            arr[i] = sc.nextInt();
        }
        System.out.println("Original array: " + Arrays.toString(arr));
        System.out.println("Which element do you want to remove?");
        int removeElement = sc.nextInt();
        int[] newArray = removeElement(arr, removeElement);
        System.out.println("Array after removal: " + Arrays.toString(newArray));
        sc.close();
    }
    private static int[] removeElement(int[] arr, int removeElement) {
        // Count occurrences of removeElement
        int count = 0;
        for (int value : arr) {
            if (value == removeElement) {
                count++;
            }
        }
        if (count == 0) {
            System.out.println("Element not found! Returning the original array.");
            return arr; // Return original array if the element is not found
        }
        // Create a new array with reduced size
        int[] newArray = new int[arr.length - count];
        int index = 0;
        for (int value : arr) {
            if (value != removeElement) {
                newArray[index++] = value; // Copy only non-matching elements
            }
        }
        return newArray;
    }
}
```

File: ReverseArray.java

```
package ArrayExample;
import java.util.Arrays;
public class ReverseArray {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        System.out.println("Original Array: " + Arrays.toString(arr));

        int[] newArray =reverseArrayUsingForLoop(arr);
        System.out.println("Reversed using For loop Array: " + Arrays.toString(newArray));
    }
 private static int[] reverseArrayUsingForLoop(int[] arr) {
  int [] newArray=new int[arr.length];
  for (int i = 0; i < arr.length; i++) {
   newArray[arr.length - 1 - i] = arr[i];
  }
  return newArray;
 }
}
```

File: SecondLargestNo.java

```
package ArrayExample;
import java.util.Arrays;
public class SecondLargestNo {
 public static void main(String[] args) {
  int array[] = {10,5,20,49,45,58};
  System.out.println("Original Array : "+Arrays.toString(array));

  int largeNo=findSecondLargestNo(array);
  System.out.println("Second Larget No is "+largeNo);
 }
 private static int findSecondLargestNo(int[] array) {
  for(int i = 0; i<array.length-1 ; i++) {
   for(int j =i+1; j<array.length; j++) {
    if(array[i] > array[j]) {
     int temp = array[i];
     array[i] = array[j];
     array[j] = temp;
    }
   }
  }
  return array[array.length-2];
 }
}
```

File: StringLengthArray.java

```
package ArrayExample;
import java.util.Arrays;
public class StringLengthArray {
 public static void main(String[] args) {
  String[] array = {"RAVI", "KIRAN", "ATUL", "CHAVAN"};
  System.out.println("Original Array "+Arrays.asList(array));

  int intArray[] = new int[array.length];

  for(int i=0; i<array.length; i++) {
   intArray[i] = array[i].length();
  }
  System.out.println("Original Array "+Arrays.toString(intArray));

 }
}
```

File: SumOfAllElement.java

```
package ArrayExample;
import java.util.Arrays;
import java.util.stream.Collectors;
public class SumOfAllElement {
 public static void main(String[] args) {
  int array[] = {10,5,20,49,45,58};
  System.out.println("Original Array : "+Arrays.toString(array));

  int sumOfAllElement = sumOfAllElement(array);
  System.out.println("Sum of All Element "+sumOfAllElement);

  usingStream(array);
 }
 private static int sumOfAllElement(int[] array) {
  int a=0;
  for(int i=0; i<array.length; i++) {
   a += array[i];
  }
  return a;
 }
 private static void usingStream(int[] array) {
  System.out.println(Arrays.stream(array).boxed().reduce(0,(x,y)->(x+y)));
  System.out.println(Arrays.stream(array).sum());
  System.out.println(Arrays.stream(array).summaryStatistics().getSum());
 }
}
```

File: UserInput2DArray.java

```
package ArrayExample;
import java.util.Scanner;
public class UserInput2DArray {

 public static void main(String[] args) {
  Scanner scanner = new Scanner(System.in);
  // Take array dimensions from user
  System.out.print("Enter number of rows: ");
  int rows = scanner.nextInt();
  System.out.print("Enter number of columns: ");
  int cols = scanner.nextInt();
  // Declare 2D array
  int[][] arr = new int[rows][cols];
  // Take input for 2D array
  System.out.println("Enter elements of the array:");
  for (int i = 0; i < rows; i++) {
   for (int j = 0; j < cols; j++) {
    System.out.print("Element at [" + i + "][" + j + "]: ");
    arr[i][j] = scanner.nextInt();
   }
  }
  // Print the array
  System.out.println("\n2D Array:");
  for (int i = 0; i < rows; i++) {
   for (int j = 0; j < cols; j++) {
    System.out.print(arr[i][j] + " ");
   }
   System.out.println();
  }
  scanner.close();
 }
}
```

Package: BasicQuestion

File: FormattedOutput.java

```
package BasicQuestion;
//how to formatting and displaying output in Java using printf() and format() methods.
public class FormattedOutput {
 public static void main(String[] args) {
  // Create a string literal.
  String str = "Hello Java";
  int num = 99;
  float f = 20.25555f;
  System.out.printf("String = %s %nnum = %d %nfloat = %f ", str, num, f);
 }
}
```

File: PrintWithouMainMethod.java

```
package BasicQuestion;
//Print Statement Without Main Method write
//but 7 requred main method but not from java 8
public class PrintWithouMainMethod {
 //Instance Block
 static {
  System.out.println("Print Statement Without Main Method write using Instance Block");
  System.exit(0);//JVM shutdown
 }
}
```

File: ThisSuperKeyword.java

```
package BasicQuestion;
//This :- This keyword is refer to current instance variable/Object
//   1)Why this keyword is used?
//   -->because it differentiat Local and Instance variable.(When Local and instance variable  is
same name)
//   2)where to use this?
//   -->* This can use ti refer current class instance variable
//      * This to refer current class method
//      * This to refer current class constructor
//Super :- This super keyword is refer to immediate parent class object.
//   Is-A-Relationship required
public class ThisSuperKeyword {
 public static void main(String[] args) {
  Dog d = new Dog("Buddy");
  d.display();
 }
}
class Animal {
 String name = "Animal"; // instance variable in superclass
 Animal() {
  System.out.println("Animal constructor called");//1st
 }
 void showName() {
  System.out.println("Name from Animal: " + name);//5th --Animal
 }
}
class Dog extends Animal {
 String name = "Dog"; // instance variable in subclass
 Dog(String name) {
  super(); // calls Animal constructor
  // local variable
  String localName = name;//Buddy
  // 'this.name' refers to current class (Dog)
  this.name = name;//Buddy set Buddy instead of Dog
  // 'super.name' refers to parent class (Animal)
  System.out.println("Local variable: " + localName);//Buddy
  System.out.println("this.name (Dog): " + this.name);//Buddy
  System.out.println("super.name (Animal): " + super.name);//Animal
 }
 void display() {
  super.showName(); // calls method from Animal
  System.out.println("Name from Dog class: " + this.name);//6th Buddy
 }
}
```

Package: BeginnerExample

File: ArmstrongNumber.java

```
package BeginnerExample;
import java.util.Scanner;
//153 = 1³ + 5³ + 3³ = 153 (Armstrong Number)
//9474 = 9 + 4 + 7 + 4 = 9474 (Armstrong Number)
public class ArmstrongNumber {
 public static void main(String[] args) {
  int arm = 0, a, b, c, temp, no;
  Scanner scanner = new Scanner(System.in);
  System.out.println("Enter any number : ");
  no = scanner.nextInt();
  temp = no;
  while (no > 0) {
   a = no % 10;
   no = no / 10;
   arm = arm + a * a * a;
  }
  if (arm == temp) {
   System.out.println("Armstrong number");
  } else {
   System.out.println("Not Armstrong number");
  }
 }
}
```

File: CalculateSumOfAllDigit.java

```
package BeginnerExample;
import java.util.Scanner;
//Calculate Sum of All Digit like 1234 sum is = 10
//1+2+3+4=10
public class CalculateSumOfAllDigit {
 public static void main(String[] args) {

  Scanner sc = new Scanner(System.in);
  System.out.println("Enter Given No for Calculate Sum ");
  Integer num=sc.nextInt();

  //Gavathi way
  Integer sum = calculateSumOfDigit(num);
  System.out.println("Gavathi way to Calculate sum "+sum);

  Integer sum1 = calculateSumOfDigitStandardWay(num);
  System.out.println("Standard Way to Calculate sum "+sum1);
 }
 private static Integer calculateSumOfDigit(Integer num) {
  String str=num.toString();
  Integer temp=0;
  for(int i=0; i <=str.length()-1; i++) {
   temp+=Character.getNumericValue(str.charAt(i));
  }
  return temp;
 }

 private static Integer calculateSumOfDigitStandardWay(Integer num) {
     int sum = 0;
     while (num > 0) {
         sum += num % 10; // Extract last digit
         num =num / 10; // Remove last digit
     }
     return sum;
 }
}
```

File: CountDigitsString.java

```
package BeginnerExample;
import java.util.Scanner;
public class CountDigitsString {
  public static int countDigits(int number) {
   int count = String.valueOf(number).length();
   System.out.println("count "+count);
         return String.valueOf(Math.abs(number)).length(); // Convert number to string and get
length
     }
     public static void main(String[] args) {
         Scanner scanner = new Scanner(System.in);
         System.out.print("Enter a number: ");
         int num = scanner.nextInt();
         scanner.close();
         System.out.println("Number of digits in " + num + " is: " + countDigits(num));
     }
}
```

File: DuplicateCharacterCount.java

```
package BeginnerExample;
import java.util.HashMap;
import java.util.Scanner;
import java.util.Set;
public class DuplicateCharacterCount {

 public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  System.out.println("Enter Original String");
  String string=sc.nextLine();//nextLine() -Take Line From Uses ,next()--OnlyTake Single Word
  duplicateCharacterCount(string);

 }
 private static void duplicateCharacterCount(String string) {
  HashMap<Character, Integer> charCountMap = new HashMap<>();

  char [] strArray =string.toCharArray();
  for (char c : strArray) {
   if(charCountMap.containsKey(c)) {
    charCountMap.put(c, charCountMap.get(c) + 1);
   }else {
    charCountMap.put(c, 1);
   }
  }
  System.out.println("Duplicate Characters in : " + charCountMap);

  Set<Character> charsInString = charCountMap.keySet();
  System.out.println("charsInString"+charsInString);

  for (Character ch : charsInString) {
   if (charCountMap.get(ch) > 1) {
    System.out.println(ch + " : " + charCountMap.get(ch));
   }
  }
 }
}
```

File: FactorialExample.java

```
package BeginnerExample;
import java.util.Scanner;
//5!
//5*4*2*2*1
//=120
public class FactorialExample {
 public static void main(String[] args) {

  Scanner sc = new Scanner(System.in);
  System.out.println("Please Enter Any Number ");
  int number = sc.nextInt();

  int fact = factorial(number);
  System.out.println(fact);
 }
 private static int factorial(int number) {
  int temp=1;
  for(int i=1;  i<=number; i++) {
   temp=temp*i;
  }
  return temp;
 }
}
```

File: FindEvenOrOddTillEnd.java

```
package BeginnerExample;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class FindEvenOrOddTillEnd {
 public static void main(String[] args) {
  List<Integer> even = new ArrayList<>();
  List<Integer> odd = new ArrayList<>();

  Scanner sc = new Scanner(System.in);
  System.out.println("Please Enter Any Number ");
  int number = sc.nextInt();

  for(int i= 1; i <=number; i++) {
   if(isEven(i)) {//even
    even.add(i);
   }else {//odd
    odd.add(i);
   }
  }
  System.out.println("Event List"+even);
  System.out.println("Odd List "+odd);
 }
 private static boolean isEven(int i) {
  if(i % 2 == 0) {
   return true;
  }
  return false;
 }
}
```

File: FindLongestWordIinSentence.java

```
package BeginnerExample;
import java.util.HashMap;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Scanner;
import java.util.Set;
public class FindLongestWordIinSentence {

 public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  System.out.println("Enter Original String");
  String string=sc.nextLine(); //nextLine() -Take Line From Uses ,next()--OnlyTake Single Word
  findLongestWord(string);
 }

 private static void findLongestWord(String string) {
  HashMap<String, Integer> map = new HashMap<>();
  String[] afterSplit = string.split(" ");
  for (String word : afterSplit) {
   map.put(word, word.length());
  }

  Set<Entry<String, Integer>> set =map.entrySet();

  String maxKey = null;
        int maxValue = Integer.MIN_VALUE;

        for (Entry<String, Integer> entry : set) {
            if (entry.getValue() > maxValue) {
                maxValue = entry.getValue();
                maxKey = entry.getKey();
            }
        }

        System.out.println("max key -->"+maxKey + " : "+ maxValue);
//  System.out.println(map.values());

 }
}
```

File: FindOutPrimeNumberTillNo.java

```
package BeginnerExample;
import java.util.ArrayList;
public class FindOutPrimeNumberTillNo {
 public static void main(String[] args) {
        System.out.println("Prime numbers from 1 to 100:");
        ArrayList<Integer> list = new ArrayList<>();

        for (int num = 1; num <= 100; num++) {
            if (isPrime(num)) {
                list.add(num);
            }
        }
        System.out.println(list);
    }
    private static boolean isPrime(int number) {
     for (int i = 2; i <= number - 1; i++) {
            if (number % i == 0) {
                return false;
            }
        }
        return true;
    }
}
```

File: FindOutPrimeOrNormalSeperatly.java

```
package BeginnerExample;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class FindOutPrimeOrNormalSeperatly {
 public static void main(String[] args) {
  List<Integer> prime = new ArrayList<>();
  List<Integer> normal = new ArrayList<>();

  Scanner sc = new Scanner(System.in);
  System.out.println("Please Enter Any Number ");
  int number = sc.nextInt();
  for (int num = 1; num <= number; num++) {
   if (isPrime(num)) {
    prime.add(num);
            }else {
             normal.add(num);
            }
  }
  System.out.println("Event List"+prime);
  System.out.println("Odd List "+normal);
 }

 private static boolean isPrime(int num) {
     for (int i = 2; i <= num - 1; i++) {
            if (num % i == 0) {
                return false;
            }
        }
        return true;
    }
}
```

File: PalindromeNumber.java

```
package BeginnerExample;
import java.util.Scanner;
//121 ->Number is palindrome
//123 ->Number is not palindrome
public class PalindromeNumber {
 public static void main(String[] args) {

  Scanner sc = new Scanner(System.in);
  System.out.println("Please Enter Any Number ");
  Integer number = sc.nextInt();
  Integer reverge=0;
  if(number.toString().length()>=2) {
   reverge=reverseNumberMethod(number);
   System.out.println("reverge "+reverge);
  }
  if(reverge == number) {
   System.out.println("Given no. is Palindrome");
  }else {
   System.out.println("Given no. is not Palindrome");
  }
 }
 private static Integer reverseNumberMethod(Integer number) {
  Integer rev=0,temp;
  while(number>0) {
   temp = number % 10;//last value bhetate
   rev = rev * 10 + temp;//last chi value bhetate
   number = number / 10;//ani number madhun last digit gayab
  }
  return rev;
 }
}
```

File: PrimeNoOrNot.java

```
package BeginnerExample;
import java.util.Scanner;
public class PrimeNoOrNot {
 public static void main(String[] args) {

  Scanner sc = new Scanner(System.in);
  System.out.println("Please Enter Any Number ");
  int number = sc.nextInt();

  if(findOutPrimeNo(number)) {
   System.out.println("Given No Is "+ number +" prime no !!!");
  }else {
   System.out.println("Given No Is not "+ number +" prime no !!!");
  }
 }
 private static boolean findOutPrimeNo(int number) {
  for (int i = 2; i <= number - 1; i++) {
            if (number % i == 0) {
                return false;
            }
        }
        return true;
 }
}
```

File: ReverseEachWordOfString.java

```
package BeginnerExample;
import java.util.Scanner;
public class ReverseEachWordOfString {
 public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  System.out.println("Enter Original String");
  String string=sc.nextLine();//nextLine() -Take Line From Uses ,next()--OnlyTake Single Word

  String reverseString = reverseString(string);
  System.out.println("Reverse String "+reverseString);
 }

 private static String reverseString(String string) {
  String [] words = string.split(" ");

  String reverseString = "" ;
  for(int i=0; i< words.length; i++) {
   String word = words[i];
   char ch;
   String nstr = "";

   for(int j=0; j < word.length(); j++) {
    ch = word.charAt(j);
    nstr = ch + nstr;
   }
   reverseString = reverseString + nstr+ " ";
  }
  return reverseString;
 }
}
```

File: ReverseNumber.java

```
package BeginnerExample;
import java.util.Scanner;
public class ReverseNumber {
 public static void main(String[] args) {

  Scanner sc = new Scanner(System.in);
  System.out.println("Please Enter Any Number ");
  Integer number = sc.nextInt();
  if(number.toString().length()>=2) {
   Integer reverge =reverseNumberMethod(number);
   System.out.println("reverge "+reverge);
  }
 }
 private static Integer reverseNumberMethod(Integer number) {
  Integer rev=0,temp;
  while(number>0) {
   temp = number % 10;//last value bhetate
   rev = rev * 10 + temp;//last chi value bhetate
   number = number / 10;//ani number madhun last digit gayab
  }
  return rev;
 }
}
```

File: ReverseStringExample.java

```
package BeginnerExample;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;
//Ravikiran  ->narikivaR
public class ReverseStringExample {
 public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  System.out.println("Enter Original String");
  String string=sc.next();

  String reverseString = reverseString(string);
  System.out.println("Reverse String "+reverseString);

  StringBuilder sb = new StringBuilder(string);
  StringBuffer sf = new StringBuffer(string);
  System.out.println("Reverse Using String Builder Reverse Method "+sb.reverse());
  System.out.println("Reverse Using String Buffer  Reverse Method "+sf.reverse());

  usingStream(string);
 }
 private static String reverseString(String string) {
  int size = string.length();
  char ch;
  String nstr = "";
  for(int i=0; i < size; i++) {
   ch = string.charAt(i);
   nstr = ch + nstr;
  }
  return nstr;
 }

 private static void usingStream(String string) {
  List<String> list = Arrays.asList(string.split(""));
  Collections.reverse(list);
  System.out.println("----------"+String.join("", list));
 }



}
```

File: ReverseStringLoop.java

```
package BeginnerExample;
import java.util.Scanner;
public class ReverseStringLoop {
 public static String reverseString(String str) {
        StringBuilder reversed = new StringBuilder();

        for (int i = str.length() - 1; i >= 0; i--) {
            reversed.append(str.charAt(i));
        }

        return reversed.toString();
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String input = scanner.nextLine();
        scanner.close();
        System.out.println("Reversed String: " + reverseString(input));

        //Another Way
        String reversed = new StringBuilder(input).reverse().toString();
        System.out.println("Direct way of Reversed String: " + reversed);
    }
}
```

File: WordCountInParagraph.java

```
package BeginnerExample;
import java.util.HashMap;
import java.util.Scanner;
public class WordCountInParagraph {
 public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  System.out.println("Enter Original String");
  String string=sc.nextLine();//nextLine() -Take Line From Uses ,next()--OnlyTake Single Word
  wordCountInPara(string);
 }
 private static void wordCountInPara(String string) {
  String [] words = string.split(" ");
  HashMap<String,Integer> charCountMap = new HashMap<>();

  for (String string2 : words) {
   if(charCountMap.containsKey(string2)) {
    charCountMap.put(string2, charCountMap.get(string2)+1);
   }else {
    charCountMap.put(string2,1);
   }
  }
  System.out.println("Count of Characters in a given string : " + charCountMap);

 }
}
```

Package: Collection_ArrayDeque

File: ArrayDequeExample.java

```
package Collection_ArrayDeque;
import java.util.ArrayDeque;
//Maintain insertion order,
//Duplicate allowed
//not thread Safe
//not allowed null element
//Faster as compare to LinkedList and Stack
public class ArrayDequeExample {
 public static void main(String[] args) {
  ArrayDeque<Integer> deq = new ArrayDeque<>();

  deq.add(40);
  deq.offer(20);
  deq.add(82);
  deq.add(82);
  deq.add(52);
//  deq.add(null);

  System.out.println(deq);
 }
}
```

File: ArrayDequeOperations.java

```
package Collection_ArrayDeque;
import java.util.ArrayDeque;
public class ArrayDequeOperations {
 public static void main(String[] args) {
        // Creating an ArrayDeque
        ArrayDeque<Integer> deque = new ArrayDeque<>();
        // Adding elements
        deque.add(10);  // Adds to the end
        deque.addFirst(20); // Adds to the front
        deque.addLast(30); // Adds to the end
        deque.offer(40);  // Adds to the end
        deque.offerFirst(50); // Adds to the front
        deque.offerLast(60); // Adds to the end
        System.out.println("ArrayDeque: " + deque); // [50, 20, 10, 30, 40, 60]
        // Removing elements
        System.out.println("Removing first: " + deque.pollFirst()); // 50
        System.out.println("Removing last: " + deque.pollLast()); // 60
        System.out.println("ArrayDeque after removals: " + deque); // [20, 10, 30, 40]
        // Peek elements
        System.out.println("First Element: " + deque.peekFirst()); // 20
        System.out.println("Last Element: " + deque.peekLast()); // 40
        // Checking size
        System.out.println("Size: " + deque.size()); // 4
        // Checking if it contains an element
        System.out.println("Contains 30? " + deque.contains(30)); // true
        // Removing a specific element
        deque.remove(30);
        System.out.println("ArrayDeque after remove(30): " + deque); // [20, 10, 40]
        // Checking if empty
        System.out.println("Is Empty? " + deque.isEmpty()); // false
        // Iterating through the elements
        System.out.print("Elements using for-each: ");
        for (Integer num : deque) {
            System.out.print(num + " ");
        }
        System.out.println();
        // Polling all elements until empty
        while (!deque.isEmpty()) {
            System.out.println("Polling: " + deque.poll());
        }
        // Checking again if empty
        System.out.println("Is Empty after polling all? " + deque.isEmpty()); // true
    }
}
```

Package: Collection_HashMap

File: CharFrequency.java

```
package Collection_HashMap;
import java.util.HashMap;
import java.util.Map;
//Write a program to count the frequency of characters in a string using Map.
public class CharFrequency {
 public static void main(String[] args) {

  String input="Ravikiran";

  Map<Character, Integer> freqMap = new HashMap<>();

        for (char ch : input.toCharArray()) {
            freqMap.put(ch, freqMap.getOrDefault(ch, 0) + 1);
        }
        System.out.println(freqMap);
 }
}
```

File: ClinicAppointmentSystem.java

```
package Collection_HashMap;
import java.util.HashMap;
import java.util.Map.Entry;
import java.util.Set;
public class ClinicAppointmentSystem {
    public static void main(String[] args) {
        // Create a HashMap to store Doctor ID and Name
        HashMap<Integer, String> doctorMap = new HashMap<>();
        // 1. Adding doctors to the system
        doctorMap.put(101, "Dr. Smith - Cardiologist");
        doctorMap.put(102, "Dr. Johnson - Neurologist");
        doctorMap.put(103, "Dr. Brown - Dermatologist");
        for (Entry<Integer, String> entry : doctorMap.entrySet()) {
         System.out.println("Doctor ID: " + entry.getKey() + ", Name: " + entry.getValue());
  }

        // 2. Display available doctors
        System.out.println("Available Doctors:");
        doctorMap.forEach((id, name) -> System.out.println("Doctor ID: " + id + ", Name: " +
name));
        // 3. Checking if a doctor is available
        int searchId = 102;
        if (doctorMap.containsKey(searchId)) {
            System.out.println("\nDoctor with ID " + searchId + " is available: " +
doctorMap.get(searchId));
        } else {
            System.out.println("\nDoctor with ID " + searchId + " is NOT available.");
        }
        // 4. Updating doctor details
        doctorMap.replace(103, "Dr. Brown - Senior Dermatologist");
        // 5. Removing a doctor from the system
        doctorMap.remove(101); // Dr. Smith is no longer available
        // 6. Booking an appointment
        HashMap<Integer, String> appointmentMap = new HashMap<>();
        appointmentMap.put(1001, "Patient: Alice, Doctor: Dr. Johnson - Neurologist");
        appointmentMap.put(1002, "Patient: Bob, Doctor: Dr. Brown - Senior Dermatologist");
        // 7. Displaying all appointments
        System.out.println("\nBooked Appointments:");
        appointmentMap.forEach((id, details) -> System.out.println("Appointment ID: " + id + ",
" + details));
        // 8. Checking if a specific appointment exists
        int appointmentId = 1002;
        if (appointmentMap.containsKey(appointmentId)) {
            System.out.println("\nAppointment " + appointmentId + " exists: " +
appointmentMap.get(appointmentId));
        }
        // 9. Cancelling an appointment
        appointmentMap.remove(1001);
        System.out.println("\nUpdated Appointments after Cancellation:");
        appointmentMap.forEach((id, details) -> System.out.println("Appointment ID: " + id + ",
" + details));
        // 10. Clearing all appointments at the end of the day
```

```
        appointmentMap.clear();
        System.out.println("\nAll appointments cleared. Total remaining: " +
appointmentMap.size());
    }
}
```

File: HashMapComparison.java

```
package Collection_HashMap;
import java.util.HashMap;
import java.util.HashSet;
public class HashMapComparison {
    /**
     * @param args
     */
    public static void main(String[] args) {
        // Creating two HashMaps
        HashMap<Integer, String> map1 = new HashMap<>();
        HashMap<Integer, String> map2 = new HashMap<>();
        HashMap<Integer, String> map3 = new HashMap<>();
        // Adding values
        map1.put(1, "Apple");
        map1.put(2, "Banana");
        map1.put(3, "Cherry");
        map2.put(1, "Apple");
        map2.put(2, "Banana");
        map2.put(3, "Cherry");
        map3.put(1, "Green Apple");
        map3.put(2, "Banana");
        map2.put(3, "Cherry");

        // Comparing using equals()
        System.out.println("Are both HashMaps equal? " + map1.equals(map2)); // Output: true
        System.out.println("Are both HashMaps equal? " + map1.equals(map3)); // Output: false

        // Comparing using Key
        System.out.println("Are both HashMaps equal using KeySet ? " +
map1.keySet().equals(map2.keySet())); // Output: true
        System.out.println("Are both HashMaps equal using KeySet ? " +
map1.keySet().equals(map3.keySet())); // Output: false

     // Comparing using value
       HashSet<String> map1Set = new HashSet<>(map1.values());
       HashSet<String> map2Set = new HashSet<>(map2.values());
       HashSet<String> map3Set = new HashSet<>(map3.values());

       System.out.println("Are both HashMaps equal using value ? "+map1Set.equals(map2Set));
//true
       System.out.println("Are both HashMaps equal using value ? "+map1Set.equals(map3Set));
//false
    }
}
```

File: HashMapExample.java

```
package Collection_HashMap;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map.Entry;
import java.util.Set;
//No maintained insertion order
//Allowed to one null key
//not thread safe / not synchronized
//Performance is faster
// does not allowed duplicate key but allowed to duplicate value
//Default capacity is 16 and load factor is 0.75
public class HashMapExample {
 public static void main(String[] args) {
  HashMap<Integer, String> map = new HashMap<>();
  map.put(1,"Java");
  map.put(2,"Python");
  map.put(3,"React");
  map.put(4,"Ruby");
  map.put(null,"C++");
  map.put(5,"Angular");
  map.put(6,"Struds");
  map.put(7,"Struds");

  Set<Entry<Integer, String>>  itr = map.entrySet();
  for (Entry<Integer, String> entry : itr) {
   System.out.println(entry.getKey() + "-"+ entry.getValue());
  }
  System.out.println("---------------------------------");
  Iterator<Integer> keySetItr=map.keySet().iterator();
  for (Entry<Integer, String> entry : itr) {
   System.out.println(entry);
  }
 }
}
```

File: HashMapOperations.java

```
package Collection_HashMap;
import java.util.HashMap;
import java.util.Map;
public class HashMapOperations {
 public static void main(String[] args) {
        // 1. Creating a HashMap
        HashMap<Integer, String> map = new HashMap<>();
        // 2. Adding Elements
        map.put(1, "Alice");
        map.put(2, "Bob");
        map.putIfAbsent(3, "Charlie"); // Adds only if key is absent
        // 3. Retrieving Elements
        System.out.println("Get key 1: " + map.get(1));
        System.out.println("Get key 5 (or default): " + map.getOrDefault(5, "Default Name"));
        // 4. Checking for Existence
        System.out.println("Contains key 2? " + map.containsKey(2));
        System.out.println("Contains value 'Bob'? " + map.containsValue("Bob"));
        // 5. Updating Values
        map.replace(2, "Bobby");
        System.out.println("Get key 2: " + map.get(2));
        map.replace(2, "Bobby", "Bob Updated");
        System.out.println("Get key 2: " + map.get(2));

        // 6. Removing Elements
        map.remove(1);
        map.remove(2, "Bob Updated");
        // 7. Iterating Over HashMap
        System.out.println("\nIterating using forEach:");
        map.forEach((key, value) -> System.out.println(key + " -> " + value));
        System.out.println("\nIterating using entrySet:");
        for (Map.Entry<Integer, String> entry : map.entrySet()) {
            System.out.println(entry.getKey() + " -> " + entry.getValue());
        }
        // 8. Getting Size of HashMap
        System.out.println("Size of map: " + map.size());
        // 9. Checking if HashMap is Empty
        System.out.println("Is map empty? " + map.isEmpty());
        // 10. Clearing HashMap
        map.clear();
        System.out.println("Map cleared. Size: " + map.size());
        // 11. Cloning HashMap
        map.put(4, "David");
        map.put(5, "Eve");
        HashMap<Integer, String> clonedMap = (HashMap<Integer, String>) map.clone();
        System.out.println("Cloned Map: " + clonedMap);
        // 12. Merging Values //concatinate
        map.merge(4, " Smith", (oldValue, newValue) -> oldValue + newValue);
        System.out.println("After merge: " + map.get(4));
        // 13. Using compute, computeIfAbsent, computeIfPresent
        map.compute(5, (key, value) -> value + " Updated");
        map.computeIfAbsent(6, key -> "New Value");
        map.computeIfPresent(4, (key, value) -> value + " Modified");
```

```
        System.out.println("Map after compute operations: " + map);
        // 14. Replacing All Elements
        map.replaceAll((key, value) -> value.toUpperCase());
        System.out.println("Map after replaceAll: " + map);
    }
}
```

File: StudentManagementSystem.java

```
package Collection_HashMap;
import java.util.HashMap;
import java.util.Map;
import java.util.Objects;
public class StudentManagementSystem {
 public static void main(String[] args) {
        // Creating a HashMap to store student ID and Student object
        HashMap<Integer, Student> studentMap = new HashMap<>();
        // 1. Adding students
        studentMap.put(101, new Student(101, "Ravi", 20, "Computer Science"));
        studentMap.put(102, new Student(102, "Kiran", 22, "Mechanical Engineering"));
        studentMap.put(103, new Student(103, "Atul", 21, "Electrical Engineering"));
        // 2. Retrieving student details
        System.out.println("Student with ID 102: " + studentMap.get(102));
        // 3. Checking if a student exists
        int searchId = 105;
        if (studentMap.containsKey(searchId)) {
            System.out.println("Student with ID " + searchId + " found: " +
studentMap.get(searchId));
        } else {
            System.out.println("Student with ID " + searchId + " not found.");
        }
        // 4. Updating student course
        if (studentMap.containsKey(103)) {
            studentMap.get(103).setCourse("Civil Engineering");
            System.out.println("Updated Student 103: " + studentMap.get(103));
        }
        // 5. Removing a student
        studentMap.remove(102);
        System.out.println("\nAfter removing Student 102:");
        // 6. Displaying all students
        for (Map.Entry<Integer, Student> entry : studentMap.entrySet()) {
            System.out.println("Student ID: " + entry.getKey() + ", " + entry.getValue());
        }
        // 7. Checking if the HashMap is empty
        System.out.println("\nIs the student map empty? " + studentMap.isEmpty());
        // 8. Clearing all students
        studentMap.clear();
        System.out.println("All students removed. Total students: " + studentMap.size());
    }
}
class Student {
    private int id;
    private String name;
    private int age;
    private String course;
    // Constructor
    public Student(int id, String name, int age, String course) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.course = course;
```

```
    }
    // Getters
    public int getId() { return id; }
    public String getName() { return name; }
    public int getAge() { return age; }
    public String getCourse() { return course; }
    // Setters (if needed)
    public void setCourse(String course) {
     this.course = course;
    }
    // Overriding toString() for better printing
    @Override
    public String toString() {
        return "ID: " + id + ", Name: " + name + ", Age: " + age + ", Course: " + course;
    }
    // Overriding equals() and hashCode() for proper comparison
//    @Override
//    public boolean equals(Object obj) {
//        if (this == obj) return true;
//        if (obj == null || getClass() != obj.getClass()) return false;
//        Student student = (Student) obj;
//        return id == student.id && age == student.age &&
//               Objects.equals(name, student.name) && Objects.equals(course, student.course);
//    }
    @Override
    public int hashCode() {
        return Objects.hash(id, name, age, course);
    }
}
```

Package: Collection_HashTable

File: HashtableExample.java

```
package Collection_HashTable;
import java.util.Hashtable;
import java.util.Iterator;
// insertion order is not maintained (Unordered storage)
// no duplicate key or value
// synchronized
// default capacity is 11 or load factor 0.75
// Slower than HashMap Due to synchronization
public class HashtableExample {
 public static void main(String[] args) {

  Hashtable<Integer, String> language = new Hashtable<>();

  language.put(1, "Hindi");
  language.put(2, "English");
  language.put(3, "Marathi");
  language.put(1, "Urdu");

  System.out.println(language);

  //language.keys();
  Iterator<Integer> lang = language.keySet().iterator();
  while (lang.hasNext()) {
   System.out.println(lang.next());
  }
 }
}
```

File: HashtableOperations.java

```
package Collection_HashTable;
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.Map;
public class HashtableOperations {

 public static void main(String[] args) {
        // 1 Creating a Hashtable (Does NOT allow null keys/values)
        Hashtable<Integer, String> hashtable = new Hashtable<>();
        // 2 Adding Elements
        hashtable.put(101, "Alice");
        hashtable.put(102, "Bob");
        hashtable.put(103, "Charlie");
        hashtable.putIfAbsent(104, "David"); // Adds only if key 104 is absent
        System.out.println("Hashtable: " + hashtable);
        // 3 Retrieving Elements
        System.out.println("Value for key 102: " + hashtable.get(102));
        System.out.println("Value for key 200 (or default): " + hashtable.getOrDefault(200, "Not
Found"));
        // 4 Checking Existence
        System.out.println("Contains key 101? " + hashtable.containsKey(101));
        System.out.println("Contains value 'Eve'? " + hashtable.containsValue("Eve"));
        // 5 Updating Elements
        hashtable.replace(103, "Charlie", "Catherine"); // Replaces only if value matches
        hashtable.replace(104, "David");
        // 6 Removing Elements
        hashtable.remove(101); // Removes key 101
        hashtable.remove(102, "Bob"); // Removes only if key-value pair matches
        // 7 Iterating Over Entries
        System.out.println("\nIterating using entrySet:");
        for (Map.Entry<Integer, String> entry : hashtable.entrySet()) {
            System.out.println(entry.getKey() + " -> " + entry.getValue());
        }

        // Iterating using Enumeration
        System.out.println("\nIterating using Enumeration:");
        Enumeration<Integer> keys = hashtable.keys();
        while (keys.hasMoreElements()) {
            Integer key = keys.nextElement();
            System.out.println(key + " -> " + hashtable.get(key));
        }
        // 8 Getting the Size
        System.out.println("\nSize of Hashtable: " + hashtable.size());
        // 9 Checking If Empty
        System.out.println("Is Hashtable empty? " + hashtable.isEmpty());
        // 10 Clearing the Hashtable
        hashtable.clear();
        System.out.println("Hashtable after clear(): " + hashtable);
    }
}
```

Package: Collection_InterviewQuestions

File: FindOutExtraElementFromTwoArrayList.java

```
package Collection_InterviewQuestions;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
//Find Out Extra and common Element From Two ArrayList
public class FindOutExtraElementFromTwoArrayList {
 public static void main(String args[]) {
  List<String> list1 = new ArrayList<>(
    Arrays.asList("Pen", "Pencil", "Notebook", "Workbook", "Rubber", "Shopnar"));
  List<String> list2 = new ArrayList<>(Arrays.asList("Workbook", "Rubber", "Scale", "Pen",
"Pencil", "Notebook"));
  usingCollection(list1, list2);
  usingString(list1, list2);
 }
 private static void usingCollection(List<String> list1, List<String> list2) {
  // Find out Extra Element in list2--removeAll
  List<String> extra = new ArrayList<>(list2);
  extra.removeAll(list1); // remove elements present in list1
  System.out.println("Extra elements in list2: " + extra);
  // Find out common from list1
  List<String> common = new ArrayList<>(list2);
  common.retainAll(list1);
  System.out.println("Common Element Print " + common);
 }
 private static void usingString(List<String> list1, List<String> list2) {
  // Extra elements in list2 (not in list1)
  List<String> extraInList2 = list2.stream().filter(e ->
!list1.contains(e)).collect(Collectors.toList());
  System.out.println("Extra elements in list2: " + extraInList2);

  // Common elements in both lists
  List<String> commonElements = list1.stream()
//                .filter(list2::contains)
    .filter(e -> list2.contains(e)).distinct().collect(Collectors.toList());
  System.out.println("Common elements in both lists: " + commonElements);

 }
}
```

File: MissingNumberFinderInArrayList.java

```
package Collection_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
//How do you identify a missing integer in an array of 1 to 100?
public class MissingNumberFinderInArrayList {
 public static void main(String[] args) {
  int[] numbers = new int[99]; // Array from 1 to 100 with one number missing
  // Example: number 57 is missing
  Integer missing = 57;
  Integer index = 0;
  Integer actualSum = 0;

  for (int i = 1; i <= 100; i++) {
   if (i != missing) {
    numbers[index++] = i;
    actualSum += i;
   }
  }
  Integer expectedSum = 100 * (100 + 1) / 2;
  List<Integer> intList = Arrays.stream(numbers).boxed().collect(Collectors.toList());

  usingCollection(intList,actualSum,expectedSum);
  usingStream(intList,actualSum,expectedSum);
 }
 private static void usingCollection(List<Integer> intList,Integer actualSum,Integer expectedSum)
{
  Integer missingNumber = expectedSum - actualSum;
  System.out.println("Missing number is: " + missingNumber);
 }
 private static void usingStream(List<Integer> intList, int actualSum, Integer expectedSum) {
     Integer streamSum = intList.stream()
                       .mapToInt(Integer::intValue)
                       .sum();
     Integer usingReduce = intList.stream()
       .reduce(0, Integer::sum);
//         .reduce(0, (a, b) -> a + b);
     System.out.println("Missing number (using stream): " + (expectedSum - streamSum));
     System.out.println("Missing number (using stream): " + (expectedSum - usingReduce));
 }
}
```

File: RemoveDuolicateElement.java

```
package Collection_InterviewQuestions;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.Set;
import java.util.stream.Collectors;
//Remove Duplicate Element from ArrayList
public class RemoveDuolicateElement {
 public static void main(String args[]) {
  ArrayList<Integer> list = new ArrayList<>();
  list.add(2);
  list.add(3);
  list.add(2);
  list.add(4);
  list.add(5);
  list.add(3);
  list.add(5);

  //Print Using Iterator
  Iterator obj = list.iterator();
  while(obj.hasNext()) {
   System.out.println(obj.next());
  }

  usingLinkedHashSet(list);
  usingStream(list);
 }
 private static void usingLinkedHashSet(ArrayList<Integer> list) {//Remove Duplicate Element
  Set<Integer> set = new LinkedHashSet<>(list);

  System.out.println("Only Print Unique Element from ArrayList using LinkedHashSet");
  Iterator obj1 = set.iterator();
  while(obj1.hasNext()) {
   System.out.println(obj1.next());
  }
 }
 private static void usingStream(ArrayList<Integer> list) {
  System.out.println("Only Print Unique Element from ArrayList Using Stream");
  System.out.println(list.stream()
  .distinct()
  .collect(Collectors.toList()));
 }
}
```

File: RemoveElementStream.java

```
package Collection_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
public class RemoveElementStream {
 public static void main(String[] args) {
        Integer[] arr = {1, 2, 3, 4, 5, 3};
        List<Integer> result = Arrays.stream(arr)
                .filter(x -> x != 3)
                .collect(Collectors.toList());
        System.out.println(result); // Output: [1, 2, 4, 5]
 }
}
```

Package: Collection_LinkedHashMap

File: LinkedHashMapExample.java

```
package Collection_LinkedHashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
//No duplicate
//maintain insertio order
//only one null key is allowed or multiple null value
//non-synchronized
//curson is not applicable so need to retrive data using object.keySet().iterator()
public class LinkedHashMapExample {
 public static void main(String[] args) {
  LinkedHashMap<Integer, String> name = new LinkedHashMap<>();

  name.put(1,"Ravi");
  name.put(3,"avinash");
  name.put(2,"Ravikant");
  name.put(4,"virat");
  name.put(4,"virat");
  name.put(null,"Rohit");

  System.out.println("get value specified Element "+name.get(4));//pass key

  Iterator<Integer> itr= name.keySet().iterator();
  while (itr.hasNext()) {
   System.out.println(itr.next());
  }


 }
}
```

File: LinkedHashMapOperations.java

```
package Collection_LinkedHashMap;
import java.util.LinkedHashMap;
import java.util.Map;
public class LinkedHashMapOperations {
  public static void main(String[] args) {
         // 1. Creating a LinkedHashMap (Maintains Insertion Order)
         LinkedHashMap<Integer, String> linkedHashMap = new LinkedHashMap<>();
         // 2. Adding Elements
         linkedHashMap.put(1, "Apple");
         linkedHashMap.put(2, "Banana");
         linkedHashMap.put(3, "Cherry");
         linkedHashMap.putIfAbsent(4, "Date");
         // 3. Retrieving Elements
         System.out.println("Value for key 2: " + linkedHashMap.get(2));
         System.out.println("Value for key 5 (or default): " + linkedHashMap.getOrDefault(5, "Not
Found"));
         // 4. Checking Existence
         System.out.println("Contains key 3? " + linkedHashMap.containsKey(3));
         System.out.println("Contains value 'Mango'? " + linkedHashMap.containsValue("Mango"));
         // 5. Updating Elements
         linkedHashMap.replace(2, "Blueberry");
         linkedHashMap.replace(3, "Cherry", "Coconut");
         // 6. Removing Elements
         linkedHashMap.remove(1);
         linkedHashMap.remove(4, "Date"); // Removes only if key-value pair matches
         // 7. Iterating Over Entries
         System.out.println("\nIterating using forEach:");
         linkedHashMap.forEach((key, value) -> System.out.println(key + " -> " + value));
         System.out.println("\nIterating using entrySet:");
         for (Map.Entry<Integer, String> entry : linkedHashMap.entrySet()) {
             System.out.println(entry.getKey() + " -> " + entry.getValue());
         }

         // 8. Getting Size
         System.out.println("\nSize of LinkedHashMap: " + linkedHashMap.size());
         // 9. Checking If Empty
         System.out.println("Is LinkedHashMap empty? " + linkedHashMap.isEmpty());
         // 10. Cloning a LinkedHashMap
         LinkedHashMap<Integer, String> clonedMap = (LinkedHashMap<Integer, String>)
linkedHashMap.clone();
         System.out.println("Cloned LinkedHashMap: " + clonedMap);
         // 11. Using compute(), computeIfAbsent(), computeIfPresent()
         linkedHashMap.compute(2, (key, value) -> value + " Updated");
         linkedHashMap.computeIfAbsent(5, key -> "Elderberry");
         linkedHashMap.computeIfPresent(3, (key, value) -> value + " Modified");
         System.out.println("\nLinkedHashMap after compute operations: " + linkedHashMap);
         // 12. Replacing All Elements
         linkedHashMap.replaceAll((key, value) -> value.toUpperCase());
         System.out.println("LinkedHashMap after replaceAll: " + linkedHashMap);
         // 13. Clearing All Elements
         linkedHashMap.clear();
         System.out.println("LinkedHashMap cleared. Size: " + linkedHashMap.size());
```

```
        }
}
```

Package: Collection_LinkedHashSet

File: LinkedHashSetExample.java

```
package Collection_LinkedHashSet;
import java.util.Iterator;
import java.util.LinkedHashSet;
// No duplicate allowed
// Maintain Insertion Order
// allowed Only one null value
// Non Synchonize
// Slower as compare to Hashset
// Only Iterator is applicable for LinkedHashSet but in below i have to use for-each because
for-each internally use iterator
public class LinkedHashSetExample {
 public static void main(String[] args) {
        LinkedHashSet<Integer> set = new LinkedHashSet<>();

        set.add(5);
        set.add(10);
        set.add(15);
        set.add(5); // Duplicate, will be ignored
        System.out.println("LinkedHashSet: " + set);

        for (Integer itr : set) {
         System.out.println("itr "+itr);
  }

        Iterator<Integer> itr = set.iterator();
        while (itr.hasNext()) {
            System.out.println("itr " + itr.next());
        }
    }

}
```

File: LinkedHashSetExample2.java

```
package Collection_LinkedHashSet;
import java.util.LinkedHashSet;
import java.util.Set;
public class LinkedHashSetExample2 {

 public static void main(String[] args) {
        // Create LinkedHashSet of Employee objects
        Set<Employee> employees = new LinkedHashSet<>();
        // Adding employees
        employees.add(new Employee(1, "Alice", 50000));
        employees.add(new Employee(2, "Bob", 60000));
        employees.add(new Employee(1, "Alice", 50000)); // Duplicate? No, because equals() is
not overridden!
        employees.add(new Employee(3, null, 8353));
        employees.add(new Employee(4, "Alice", 50000)); // Duplicate? No, because equals() is
not overridden!
        // Print elements
        for (Employee emp : employees) {
            System.out.println(emp);
        }
    }
}
class Employee {
    int id;
    String name;
    double salary;
    // Constructor
    public Employee(int id, String name, double salary) {
        this.id = id;
        this.name = name;
        this.salary = salary;
    }
    // toString() for readable output
    @Override
    public String toString() {
        return "Employee{id=" + id + ", name='" + name + "', salary=" + salary + "}";
    }
}
```

File: LinkedHashSetOperations.java

```
package Collection_LinkedHashSet;
import java.util.Iterator;
import java.util.LinkedHashSet;
public class LinkedHashSetOperations {
    public static void main(String[] args) {
        // 1 Creating a LinkedHashSet
        LinkedHashSet<String> names = new LinkedHashSet<>();
        // 2 Adding elements
        names.add("Alice");
        names.add("Bob");
        names.add("Charlie");
        names.add("David");
        names.add("Alice"); // Duplicate, will not be added
        System.out.println("LinkedHashSet after additions: " + names);
        // 3 Checking if an element exists
        System.out.println("Contains 'Charlie'? " + names.contains("Charlie"));
        System.out.println("Contains 'Eve'? " + names.contains("Eve"));
        // 4 Removing an element
        names.remove("Bob");
        System.out.println("After removing 'Bob': " + names);
        // 5 Get size of LinkedHashSet
        System.out.println("Size of LinkedHashSet: " + names.size());
        // 6 Iterating over elements
        System.out.println("Iterating using for-each:");
        for (String name : names) {
            System.out.println(name);
        }
        // Using Iterator
        System.out.println("Iterating using Iterator:");
        Iterator<String> iterator = names.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
        // 7 Convert LinkedHashSet to Array
        Object[] nameArray = names.toArray();
        System.out.println("Converted to Array:");
        for (Object obj : nameArray) {
            System.out.println(obj);
        }
        // 8 Check if empty
        System.out.println("Is LinkedHashSet empty? " + names.isEmpty());
        // 9 Clearing all elements
        names.clear();
        System.out.println("After clear(): " + names);
        System.out.println("Is LinkedHashSet empty now? " + names.isEmpty());
    }
}
```

Package: Collection_LinkedList

File: LinkedListCustomerSupportAppEx.java

```
package Collection_LinkedList;
public class LinkedListCustomerSupportAppEx {
  public static void main(String[] args) {

    TicketingSystem system = new TicketingSystem();
         // Adding sample tickets
         system.addTicket(new SupportTicket(101, "Alice", "Cannot log in", 2));
         system.addTicket(new SupportTicket(102, "Bob", "Payment issue", 1));
         system.addTicket(new SupportTicket(103, "Charlie", "App crash on startup", 3));
         // Viewing all tickets
         system.viewAllTickets();
         // Sorting tickets by priority
         system.sortTicketsByPriority();
         system.viewAllTickets();
         // Checking next ticket without removing
         system.peekNextTicket();
         // Processing a ticket
         system.processNextTicket();
         system.viewAllTickets();
     }
}
```

File: LinkedListExample1.java

```
package Collection_LinkedList;
import java.util.LinkedList;
public class LinkedListExample1 {
public static void main(String[] args) {

  // Creating a list of names
        LinkedList<String> names = new LinkedList<>();
        // Adding Names
        names.add("Ravi");
        names.add("Kiran");
        names.add("Atul");
        names.add("Ashwini");
        names.add("Pooja");
        // Displaying the list
        System.out.println("Names List: " + names);
        // Accessing an element
        System.out.println("First Name: " + names.get(0));
        // Removing an element
        names.remove("Atul");
        System.out.println("After removing Atul: " + names);
        // Iterating over the list
        System.out.println("Iterating over the list:");
        for (String name : names) {
            System.out.println(name);
        }
    }
}
```

File: LinkedListExample2.java

```
package Collection_LinkedList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.LinkedList;
import java.util.List;
public class LinkedListExample2 {
 public static void main(String[] args) {

  // Creating a LinkedList of integers
        LinkedList<Integer> list = new LinkedList<>();
        // Adding elements to the LinkedList
        list.add(10);
        list.add(20);
        list.add(30);
        list.add(40);
        list.add(50);
        System.out.println("Initial LinkedList: " + list);
        // Adding elements at specific positions
        list.push(23);//add in first
        list.addFirst(5);//add in first
        list.addLast(60);
        System.out.println("After adding First & Last: " + list);
        // Removing elements
        list.remove(2); // Removes element at index 2
        list.removeFirst(); // Removes first element
        list.removeLast(); // Removes last element
        System.out.println("After removing elements: " + list);
        // Peek (Retrieve but do not remove first element)
        System.out.println("Peek (first element): " + list.peek());
        // Peek Last
        System.out.println("Peek Last (last element): " + list.peekLast());
        // Pop (removes and returns first element)
        System.out.println("Pop (removes first element): " + list.pop());
        System.out.println("After pop operation: " + list);
        // Adding a collection (addAll)
        LinkedList<Integer> newList = new LinkedList<>(Arrays.asList(70, 80, 90));
        list.addAll(newList);
        System.out.println("After addAll(): " + list);
        // Getting a sublist
        List<Integer> subList = list.subList(2, 5);
        System.out.println("SubList (index 2 to 5): " + subList);
        // Sorting the list
        Collections.sort(list);
        System.out.println("After sorting: " + list);
//        Collections.sort(list,Collections.reverseOrder());
        // Sorting with custom comparator (Descending order)
        list.sort(Comparator.reverseOrder());
        System.out.println("After sorting in descending order: " + list);
        // Retrieving elements without removal
        System.out.println("Retrieve first element: " + list.getFirst());
        System.out.println("Retrieve last element: " + list.getLast());
```

```
        list.set(0,100);
        System.out.println("update element at 0th index: " + list);
    }
}
```

File: LinkedListExample3.java

```
package Collection_LinkedList;
import java.util.LinkedList;
import java.util.ListIterator;
public class LinkedListExample3 {

 public static void main(String[] args)
 {
   LinkedList<String> list = new LinkedList<String>();
   ListIterator<String> litr = list.listIterator();
     list.add("Harry");
     list.add("John");
     list.add("Tom");
     list.add("Deep");

    litr = list.listIterator();
    if(litr.next().equals("John"))
       litr.remove();
    while(litr.hasNext())
       System.out.println(litr.next());
    }
}
```

File: LinkedListExample4.java

```
package Collection_LinkedList;
import java.util.LinkedList;
public class LinkedListExample4 {
 public static void main(String[] args)
 {
   LinkedList<Integer> list = new LinkedList<Integer>();
   list.add(20);
   list.add(1,30);
   list.add(0, 40);
   list.add(50);
   list.add(2, 60);
   list.add(60);
   //40,20,60,30,50,60
   System.out.println("original Linked List :"+list); ////40,20,60,30,50,60
   System.out.println(list.peekFirst());//40
   System.out.println(list.peekLast());//60
   System.out.println(list.peek());//40


   System.out.println(list);
   System.out.println(list.lastIndexOf(60));//return index
   System.out.println(list.element());//Retrieves, but does not remove, the head (first element)
of this list.
   }
}
```

File: LinkedListExample5.java

```
package Collection_LinkedList;
import java.util.LinkedList;
import java.util.ListIterator;
public class LinkedListExample5 {
 public static void main(String[] args){
  LinkedList<Integer> list = new LinkedList<Integer>();
     list.add(20);
     list.add(1,30);
     list.add(0, 40);
     list.add(2, 60);

     System.out.println("Original LinkedList "+list);//40,20,60,30
     ListIterator<Integer> litr = list.listIterator(2);
     while(litr.hasNext())
      System.out.println(litr.next());
     }
}
```

File: LinkedListMethodsExample.java

```
package Collection_LinkedList;
import java.util.LinkedList;
import java.util.ListIterator;
public class LinkedListMethodsExample {

 public static void main(String[] args) {
  LinkedList<String> list = new LinkedList<>();

  System.out.println("=== Adding Elements ===");

  // 1. addFirst(Object o)
        list.addFirst("First");
        System.out.println("After addFirst: " + list);
        // 2. addLast(Object o)
        list.addLast("Last");
        System.out.println("After addLast: " + list);
        // 7. offerFirst(Object o)-It is used to insert the specified element at the front of a
list.
        list.offerFirst("OfferFirst");
        System.out.println("After offerFirst: " + list);
        // 8. offerLast(Object o)-It is used to insert the specified element at the end of a
list.
        list.offerLast("OfferLast");
        System.out.println("After offerLast: " + list);
        // 16. push(Object o)--This method is used to push an element onto the stack represented
by a list.
        list.push("PushElement");
        System.out.println("After push: " + list);
        //iterate
        System.out.println("\n === Iterator using ListIterator ===" );
        ListIterator<String> itr = list.listIterator();
        while (itr.hasNext()) {
         System.out.println(itr.next());
  }
        System.out.println("\n=== Accessing Elements ===");
        // 3. getFirst()
        System.out.println("getFirst(): " + list.getFirst());
        // 4. getLast()
        System.out.println("getLast(): " + list.getLast());
        // 9. peek()--It retrieves the first element from the linked list
        System.out.println("peek(): " + list.peek());
        // 10. peekFirst()--It is used to retrieve the first element from the linked list. It
will return null if the list is empty.
        System.out.println("peekFirst(): " + list.peekFirst());
        // 11. peekLast()--It is used to retrieve the last element from the linked list. It will
return null if the list is empty.
        System.out.println("peekLast(): " + list.peekLast());
        System.out.println("\n=== Removing Elements ===");
        // 5. removeFirst()
        System.out.println("removeFirst(): " + list.removeFirst());
        System.out.println("List after removeFirst: " + list);
        // 6. removeLast()
```

```
        System.out.println("removeLast(): " + list.removeLast());
        System.out.println("List after removeLast: " + list);
        // 12. poll()--It retrieves and removes the first element from the linked list.
        System.out.println("poll(): " + list.poll());
        System.out.println("List after poll: " + list);
        // 13. pollFirst()
        System.out.println("pollFirst(): " + list.pollFirst());
        System.out.println("List after pollFirst: " + list);
        // 14. pollLast()
        System.out.println("pollLast(): " + list.pollLast());
        System.out.println("List after pollLast: " + list);
        // Adding elements to test pop
        list.push("Element1");
        list.push("Element2");
        list.push("Element3");
        System.out.println("\nList before pop: " + list);
        // 15. pop()--This method is used to pop an element from the stack represented by a
list.
        System.out.println("pop(): " + list.pop());
        System.out.println("List after pop: " + list);


 }
}
```

File: SupportTicket.java

```
package Collection_LinkedList;
public class SupportTicket {
 private int id;
    private String customerName;
    private String issue;
    private int priority; // Lower number = Higher priority
    public SupportTicket(int id, String customerName, String issue, int priority) {
        this.id = id;
        this.customerName = customerName;
        this.issue = issue;
        this.priority = priority;
    }
    public int getPriority() {
        return priority;
    }
    @Override
    public String toString() {
        return "Ticket ID: " + id + ", Customer: " + customerName + ", Issue: " + issue + ",
Priority: " + priority;
    }
}
```

File: TicketingSystem.java

```
package Collection_LinkedList;
import java.util.Comparator;
import java.util.LinkedList;
public class TicketingSystem {
  private LinkedList<SupportTicket> ticketQueue;
     public TicketingSystem() {
         this.ticketQueue = new LinkedList<>();
     }
     // Add a new support ticket
     public void addTicket(SupportTicket ticket) {
         ticketQueue.add(ticket);
         System.out.println("Added new ticket: " + ticket);
     }
     // Peek at the next ticket to be processed
     public void peekNextTicket() {
         if (!ticketQueue.isEmpty()) {
             System.out.println("Next Ticket: " + ticketQueue.peek());
         } else {
             System.out.println("No tickets available.");
         }
     }
     // Process (remove) the highest-priority ticket
     public void processNextTicket() {
         if (!ticketQueue.isEmpty()) {
             SupportTicket processedTicket = ticketQueue.poll();
             System.out.println("Processing ticket: " + processedTicket);
         } else {
             System.out.println("No tickets to process.");
         }
     }
     // View all pending tickets
     public void viewAllTickets() {
         if (ticketQueue.isEmpty()) {
             System.out.println("No pending tickets.");
         } else {
             System.out.println("Pending Tickets:");
             for (SupportTicket ticket : ticketQueue) {
                 System.out.println(ticket);
             }
         }
     }
     // Sort tickets by priority (lower number = higher priority)
     public void sortTicketsByPriority() {
         ticketQueue.sort(Comparator.comparingInt(SupportTicket::getPriority));
         System.out.println("Tickets sorted by priority.");
     }
}
```

Package: Collection_PriorityQueue

File: PriorityQueueExample.java

```
package Collection_PriorityQueue;
import java.util.PriorityQueue;
//Orders elements based on priority
//Does not allow null elements.
//Duplicates are allowed.
//Not thread-safe
//smallest element is at the head
//Priority Queue Does not follow the FIFO Rule.
public class PriorityQueueExample {
 public static void main(String[] args) {
        // Creating a PriorityQueue (Min-Heap)
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        // Adding elements
        pq.add(30);
        pq.add(10);
        pq.add(50);
        pq.add(20);
        pq.add(20);
        System.out.println("PriorityQueue: " + pq); // Order may not be sorted when printed
        // Removing elements (smallest first)
        System.out.println("Polling: " + pq.poll()); // 10 (smallest)
        System.out.println("Polling: " + pq.poll()); // 20
        System.out.println("Remaining Queue: " + pq);// [30,50]
    }
}
```

File: PriorityQueueMaxHeap.java

```
package Collection_PriorityQueue;
import java.util.Comparator;
import java.util.PriorityQueue;
public class PriorityQueueMaxHeap {
  public static void main(String[] args) {
        // Max-Heap (Descending Order)
        PriorityQueue<Integer> pq = new PriorityQueue<>(Comparator.reverseOrder());
        pq.add(30);
        pq.add(10);
        pq.add(50);
        pq.add(20);
        System.out.println("PriorityQueue (Max-Heap): " + pq);
        // Removing elements (largest first)
        System.out.println("Polling: " + pq.poll()); // 50
        System.out.println("Polling: " + pq.poll()); // 30
        pq.add(100);
        System.out.println("Remaining Queue: " + pq);
  }
}
```

File: PriorityQueueOperations.java

```
package Collection_PriorityQueue;
import java.util.PriorityQueue;
public class PriorityQueueOperations {
    public static void main(String[] args) {
     // Creating a Min-Heap PriorityQueue (Default: Smallest element has highest priority)
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        // Adding elements
        pq.add(30);
        pq.add(10);
        pq.add(50);
        pq.add(20);
        pq.offer(40); // offer() also adds elements
        System.out.println("PriorityQueue: " + pq);
        // Peek (Get highest priority element without removing)
        System.out.println("Peek (Top Element): " + pq.peek()); // 10 (smallest)
        // Poll (Remove and return highest priority element)
        System.out.println("Polling: " + pq.poll()); // Removes 10
        System.out.println("PriorityQueue after poll: " + pq);
        // Remove a specific element
        boolean removed = pq.remove(30); // Removes 30
        System.out.println("Was 30 removed? " + removed);
        System.out.println("PriorityQueue after remove(30): " + pq);
        // Check if a specific element exists
        System.out.println("Contains 50? " + pq.contains(50)); // true
        // Get the size of the queue
        System.out.println("Size of PriorityQueue: " + pq.size()); // 3 elements left
        // Convert PriorityQueue to Array
        Object[] arr = pq.toArray();
        System.out.print("Elements in array format: ");
        for (Object num : arr) {
            System.out.print(num + " ");
        }
        System.out.println();
        // Polling all elements until empty
        while (!pq.isEmpty()) {
            System.out.println("Polling: " + pq.poll());
        }
        // Check if empty
        System.out.println("Is PriorityQueue empty? " + pq.isEmpty()); // true
    }
}
```

Package: Collection_Set

File: DistinctCheck.java

```
package Collection_Set;
import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;
//in a given array return true if it contain distinct value and false otherwise int
a[]={1,3,3,2,4,6,7,9,0,0}
public class DistinctCheck {
 public static void main(String[] args) {
  int[] a = {1, 3, 3, 2, 4, 6, 7, 9, 0, 0};
//  int[] a = {1, 2, 3, 4, 5, 6, 7, 8};

  hasAllDistinct(a);
  usingStream(a);
 }
 private static void hasAllDistinct(int[] a) {
  int originalSize = a.length;
  Set<Integer> set = new HashSet<>();
  for (int no : a) {
            set.add(no);
        }
  if(set.size() == originalSize) {
   System.out.println("Array Contains Unique Values");
  }else {
   System.out.println("Array doesnot Contains Unique Values");
  }
 }
 private static void usingStream(int[] a) {
  Boolean b= Arrays.stream(a)
     .boxed()
     .distinct()
     .count() == a.length;

  if(b) {
   System.out.println("Array Contains Unique Values");
  }else {
   System.out.println("Array doesnot Contains Unique Values");
  }
 }
}
```

File: HashSetBasicOperation.java

```
package Collection_Set;
import java.util.HashSet;
import java.util.Iterator;
// No Insertion Order Maintain
// No Duplicate Element
// Null Allowed but for only one time
// Faster for Searching Operation
// Non Synchronized
public class HashSetBasicOperation {
 public static void main(String[] args) {

  HashSet<String> set = new HashSet<>();
  set.add("Apple");
        set.add("Banana");
        set.add("Mango");
        set.add("Apple"); // Duplicate, will be ignored
        set.add(null); // Only One time allowed to set null value

        Iterator<String> itr = set.iterator();
        while (itr.hasNext()) {
   System.out.println(itr.next());
  }
 }
}
```

File: HashSetClassBaseExample.java

```
package Collection_Set;
import java.util.HashSet;
import java.util.Objects;
public class HashSetClassBaseExample {
  public static void main(String[] args) {
         // Creating a HashSet of Student objects
         HashSet<ClgStudent> students = new HashSet<>();
         // Adding students to HashSet
         students.add(new ClgStudent(1, "Alice", 20));
         students.add(new ClgStudent(2, "Bob", 22));
         students.add(new ClgStudent(3, "Charlie", 21));
         students.add(new ClgStudent(1, "Alice", 20)); // Duplicate based on 'id', won't be added
         // Displaying students in HashSet
         for (ClgStudent s : students) {
             System.out.println(s);
         }
     }
}
class ClgStudent{
 private int id;
    private String name;
    private int age;

    public ClgStudent(int id, String name, int age) {
        this.id = id;
        this.name = name;
        this.age = age;
    }

    // Overriding equals() to compare objects based on id
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        ClgStudent student = (ClgStudent) obj;
        return id == student.id;  // Assuming id uniquely identifies a student
    }

 // Overriding hashCode() to ensure unique hashing based on id
    @Override
    public int hashCode() {
        return Objects.hash(id);
    }
    // toString() method to print student details
    @Override
    public String toString() {
        return "Student{id=" + id + ", name='" + name + "', age=" + age + "}";
    }
}
```

File: HashSetRealTimeExample.java

```
package Collection_Set;
import java.util.HashSet;
import java.util.Scanner;
public class HashSetRealTimeExample {
 public static void main(String[] args) {

        HashSet<String> registeredUsers = new HashSet<>();
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.print("Enter a username to register (or type 'exit' to stop): ");
            String username = scanner.nextLine();

            if (username.equalsIgnoreCase("exit")) {
                break;
            }
            if (registeredUsers.add(username)) {
                System.out.println("User registered successfully!");
            } else {
                System.out.println("Username already taken. Please choose another.");
            }
        }
        System.out.println("Final registered users: " + registeredUsers);
        scanner.close();
    }
}
```

Package: Collection_TreeMap

File: TreeMapExample.java

```
package Collection_TreeMap;
import java.util.TreeMap;
//Key value pair
//No duplicate Element
//key is not allowed
//no Null key but multiple null value
//Non Synchronize
//arrange Ascending Order
public class TreeMapExample {
 public static void main(String[] args) {

  TreeMap<Integer, String> tree = new TreeMap<>();
  tree.put(1, "Apple");
  tree.put(2,"Banana");
  tree.put(5,"Greps");
  tree.put(6,"watermelon");
  tree.put(4,null);//null value allowed but null key is not allowed
//  tree.put(null,null);//null key is not allowed

  System.out.println(tree);//Asceding order
 }
}
```

File: TreeMapOperations.java

```
package Collection_TreeMap;
import java.util.Map;
import java.util.NavigableMap;
import java.util.TreeMap;
public class TreeMapOperations {
    public static void main(String[] args) {
        // 1 Creating a TreeMap (Maintains Sorted Order by Key)
        TreeMap<Integer, String> treeMap = new TreeMap<>();
        // 2 Adding Elements
        treeMap.put(3, "Apple");
        treeMap.put(1, "Banana");
        treeMap.put(4, "Cherry");
        treeMap.put(2, "Date");
        treeMap.putIfAbsent(5, "Elderberry");
        System.out.println("TreeMap: " + treeMap);
        // 3 Retrieving Elements
        System.out.println("Value for key 2: " + treeMap.get(2));
        System.out.println("Value for key 6 (or default): " + treeMap.getOrDefault(6, "Not
Found"));
        // 4 Checking Existence
        System.out.println("Contains key 3? " + treeMap.containsKey(3));
        System.out.println("Contains value 'Mango'? " + treeMap.containsValue("Mango"));
        // 5 Updating Elements
        treeMap.replace(2, "Blueberry");
        treeMap.replace(3, "Apple", "Coconut");//key , oldValue , newValue
        // 6 Removing Elements
        treeMap.remove(1);
        treeMap.remove(5, "Elderberry"); // Removes only if key-value pair matches
        // 7 Iterating Over Entries
        System.out.println("\nIterating using entrySet:");
        for (Map.Entry<Integer, String> entry : treeMap.entrySet()) {
            System.out.println(entry.getKey() + " -> " + entry.getValue());
        }
        // 8 Retrieving First & Last Entries
        System.out.println("\nFirst Entry: " + treeMap.firstEntry());
        System.out.println("Last Entry: " + treeMap.lastEntry());
        // 9 Retrieving Floor, Ceiling, Lower, and Higher Entries
        System.out.println("Floor Entry (<= 3): " + treeMap.floorEntry(3));//self or lower
        System.out.println("Ceiling Entry (>= 3): " + treeMap.ceilingEntry(3));//self or higher
        System.out.println("Lower Entry (< 3): " + treeMap.lowerEntry(3));//lower
        System.out.println("Higher Entry (> 3): " + treeMap.higherEntry(3));//higher
        // 10 Navigating using subMap, headMap, tailMap
        System.out.println("SubMap (2 to 4): " + treeMap.subMap(2, 5)); // Exclusive of 5
        System.out.println("HeadMap (< 3): " + treeMap.headMap(3)); // All keys < 3
        System.out.println("TailMap (>= 3): " + treeMap.tailMap(3)); // All keys >= 3
        // 11 Reverse Order Iteration
        NavigableMap<Integer, String> reversedMap = treeMap.descendingMap();
        System.out.println("\nTreeMap in Reverse Order:");
        for (Map.Entry<Integer, String> entry : reversedMap.entrySet()) {
            System.out.println(entry.getKey() + " -> " + entry.getValue());
        }
        // 12 Using compute(), computeIfAbsent(), computeIfPresent()
```

```
        treeMap.compute(2, (key, value) -> value + " Updated");
        treeMap.computeIfAbsent(6, key -> "Fig");
        treeMap.computeIfPresent(3, (key, value) -> value + " Modified");
        System.out.println("\nTreeMap after compute operations: " + treeMap);
    }
}
```

Package: Collection_TreeSet

File: TreeSetCustomOrder.java

```
package Collection_TreeSet;
import java.util.Comparator;
import java.util.Iterator;
import java.util.TreeSet;
public class TreeSetCustomOrder {
 //Case-sensitive sorting in descending order that why duplicate Delhi or delhi
 public static void main(String[] args) {

  TreeSet<String> city = new TreeSet<>(Comparator.reverseOrder());
  city.add("Pune");
  city.add("Mumbai");
  city.add("Chennai");
  city.add("Delhi");
  city.add("delhi");
  city.add("Bangalore");

  Iterator<String> itr = city.iterator();
  while (itr.hasNext()) {
   System.out.println(itr.next());

  }
 }
}
```

File: TreeSetExample.java

```
package Collection_TreeSet;
import java.util.TreeSet;
//Contain Unique Elements
//No Null element
//does not maintained Insertion order - it maintain ascending order
//Tree set is non-synchronous
//accessing and retriving is very Faster
//Uses Red-Black Tree internally for balancing.
public class TreeSetExample {
 public static void main(String[] args) {
  TreeSet tree = new TreeSet<>();

//  tree.add(null);// null is not allowed
  tree.add(12);
  tree.add(1);
  tree.add(15);
  tree.add(15);//Not allowed to duplicate element

  System.out.println(tree);//Asceding order is maintained
  System.out.println(tree.descendingSet());//descending order

//  for (Object obj : tree) {
//   System.out.println(obj);
//  }
 }
}
```

File: TreeSetOperations.java

```
package Collection_TreeSet;
import java.util.TreeSet;
public class TreeSetOperations {
 public static void main(String[] args) {
        TreeSet<Integer> numbers = new TreeSet<>();

        // Adding elements
        numbers.add(10);
        numbers.add(20);
        numbers.add(30);
        numbers.add(40);
        numbers.add(50);
        System.out.println("TreeSet: " + numbers);

        // Higher (Finds the next higher element)
        System.out.println("Higher than 20: " + numbers.higher(20));  // 30
        // Lower (Finds the next lower element.)
        System.out.println("Lower than 20: " + numbers.lower(20));    // 10

        // Ceiling ( given element - higher)
        System.out.println("Ceiling of 25: " + numbers.ceiling(25));  // 30

        // Floor ( given element below)
        System.out.println("Floor of 25: " + numbers.floor(25));      // 20

        // First and Last elements
        System.out.println("First Element: " + numbers.first());      // 10
        System.out.println("Last Element: " + numbers.last());        // 50

        // Checks if the set is empty
        System.out.println("Is TreeSet empty? " + numbers.isEmpty()); // false

        // Getting size
        System.out.println("Size of TreeSet: " + numbers.size());     // 5

        // Removing elements using pollFirst and pollLast--Removes & returns the first or last
element
        System.out.println("Removed first element: " + numbers.pollFirst()); // Removes 10
        System.out.println("Removed last element: " + numbers.pollLast());   // Removes 50

        numbers.add(60);
        System.out.println("Removed function element 40: " + numbers.remove(40)); // true

        // Displaying final TreeSet
        System.out.println("Updated TreeSet: " + numbers);

 }
}
```

Package: Collections_ArrayList

File: AddDuplicateAndThenRemoveDuplicateElement.java

```
package Collections_ArrayList;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;
//Can we store duplicate elements in an ArrayList? If yes, what are the ways to remove duplicate
elements from ArrayList?
public class AddDuplicateAndThenRemoveDuplicateElement {
 public static void main(String[] args) {
  List<Integer> number = new ArrayList<>();
  number.add(null);
  number.add(1);
  number.add(2);
  number.add(1);
  number.add(1);
  number.add(3);

  Collections.reverse(number);
  System.out.println("Original ArrayLisr "+number);

  //using HashSet
  Set<Integer> set = new HashSet<>(number);
  System.out.println("Remove Duplicate Using HashSet "+set);

  //using LinkedHashSet
  HashSet<Integer> hashSet = new LinkedHashSet<>(number);
  System.out.println("Remove Duplicate Using LinkedHashSet "+hashSet);

  //Using Stream
  List<Integer> stream = number.stream()
   .distinct()
   .collect(Collectors.toList());

  System.out.println("Remove Duplicate Using Stream "+stream);
 }
}
```

File: ArrayListCollection1.java

```
package Collections_ArrayList;
import java.util.ArrayList;
import java.util.List;
//ArrayList is non-synchronized
//ArrayList available in Java.util package
//Implement from List Interface
//Based on Array data Structure
//ArrayList is resizable (Dyanimc Size)
//Array is fix size but ArrayList is size grow /shring  at runtime
//It Maintained Insertion Order
//Random Access is Possible
//It can store duplicate Element
//Array List is Slow as compare to LinkedList
public class ArrayListCollection1 {
 public static void main(String[] args) {
  List<String> al = new ArrayList<String>();
  // Adding elements in the list1.
  al.add("Apple");
  al.add("Mango");
  al.add("Orange");
  al.add("Grapes");
  System.out.println("List1 contain: " + al);
  // Create another list2 of String type.
  List<String> al2 = new ArrayList<String>();
  al2.add("11");
  al2.add("12");
  al2.add("13");
  System.out.println("List2 contain :-" + al2);
  // Adding list2 in list1 at 2nd position (i.e. index = 2) using addAll() method.
  al.addAll(2, al2);
  System.out.println("List1 after adding List2 at 2nd position: " + al);
 }
}
```

File: ArrayListExample1.java

```
package Collections_ArrayList;
import java.util.ArrayList;
import java.util.List;
public class ArrayListExample1 {
 public static void main(String[] args) {

  // Creating a list of names
        List<String> names = new ArrayList<>();
        // Adding Names
        names.add("Ravi");
        names.add("Kiran");
        names.add("Atul");
        names.add("Ashwini");
        names.add("Pooja");
        // Displaying the list
        System.out.println("Names List: " + names);
        // Accessing an element
        System.out.println("First Name: " + names.get(0));
        //add at given Possition
        names.add(1,"ravikiran");
        // Removing an element
        names.remove("Atul");
        System.out.println("After removing Atul: " + names);
        // Iterating over the list
        System.out.println("Iterating over the list:");
        for (String name : names) {
            System.out.println(name);
        }

    }
}
```

File: ArrayListExample2.java

```
package Collections_ArrayList;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;
public class ArrayListExample2 {
 public static void main(String args[]) {

  //1. Creating an ArrayList
  ArrayList<String> lists = new ArrayList<>();

   // 2. Adding elements
  lists.add("Java");
  lists.add("Spring MVC");
  lists.add("Spring Boot");
  lists.add("Rubby");
  lists.add("C#");
  lists.add("PHP");
  lists.add(".Net");
  lists.add("wicket");
  lists.add("C++");

  // 3. Adding an element at a specific index
        lists.add(1, "Python");
        System.out.println("After adding at index 1: " + lists);

        // 4. Adding another list
        ArrayList<String> newFruits = new ArrayList<>();

        newFruits.add("Elderberry");
        newFruits.add("Fig");
        lists.addAll(newFruits);
        System.out.println("After adding another list: " + lists);

        // 5. Accessing elements
        System.out.println("Element at index 2: " + lists.get(2));

        // 6. Updating an element
        lists.set(3, "Dragonfruit");
        System.out.println("After updating index 3: " + lists);

        // 7. Removing elements
        lists.remove("Banana");
        System.out.println("After removing 'Banana': " + lists);
        lists.remove(2);
        System.out.println("After removing index 2: " + lists);

        // 8. Checking if an element exists
        System.out.println("Contains 'Apple'? " + lists.contains("Apple"));
        // 9. Finding index of an element
        System.out.println("Index of 'Fig': " + lists.indexOf("Fig"));
```

```
      // 10. Sorting the list
      Collections.sort(lists);
      System.out.println("Sort ArrayList " + lists);

      // 11. Reversing the list
      Collections.reverse(lists);
      System.out.println("After reversing: " + lists);

      // 12. Iterating using a loop
for(String list : lists) {
 System.out.println("Print Using For Loop "+list);
}

// 13. Iterating using an iterator
      System.out.println("Iterating using iterator:");
      Iterator<String> iterator = lists.iterator();
      while (iterator.hasNext()) {
          System.out.println(iterator.next());
      }
      // 14. SubList
      ArrayList<String> sublist = new ArrayList<String>(lists.subList(1, 7));
      System.out.println("Show Sublist from 2 to 4" + sublist);

      // 15. Converting to an array
      String[] array = lists.toArray(new String[0]);
      System.out.println("Converted to array: ");
      for (String s : array) {
          System.out.print(s +"\t");
      }

      System.out.println();

      // 16. value swap
      Collections.swap(sublist, 2, 5);
      System.out.println("Swap array value" + sublist);

      // 17. lastIndex print
      System.out.println("print Last Index" + sublist.lastIndexOf(iterator));

      // 18. toArray() - Convert ArrayList to Array
      String[] fruitArray = lists.toArray(new String[0]);
      System.out.println("Array from ArrayList: " + Arrays.toString(fruitArray));
      // 19. asList() - Convert Array to List (Fixed-size list)
      List<String> fruitList = Arrays.asList(fruitArray);
      System.out.println("List from Array: " + fruitList);
      // 20. toString() - Convert ArrayList to String
      System.out.println("ArrayList as String: " + lists.toString());
      // 21. lastIndexOf() - Find the last occurrence of an element
      int lastIndex = lists.lastIndexOf(".Net");
      System.out.println("Last index of '.Net': " + lastIndex);
      // 22. clone() - Create a shallow copy of ArrayList
      ArrayList<String> clonedList = (ArrayList<String>) lists.clone();
      System.out.println("Cloned ArrayList: " + clonedList);
```

```
        // 23. Clearing the list
        lists.clear();
        System.out.println("After clearing: " + lists);
        // 24. Checking if the list is empty
        System.out.println("Is the list empty? " + lists.isEmpty());
 }
}
```

File: ArrayListExample3.java

```
package Collections_ArrayList;
import java.util.ArrayList;
import java.util.Collections;
public class ArrayListExample3 {
 public static void main(String[] args) {
  ArrayList<String> lists = new ArrayList<>();

  lists.add("JAVA");
  lists.add("React");
  lists.add("Spring MVC");
  lists.add("Spring Boot");

  for (String list : lists) {
   System.out.println(list.toUpperCase());
  }
  System.out.println("-------------------Below is Reverse Array List---------------------");
  Collections.reverse(lists);
  for (String list : lists) {
      System.out.println(list.toUpperCase());
  }

  System.out.println("--------Update------");
  lists.set(2, "Spring AOP");
  System.out.println(lists);

  System.out.println("--------Remove------");
  lists.remove(1);
  System.out.println(lists);

  System.out.println("--------Sort by Asc Order ------");
  Collections.sort(lists);
  System.out.println(lists);

  //clone
  ArrayList<String> newList = new ArrayList<>();
  System.out.println("To Check if arrayList is empty or not");
  if(newList.isEmpty()) {
   System.out.println("Given List is Empty !!! ");
  }else {
   System.out.println("Given List is not Empty !!! ");
  }
  newList.addAll(lists);

  System.out.println("To Check if arrayList is empty or not");
  if(newList.isEmpty()) {
   System.out.println("Given List is Empty !!! ");
  }else {
   System.out.println("Given List is not Empty !!! ");
  }


 }
```

```
}
```

File: ArrayListExample3RealEx.java

```
package Collections_ArrayList;
import java.util.ArrayList;
public class ArrayListExample3RealEx {
 public static void main(String[] args) {
  Employee e1 = new Employee(1, "Ravikiran");
  Student s1 = new Student(1, "Chavan");

  ArrayList a1 = new ArrayList<>();
  a1.add(e1);
  a1.add(s1);

  for (Object a : a1) {
   System.out.println("Array List Print - "+a);
  }
  for(Object obj : a1) {
   if(obj instanceof Employee) {
    Employee ee = (Employee)obj;
    System.out.println(ee.empId +" - "+ee.empName);
   }else if(obj instanceof Student){
    Student ss = (Student)obj;
    System.out.println(ss.studentId +" - "+ss.studentName);
   }
  }
 }
}
```

File: ArrayListExample4.java

```
package Collections_ArrayList;
import java.util.ArrayList;
public class ArrayListExample4 {
 public static void main(String[] args)
 {
    ArrayList<String> list = new ArrayList<String>();
      list.add(null);
      list.add(1, null);
      list.add(1, null);
      list.add(1, null);
      System.out.println(list);

      ArrayList<String> list1 = new ArrayList<String>();
      list1.add(null);
      list1.add(0, "A");
      list1.add(2, "B");
      list1.add(1, "C");
    //[null]
    //[A,null,B]
    //[A,C,null,B]

      ArrayList<String> list2 = new ArrayList<String>();
      list.add(null);
      list.add(0, "A");
      list.add(null);
      list.add(2, "B");
      list.add("20");
      list.add(1, "C");
     System.out.println(list);
   }
}
```

File: AscendingAndDescendingArrayList.java

```
package Collections_ArrayList;
import java.util.ArrayList;
import java.util.Collections;
public class AscendingAndDescendingArrayList {
 public static void main(String[] args) {
  ArrayList<Integer> aList = new ArrayList<>();
  aList.add(1);
  aList.add(11);
  aList.add(3);
  aList.add(193);
  aList.add(43);
  aList.add(8);

  System.out.println("Original Array List");
  for (Integer itr : aList) {
   System.out.println(itr);
  }
  ascending(aList);
  descending(aList);
 }
 private static void ascending(ArrayList<Integer> aList) {
  Collections.sort(aList);//Automatically Ascending Order
  System.out.println("After Ascending Order");
  for (Integer itr : aList) {
   System.out.println(itr);
  }
 }
 private static void descending(ArrayList<Integer> aList) {
  System.out.println("After Descending Order - Approch 1");

  Collections.reverse(aList);
//  Collections.sort(aList,Collections.reverseOrder());
  for (Integer itr : aList) {
   System.out.println(itr);
  }

  System.out.println("After Descending Order - Approch -2");
  Collections.sort(aList,Collections.reverseOrder());
  for (Integer itr : aList) {
   System.out.println(itr);
  }
 }
}
```

File: Employee.java

```
package Collections_ArrayList;
public class Employee {
 int empId;
 String empName;

 public Employee(int empId, String empName) {
//  super();
  this.empId = empId;
  this.empName = empName;
 }
}
```

File: FindCommonElementsFromTwoArrayList.java

```
package Collections_ArrayList;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
//find common elements
public class FindCommonElementsFromTwoArrayList {
 public static void main(String[] args) {
  List<Integer> list1 = new ArrayList<>(Arrays.asList(1, 2, 3, 4, 5));
  List<Integer> list2 = new ArrayList<>(Arrays.asList(3, 4, 5, 6, 7));

  //Using Stream
  List<Integer> common = usingStream(list1, list2);
  System.out.println("Common elements: " + common);

  //Using arrayList-Traditional Way
  List<Integer> common1 = usingTraditionalWay(list1, list2);
  System.out.println("Common elements: " + common1);

 }
 private static List<Integer> usingStream(List<Integer> list1, List<Integer> list2) {
        return list1.stream()
                .filter(x->list2.contains(x)) // Check if list2 contains each element from list1
                .collect(Collectors.toList());
    }

 private static List<Integer> usingTraditionalWay(List<Integer> list1, List<Integer> list2) {
  List<Integer> common = new ArrayList<>(list1);
        common.retainAll(list2);//Keep only elements also in list2
  return common;
 }

}
```

File: FindOutMissingNumber.java

```
package Collections_ArrayList;
import java.util.ArrayList;
//Calculate expected sum using the sum formula: s=n*(n+1)/2
//Calculate actual sum of elements in the list.
//Find the missing number using :-Missing Number=Expected Sum - Actual Sum
public class FindOutMissingNumber {
 public static void main(String[] args) {
  addinArray(1,100);
 }
 private static void addinArray(int startElement, int endElement) {
  ArrayList<Integer> list = new ArrayList<>();
  int actualSum = 0;
  for(int i=startElement; i<= endElement; i++) {
   if(i != 40) {
    list.add(i);
    actualSum += i;
   }
  }
  System.out.println("List"+list);
  int expectedSum = (endElement * (endElement + 1)) / 2;
  int missingNumber = expectedSum - actualSum;

  System.out.println("Missing Number is : "+missingNumber);
 }
}
```

File: FindSecondLargestElement.java

```
package Collections_ArrayList;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import java.util.Optional;
import java.util.Set;
import java.util.TreeSet;
//Find the second largest number
public class FindSecondLargestElement {
 public static void main(String[] args) {
  ArrayList<Integer> array = new ArrayList<>();
  ArrayList<Integer> array1 = new ArrayList<>(Arrays.asList(12,23));
  array.add(40);
  array.add(13);
  array.add(44);
  array.add(25);
  array.add(74);

  System.out.println("Original Array "+array);

  //Using Collection
  Integer number = usingTree(array);
  System.out.println("Second Largest Element is "+number);

  //Using Stream
  Integer number1 = usingStream(array);
  System.out.println("Second Largest Element is "+number1);

 }
 private static Integer usingTree(ArrayList<Integer> array) {
  Set<Integer> set = new TreeSet<>();
  set.addAll(array);
  List<Integer> sortedList = new ArrayList<>();
  sortedList.addAll(set);
  return sortedList.get(set.size()-2);
 }
 private static Integer usingStream(ArrayList<Integer> array) {
  return array.stream()
    .distinct()
    .sorted(Collections.reverseOrder())
    .skip(1)
    .findFirst().get();
 }
}
```

File: ForwardBackwordDirectionPrint.java

```
package Collections_ArrayList;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.ListIterator;
public class ForwardBackwordDirectionPrint {
 public static void main(String[] args) {
  List<String> lists = new ArrayList<>();
  lists.add("Ravi");
  lists.add("Kiran");
  lists.add("Chavan");

  // Using for loop
  System.out.println("********* Forward using Traditional Way: *********");
  for (String list : lists) {
   System.out.println(list);
  }

  //Forward -  Using Iterator
        System.out.println("********* Forward using Iterator: *********");
        Iterator<String> it = lists.iterator();
        while (it.hasNext()) {
            System.out.println(it.next());
        }

        // Backward - Using Iterator
        System.out.println("********* Backword using Iterator: *********");
        ListIterator<String> listIterator = lists.listIterator(lists.size());
        while (listIterator.hasPrevious()) {
            System.out.println(listIterator.previous());
        }
 }
}
```

File: GenericsExample.java

```
package Collections_ArrayList;
import java.util.ArrayList;
// Specifing type of element to be hold in ArrayList is called as Generics
public class GenericsExample {

 public static void main(String[] args) {
  ArrayList<String> list = new ArrayList<>();//String Generic Collection

  list.add("Ravikiran");
  list.add("Chavan");
//  list.add(100);//ClassCastException run time error

  for (Object objStr : list) {
   System.out.println(objStr);
  }

  System.out.println("-----------------------------");

  ArrayList<Integer> listInt = new ArrayList<>();//Integer Generic Collection
  listInt.add(103);
  listInt.add(193);
  listInt.add(191);
//  listInt.add("ravikiran");// ClassCastException run time error

  for (Object objInt : listInt) {
   System.out.println(objInt);
  }
 }
}
```

File: GenericStudentExample.java

```
package Collections_ArrayList;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
class Student1 {
    int id;
    String name;
    double percentage;
    int age;
    // Constructor
    public Student1(int id, String name, double percentage, int age) {
        this.id = id;
        this.name = name;
        this.percentage = percentage;
        this.age = age;
    }
    // Overriding toString() for better printing
    @Override
    public String toString() {
        return "ID: " + id + ", StudentName: " + name + ", Percentage: " + percentage + "%, Age:
" + age;
    }
}
public class GenericStudentExample {
 public static void main(String[] args) {
        // Creating a generic collection (ArrayList of Students)
        ArrayList<Student1> studentList = new ArrayList<>();
        // Adding student objects to the list
        studentList.add(new Student1(1, "Ravi", 85.5, 20));
        studentList.add(new Student1(2, "Kiran", 78.2, 21));
        studentList.add(new Student1(3, "Chavan", 92.7, 22));
        // Iterating through the list and printing student details
        for (Student1 student : studentList) {
            System.out.println(student); // toString() is automatically called
        }

        //Iterate using Stream API
        List<String> names= studentList.stream()
             .map(s -> s.name)
             .collect(Collectors.toList());

        System.out.println("Student Names: " + names);
    }
}
```

File: GivenArrayListIsDivideByTwoOrNot.java

```
package Collections_ArrayList;
import java.util.ArrayList;
public class GivenArrayListIsDivideByTwoOrNot {
 public static void main(String[] args) {
  ArrayList<Integer> aList = new ArrayList<>();
  aList.add(4);
  aList.add(12);
  aList.add(15);//Not divide
  aList.add(8);

  System.out.println("Original Array List");
  for (Integer itr : aList) {
   System.out.println(itr);
  }

  if(toCheckGivenArrayListIsDivideByTwoOrNot(aList)) {
   System.out.println("Given All Array Element is divided by 2");
  }else {
   System.out.println("Given All Array Element is not divided by 2");
  }
 }
 private static boolean toCheckGivenArrayListIsDivideByTwoOrNot(ArrayList<Integer> aList) {
        for (Integer number : aList) {
            if (number % 2 != 0) {
                return false; // If any number is not divisible by 2, return false
            }
        }
        return true; // All elements are divisible by 2
    }
}
```

File: NonGenericExample.java

```
package Collections_ArrayList;
import java.util.ArrayList;
public class NonGenericExample {
  public static void main(String[] args) {
         ArrayList list = new ArrayList();  // Raw collection (not recommended)
         list.add("Java");
         list.add(100);  // Allowed, but risky
         list.add(3.14);
         for (Object obj : list) {
             String str = (String) obj;  // Type casting required (unsafe)
             System.out.println(str);
         }
     }
}
```

File: RemoveDuplicateElement.java

```
package Collections_ArrayList;
import java.util.ArrayList;
import java.util.LinkedHashSet;
import java.util.stream.Collectors;
public class RemoveDuplicateElement {
 public static void main(String[] args) {
  ArrayList<Integer> aList = new ArrayList<>();
  aList.add(1);
  aList.add(11);
  aList.add(1);
  aList.add(193);
  aList.add(43);
  aList.add(43);
  aList.add(8);

  System.out.println("Original "+aList);

  ArrayList<Integer> secondCopy = new ArrayList<>();
  //Copy Array 1st Approch
  secondCopy.addAll(aList);
  //Copy Array 2nd Approch
  //ArrayList<Integer> secondCopy = new ArrayList<>(aList);

  //Copy Array 3rd Approch
//  for (Integer num : aList) {
//      secondCopy.add(num);
//  }
  //Using LinkedHashSet
  removeDuplicateElelement(aList);

  //Without LinkedHashSet
  removeDuplicate(aList);

  //Using Stream
  removeDuplicateUsingStream(aList);
 }
 private static void removeDuplicateElelement(ArrayList<Integer> aList) {
  LinkedHashSet<Integer> linkedHashSet1 = new LinkedHashSet<>();
  linkedHashSet1.addAll(aList);
  System.out.println("Using LinkedHashSet 1st Way "+linkedHashSet1);

  LinkedHashSet<Integer> linkedHashSet = new LinkedHashSet<>(aList);
  System.out.println("Using LinkedHashSet Way 2nd "+linkedHashSet);
 }

 private static void removeDuplicate(ArrayList<Integer> aList) {
     ArrayList<Integer> uniqueList = new ArrayList<>();
     ArrayList<Integer> duplicateList = new ArrayList<>();
     for (Integer num : aList) {
         if (uniqueList.contains(num)) {
          duplicateList.add(num);
         }else {
```

```
          uniqueList.add(num);
         }
     }
     System.out.println("Using Manual Code Unique "+uniqueList);
     System.out.println("Using Manual Code Duplicate "+duplicateList);
 }
 private static void removeDuplicateUsingStream(ArrayList<Integer> aList) {
  System.out.println(aList.stream()
    .distinct()
    .collect(Collectors.toList())
  );
 }
}
```

File: Student.java

```
package Collections_ArrayList;
public class Student {
 int studentId;
 String studentName;

 public Student(int studentId, String studentName) {
//  super();
  this.studentId = studentId;
  this.studentName = studentName;
 }
}
```

Package: Collections_Stack

File: StackExample1.java

```
package Collections_Stack;
import java.util.Stack;
public class StackExample1 {
 public static void main(String[] args) {

  // Creating a Stack
        Stack<Integer> stack = new Stack<>();
        // 1. Pushing elements onto the stack
        stack.push(10);
        stack.push(60);
        stack.push(50);
        stack.push(40);
        System.out.println("Stack after push operations: " + stack);
        // 2. Peeking at the top element
        System.out.println("Top element (peek): " + stack.peek());
        // 3. Popping an element (removes top)
        int poppedElement = stack.pop();
        System.out.println("Popped element: " + poppedElement);
        System.out.println("Stack after pop: " + stack);
        // 4. Searching for an element
        int position = stack.search(60);
        if (position != -1) {
            System.out.println("Position of 20 in stack (from top): " + position);
        } else {
            System.out.println("Element not found");
        }
        // 5. Checking if stack is empty
        System.out.println("Is stack empty? " + stack.isEmpty());
        // 6. Getting the size of the stack
        System.out.println("Stack size: " + stack.size());
        // 7. Iterating through the stack
        System.out.println("Stack elements:");
        for (int num : stack) {
            System.out.println(num);
        }
    }
}
```

Package: Cursors_For_Iteration

File: Enumaration_Cursor_Ex.java

```
package Cursors_For_Iteration;
import java.util.Arrays;
import java.util.Enumeration;
import java.util.Vector;
//Enumaration work with only vector and stack--Means work with legacy classes
//No edit /no add or no remove operation
//only forward direction
public class Enumaration_Cursor_Ex {
 public static void main(String[] args) {
  Vector<String> techStack = new Vector<>(Arrays.asList("C","Java","Python","React"));

  Enumeration<String> e= techStack.elements();

  while(e.hasMoreElements()){
   System.out.println(e.nextElement());
  }
 }
}
```

File: Iterator_Ex.java

```
package Cursors_For_Iteration;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Iterator;
import java.util.List;
//Introduce in JDK 1.2
//Remove Operation only perform does not perform add or modified
//Forword Direction
//Work With List Set,Queue
//It also Called As Universal Cursor
public class Iterator_Ex {
 public static void main(String[] args) {
  List<String> techStack = new ArrayList<>(Arrays.asList("C","Java","Python","React"));

  Iterator<String> it = techStack.iterator();
  while(it.hasNext()) {
   System.out.println(it.next());
  }

  //Remove
  while(it.hasNext()) {
   if(it.next().equals("C")) {
    techStack.remove(it);
   }
  }
 }

}
```

File: LitsIterator_Ex.java

```
package Cursors_For_Iteration;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.ListIterator;
//Introduce in JDK 1.2
//Remove Operation, add and modified operation Perform
//Forword Direction As well AS Backword Direction
//Works only List (ArrayList and LinkedList)
public class LitsIterator_Ex {
  public static void main(String[] args) {
        List<String> techStack = new ArrayList<>(Arrays.asList("C", "Java", "Python", "React"));
        ListIterator<String> list = techStack.listIterator();
        // Forward Direction
        System.out.println("Forward Direction-->");
        while (list.hasNext()) {
            System.out.print(list.next() +",");
        }

        System.out.println();
        // Backward Direction
        System.out.println("Backward Direction-->");
        while (list.hasPrevious()) {
            System.out.print(list.previous() +",");
        }

        System.out.println();
        // Reset Iterator for Forward Traversal Again
        list = techStack.listIterator();
        System.out.println("Forward Direction Again-->");
        while (list.hasNext()) {
            System.out.println(list.next());
        }
        System.out.println();
        // Add "Ruby" using ListIterator
        System.out.println("Adding Ruby...");
        list.add("Ruby");
        // Reset Iterator for Modification
        list = techStack.listIterator();
        System.out.println("\nModification Java = Spring Boot -->");
        while (list.hasNext()) {
            if (list.next().equals("Java")) {
                list.set("Spring Boot");
            }
        }
        // Display Updated List
        System.out.println("Updated List: " + techStack);
        // Reset Iterator for Removal
        list = techStack.listIterator();
        while (list.hasNext()) {
            if (list.next().length() == 1) { // Removing single-character elements like "C"
                list.remove();
```

```
            }
        }
        // Display Final List
        System.out.println("Final List after Removal: " + techStack);
    }
}
```

Package: FileRecordsCounts

File: ExcelRowCounter.java

```
package FileRecordsCounts;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
public class ExcelRowCounter {
 public static void main(String[] args) {
        File folder = new File("C:\\Users\\DELL\\Desktop\\Test"); //  Change this path
        File[] files = folder.listFiles();
        if (files == null) {
            System.out.println("Folder not found or is empty.");
            return;
        }
        int totalFilesProcessed = 0;
        int totalNonEmptyFirstColCells = 0;
        for (File file : files) {
            if (file.isFile()) {
                String fileName = file.getName().toLowerCase();
                if (fileName.endsWith(".csv") || fileName.endsWith(".xls") ||
fileName.endsWith(".xlsx")) {
                    int count = countFirstColumnNonEmptyCells(file);
                    System.out.println(file.getName() + " -> Non-empty first column cells: " +
count);
                    totalFilesProcessed++;
                    totalNonEmptyFirstColCells += count;
                } else {
                    System.out.println(file.getName() + " -> Skipped (unsupported file type)");
                }
            }
        }
        System.out.println("====================================");
        System.out.println("Total CSV files processed: " + totalFilesProcessed);
        System.out.println("Total non-empty cells in first column: " +
totalNonEmptyFirstColCells);
    }
    private static int countFirstColumnNonEmptyCells(File file) {
        int count = 0;
        try (BufferedReader reader = new BufferedReader(new FileReader(file))) {
            String line;
            boolean isFirstLine = true;
            while ((line = reader.readLine()) != null) {
                if (isFirstLine) {
                    isFirstLine = false; // Skip header
                    continue;
                }
                String[] columns = line.split(",", -1); // -1 to include trailing empty columns
                if (columns.length > 0 && !columns[0].trim().isEmpty()) {
                    count++;
                }
```

```
            }
        } catch (IOException e) {
            System.out.println("Error reading file: " + file.getName());
        }
        return count;
    }
}
```

Package: FileUpload

File: TakeInput.java

```
package FileUpload;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class TakeInput {
 public static void main(String[] args) throws IOException
 {
 // Create an InputStreamReader object using a standard input stream.
    InputStreamReader isr = new InputStreamReader(System.in);
 // Create BufferedReader object to take input from the keyboard.
    BufferedReader br = new BufferedReader(isr); // Wrapping input stream reader into buffered
reader.

 // Prompt to enter a string from the user.
    System.out.println("Enter your first name: ");
    String firstName = br.readLine(); // reading data.
    System.out.println("Enter your last name: ");
    String lastName = br.readLine();
    String fullName = firstName +" " + lastName;
    System.out.println("Full name: " +fullName);
  }
}
```

Package: InnerClass

File: LocalInnerClass.java

```
package InnerClass;
public class LocalInnerClass {
  public static void main(String[] args) {
         Outer2 outer = new Outer2();
         outer.outerMethod();
     }
}
class Outer2 {
    void outerMethod() {
     System.out.println("Line 12");
        class Inner {
            void display() {
                System.out.println("Local Inner Class");
            }
        }
        System.out.println("Line 18");
        Inner inner = new Inner();
        inner.display();
    }
}
```

File: MemberInnerClass.java

```
package InnerClass;
public class MemberInnerClass {
 public static void main(String[] args) {
        Outer outer = new Outer();
        Outer.Inner inner = outer.new Inner();  // Creating Inner class object
        inner.display();
    }
}
class Outer {
    private String message = "Hello from Outer!";
    class Inner {
        void display() {
            System.out.println(message); // Accessing private member of Outer class
        }
    }
}
```

File: StaticNestedOrInnerClass.java

```
package InnerClass;
public class StaticNestedOrInnerClass {
  public static void main(String[] args) {
        Outer1.Inner inner = new Outer1.Inner();  // No need for Outer class object
        inner.display();
    }
}
class Outer1 {
    static class Inner {
        void display() {
            System.out.println("Static Inner Class");
        }
    }
}
```

Package: InterviewQuestionsOnCollections

File: ArrayIntersection.java

```
package InterviewQuestionsOnCollections;
import java.util.Arrays;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;
// Question:-Find Intersection of Two Arrays using Set
// Output :-Intersection: 2 3 5
// Common Value Reterns
public class ArrayIntersection {
  public static void main(String[] args) {
        int[] arr1 = {1, 2, 2, 3, 4, 5};
        int[] arr2 = {2, 3, 5, 6};
        Set<Integer> intersection = findIntersection(arr1, arr2);
        // Print the result
        System.out.print("Intersection: ");
        for (int num : intersection) {
            System.out.print(num + " ");
        }
        findIntersectionUsingStream(arr1,arr2);
    }
 private static Set<Integer> findIntersection(int[] arr1, int[] arr2) {
  Set<Integer> set1 = new HashSet<>();
        Set<Integer> resultSet = new HashSet<>();
        // Step 1: Add elements of arr1 to the set
        for (int num : arr1) {
            set1.add(num);
        }
        // Step 2: Check for common elements in arr2
        for (int num : arr2) {
            if (set1.contains(num)) {
                resultSet.add(num); // Add to result set to avoid duplicates
            }
        }

  return resultSet;
 }
 private static void findIntersectionUsingStream(int[] arr1, int[] arr2) {
  List<Integer> list1 = Arrays.stream(arr1)
                .boxed()
                .collect(Collectors.toList());

  List<Integer> list2 = Arrays.stream(arr2)
                .boxed()
                .collect(Collectors.toList());

  System.out.println("\nUsing Stream :"+
    list1.stream()
    .filter(x->list2.contains(x))
```

```
    .collect(Collectors.toList())
  );
 }
}
```

File: CharFrequency.java

```
package InterviewQuestionsOnCollections;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;
import java.util.stream.Collectors;
public class CharFrequency {
 public static void main(String[] args) {
  String input="Ravikiran";

  Map<Character, Integer> freqMap = usingTradionalWay(input);
  System.out.println("Using traditional way "+freqMap);

  Map<String, Long> usingStream = usingStream(input);
  System.out.println("Using usingStream way "+usingStream);

 }
 private static Map<Character, Integer> usingTradionalWay(String input) {
  Map<Character, Integer> freqMap = new HashMap<>();

        for (char ch : input.toCharArray()) {
            freqMap.put(ch, freqMap.getOrDefault(ch, 0) + 1);
        }
        return freqMap;
 }
 private static Map<String, Long> usingStream(String input) {
  return Arrays.stream(input.split(""))
  .collect(Collectors.groupingBy(x->x,Collectors.counting()));

 }
}
```

File: CollectorsMethodTerminalOperation.java

```
package InterviewQuestionsOnCollections;
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;
public class CollectorsMethodTerminalOperation {
 public static void main(String[] args) {

  List<String> list =
Arrays.asList("one","two","three","four","five","six","seven","eight","nine","ten","elevent","tw
enty");

  //toList
  List<String> toList= list.stream()
    .filter(x->x.length() >3)
    .collect(Collectors.toList());
  System.out.println(toList);

  //toSet
  Set<String> toSet=list.stream()
    .filter(x -> x.length() > 4 && x.startsWith("t"))
    .collect(Collectors.toSet());
  System.out.println(toSet);

  //toMap
  Map<String,Integer> toMap= list.stream()
    .filter(x ->x.startsWith("t"))
    .collect(Collectors.toMap(x->x, x->x.length()));
  System.out.println(toMap);

  //toMap but duplicate key
  //to Work with how to avoid collison while working with map
  Map<Integer,String> toMap1= list.stream()
    .collect(Collectors.toMap(x->x.length(), x->x,(oldValue,newValue)->oldValue +" : "+newValue));
  System.out.println(toMap1);

  //count from stream class and counting from Collectors class
  Long count=list.stream().count();
  Long counting=list.stream().collect(Collectors.counting());
  System.out.println("Count using stream class : "+count +" Counting from Collectors class :
"+counting);


  //Joining
  System.out.println(list.stream().collect(Collectors.joining(",")));
  System.out.println(list.stream().collect(Collectors.joining(",", "!", "@")));

  //partitioningBy
  System.out.println(list.stream()
    .collect(Collectors.partitioningBy(x->x.length()>3)));
  //Or
```

```
  Map<Boolean, List<String>>partitioningBy = list.stream()
      .collect(Collectors.partitioningBy(x->x.length()>3));
  System.out.println(partitioningBy);

  //partitioningBy downstream
  Map<Boolean, Long> downStreamPartitioning= list.stream()
  .collect(Collectors.partitioningBy(x->x.length()>3, Collectors.counting()));
  System.out.println(downStreamPartitioning);

  //groupingBy
  Map<Object, List<String>> grouping = list.stream()
  .collect(Collectors.groupingBy(x->x.length()));
  System.out.println(grouping);

  //grouping 2nd example
  Map<Object, String> grouping1 = list.stream()
  .collect(Collectors.groupingBy(x->x.length(),Collectors.joining("_")));
  System.out.println(grouping1);

  Map<Object, List<Object>> grouping2 =list.stream()
  .collect(Collectors.groupingBy(x->x.length(),Collectors.mapping(x->x.toUpperCase(),
Collectors.toList())));
  System.out.println(grouping2);
 }
}
```

File: CollectorsUsingToSet.java

```
package InterviewQuestionsOnCollections;
import java.util.LinkedList;
import java.util.List;
import java.util.stream.Collectors;
public class CollectorsUsingToSet {
 public static void main(String[] args) {
  List<String> language = new LinkedList<>();
  language.add("Java");
  language.add("Spring");
  language.add("Spring Boot");
  language.add("Spring MVC");
  language.add("Spring Data JPA");
  language.add("Spring Batch");
  language.add("Spring Batch");
  language.add("Java");

  System.out.println(""+language);

  System.out.println(language.stream()
  .collect(Collectors.toSet()));
 }
}
```

File: FindDuplicateElementUsingHashSet.java

```
package InterviewQuestionsOnCollections;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Scanner;
//Find Duplicate Elements in an Array using HashSet
public class FindDuplicateElementUsingHashSet {

 public static void main(String[] args) {
  Scanner scanner = new Scanner(System.in);
   System.out.print("Enter the number of elements: ");
         int size = scanner.nextInt();
         // Step 2: Declare the array
         int[] numbers = new int[size];
         // Step 3: Input elements
         System.out.println("Enter " + size + " numbers:");
         for (int i = 0; i < size; i++) {
             numbers[i] = scanner.nextInt();
         }
         // Step 4: Display the array
         System.out.print("You entered: ");
         for (int num : numbers) {
             System.out.print(num + " ");
         }
         scanner.close();
         System.out.println();
         fndDuplicateElement(numbers);
 }
 private static void fndDuplicateElement(int[] numbers) {
  HashSet<Integer> duplicateEle = new HashSet<>();

  for (int i : numbers) {
   if(!duplicateEle.isEmpty() && duplicateEle.contains(i)) {
    System.out.println("Duplicate Element is like that "+i);

   }else {
    duplicateEle.add(i);
   }
  }

 }


}
```

File: FirstNonRepeatingCharacter.java

```
package InterviewQuestionsOnCollections;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
// Find First Non-Repeating Character in a String using HashMap
public class FirstNonRepeatingCharacter {
 public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  System.out.println("Enter String");
  String word=sc.next();

  System.out.println(findFirstNonRepitingChar(word));

 }
 private static char findFirstNonRepitingChar(String word) {
  HashMap<Character, Integer> map = new HashMap<>();
  for(char ch : word.toCharArray()) {
   if (map.containsKey(ch)) {
    map.put(ch, map.get(ch) + 1);
   } else {
    map.put(ch, 1);
   }
   //Another Way using getOrDefault()
   //map.put(ch, map.getOrDefault(ch, 0)+1);
   //getOrDefault(ch, 0) ensures that new characters start with a count of 0.
   //put(ch, count + 1) updates the count in a concise way.
  }

  for (Map.Entry<Character, Integer> entry : map.entrySet()) {
            if (entry.getValue() == 1) {
                return entry.getKey();
            }
        }
  return 0;
 }
}
```

File: SortingOfArray.java

```
package InterviewQuestionsOnCollections;
import java.util.Arrays;
import java.util.Collections;
import java.util.stream.Collectors;
public class SortingOfArray {
 public static void main(String[] args) {
  Integer arr[]= {10,12,4,5,7,3,4,2};

  usingTraditionalWay(arr);
  usingStream(arr);
 }
 private static void usingTraditionalWay(Integer[] arr) {

  Arrays.sort(arr);
  System.out.println(Arrays.toString(arr));

  Arrays.sort(arr, Collections.reverseOrder()); // Descending order
        System.out.println("Descending: " + Arrays.toString(arr));

 }

 private static void usingStream(Integer[] arr) {
  System.out.println("Ascending (Stream): " +
          Arrays.stream(arr)                      // Convert to Stream<Integer>
                .sorted()                         // Sort ascending
                .collect(Collectors.toList()));

  System.out.println("Descending (Stream): " +
          Arrays.stream(arr)
                .sorted(Collections.reverseOrder()) // Sort descending
                .collect(Collectors.toList()));

 }
}
```

File: WordCount.java

```
package InterviewQuestionsOnCollections;
import java.util.Arrays;
import java.util.stream.Collectors;
public class WordCount {
 public static void main(String[] args) {
  String sentence = "java spring java docker spring boot java";

  System.out.println(Arrays.stream(sentence.split(" "))
  .collect(Collectors.groupingBy(x->x,Collectors.counting())));

 }
}
```

File: WordFrequencyCounterInGivenSentence.java

```
package InterviewQuestionsOnCollections;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
import java.util.function.Function;
import java.util.stream.Collectors;
public class WordFrequencyCounterInGivenSentence {
 public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  System.out.println("Enter Original String");
  String string=sc.nextLine();//nextLine() -Take Line From Uses ,next()--OnlyTake Single Word
  wordCountInPara(string.toLowerCase());

  withStream(string.toLowerCase());
 }
 private static void wordCountInPara(String string) {
  String [] words = string.split(" ");
  HashMap<String,Integer> charCountMap = new HashMap<>();

  for (String string2 : words) {
   if(charCountMap.containsKey(string2)) {
    charCountMap.put(string2, charCountMap.get(string2)+1);
   }else {
    charCountMap.put(string2,1);
   }
  }
  System.out.println("Count of Characters in a given string : " + charCountMap);

 }
 private static void withStream(String lowerCase) {
  String[] list =lowerCase.split(" ");
  Map<String, Long> wordFrequency = Arrays.stream(list)
                .collect(Collectors.groupingBy(x->x, Collectors.counting()));

  System.out.println("Count of Characters in a given string : " + wordFrequency);
 }
}
```

Package: Java8Feature_FunctionInterface

File: BiFunctionExample1.java

```
package Java8Feature_FunctionInterface;
import java.util.function.BiFunction;
//Accept two input parameters
//It Accept Two input Parameters and return result
// R apply(T t,U u);
public class BiFunctionExample1 {
 public static void main(String[] args) {
  //1st Concatinate String take both input string type and return also String type
  BiFunction<String, String, String> biFunction1 =(x,y) -> x.concat(y);
  System.out.println("Concatinate String -"+biFunction1.apply("Ravi", "Kiran"));//Ravikiran

  //2nd Find out of length of given concatinated string take both input parameter is String and
return integer.
  BiFunction<String, String, Integer> biFunction2 = (x,y) ->x.concat(y).length();
  System.out.println("Length of given Concatinate String "+biFunction2.apply("Ravi", "Kiran"));//9

 }
}
```

File: BinaryOperatorExample.java

```
package Java8Feature_FunctionInterface;
import java.util.function.BiFunction;
import java.util.function.BinaryOperator;
//Input & output type are same that time used UnaryOperator and BinaryOperator
//They are derived from Function and BiFunction
//Take two Input and Return Same Output type
//T apply(T t1, T t2);
public class BinaryOperatorExample {
 public static void main(String[] args) {
  //1st Concatinate String take both input string type and return also String type
  BiFunction<String, String, String> biFunction1 =(x,y) -> x.concat(y);
  System.out.println("Concatinate String -"+biFunction1.apply("Ravi", "Kiran"));//Ravikiran

  BinaryOperator<String> biFunction2 =(x,y) -> x.concat(y);
  System.out.println("Concatinate String -"+biFunction2.apply("Ravi", "Kiran"));//Ravikiran

  BinaryOperator<String> biFunction3 = (str1,str2) -> str1 +" "+ str2;
  System.out.println("Concatinate String -"+biFunction3.apply("Ravi", "Kiran"));//Ravi kiran
 }
}
```

File: CombinePredicateAndFunctionEx.java

```
package Java8Feature_FunctionInterface;
import java.util.Arrays;
import java.util.List;
import java.util.function.Function;
import java.util.function.Predicate;
import java.util.stream.Collectors;
public class CombinePredicateAndFunctionEx {
 public static void main(String[] args) {
  List<String> names=Arrays.asList("Ravi","Ravikiran","Kiran","Atul");

  System.out.println("---------------------Predicate-------------");
  Predicate<String> startsWithRavi = x -> x.startsWith("Ra");
  for (String name : names) {
   System.out.println(name+"-"+startsWithRavi.test(name));
  }

  System.out.println("---------------------Function-------------");
  Function<String, Integer> toLength= str -> str.length();
  for (String name : names) {
   System.out.println(name+"-"+toLength.apply(name));
  }


  System.out.println("--------Combination of Predicates and Function-----------");
  List<Integer> result = names.stream()
      .filter(startsWithRavi)
      .map(toLength)
      .collect(Collectors.toList());

  System.out.println(result);
 }
}
```

File: FunctionInterfaceCustomObject.java

```
package Java8Feature_FunctionInterface;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.function.Function;
public class FunctionInterfaceCustomObject {
 public static void main(String[] args) {
  Student s1 = new Student(1, "Ravi", 20L);
  Student s2 = new Student(2, "Ravikishan", 23L);
  Student s3 = new Student(3, "RaviKiran", 26L);
  Student s4 = new Student(4, "kiran", 50L);
  Student s5 = new Student(5, "Aravi", 10L);

  List<Student> students = Arrays.asList(s1, s2, s3, s4, s5);

  for (Student student : students) {
   System.out.println(student);
  }

  //find out Student List which is name starts with Ravi
  Function<List<Student>, List<Student>> startWithRaviName= x ->{
   List<Student> result =new ArrayList<>();
   for (Student student : students) {
    if (student.getName().startsWith("Ravi")) {
     result.add(student);
    };
   }
   return result;
  };
  System.out.println("Name Start With Ravi -"+startWithRaviName.apply(students));

 }

}
class Student{

 public Integer id;
 public String name;
 public Long age;

 public Integer getId() {
  return id;
 }

 public void setId(Integer id) {
  this.id = id;
 }

 public String getName() {
  return name;
 }
```

```
 public void setName(String name) {
  this.name = name;
 }

 public Long getAge() {
  return age;
 }

 public void setAge(Long age) {
  this.age = age;
 }

 public Student(Integer id, String name, Long age) {
  super();
  this.id = id;
  this.name = name;
  this.age = age;
 }
 public Student() {
  super();
 }

 @Override
 public String toString() {
  return "Student [id=" + id + ", name=" + name + ", age=" + age + "]";
 }

}
```

File: FunctionInterfaceExample1.java

```
package Java8Feature_FunctionInterface;
import java.util.function.Function;
//The Function<T, R> interface in Java 8 is a part of java.util.function package
//and represents a function that accepts one argument and produces a result.
//Function
public class FunctionInterfaceExample1 {
 public static void main(String[] args) {
//  Function<T, R>

  //Pass String and Return Result -Length of String
  String name = "Ravikiran";
  Function<String, Integer> resultLength = x ->x.length();
  System.out.println("Length of String is - "+resultLength.apply(name));

  //Return Substring--First 4 Leters
   Function<String, String> subString=x->x.substring(0, 4);
   System.out.println("Return first 4 Charecters is -"+subString.apply(name));

 }
}
```

File: UnaryOperatorExample.java

```
package Java8Feature_FunctionInterface;
import java.util.function.Function;
import java.util.function.UnaryOperator;
//Input & output type are same that time used UnaryOperator and BinaryOperator
//They are derived from Function and BiFunction
//Take one Input and Return Same Output type
//T apply(T t);
public class UnaryOperatorExample {
 public static void main(String[] args) {

  String name="Ravikiran";
  //Return Substring--First 4 Leters
   Function<String, String> subString = x-> x.substring(0, 4);
   System.out.println("Return first 4 Charecters is -"+subString.apply(name));
  //Above Example is Old Way when Both input are same,
   UnaryOperator<String> subString1 = x -> x.substring(0, 4);
   System.out.println("Return first 4 Charecters is -"+subString.apply(name));
 }
}
```

Package: Java8Feature_InterviewQuestions

File: AscendingAndDescendingNumber.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.stream.Collectors;
public class AscendingAndDescendingNumber {
 public static void main(String[] args) {
  int a[]= {2,4,2,5,1};

  List<Integer> arrayAsc = Arrays.stream(a).boxed()
   .sorted()
   .collect(Collectors.toList());
  System.out.println(arrayAsc);

  List<Integer> arrayReverse = Arrays.stream(a).boxed()
    .sorted(Collections.reverseOrder())
    .collect(Collectors.toList());
  System.out.println(arrayReverse);
 }
}
```

File: AverageOfListOfInteger.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
//Calculate the average of all the numbers
public class AverageOfListOfInteger {
 public static void main(String[] args) {
  List<Integer> list = Arrays.asList(1,2,3,4,5);

  System.out.println(list.stream().mapToDouble(x->x).average().getAsDouble());
  System.out.println(list.stream().mapToInt(x->x).sum());

 }
}
```

File: ConverStringToInteger.java

```
package Java8Feature_InterviewQuestions;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.stream.Collectors;
public class ConverStringToInteger {
 public static void main(String[] args) {
  String []a = {"2","3","7","1","8"};
  ArrayList<String> str = new ArrayList<>(Arrays.asList(a));

  System.out.println(
    str.stream()
    .map(x -> Integer.parseInt(x))
    .collect(Collectors.toList())

  );
 }
}
```

File: ConvertListToMap.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
public class ConvertListToMap {
 public static void main(String[] args) {
  List<String> singer = Arrays.asList("RA Rehaman","Lata Mangeshkar","Kumar Sanu","Sonu
Nigam","Arijit Singh");

  System.out.println(
    singer.stream()
    .collect(Collectors.toMap(x->x,x->x.length()))
  );
 }

}
```

File: ConvertStringToUpperCase.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
//Convert List of String to Uppercase
public class ConvertStringToUpperCase {
 public static void main(String[] args) {
  List<String> list = Arrays.asList("kgf","panchayat","family man","money hiest");

  System.out.println(
    list.stream()
    .map(x->x.toUpperCase())
    .collect(Collectors.toList())
  );
 }
}
```

File: ConvertToUppercaseAndContact.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;
//Convert to uppercase and concat
// Reduce:- To combine all elements of a stream into a single result.
//Why use reduce()?
// Because it returns one single result like (e.g., total, product, min, max, string join, etc.)
public class ConvertToUppercaseAndContact {
 public static void main(String[] args) {
  List<String> str = Arrays.asList("a","b","c","d","e","f");

  Optional<String> result=str.stream()
    .reduce((x,y)->(x+" "+y).toUpperCase());
  System.out.println(result.get());

 }
}
```

File: CountOfParticularSubstring.java

```
package Java8Feature_InterviewQuestions;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.IntStream;
//Count of particular substring
public class CountOfParticularSubstring {
 public static void main(String[] args) {
  String str = "whatisgoingowhatareyoudoing";
  String find= "what";

  Long count = IntStream.range(0, str.length()-find.length()-1)
  .filter(x->str.substring(x,x+find.length()).equals(find))
  .count();

  System.out.println(count);

  int count1 = anotherWay(str,find);
  System.out.println(count1);
 }
 private static int anotherWay(String str, String find) {
  List<String> newString = new ArrayList<>();
  for (int i = 0; i <= str.length()- find.length(); i++) {
   newString.add(str.substring(i,i+find.length()));
  }
  System.out.println(newString);

  return newString.stream()
    .filter(x->x.equals(find))
    .collect(Collectors.toList()).size();
 }
}
```

File: DistinctOddNumber.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
//Find Distinct Odd Number from Given List of array
public class DistinctOddNumber {
 public static void main(String[] args) {
  List<Integer> list = Arrays.asList(1,2,3,4,5,6,7,8,9,10,2,3,11,12,4,5,6);

  System.out.println(
  list.stream()
  .filter(x -> x%2 != 0)
  .distinct()
  .collect(Collectors.toList()));
 }
}
```

File: DomainCounter.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
//Find the occurrence of domains using Java streams
public class DomainCounter {
 public static void main(String[] args) {
  List<String> emails = Arrays.asList(
             "john@gmail.com",
             "alice@yahoo.com",
             "bob@gmail.com",
             "eve@hotmail.com",
             "jane@yahoo.com",
             "mark@gmail.com"
         );

  System.out.println(emails.stream()
  .map(x->x.substring(x.indexOf("@")))
  .collect(Collectors.groupingBy(x->x,Collectors.toList())));
 }
}
```

File: DuplicateFinder.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;
// find duplicate elements in a list and count their occurrences
public class DuplicateFinder {
 public static void main(String[] args) {
  List<Integer> numbers = Arrays.asList(1, 2, 3, 1, 2, 4, 5, 3, 2, 2);
   Map<Integer, Long> countMap = numbers.stream()
               .collect(Collectors.groupingBy(n -> n, Collectors.counting()));
   Map<Integer, Long> duplicates1  = countMap.entrySet().stream()
   .filter(e->e.getValue()>1)
   .collect(Collectors.toMap(e->e.getKey(), e->e.getValue()));

   System.out.println(duplicates1); // Output: {1=2, 2=4, 3=2}
 }
}
```

File: ExtractIntegersFromStringList.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.stream.Collectors;
//To extract only integer values from a List<String>, you can use Java 8 Streams and filter
strings that are valid integers.
public class ExtractIntegersFromStringList {
 public static void main(String[] args) {
  String[] s= {"ravi","123","kiran","223","23r"};

  System.out.println(
    Arrays.stream(s)
    .filter(x->x.matches("[0-9]+")) //this or below this
//    .filter(x->x.matches("\\d+")) //this is alos work
    .collect(Collectors.toList())
  );

  //Return only two first element
  System.out.println(Arrays.stream(s).limit(2).collect(Collectors.toList()));
 }
}
```

File: FibonacciExample.java

```
package Java8Feature_InterviewQuestions;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.Stream;
//Generate the first 10 Fibonacci numbers using both the traditional way and Java Streams
public class FibonacciExample {
  public static void main(String[] args) {
         System.out.println("Traditional Fibonacci:");
         printFibonacciTraditional(10);
         System.out.println();
         System.out.println("Stream-based Fibonacci:");
         printFibonacciWithStream(10);
     }
     //  Method 1: Traditional iterative way
     public static void printFibonacciTraditional(int n) {
         int a = 0, b = 1;
         for (int i = 0; i < n; i++) {
             System.out.print(a + " ");
             int next = a + b;
             a = b;
             b = next;
         }
     }
     //  Method 2: Stream-based approach
     public static void printFibonacciWithStream(int n) {
         System.out.println(
           Stream.iterate(new int[]{0, 1}, f -> new int[]{f[1], f[0] + f[1]})
              .limit(n)
              .map(f -> f[0])
              .collect(Collectors.toList())
        );
     }
}
```

File: FilterExample.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
public class FilterExample {
 public static void main(String[] args) {
  List<String> names =
Arrays.asList("Ravi","chavan","Kiran","Ravikiran","Suryakiran","ravikishor");
  System.out.println(
    names.stream()
   .filter(x->x.toLowerCase().startsWith("r"))
   .collect(Collectors.toList())
  );
 }
}
```

File: FindDuplicateElementUsingStream.java

```
package Java8Feature_InterviewQuestions;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;
public class FindDuplicateElementUsingStream {
  public static void main(String[] args) {
   List<Integer> number = new ArrayList<>(Arrays.asList(1,2,3,3,4,4,2,6));
   usingStream(number);
  }
  private static void usingStream(List<Integer> number) {
   Map<Object, Long> map = number.stream()
   .collect(Collectors.groupingBy(x->x,Collectors.counting()));

   System.out.println(
     map.entrySet().stream()
     .filter(x->x.getValue()>1)
     .collect(Collectors.toList())
   );
  }
}
```

File: FindMiddleCharacter.java

```
package Java8Feature_InterviewQuestions;
import java.util.stream.IntStream;
public class FindMiddleCharacter {
    public static void main(String[] args) {
        String input = "ravikiran";  // Example string
        int length = input.length();
        int middle = length / 2;

        //Traditional Way
        printMiddleCharacter(input,length,middle);
        //Using Stream
        printMiddleCharacterUsingStream(input,length,middle);
    }
 public static void printMiddleCharacter(String str,int length,int middle) {
        if (length % 2 == 0) {
            // Even length: print two middle characters
            System.out.println("Middle characters: " + str.charAt(middle - 1) +
str.charAt(middle));
        } else {
            // Odd length: print one middle character
            System.out.println("Middle character: " + str.charAt(middle));
        }
    }
 private static void printMiddleCharacterUsingStream(String input, int length, int middle) {
  IntStream.range(0, length)
        .filter(i -> (length % 2 == 0) ? (i == middle - 1 || i == middle) : (i == middle))
        .mapToObj(input::charAt)
        .forEach(System.out::print);
 }
}
```

File: FirstNonRepeatedCharacter.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
//Find First Non Repeated Character
public class FirstNonRepeatedCharacter {
 public static void main(String[] args) {
//  String word = "swiss";
  String word = "Hello World";
  System.out.println(
    Arrays.stream(word.split(""))
    .filter(x->word.indexOf(x) == word.lastIndexOf(x))
    .findFirst().get()
  );

  char result = firstNonRepeatedCharacter(word);
        System.out.println(result);
 }

 public static char firstNonRepeatedCharacter(String word) {
        for (int i = 0; i < word.length(); i++) {
            char currentChar = word.charAt(i);
            if (word.indexOf(currentChar) == word.lastIndexOf(currentChar)) {
                return currentChar;
            }
        }
        return '\0';  // No non-repeated character
    }
}
```

File: FirstNonRepeatingIneger.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.Collections;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.function.Function;
import java.util.stream.Collectors;
public class FirstNonRepeatingIneger {
 public static void main(String[] args) {
  Integer[] numbers = {4, 2, 5, 4, 2, 3, 5, 8, 9};
   Integer result = Arrays.stream(numbers)
               .filter(x -> Collections.frequency(Arrays.asList(numbers), x) == 1)
               .findFirst()
               .orElse(null); // use orElseThrow() if you want to throw exception
   System.out.println("First non-repeating number: " + result);

   Map<Integer, Long> countMap = Arrays.stream(numbers)
               .collect(Collectors.groupingBy(
                   Function.identity(), LinkedHashMap::new, Collectors.counting()
               ));
          Integer result1 = countMap.entrySet().stream()
               .filter(e -> e.getValue() == 1)
               .map(Map.Entry::getKey)
               .findFirst()
               .orElse(null);
          System.out.println("First non-repeating number: " + result1);

 }
}
```

File: FirstRepeatedCharacter.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
//Find First Repeaded Charectors from given string
public class FirstRepeatedCharacter {
 public static void main(String[] args) {
  String str="Hello World";

  System.out.println(Arrays.stream(str.split(""))
  .filter(x->str.indexOf(x) != str.lastIndexOf(x))
  .findFirst().get());
 }
}
```

File: FlatMapExample.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.IntStream;
public class FlatMapExample {
 public static void main(String[] args) {
  List<String> list = Arrays.asList("Hello world", "Java streams");

  System.out.println(
    list.stream()
    .flatMap(x->Arrays.stream(x.split(" ")))
    .collect(Collectors.toList())
  );
 }
}
```

File: GetProductOfIntArray.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
//Get Product of first 2 element
public class GetProductOfIntArray {
 public static void main(String[] args) {
  int [] a = {1,2,3,4,5,6};

  int product = Arrays.stream(a).boxed()
                .limit(2)
                .reduce(1, (x, y) -> x * y);
  System.out.println(product);
 }
}
```

File: GroupByFirstChar.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
//Group list of strings by their first character and count the number of strings
public class GroupByFirstChar {
 public static void main(String[] args) {
        List<String> words = Arrays.asList("apple", "banana", "apricot", "blueberry", "cherry",
"avocado");

        System.out.println(words.stream()
        .map(x->x)
        .collect(Collectors.groupingBy(x->x.charAt(0),Collectors.counting())));
 }
}
```

File: GroupByMiddleCharector.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.stream.Collectors;
import java.util.stream.Stream;
//Given the String[] group the string based on the middle charectors
// i/p String[] str = {"ewe","jhj","rwd","gwj","dhj","gjs","djg","fsg"};
// O/p {s=[fsg], w=[ewe, rwd, gwj], h=[jhj, dhj], j=[gjs, djg]}
public class GroupByMiddleCharector {
 public static void main(String[] args) {
  String[] str = {"ewe","jhj","rwd","gwj","dhj","gjs","djg","fsg"};

  System.out.println(Arrays.stream(str).collect(Collectors.groupingBy(x->x.toString().substring(1,
2))));


  System.out.println(Stream.of(str).collect(Collectors.groupingBy(x->x.toString().substring(1,
2))));
 }
}
```

File: IntersectionElementFromTwoList.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
//Find the intersection of two lists using Java streams
public class IntersectionElementFromTwoList {
 public static void main(String[] args) {
  List<Integer> list1 = Arrays.asList(1,2,3,4,5);
  List<Integer> list2 = Arrays.asList(2,4,5,6,7);

  System.out.println(list1.stream().filter(x->list2.contains(x)).collect(Collectors.toList()));
 }
}
```

File: IsArrayDistinctReturnTrueFalse.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
//in a given of integers,return true if it contains disticnt values and false otherwise
public class IsArrayDistinctReturnTrueFalse {
 public static void main(String[] args) {
  int a[]= {0,2,3,5,0};

  boolean isDistinct = Arrays.stream(a).boxed().distinct().count() == a.length;
  System.out.println(isDistinct);
 }
}
```

File: ListToMapExample.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;
//Convert string list to map<String,Integer>and its equivalent lenght
public class ListToMapExample {
 public static void main(String[] args) {
  List<String> list = Arrays.asList("Apple","Banana","Kiwi","Mango");

  Map<String,Integer> map = list.stream()
  .collect(Collectors.toMap(x->x, x->x.length()));

  System.out.println(map);
 }
}
```

File: ListToMapExample1.java

```
package Java8Feature_InterviewQuestions;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;
//To Convert List to Map
public class ListToMapExample1 {
 public static void main(String[] args) {
  List<Integer> numbers = new ArrayList<>();
  numbers.add(1);
  numbers.add(2);
  numbers.add(13);
  numbers.add(12);
  numbers.add(14);

  Map<Integer,Integer> num = numbers.stream()
         .collect(Collectors.toMap(x->x, x->x));

  System.out.println(num);
 }
}
```

File: LongestWordFinder.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Map;
import java.util.Optional;
import java.util.stream.Collectors;
//Question:-Java finds the longest word easily using streams or normal way
public class LongestWordFinder {
 public static void main(String[] args) {
  String sentence = "Java finds the longest word easily using streams or normal way";

  //Old Techniques
  String oldTechniques=usingOldTechnique(sentence);
  System.out.println(oldTechniques);

  //New Techniques -using Stream
  String newTechniques=usingNewTechnique(sentence);
  System.out.println(newTechniques);

  //New Techniques -- Using Stream
  String newTechniques1=usingNewTechniqueNew(sentence);
  System.out.println(newTechniques1);
 }
 private static String usingOldTechnique(String sentence) {
  String longestWord ="";
  String[] words = sentence.split(" ");
  for (String word : words) {
   if(word.length() >longestWord.length()) {
    longestWord=word;
   }
  }
  return "longest Word is '"+longestWord +"' and lenght of this word is "+longestWord.length();
 }
 private static String usingNewTechnique(String sentence) {
  Optional<String> longestWord = Arrays.stream(sentence.split(" "))
    .max((x,y) -> Integer.compare(x.length(), y.length()));

  return "longest Word is '"+longestWord.get() +"' and lenght of this word is
"+longestWord.get().length();
 }
 private static String usingNewTechniqueNew(String sentence) {
  Map<Integer,String> map = Arrays.stream(sentence.split(" "))
  .collect(Collectors.toMap(x->x.length(), x->x,(existing, replacement) -> existing));

  System.out.println("---------"+map.entrySet().stream()
     .sorted((e1, e2) -> e2.getKey() - e1.getKey()) // sort by key descending
//     .sorted(Map.Entry.<Integer, String>comparingByKey(Comparator.reverseOrder()))
     .findFirst());


  return null;
 }
```

```
}
```

File: MissingElementWithStream.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.Set;
import java.util.stream.Collectors;
import java.util.stream.IntStream;
public class MissingElementWithStream {
 public static void main(String[] args) {
  int a[]= {1,2,6,7,6,3,7,8,9};

  Set<Integer> set = Arrays.stream(a)
      .boxed()
      .collect(Collectors.toSet());

  System.out.println(
    IntStream.rangeClosed(1, 9)
    .filter(x->!set.contains(x))
    .boxed().toList()
  );

 }
}
```

File: MoveAllZeroTOBeginning.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.LinkedList;
import java.util.List;
import java.util.stream.Collectors;
//wite a stream program to move all zero's to beginning of array int[]
//and in maintened insertion order then
public class MoveAllZeroTOBeginning {
 public static void main(String[] args) {
  int a[]= {0,1,3,5,0,5,0,3,0};

  System.out.println(Arrays.stream(a)
  .boxed()
  .sorted().collect(Collectors.toList()));


 List<Integer>list1 = Arrays.stream(a).boxed().filter(x -> x == 0).collect(Collectors.toList());
 List<Integer>list2 = Arrays.stream(a).boxed().filter(x -> x != 0).collect(Collectors.toList());

 List<Integer> finalList = new LinkedList<>();
 finalList.addAll(list1);
 finalList.addAll(list2);

 System.out.println(finalList);

 }
}
```

File: MultiplyArrayElementsUsingReduce.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.Optional;
//Using Reduce to return single result from multple stream element
//like that to -sum ,product,maximum,concatinate
public class MultiplyArrayElementsUsingReduce {
  public static void main(String[] args) {
   List<Integer> numbers = Arrays.asList(1,2,3,4,5);

   //sum
   Optional<Integer> sum = numbers.stream()
   .reduce((x,y)->(x+y));

   System.out.println(sum.get());

   //Product
   Optional<Integer> product = numbers.stream()
     .reduce((x,y)->(x*y));

   System.out.println(product.get());

   //maximum
   Optional<Integer> max = numbers.stream()
               .reduce((x, y) -> x > y ? x : y);
          System.out.println("Maximum: " + max.get());

   //Concatinate
         Optional<String> concatenated = numbers.stream()
                 .map(String::valueOf)
                 .reduce((x, y) -> x + y);
             System.out.println("Concatenated: " + concatenated.get());
  }

}
```

File: NthLongestWord.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.stream.Collectors;
//Given a sentence ,find the word that has the 2nd  nth highest lenght
public class NthLongestWord {
 public static void main(String[] args) {
  String sentence = "I am Learing Stream API in Java";
  //Largest
  String nthLargest = Arrays.stream(sentence.split("
")).sorted(Comparator.comparing(String::length).reversed()).skip(1).findFirst().get();
  System.out.println(nthLargest +" Lenght is "+nthLargest.length());
  //Smallest
  System.out.println(Arrays.stream(sentence.split("
")).sorted(Comparator.comparing(String::length)).skip(1).findFirst().get());

  System.out.println(Arrays.stream(sentence.split(" "))
  .sorted(Collections.reverseOrder(Comparator.comparingInt(String::length)))
  .skip(1)
  .findFirst()
  .get()
  );
 }
}
```

File: OccurrenceOfEachWord.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.Map;
import java.util.function.Function;
import java.util.stream.Collectors;
//Given a Sentence,find the Occurrence Of Each Word
//Ex:-I am learing Stream API in Java Java.
//O/P {Java=2,in=1,API=1,learning=1,am=1 ,Stream=1}
public class OccurrenceOfEachWord {
 public static void main(String[] args) {
  String sentence = "I am learing Stream API in Java Java";

  Map<String, Long> map = Arrays.stream(sentence.split(" "))
    .collect(Collectors.groupingBy(Function.identity(),Collectors.counting()));
  System.out.println(map);

  //Another Way
  Map<String, Long> map1 = Arrays.stream(sentence.split(" "))
     .collect(Collectors.groupingBy(x->x,Collectors.counting()));
  System.out.println(map1);
 }
}
```

File: PalindromeStreamEasy.java

```
package Java8Feature_InterviewQuestions;
import java.util.stream.IntStream;
//Palindrome Example
public class PalindromeStreamEasy {
 public static void main(String[] args) {
  String input = "madam";

  usingTraditionalWay(input);

  usingStream(input);
 }
 private static void usingTraditionalWay(String input) {
  String reversed = new StringBuilder(input).reverse().toString();
        if (input.equals(reversed)) {
            System.out.println("Palindrome");
        } else {
            System.out.println("Not a palindrome");
        }
 }
 private static void usingStream(String input) {
  boolean isPalindrome = IntStream.range(0, input.length() / 2)
              .allMatch(i -> input.charAt(i) == input.charAt(input.length() - 1 - i));
         System.out.println(isPalindrome ? "Palindrome" : "Not a palindrome");
 }
}
```

File: PalindromeStreamEasy2Ex.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
public class PalindromeStreamEasy2Ex {

 public static void main(String[] args) {
  List<String> words = Arrays.asList("madam", "cat", "racecar", "java", "level", "world");
  List<String> palindromes = words.stream()
         .filter(PalindromeStreamEasy2Ex::isPalindrome)
         .collect(Collectors.toList());
  System.out.println("Palindromes: " + palindromes);
 }
 private static boolean isPalindrome(String str) {
  String reversed = new StringBuilder(str).reverse().toString();
  return str.equalsIgnoreCase(reversed);
 }
}
```

File: ParttitioningExample.java

```
package Java8Feature_InterviewQuestions;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.stream.Collectors;
public class ParttitioningExample {
 public static void main(String[] args) {
  ArrayList<Integer> list = new ArrayList<>(Arrays.asList(1,2,3,4,5,6,7,8));

  System.out.println(list.stream()
   .collect(Collectors.partitioningBy(x->x % 2 == 0))
   .entrySet().stream()
   .collect(Collectors.toMap(x->x.getKey() == false ? "odd" : "even", y->y.getValue()))
  );
 }
}
```

File: Person.java

```
package Java8Feature_InterviewQuestions;
public class Person {

 private String name;
 private Long age;
 private String address;

 public String getName() {
  return name;
 }
 public void setName(String name) {
  this.name = name;
 }
 public Long getAge() {
  return age;
 }
 public void setAge(Long age) {
  this.age = age;
 }
 public String getAddress() {
  return address;
 }
 public void setAddress(String address) {
  this.address = address;
 }

 public Person(String name, Long age, String address) {
  super();
  this.name = name;
  this.age = age;
  this.address = address;
 }

 @Override
 public String toString() {
  return "Person [name=" + name + ", age=" + age + ", address=" + address + "]";
 }

}
```

File: RangeBelongsToNumber.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;
//given an array of integers group the numbers by the range in which they belong.
//int a[]={1,2,3,11,15,24,25,36,47,46};
//o/p :-0=[1,2,3],10=[11,15],20=[24,25],30=[36],40=[47,46]
public class RangeBelongsToNumber {
 public static void main(String[] args) {
  int a[]= {1,2,3,11,15,24,25,36,47,46};

  Map<Object, List<Integer>>list1=Arrays.stream(a).boxed()
    .collect(Collectors.groupingBy(x->x/10*10, Collectors.toList()));
  System.out.println(list1);

  //Maintained Insertion Order
  Map<Object, List<Integer>>list2=Arrays.stream(a).boxed()
    .collect(Collectors.groupingBy(x->x/10*10,LinkedHashMap::new, Collectors.toList()));
  System.out.println(list2);
 }
}
```

File: RemoveDuplicates.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;
//To remove duplicate characters from a string while maintaining the original order, you can do
it both in a normal (traditional) way and using Java 8 Streams.
public class RemoveDuplicates {
 public static void main(String[] args) {
  String word = "ravikiran";
  //Traditional Way
  usingTraditional(word);
  System.out.println();
  System.out.println("-------------------");

  //Using HashSet
  usingHashSet(word);
  System.out.println("-------------------");

  //using Stream
  usingStream(word);
 }

 private static void usingTraditional(String word) {
  Arrays.stream(word.split("")).distinct().forEach(System.out::print);
 }
 private static void usingStream(String word) {
  String newString = "";
  for (int i = 0; i < word.length(); i++) {
   char ch = word.charAt(i);//return single charactor from string
   if (newString.indexOf(ch) == -1) {//Return Index and pass char
    newString += ch;
            }
  }
  System.out.println("Result: " +newString);
 }

 private static void usingHashSet(String word) {
  StringBuilder newString = new StringBuilder();
  Set<Character> seen = new HashSet<>();
  for (char ch : word.toCharArray()) {
      if (seen.add(ch)) { // returns false if already exists
          newString.append(ch);
      }
  }
  System.out.println("Result: " + newString.toString());
 }


}
```

File: RemoveNonNumaricFromList.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
//Remove all non-numeric characters from a list.
public class RemoveNonNumaricFromList {
 public static void main(String[] args) {
  List<String> list = Arrays.asList("12dsjs3","83dsjs3","932fssjs8","022dops");

  //Get only Charectors
  System.out.println(list.stream().map(x->x.replaceAll("[aA-
zZ]","")).collect(Collectors.toList()));

  //Get Only numbers
  System.out.println(list.stream().map(x->x.replaceAll("[0-
9]+","")).collect(Collectors.toList()));
 }
}
```

File: RemoveNullEmptyElementFromArray.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
public class RemoveNullEmptyElementFromArray {
 public static void main(String[] args) {
  String[] input = {"java", "", " ", null, "spring", "Ruby"};

  List<String> cleaned = Arrays.stream(input)
    .filter(s -> s != null && !s.trim().isEmpty())
    .collect(Collectors.toList());

  System.out.println(cleaned);  // Output: [java, spring, Ruby]
 }
}
```

File: RotateLeftSimple.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.Stream;
//how to rotate an array to the left (anti-clockwise)
public class RotateLeftSimple {
 public static void main(String[] args) {
  int[] arr = {1, 2, 3, 4, 5};
        int d = 3; // number of positions to rotate

        usingTraditionalWay(arr,d);
        usingStreamWay(arr,d);
 }
 private static void usingTraditionalWay(int[] arr, int d) {
  int n = arr.length;
        int[] rotated = new int[n];
        for (int i = 0; i < n; i++) {
            rotated[i] = arr[(i + d) % n];
        }
        // Print rotated array
        for (int num : rotated) {
            System.out.print(num + " ");
        }
 }

 private static void usingStreamWay(int[] arr, int d) {
  List<Integer> list = Arrays.stream(arr)
                .boxed()
                .collect(Collectors.toList());

  List<Integer> rotated = Stream.concat(
                list.subList(d, list.size()).stream(),
                list.subList(0, d).stream()
        ).collect(Collectors.toList());

  System.out.println(rotated);
 }


}
```

File: SkipNthElementFromList.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
//Find the kth smallest element in a list of integers.
public class SkipNthElementFromList {

 public static void main(String[] args) {
  List<Integer> list = Arrays.asList(4,2,5,7,3,2);

  int secondElement = list.stream().sorted().skip(2).findFirst().get();
  System.out.println(secondElement);

 }
}
```

File: SortListOfStringsInAlphabeticalOrder.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.stream.Collectors;
public class SortListOfStringsInAlphabeticalOrder {
 public static void main(String[] args) {
  List<String> str = Arrays.asList("Puna","Zudio","Addidas","Nike","New Balance","H&M");

  System.out.println(str.stream()
    .sorted()
    .collect(Collectors.toList())
  );

  //Reverse Order
  System.out.println(str.stream()
    .sorted(Collections.reverseOrder())
    .collect(Collectors.toList())
  );
 }
}
```

File: SquaresOfAllIntList.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
//Square of All elelemtns from Integer List
public class SquaresOfAllIntList {
 public static void main(String[] args) {
  List<Integer> list = Arrays.asList(1,2,3,4,5);

  System.out.println(list.stream()
  .map(x->x*x)
  .collect(Collectors.toList()));
 }
}
```

File: StringJoinUsingObject.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
//Transform Person object stream into a single string
public class StringJoinUsingObject {
 public static void main(String[] args) {
  List<Person> people = Arrays.asList(
    new Person("Ravi", 29L, "Kasarsai pune"),
    new Person("kiran", 28L, "Thergoan pune"),
    new Person("Ravikiran", 26L, "Thergoan pune"),
    new Person("Sham", 32L, "Mumbai"),
    new Person("ram", 27L, "Dubai")
   );


  String result = people.stream()
       .map(Person::toString)
       .collect(Collectors.joining(" | "));
  System.out.println(result);

  String getOnlyName = people.stream()
       .map(Person::getName)
       .collect(Collectors.joining(" | "));
  System.out.println(getOnlyName);

  //get Age is greater than 29
  System.out.println(people.stream()
   .filter(p -> p.getAge() >= 29)
   .map(Person::getName)
   .collect(Collectors.toList())
  );
 }
}
```

File: StringsContainingOnlyDigits.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
//Find and print strings containing only digits
public class StringsContainingOnlyDigits {
 public static void main(String[] args) {
  List<String> list = Arrays.asList("123","sd233","gd35","23","sdsgd");

  System.out.println(list.stream().filter(x->x.matches("[0-9]+"))//  [\\d+]
    //.map(x->x)
    .collect(Collectors.toList()));
 }
}
```

File: SumOfAllElementsInList.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.IntSummaryStatistics;
import java.util.List;
import java.util.stream.Collectors;
//Find the sum of all the elements in a list.
public class SumOfAllElementsInList {
 public static void main(String[] args) {
  List<Integer> list = Arrays.asList(1,2,3,4,5);

  int sum=list.stream().mapToInt(x->x).sum();
  System.out.println(sum);

  IntSummaryStatistics sum1=list.stream().collect(Collectors.summarizingInt(x->x));
  System.out.println(sum1.getSum());
 }
}
```

File: SumOfUniqueElement.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.stream.Collectors;
public class SumOfUniqueElement {

 public static void main(String[] args) {
  int a[]= {1,4,2,5,2,1,2,3};
  System.out.println(Arrays.stream(a).boxed().distinct().collect(Collectors.summingInt(x->x)));
  System.out.println(Arrays.stream(a).distinct().sum());
 }
}
```

File: Test.java

```
package Java8Feature_InterviewQuestions;
import java.util.ArrayList;
import java.util.Map;
import java.util.stream.Collectors;
public class Test {
 public static void main(String[] args) {
  ArrayList<Users> userList = new ArrayList<>();
  userList.add(new Users("Ravikiran","621360932898","BRSPC1159C"));
  userList.add(new Users("Ravi","62136093234","GSPC1159C"));
  userList.add(new Users("kiran","62136093234","BFJPC1159C"));

  Map<Object, Long> map =userList.stream()
  .collect(Collectors.groupingBy(x->x.getUuid(),Collectors.counting()));

  System.out.println(map.entrySet().stream().sorted()
   .filter(x->x.getValue() > 1)
//   .distinct()
   .collect(Collectors.toList())
  );

 }
}
//select uuid ,count(*) from users group by uuid having count(*)>1;
```

File: UnionOfTwoList.java

```
package Java8Feature_InterviewQuestions;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.Stream;
public class UnionOfTwoList {
 public static void main(String[] args) {
  List<Integer> list1 = Arrays.asList(1,2,3,4,5);
  List<Integer> list2 = Arrays.asList(6,7,8,9,10);

  System.out.println(
    Stream.concat(list1.stream(),list2.stream())
    .collect(Collectors.toList())
  );
 }
}
```

File: Users.java

```
package Java8Feature_InterviewQuestions;
public class Users{

 private String name;
 private String uuid;
 private String panNo;

 public Users(String name, String uuid, String panNo) {
  super();
  this.name = name;
  this.uuid = uuid;
  this.panNo = panNo;
 }
 public String getName() {
  return name;
 }
 public void setName(String name) {
  this.name = name;
 }
 public String getUuid() {
  return uuid;
 }
 public void setUuid(String uuid) {
  this.uuid = uuid;
 }
 public String getPanNo() {
  return panNo;
 }
 public void setPanNo(String panNo) {
  this.panNo = panNo;
 }
 @Override
 public String toString() {
  return "Users [name=" + name + ", uuid=" + uuid + ", panNo=" + panNo + "]";
 }


}
```

Package: Java8Feature_Lambda

File: ConvertListToMapUserEntity.java

```
package Java8Feature_Lambda;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
public class ConvertListToMapUserEntity {
 public static void main(String[] args) {
  User user1 = new User(1, "Ravi", 25L);
        User user2 = new User(2, "Kiran", 30L);
        User user3 = new User(3, "Chavan", 28L);
        User user4 = new User(4, "Ravi", 28L);

        List<User> userList = new ArrayList<>();
        userList.add(user1);
        userList.add(user2);
        userList.add(user3);
        userList.add(user4);
        System.out.println(
          userList.stream()
          .collect(Collectors.toMap(x->x.getName(),x->x.getAge(),(existing,duplicate)->duplicate))
        );

 }
}
class User{

 public Integer id;
 public String name;
 public Long age;

 public User(Integer id, String name, Long age) {
  super();
  this.id = id;
  this.name = name;
  this.age = age;
 }
 public User() {
  super();
 }
 public Integer getId() {
  return id;
 }
 public void setId(Integer id) {
  this.id = id;
 }
 public String getName() {
  return name;
 }
 public void setName(String name) {
  this.name = name;
```

```
 }
 public Long getAge() {
  return age;
 }
 public void setAge(Long age) {
  this.age = age;
 }


}
```

File: LambdaExample1.java

```
package Java8Feature_Lambda;
//What is a Lambda Expression?
//A lambda expression is a shorter way to write an anonymous method (function).
//it is use with functional interfaces (interfaces with only one abstract method and multiple
default and static method).
//What is anonymous class?
//A nested class doen't have any name is know as anonymous class.
public class LambdaExample1 {

 public static void main(String[] args) {
        MyPrinter printer = msg -> System.out.println("Message: " + msg);
        printer.print("Hello, Lambda!");
    }
}
@FunctionalInterface
interface MyPrinter {
    void print(String message);
}
```

File: LambdaExpresionExample2.java

```
package Java8Feature_Lambda;
public class LambdaExpresionExample2 {
 public static void main(String[] args) {
  //1st
  Sum sum = () -> {
         int a = 4, b = 4;
         System.out.println(a + b);
     };
     sum.add(); // invoke the lambda

     //2nd
     Sum sum1 = () ->{
   System.out.println(4+7);
  };
  sum1.add();
 }


}
@FunctionalInterface
interface Sum {
    void add(); // no parameters, no return
}
```

File: LambdaExpresionExample3.java

```
package Java8Feature_Lambda;
public class LambdaExpresionExample3 {
 public static void main(String[] args) {
  SumOperation sumOperation =(a,b)->{
   System.out.println(a+b);
  };
  sumOperation.addOperation(20L, 5L);
 }
}
@FunctionalInterface
interface SumOperation {
    void addOperation(Long a,Long b); // no parameters, no return
}
```

File: LambdaWithMultipleParameterExample4.java

```
package Java8Feature_Lambda;
public class LambdaWithMultipleParameterExample4 {
 public static void main(String[] args) {


  MyInfterface add = (a, b) -> a + b;
  MyInfterface multiply = (a, b) -> a * b;
  MyInfterface substract = (a, b) -> a - b;

        // Using the operations
        System.out.println(add.operation(6, 3));
        System.out.println(multiply.operation(4, 5));
        System.out.println(substract.operation(4, 5));

 }
}
@FunctionalInterface
interface MyInfterface {
    int operation(int a, int b);
}
```

File: LambdaWithSingleParameterExample3.java

```
package Java8Feature_Lambda;
import java.util.ArrayList;
public class LambdaWithSingleParameterExample3 {

 public static void main(String[] args) {
   ArrayList<Integer> al = new ArrayList<Integer>();
         al.add(1);
         al.add(2);
         al.add(3);
         al.add(4);

      // Using lambda expression to print all elements of al
      System.out.println("Elements of the ArrayList: ");
      al.forEach(x->System.out.println(x));

      // Using lambda expression to print even elements of al
      System.out.println("Even elements of the ArrayList: ");
      al.forEach(y->{
       if( y % 2 == 0) {
        System.out.println("event No "+y);
       }
      });
 }
}
```

File: LambdaWithZeroParameterExample2.java

```
package Java8Feature_Lambda;
public class LambdaWithZeroParameterExample2 {
 public static void main(String[] args)
    {
        // Lambda expression with zero parameters
        ZeroParameter zeroParamLambda = () -> System.out.println("This is a zero-parameter
lambda expression!");
        // Invoke the method
        zeroParamLambda.display();
    }
}
@FunctionalInterface
interface ZeroParameter {
    void display();
}
```

Package: Java8Feature_MethodReference

File: ConstructorReferenceEx5.java

```
package Java8Feature_MethodReference;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
//Why Use Method Reference ?
//Instead of writing a lambda expression, you can refer to an existing method if it matches the
signature
//Reduce boilerplate code.
//Improve code Readability.
//Type of method Reference
// 1)Static Method Reference
// 2)Instance Method Reference
// 3)Constructor Method Reference
//2)Constructor Method Reference
public class ConstructorReferenceEx5 {
 public static void main(String[] args) {
  List<String> names = Arrays.asList("Ravi","Kiran","Atul");
  List<Student> result =names.stream()
    .map(Student::new)
    .collect(Collectors.toList());

  System.out.println(result);
 }
}
class Student{
 public String name;
 public String getName() {
  return name;
 }
 public void setName(String name) {
  this.name = name;
 }
 public Student(String name) {
  super();
  this.name = name;
 }
 @Override
 public String toString() {
  return  name ;
 }
 public Student() {
  super();
  // TODO Auto-generated constructor stub
 }
}
```

File: InstanceMethodReferenceEx3.java

```
package Java8Feature_MethodReference;
//Method Reference:-A method reference is a shorthand way of calling a method using the ::
operator .Without creating new Opbject.
//Or Method Reference is java provide a way to refer to method without invoke / executing them.
//They are a shorthand notation for lambda expresion and improve code Readability.
import java.util.Arrays;
import java.util.List;
//Why Use Method Reference ?
//Instead of writing a lambda expression, you can refer to an existing method if it matches the
signature
//Reduce boilerplate code.
//Improve code Readability.
//Type of method Reference
// 1)Static Method Reference
// 2)Instance Method Reference
// 3)Constructor Method Reference
//2)Instance Method Reference on a Object
public class InstanceMethodReferenceEx3 {

 public void print(String msg) {
  System.out.println(msg);
 }
 public static void main(String[] args) {
  InstanceMethodReferenceEx3 obj = new InstanceMethodReferenceEx3();
  List<String> list = Arrays.asList("Ravi","Kiran","Atul","Chavan");
  list.forEach(obj::print);
 }
}
```

File: InstanceMethodReferenceEx4.java

```
package Java8Feature_MethodReference;
//Method Reference:-A method reference is a shorthand way of calling a method using the ::
operator .Without creating new Opbject.
//Or Method Reference is java provide a way to refer to method without invoke / executing them.
//They are a shorthand notation for lambda expresion and improve code Readability.
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
//Why Use Method Reference ?
//Instead of writing a lambda expression, you can refer to an existing method if it matches the
signature
//Reduce boilerplate code.
//Improve code Readability.
//Type of method Reference
// 1)Static Method Reference
// 2)Instance Method Reference
// 3)Constructor Method Reference
//2)Instance Method Reference on a class
public class InstanceMethodReferenceEx4 {
 public static void main(String[] args) {
  List<String> list = Arrays.asList("Ravi","Kiran","Atul","Chavan");

  List<String> result = list.stream()
    .map(String::toLowerCase)
    .collect(Collectors.toList());

  System.out.println("Result is -"+result);
 }
}
```

File: StaticMethodReferenceEx1.java

```
package Java8Feature_MethodReference;
//Method Reference:-A method reference is a shorthand way of calling a method using the ::
operator .Without creating new Opbject.
//Or Method Reference is java provide a way to refer to method without invoke / executing them.
//They are a shorthand notation for lambda expresion and improve code Readability.
//Why Use Method Reference ?
//Instead of writing a lambda expression, you can refer to an existing method if it matches the
signature
//Reduce boilerplate code.
//Improve code Readability.
//Type of method Reference
// 1)Static Method Reference
// 2)Instance Method Reference
// 3)Constructor Method Reference
//Static Method Reference:-
public class StaticMethodReferenceEx1 {
 public static void printMessage() {
        System.out.println("Hello from static method");
    }
    public static void main(String[] args) {
        Runnable r = StaticMethodReferenceEx1::printMessage; // Method Reference
        r.run(); // Output: Hello from static method
    }
}
```

File: StaticMethodReferenceEx2.java

```
package Java8Feature_MethodReference;
//Method Reference:-A method reference is a shorthand way of calling a method using the ::
operator .Without creating new Opbject.
//Or Method Reference is java provide a way to refer to method without invoke / executing them.
//They are a shorthand notation for lambda expresion and improve code Readability.
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
//Why Use Method Reference ?
//Instead of writing a lambda expression, you can refer to an existing method if it matches the
signature
//Reduce boilerplate code.
//Improve code Readability.
//Type of method Reference
// 1)Static Method Reference
// 2)Instance Method Reference
// 3)Constructor Method Reference
//Static Method Reference:-
public class StaticMethodReferenceEx2 {

 public static boolean isEven(Integer no) {
  return no % 2 ==0;
 }

 public static void main(String[] args) {
  List<Integer> no = Arrays.asList(1,2,3,4,5,6,7,8);
  List<Integer> result=no.stream()
        .filter(StaticMethodReferenceEx2::isEven)
        .collect(Collectors.toList());

  System.out.println("Result is "+result);
 }
}
```

Package: Java8Feature_Predicates

File: BiPredicatesExample1.java

```
package Java8Feature_Predicates;
import java.util.function.BiPredicate;
//Accept two input parameters
//It Accept Two input Parameters and return boolean
// boolean test(T t , U u)
public class BiPredicatesExample1 {
 public static void main(String[] args) {

  //1st Example
  BiPredicate<String, String> biPredicates = (x,y)->x.length() == y.length();
  System.out.println("Both String are same or not - "+biPredicates.test("Ravi", "kiran"));//false
  System.out.println("Both String are same or not - "+biPredicates.test("Ravi", "Atul"));//True

  //2nd Example
  BiPredicate<String, String> biPred = (x,y) -> x.startsWith(y);
  System.out.println("1st String Starts With 'Ravi' - "+biPred.test("Ravikiran", "Ravi"));//True
  System.out.println("1st String Starts With 'Ravi' - "+biPred.test("Ravikiran", "kiran"));//fasle
 }
}
```

File: PredicatesExample1.java

```
package Java8Feature_Predicates;
import java.util.function.Predicate;
//Use in if condition instead of if we can use predicates
//
public class PredicatesExample1 {
 public static void main(String[] args) {

  Predicate<Integer> predicate = x -> x % 2 ==0;
  System.out.println(predicate.test(20));

  Predicate<Integer> predi=salary -> salary > 10000;
  System.out.println("Sallery is greater than 10000 :-"+predi.test(1000));
  //1st Way
  String name="Ravikiran";
  Predicate<String> startsWithR= x->x.startsWith("R");
  System.out.println("If given Name Start with R -"+startsWithR.test(name));

  //2nd Way
  Predicate<String> startsWithRUsingCharAt = x->x.toLowerCase().charAt(0) == 'k';
  System.out.println("If given Name Start with K -"+startsWithRUsingCharAt.test(name));

  //End With
  Predicate<String> endWithRUsingCharAt =x -> x.toLowerCase().charAt(name.length()-1)=='n';
  System.out.println("End With -"+endWithRUsingCharAt.test(name));

  //And default Method--Need both True
  Predicate<String> andPredicates=startsWithRUsingCharAt.and(endWithRUsingCharAt);
  System.out.println("'AND' - Start With K and End with N-" +andPredicates.test(name));//False

  Predicate<String> andPredicates1=startsWithR.and(endWithRUsingCharAt);
  System.out.println("'AND' - Start With R and End with N-" +andPredicates1.test(name));//True

  //Or default Method--Need any one true
  Predicate<String> orPredicates=startsWithRUsingCharAt.or(endWithRUsingCharAt);
  System.out.println("'OR' - Start With K and End with N-" +orPredicates.test(name));//true

  Predicate<String> orPredicates1=startsWithR.or(endWithRUsingCharAt);
  System.out.println("'OR' - Start With R and End with N-" +orPredicates1.test(name));//True

  //negate
  Predicate<String> negatePredicates1=startsWithR.negate();
  System.out.println("'NEGATE' -" +negatePredicates1.test(name));//True asel tr false krte
 }
}
```

Package: Java8Feature_Stream

File: StreamExample1.java

```
package Java8Feature_Stream;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
public class StreamExample1 {
 public static void main(String[] args) {
  List<String> techStack = Arrays.asList("Java","Spring Boot","Spring MVC","Spring
AOP","Angular","Postgresql","Oracle");

  List<String> result1 = techStack.stream()
  .filter(x ->x.length()>6)//Use Predicates
  .collect(Collectors.toList());
  System.out.println("result1"+result1);

  //Operation
  Long count=techStack.stream().count();
  System.out.println("count "+count);

  //distinct() and start with Spring-use filter for predicates
  List<String >distinctListStartWithS = techStack.stream()
  .filter(x -> x.startsWith("Spring"))
  .distinct()
  .collect(Collectors.toList());
  System.out.println("distinctListStartWithS :-"+distinctListStartWithS);

  //map for function() interface
  //All values to convert to uppercase
  List<String> toUpperCase= techStack.stream()
    .map(x ->x.toUpperCase())
    .collect(Collectors.toList());
  System.out.println("toUpperCase :- "+toUpperCase);

  List<String> filterAndMap = techStack.stream()
    .filter(x -> x.length()<5)
    .map(x -> x.toUpperCase())
    .collect(Collectors.toList());

  System.out.println("filterAndMap "+filterAndMap);
 }
}
```

File: StreamExample2.java

```
package Java8Feature_Stream;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
public class StreamExample2 {
 public static void main(String[] args) {

  List<Integer> number = Arrays.asList(1,2,3,4,5,6,7,8,9);

  List<Integer> square=number.stream()
   .map(x -> x * x )
   .collect(Collectors.toList());

  System.out.println("square :- "+square);

  //print one by one and iterate
  number.stream().forEach(x->System.out.println(x));



 }
}
```

File: StreamExample3_FlatMap.java

```
package Java8Feature_Stream;
import java.util.Arrays;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;
import java.util.stream.Stream;
//FlatMap--To convert Multiple List to one Single List(flatend list)
public class StreamExample3_FlatMap {

 public static void main(String[] args) {
  List<Integer> list1 = Arrays.asList(1,2,3,4);
  List<Integer> list2 = Arrays.asList(3,6,8);
  List<Integer> list3 = Arrays.asList(2,3,5,3,7);

  System.out.println(
    Stream.of(list1,list2,list3)
    .flatMap(List::stream)
    .collect(Collectors.toList())
  );

  //Distinct value
  System.out.println(
    Stream.of(list1,list2,list3)
    .flatMap(List::stream).distinct()
    .collect(Collectors.toList())
  );

  Map<String,List<Integer>> map= new LinkedHashMap<>();
  map.put("ravi", Arrays.asList(1,2,3,4,5));
  map.put("Kiran", Arrays.asList(2,2,4,6,8));
  map.put("Chavan", Arrays.asList(1,2));
  map.put("atul", Arrays.asList(2,8,9,4,5));

  //print all keys ussing keySet()
  System.out.println(map.keySet());

  //print all values
  System.out.println(map.values());

  //Converting multiple List to one Single List
  System.out.println(map.values().stream().flatMap(List::stream).collect(Collectors.toList()));
 }
}
```

File: StreamExample4_Reduce.java

```
package Java8Feature_Stream;
import java.util.Arrays;
import java.util.List;
import java.util.Optional;
public class StreamExample4_Reduce {

 public static void main(String[] args) {
  List<Integer> number = Arrays.asList(1,2,3,4,5);

  //Addition
  Optional<Integer> sum = number.stream()
  .reduce((x , y) -> (x + y));

  if(sum.isPresent()) {
   System.out.println("sum :- "+sum.get());
  }else {
   System.out.println("Value is not Present !!! ");
  }

  //Multiplication
  Optional<Integer> multi = number.stream()
    .reduce((x , y) -> (x * y));
  if(multi.isPresent()) {
   System.out.println("sum :- "+multi.get());
  }else {
   System.out.println("Value is not Present !!! ");
  }

  //Substraction
  Optional<Integer> sub = number.stream()
    .reduce((x , y) -> (x - y));
  if(multi.isPresent()) {
   System.out.println("sum :- "+sub.get());
  }else {
   System.out.println("Value is not Present !!! ");
  }

 }
}
```

File: StreamExample5_Limit.java

```
package Java8Feature_Stream;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
public class StreamExample5_Limit {
 public static void main(String[] args) {
  List<Integer> number = Arrays.asList(1,2,3,4,5,6,7,8,9,9,1,3,4,5,55,56,3,34,234);

  System.out.println(number.stream()
//  .sorted(Comparator.reverseOrder())//Decendingding Order
  .sorted((a,b)->(b-a))//Decendingding Order
  .collect(Collectors.toList()));

  System.out.println(number.stream()
    .sorted()//Ascending Order
    .sorted((a,b)->(a-b))//Ascending Order
    .collect(Collectors.toList()));
 }
}
```

Package: Multithreading

File: ConcurrentModificationExcpetion.java

```
package Multithreading;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CopyOnWriteArrayList;
public class ConcurrentModificationExcpetion {
 public static void main(String[] args) {

  failFastIterator();//Modification nahi karu shakat fail fast madhe--
ConcurrentModificationException throw error.Works on ArrayList HashMap,HashSet,LinkedList
  failSafeIterator();//Modification karu shakat fail safe madhe--
CopyOnWriteArrayList,ConcurrentHashMap

 }

 private static void failFastIterator() {
  List<String> list = new ArrayList<>();
  list.add("A");
  list.add("B");

  for(String s : list) {
   list.add("C");// ConcurrentModificationException
  }
  System.out.println(list);
 }
 private static void failSafeIterator() {
  List<String> list = new CopyOnWriteArrayList<>();
  list.add("A");
  list.add("B");

  for (String s : list) {
   list.add("C"); // No exception, iterates safely
  }
  System.out.println(list); // [A, B, C, C]
 }
}
```

File: ExtendsTreadClass.java

```
package Multithreading;
public class ExtendsTreadClass {
 public static void main(String[] args) {
        MyThread t1 = new MyThread();
        t1.start();  // Starts the thread
    }

}
class MyThread extends Thread {
    public void run() {
        System.out.println("Thread is running...");
    }
}
```

File: GetOrSetThreadNameEx.java

```
package Multithreading;
public class GetOrSetThreadNameEx {
 public static void main(String[] args) {

  // Creating threads and setting names
        MyThreads t1 = new MyThreads();
        MyThreads t2 = new MyThreads("Worker-2");
        // Getting thread names before starting
        System.out.println("Thread 1 Name: " + t1.getName());
        System.out.println("Thread 2 Name: " + t2.getName());
        // Start the threads
        t1.start();
        t2.start();
        // Setting new names after start
        t1.setName("Processor-1");
        t2.setName("Processor-2");
        // Getting thread names after renaming
        System.out.println("Renamed Thread 1: " + t1.getName());
        System.out.println("Renamed Thread 2: " + t2.getName());
    }
}
class MyThreads extends Thread {
    public MyThreads(String name) {
        super(name); // Set thread name using constructor
    }
    public MyThreads() {
 }
 @Override
    public void run() {
        System.out.println("Thread Running: " + Thread.currentThread().getName());
    }
}
```

File: ImplementingFromRunnableInterface.java

```
package Multithreading;
public class ImplementingFromRunnableInterface {
  public static void main(String[] args) {
  MyRunnable t =new MyRunnable();
        Thread t1 = new Thread(t);
        t1.start();
     }
}
class MyRunnable implements Runnable {
    public void run() {
        System.out.println("Thread is running...");
    }
}
```

File: JoinMethod.java

```
package Multithreading;
//The join() method waits for a thread to finish before proceeding another thread.
public class JoinMethod {
 public static void main(String[] args) {
        JoinExample t1 = new JoinExample();
        JoinExample t2 = new JoinExample();

        t1.start();
        try {
            t1.join(); // Waits for t1 to finish before starting t2
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        t2.start();
    }
}
class JoinExample extends Thread {
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(Thread.currentThread().getName() + " - Count: " + i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

File: SleepMethod.java

```
package Multithreading;
//thread sleep for some time.
//Always Throw InterruptedException
//after Some time it execute
public class SleepMethod {
 public static void main(String[] args) {
        SleepExample t1 = new SleepExample();
        t1.start();
    }
}
class SleepExample extends Thread {
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(Thread.currentThread().getName() + " - Count: " + i);
            try {
                Thread.sleep(2000); // Sleep for 2 second
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

File: StateMethod.java

```
package Multithreading;
//The getState() method returns the current state of a thread.
public class StateMethod {
 public static void main(String[] args) {
        StateExample t1 = new StateExample();
        System.out.println("State before start(): " + t1.getState()); // NEW
        t1.start();
        System.out.println("State after start(): " + t1.getState()); // RUNNABLE or TERMINATED
        try {
            Thread.sleep(100); // Allow thread to finish
            System.out.println("State after completion: " + t1.getState()); // TERMINATED
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
class StateExample extends Thread {
    public void run() {
        System.out.println(Thread.currentThread().getName() + " is running...");
    }
}
```

File: StopMethod.java

```
package Multithreading;
//It is use to stop thread
public class StopMethod {
 public static void main(String[] args) {
        StopExample t1 = new StopExample();
        t1.start();
        try {
            Thread.sleep(2000);
            t1.stop(); // Unsafe method
            System.out.println("Thread stopped.");
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
class StopExample extends Thread {
    public void run() {
        for (int i = 1; i <= 10; i++) {
            System.out.println(Thread.currentThread().getName() + " - Count: " + i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

File: ThreadLifecycle.java

```
package Multithreading;
class ThreadLife extends Thread {

    public void run() {
        System.out.println(Thread.currentThread().getName() + " - State: RUNNING");
        try {
            Thread.sleep(5000); // Moves to TIMED_WAITING state
            System.out.println(Thread.currentThread().getName() + " - State: WAITING");
            synchronized (this) {
                wait();  // Moves to WAITING state
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println(Thread.currentThread().getName() + " - State: TERMINATED");
    }
}
public class ThreadLifecycle {
    public static void main(String[] args) throws InterruptedException {
        ThreadLife thread = new ThreadLife();
        System.out.println(thread.getName() + " - State: NEW");
        thread.start(); // Moves to RUNNABLE
        System.out.println(thread.getName() + " - State: " + thread.getState()); // Should be
RUNNABLE
        Thread.sleep(10000);
        System.out.println(thread.getName() + " - State: " + thread.getState()); // Should be
TIMED_WAITING
        synchronized (thread) {
            thread.notify(); // Moves thread from WAITING to RUNNABLE
        }
        Thread.sleep(1000);
        System.out.println(thread.getName() + " - State: " + thread.getState()); // Should be
TERMINATED
    }
}
```

File: ThreadPriorityExample.java

```
package Multithreading;
public class ThreadPriorityExample {
  public static void main(String[] args) {
        PriorityThread t1 = new PriorityThread("Thread-1");//setThreadName=Thread-1
        PriorityThread t2 = new PriorityThread("Thread-2");
        PriorityThread t3 = new PriorityThread("Thread-3");
        // Set different priorities
        t1.setPriority(Thread.MIN_PRIORITY);  // Priority 1
        t2.setPriority(Thread.NORM_PRIORITY); // Priority 5 (default)
        t3.setPriority(Thread.MAX_PRIORITY);  // Priority 10
        // Start threads
        t1.start();
        t2.start();
        t3.start();
    }
}
 class PriorityThread extends Thread {
  public PriorityThread(String name) {
   super(name);
  }

  @Override
  public void run() {
   System.out.println(Thread.currentThread().getName() + " - Priority: " +
Thread.currentThread().getPriority());
  }
 }
```

File: YieldMethod.java

```
package Multithreading;
//Yield Method is use to temporarily stop current thread and give chance to another thread for
execution
public class YieldMethod {
 public static void main(String[] args) {
        YieldExample t1 = new YieldExample();
        YieldExample t2 = new YieldExample();

        t1.start();
        t2.start();
    }
}
class YieldExample extends Thread {
    public void run() {
        for (int i = 1; i <= 3; i++) {
            System.out.println(Thread.currentThread().getName() + " is running");
            Thread.yield(); // Give chance to other threads
        }
    }
}
```

Package: OOPSExample

File: AutoboxingUnboxingExample.java

```
package OOPSExample;
public class AutoboxingUnboxingExample {
 public static void main(String[] args) {

  // Autoboxing: Primitive  Wrapper
        int num = 10;       // Primitive
        Integer obj = num;  // Autoboxing (int  Integer)
        System.out.println("Autoboxed Integer: " + obj);
        // Unboxing: Wrapper  Primitive
        Integer number = new Integer(50);  // Wrapper Object
        int primitiveNum = number;  // Unboxing (Integer  int)
        System.out.println("Unboxed int: " + primitiveNum);
    }
}
```

File: InterfaceExample.java

```
package OOPSExample;
public class InterfaceExample {
 public static void main(String[] args) {
        Animal myDog = new Dog();
        myDog.sound();  // Output: Dog barks
    }
}
//Defining an interface
interface Animal {
 void sound();  // Abstract method (must be implemented)
}
//Implementing the interface
class Dog implements Animal {
 public void sound() {
     System.out.println("Dog barks");
 }
}
```

File: MultipleInterfaces_Example.java

```
package OOPSExample;
public class MultipleInterfaces_Example {
 public static void main(String[] args) {
        Car myCar = new Car();
        myCar.start();       // Output: Car is starting...
        myCar.fuelType();    // Output: Petrol engine
    }
}
interface Vehicle {
    void start();
}
interface Engine {
    void fuelType();
}
// A class implementing multiple interfaces
class Car implements Vehicle, Engine {
    public void start() {
        System.out.println("Car is starting...");
    }
    public void fuelType() {
        System.out.println("Petrol engine");
    }
}
```

Package: PatternExample

File: PatternExample1.java

```
package PatternExample;
import java.util.Scanner;
public class PatternExample1 {
 public static void main(String[] args) {
  Scanner sc= new Scanner(System.in);
  System.out.println("Please Enter Number :");
  int num = sc.nextInt();

  simplePattern(num);
 }
 private static void simplePattern(int num) {
  for(int i=0; i<num; i++) {
   for(int j=0; j<=i; j++) {
    System.out.print("*"+ " ");
   }
   System.out.println("");
  }
 }
}
```

File: PatternExample10.java

```
package PatternExample;
import java.util.Scanner;
public class PatternExample10 {
 public static void main(String[] args) {
  Scanner sc= new Scanner(System.in);
  System.out.println("Please Enter Number :");
  int size = sc.nextInt();

  simplePattern(size);
 }
 private static void simplePattern(int size) {
  if(size % 2 ==0) {
   size = size + 1;
  }
  for(int i=0; i<size; i++) {
   for(int j=0; j<size; j++) {
    if(j == size/2 || i == size/2) {
     System.out.print(" * ");
    }else {
     System.out.print("   ");
    }
   }
   System.out.println();
  }
 }
}
```

File: PatternExample11.java

```
package PatternExample;
import java.util.Scanner;
public class PatternExample11 {
 public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Please Enter Number :");
        int size = sc.nextInt();
        simplePattern(size);
    }
    private static void simplePattern(int size) {
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                if ((i == j) || (i + j == size - 1)) {  // Fix here
                    System.out.print("*");
                } else {
                    System.out.print(" ");
                }
            }
            System.out.println();
        }
    }
}
//*   *
// * *
//  *
// * *
//*   *
```

File: PatternExample2.java

```
package PatternExample;
import java.util.Scanner;
public class PatternExample2 {
 public static void main(String[] args) {
  Scanner sc= new Scanner(System.in);
  System.out.println("Please Enter Number :");
  int num = sc.nextInt();

  simplePattern(num);
  System.out.println("Another Way to Print");
  simplePattern1(num);
 }
 private static void simplePattern(int num) {
  for(int i=0; i<num; i++) {
   for(int j=num; j>i; j--) {
    System.out.print("*"+ " ");
   }
   System.out.println("");
  }
 }

 private static void simplePattern1(int num) {
        for (int i = 1; i <= num; i++) {
            for (int j = 0; j <= num - i; j++) {
                System.out.print("* ");
            }
            System.out.println();
        }
    }
}
//  j0 j1 j2 j3
//i0 * * * * *
//i1 * * * *
//i2 * * *
//i3 * *
//i4 *
```

File: PatternExample3.java

```
package PatternExample;
import java.util.Scanner;
public class PatternExample3 {
 public static void main(String[] args) {
  Scanner sc= new Scanner(System.in);
  System.out.println("Please Enter Number :");
  int num = sc.nextInt();

  simplePattern(num);
 }
 private static void simplePattern(int num) {
  for(int i=0; i<num; i++) {
   for(int j=0; j<=i; j++) {
    System.out.print(j+1+ " ");
   }
   System.out.println("");
  }
 }
}
```

File: PatternExample4.java

```
package PatternExample;
import java.util.Scanner;
public class PatternExample4 {
 public static void main(String[] args) {
  Scanner sc= new Scanner(System.in);
  System.out.println("Please Enter Number :");
  int num = sc.nextInt();

  simplePattern(num);
 }
 private static void simplePattern(int num) {
  for(int i=0; i<num; i++) {
   for(int j=0; j<i; j++) {
    System.out.print("-"+ " ");
   }
   for(int k=0; k<num-i; k++) {
    System.out.print("*"+" ");
   }
   System.out.println("");
  }

 }
}
//* * * * *
//- * * * *
//- - * * *
//- - - * *
//- - - - *
```

File: PatternExample5.java

```
package PatternExample;
import java.util.Scanner;
public class PatternExample5 {
 public static void main(String[] args) {
  Scanner sc= new Scanner(System.in);
  System.out.println("Please Enter Number :");
  int num = sc.nextInt();

  simplePattern(num);
 }
 private static void simplePattern(int num) {
  for(int i=0; i<num; i++) {
   for(int j=0; j<num-i; j++) {
    System.out.print("-"+ " ");
   }
   for(int k=0; k<i; k++) {
    System.out.print("*"+" ");
   }
   System.out.println("");
  }

 }
}
//- - - - - -
//- - - - - *
//- - - - * *
//- - - * * *
//- - * * * *
//- * * * * *
```

File: PatternExample6.java

```
package PatternExample;
import java.util.Scanner;
public class PatternExample6 {
 public static void main(String[] args) {
  Scanner sc= new Scanner(System.in);
  System.out.println("Please Enter Number :");
  int num = sc.nextInt();

  simplePattern(num);
 }
 private static void simplePattern(int num) {
  for(int i=0; i<num; i++) {
   for(int j=0; j<num-i; j++) {
    System.out.print(""+ " ");
   }
   for(int k=0; k<i; k++) {
    System.out.print("*"+" ");
   }
   System.out.println("");
  }

 }
}
```

File: PatternExample7.java

```
package PatternExample;
import java.util.Scanner;
public class PatternExample7 {
 public static void main(String[] args) {
  Scanner sc= new Scanner(System.in);
  System.out.println("Please Enter Number :");
  int num = sc.nextInt();

  simplePattern(num);
 }
 private static void simplePattern(int num) {
  for(int i=0; i<num; i++) {
   for(int j=0; j<num-i; j++) {
    System.out.print(" "+ " ");
   }
   for(int k=0; k<i; k++) {
    System.out.print("*"+" ");
   }
   for(int l=0; l<=i; l++) {
    System.out.print("*"+" ");
   }
   for(int m=0; m<i-1; m++) {
    System.out.print(" "+ " ");
   }
   System.out.println("");
  }

 }
}
//          *
//        * * *
//      * * * * *
//    * * * * * * *
//  * * * * * * * * *
//* * * * * * * * * * *
```

File: PatternExample8.java

```
package PatternExample;
import java.util.Scanner;
public class PatternExample8 {
 public static void main(String[] args) {
  Scanner sc= new Scanner(System.in);
  System.out.println("Please Enter Number :");
  int num = sc.nextInt();

  simplePattern(num);
 }
 private static void simplePattern(int num) {
  for (int i = 0; i < num; i++) {
            // Printing leading spaces
            for (int j = 0; j < i; j++) {
                System.out.print("  ");
            }
            // Printing stars
            for (int k = 0; k < (num - i) * 2 - 1; k++) {
                System.out.print("* ");
            }
            System.out.println();
        }
 }
}
//* * * * * * * * * * * * *
//  * * * * * * * * * * * *
//   * * * * * * * * * *
//     * * * * * * *
//       * * * * *
//         * * *
//           *
```

File: PatternExample9.java

```
package PatternExample;
import java.util.Scanner;
public class PatternExample9 {
 public static void main(String[] args) {
  Scanner sc= new Scanner(System.in);
  System.out.println("Please Enter Number :");
  int size = sc.nextInt();

  simplePattern(size);
 }
 private static void simplePattern(int size) {
  for (int i = 0; i < size; i++) {
           for(int j=0; j<size; j++) {
            if(j==0 || j==size-1) {
             System.out.print("*"+" ");
            }else {
             System.out.print(" ");
            }
           }
           System.out.println();
       }
 }
}
```

Package: Polymorphism

File: MethodOverloading.java

```
package Polymorphism;
public class MethodOverloading {
 // Method with two int parameters
    int add(int a, int b) {
        return a + b;
    }
    // Method with three int parameters
    int add(int a, int b, int c) {
        return a + b + c;
    }
    // Method with double parameters
    double add(double a, double b) {
        return a + b;
    }

    public static void main(String[] args) {
     MethodOverloading math = new MethodOverloading();
        System.out.println(math.add(5, 10));      // Calls method with two int parameters
        System.out.println(math.add(5, 10, 15));  // Calls method with three int parameters
        System.out.println(math.add(5.5, 10.5));  // Calls method with double parameters
    }
}
```

File: MethodOverriding.java

```
package Polymorphism;
public class MethodOverriding {
    public static void main(String[] args) {
        Animal myAnimal;  // Reference of superclass
        myAnimal = new Dog();
        myAnimal.makeSound(); // Calls Dog's version of makeSound()
        myAnimal = new Cat();
        myAnimal.makeSound(); // Calls Cat's version of makeSound()

        myAnimal = new Animal();
        myAnimal.makeSound();
    }
}
class Animal {
    void makeSound() {
        System.out.println("Animal makes a sound");
    }
}
class Dog extends Animal {
    @Override
    void makeSound() {
        System.out.println("Dog barks");
    }
}
class Cat extends Animal {
    @Override
    void makeSound() {
        System.out.println("Cat meows");
    }
}
```

File: MethodOverridingEx1.java

```
package Polymorphism;
abstract class Payment {
    abstract void pay(double amount);
}
class CreditCard extends Payment {
    @Override
    void pay(double amount) {
        System.out.println("Paid $" + amount + " using Credit Card.");
    }
}
class PayPal extends Payment {
    @Override
    void pay(double amount) {
        System.out.println("Paid $" + amount + " using PayPal.");
    }
}
public class MethodOverridingEx1 {
 public static void main(String[] args) {
        Payment payment;
        payment = new CreditCard();
        payment.pay(100.50);
        payment = new PayPal();
        payment.pay(200.75);
    }
}
```

Package: SingletonClass

File: MySingleton.java

```
package SingletonClass;
//What is singleton class?
//A Singleton class is a class that allows only one instance of itself to be created throughout
the entire application.
//Key Characteristics:
//1) Single instance: Only one object is ever created.
//2) Global access: The instance is globally accessible.
//3) Controlled instantiation: The class itself controls the instantiation process.
//Real-world Example
// Office Printer
//When to Use Singleton:?
//1) Logging services
//2) Configuration settings
//3) Database connections
//4) Cache managers
//Not thread-safe by default
//If you want to create then use (Synchronized Method)with Thread-safe Singleton
//For Example :-
 //public static synchronized Singleton getInstance() {
 //    if (instance == null) {
 //        instance = new Singleton();
 //    }
 //    return instance;
 //}
public class MySingleton {
    // Step 1: Private static instance
    private static MySingleton instance;
    // Step 2: Private constructor
    private MySingleton() {
        // Prevent instantiation
    }
    // Step 3: Public static method to return instance
    public static MySingleton getInstance() {
        if (instance == null) {//two threads can create two different instances if they both
enter the if (instance == null) block at the same time.
            instance = new MySingleton(); // create only if not already created
        }
        return instance;
    }
}
```

Package: StringExample

File: ChangeCaseAndLanguage.java

```
package StringExample;
import java.util.Locale;
//we will convert string str into lowercase letter using using TURKISH and ENGLISH languages.
public class ChangeCaseAndLanguage {
 public static void main(String[] args) {
  String str = new String("I LIKE IT");
  System.out.println("Original string: " + str);

  // Create Locales with language "tr" for turkish and "en" for english.
  Locale turkish = Locale.forLanguageTag("tr");
  Locale english = Locale.forLanguageTag("en");
  // Converting string str into lowercase letter using turkish and english
  // languages.
  String toLowerCaseTr = str.toLowerCase(turkish);
  String toLowerCaseEn = str.toLowerCase(english);
  System.out.println("Lowercase letters in turkish: " + toLowerCaseTr);
  System.out.println("Lowercase letters in english: " + toLowerCaseEn);
 }
}
```

File: CollectionsUtilityClass.java

```
package StringExample;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
public class CollectionsUtilityClass{
 public static void main(String[] args) {
   // Creating a List of integers
        List<Integer> numbers = new ArrayList<>(Arrays.asList(5, 2, 8, 1, 3, 8, 2, 5));
        System.out.println("Original List: " + numbers);
        // 1 Sorting the list in ascending order
        Collections.sort(numbers);
        System.out.println("Sorted List (Ascending): " + numbers);
        // 2 Sorting the list in descending order
        Collections.sort(numbers, Collections.reverseOrder());
        System.out.println("Sorted List (Descending): " + numbers);
        // 3 Reversing the list
        Collections.reverse(numbers);
        System.out.println("Reversed List: " + numbers);
        // 4 Shuffling the list (Random Order)
        Collections.shuffle(numbers);
        System.out.println("Shuffled List: " + numbers);
        // 5 Finding Maximum & Minimum Elements
        System.out.println("Max Element: " + Collections.max(numbers));
        System.out.println("Min Element: " + Collections.min(numbers));
        // 6 Counting Frequency of an Element
        System.out.println("Frequency of 8: " + Collections.frequency(numbers, 8));
        // 7 Performing Binary Search (List must be sorted first)
        Collections.sort(numbers);
        int index = Collections.binarySearch(numbers, 5);
        System.out.println("Index of 5 (Binary Search): " + index);
        // 8 Making the List Unmodifiable (Immutable)
        List<Integer> unmodifiableList = Collections.unmodifiableList(numbers);
        System.out.println("Unmodifiable List: " + unmodifiableList);
        // unmodifiableList.add(10);  //  Throws Exception: UnsupportedOperationException
        // 9 Making a Synchronized List (Thread-Safe)
        List<Integer> synchronizedList = Collections.synchronizedList(new ArrayList<>(numbers));
        // Accessing synchronizedList in a synchronized block
        synchronized (synchronizedList) {
            for (int num : synchronizedList) {
                System.out.print(num + " ");
            }
            System.out.println("\nSynchronized List Accessed");
        }
    }
}
```

File: CompareEqualsAndOperators.java

```
package StringExample;
public class CompareEqualsAndOperators {
  public static void main(String[] args) {
         String s1 = new String("Hello");
         String s2 = new String("Hello");
         String s3 = s1; // s3 points to the same object as s1
         // == Operator (Reference Comparison)
         System.out.println("s1 == s2 :"+s1 == s2); // false (Different objects)
         System.out.println("s1 == s1 :"+s1 == s1); // true (same objects)
         System.out.println(s1 == s3); // true (Same reference)
         // equals() Method (Value Comparison)
         System.out.println(s1.equals(s2)); // true (Same content)
         // compareTo() Method (Lexicographical Comparison)
         String s4 = "Apple";
         String s5 = "Banana";
         System.out.println(s4.compareTo(s5)); // Negative (-1) because "Apple" < "Banana"
         System.out.println(s5.compareTo(s4)); // Positive (1) because "Banana" > "Apple"
         System.out.println(s1.compareTo(s2)); // 0 (Both are "Hello")
     }
}
```

File: ConvertStringToCollectionOfStrings.java

```
package StringExample;
import java.util.Arrays;
import java.util.Collection;
//Convert a String to a Collection of Strings
public class ConvertStringToCollectionOfStrings {
 public static void main(String[] args) {
  String string = "apple,banana,orange,grape";

  Collection<String> collection=Arrays.asList(string.split(","));
  System.out.println(collection);


 }
}
```

File: ConvertStringToIntegerAndWisevarsa.java

```
package StringExample;
import java.math.BigDecimal;
//Convert String Integer
//Convert integer to String
public class ConvertStringToIntegerAndWisevarsa {
 public static void main(String[] args) {
  String str = "50";
  Integer no = Integer.parseInt(str);
  System.out.println("Convert String to Integer "+no);

  String newString = no.toString();
  System.out.println("convert Integer to String "+newString);


  // String to Double
        String doubleStr = "45.67";
        Double doubleVal = Double.parseDouble(doubleStr);
        System.out.println("Convert String to Double: " + doubleVal);

     // Double to String
        String doubleToStr = doubleVal.toString();
        System.out.println("Convert Double to String: " + doubleToStr);

        // String to Long
        String longStr = "123456789";
        Long longVal = Long.parseLong(longStr);
        System.out.println("Convert String to Long: " + longVal);
        // Long to String
        String longToStr = longVal.toString();
        System.out.println("Convert Long to String: " + longToStr);

        // String to Float
        String floatStr = "12.34";
        Float floatVal = Float.parseFloat(floatStr);
        System.out.println("String to Float: " + floatVal);
        System.out.println("Float to String: " + floatVal.toString());

     // String to BigDecimal
        String bigDecimalStr = "9876543210.12345";
        BigDecimal bigDecimalVal = new BigDecimal(bigDecimalStr);
        System.out.println("String to BigDecimal: " + bigDecimalVal);
        System.out.println("BigDecimal to String: " + bigDecimalVal.toString());
 }
}
```

File: CountNoOfWordsInList.java

```
package StringExample;
import java.util.Iterator;
//Write a Java program to count the number of words present in a string.
public class CountNoOfWordsInList {
 public static void main(String[] args) {
  String str = "I Love Java Programming!";

  int count1 = m1(str);
  System.out.println("Using TO Convert String to String [] "+count1);

  int count2 = m2(str);
  System.out.println("Using For Loop "+count2);

 }
 private static int m1(String str) {
   String[] strArray = str.split(" ");
  return strArray.length;
 }

 private static int m2(String str) {
  int count = 0;
  for (String word : str.split(" ")) {
   count++;
  }
  return count;
 }


}
```

File: CountOfOccurrencesCharacter.java

```
package StringExample;
import java.util.Iterator;
//Write a Java program to count the occurrences of each character in string.
public class CountOfOccurrencesCharacter {
 public static void main(String[] args) {
  String str = "I Love Java Programming";
  int count = countOccurrences(str,'o');
  System.out.println(count);
 }
 private static int countOccurrences(String str, char c) {
  String toLower = str.toLowerCase();
  int count = 0;
  for (int i = 0; i < toLower.length(); i++) {
   if(toLower.charAt(i) == c) {
    count++;
   }
  }
  return count;
 }
}
```

File: CountVowels.java

```
package StringExample;
public class CountVowels {
    public static void main(String[] args) {
        String str = "Ravikiran Aniruddh Chavan";
        int count = countVowels(str);
        System.out.println(count);  // Output: 3
    }
    public static int countVowels(String str) {
        int count = 0;
        for (int i = 0; i < str.length(); i++) {
            char ch = str.charAt(i);
            if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' ||
                ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U') {
                count++;
            }
        }
        return count;
    }
}
```

File: FirstNonRepeatedChar.java

```
package StringExample;
public class FirstNonRepeatedChar {
    public static void main(String[] args) {
        String str = "ravikiran";
        char result = firstNonRepeatedCharacter(str);
        System.out.println(result);  // Output: w
    }
    public static char firstNonRepeatedCharacter(String str) {
        for (int i = 0; i < str.length(); i++) {
            char currentChar = str.charAt(i);
            if (str.indexOf(currentChar) == str.lastIndexOf(currentChar)) {
                return currentChar;
            }
        }
        return '\0';  // No non-repeated character
    }
}
```

File: GetCharacter.java

```
package StringExample;
public class GetCharacter {
 public static void main(String[] args) {
  String str = "Ravikiran Chavan";
  for (int i = 0; i < str.length() ; i++) {
   System.out.println(str.charAt(i));
  }

  System.out.println("Charector position 3 "+str.charAt(3));
  System.out.println("Charector position 7 "+str.charAt(7));
  System.out.println("Charector position10 "+str.charAt(10));
//  System.out.println("Charector position 16 "+str.charAt(16)); //Exception
StringIndexOutOfBoundsException
 }
}
```

File: Palindrome.java

```
package StringExample;
public class Palindrome {
    public static void main(String[] args) {
        String str = "Madam";
        boolean result = isPalindrome(str);
        System.out.println(result);  // Output: true
    }
    public static boolean isPalindrome(String str) {
        String reversed = new StringBuilder(str).reverse().toString();
        return str.equalsIgnoreCase(reversed);
    }
}
```

File: PerformanceTestSpring_Buffer_Builder.java

```
package StringExample;
public class PerformanceTestSpring_Buffer_Builder {
 public static void main(String[] args) {
  long startTime = System.currentTimeMillis();
  String str = new String("Java");
  for (int i = 0; i < 1000; i++) {
   str.concat("Technology");
  }
  System.out.println("Time taken by String: " + (System.currentTimeMillis() - startTime) + "ms");
  startTime = System.currentTimeMillis();
  StringBuffer sbuffer = new StringBuffer("Java");
  for (int i = 0; i < 1000; i++) {
   sbuffer.append("Technology");
  }
  System.out.println("Time taken by StringBuffer: " + (System.currentTimeMillis() - startTime) +
"ms");
  startTime = System.currentTimeMillis();
  StringBuilder sbuilder = new StringBuilder("Java");
  for (int i = 0; i < 1000; i++) {
   sbuilder.append("Technology");
  }
  System.out.println("Time taken by StringBuilder: " + (System.currentTimeMillis() - startTime) +
"ms");
 }
}
```

File: ReverseStringUsingBuiltInMethod.java

```
package StringExample;
public class ReverseStringUsingBuiltInMethod {
 public static void main(String[] args) {
  String name="Ravikiran";
  System.out.println("Given Original String -->"+name);

  StringBuffer sb = new StringBuffer(name);
  sb.reverse();
  System.out.println("Given String Is Reverse using Inbuilt Method-->"+sb);
 }
}
```

File: ReverseStringUsingManualMethod.java

```
package StringExample;
public class ReverseStringUsingManualMethod {
 public static void main(String[] args) {

  String name="Ravikiran";
  System.out.println("original String is "+name);
  reverseString(name);
 }
 private static void reverseString(String name) {
  String updString = "";
  for(int i = name.length()-1; i >= 0; i--) {
   updString+=name.charAt(i);
  }
  System.out.println("Reverse String "+updString);
 }
}
```

File: StringConcatExample.java

```
package StringExample;
public class StringConcatExample {
 public static void main(String[] args) {

  String a="Ravi";
  String b="Kiran";
  String c="Chavan";

  String operator=usingPlusOperator(a,b,c);
  System.out.println(operator);

  String concat=usingconcat(a,b,c);
  System.out.println(concat);

  String stringBuilder=usingStringBuilder(a,b,c);
  System.out.println(stringBuilder);

  String join=usingJoin(a,b,c);
  System.out.println(join);
 }
 //Using + Operator (Simple and Readable)
 private static String usingPlusOperator(String a, String b, String c) {
  return a+b+c;
 }
 //Using concat() Method
 private static String usingconcat(String a, String b, String c) {
  return a.concat(b).concat(c);
 }

 //Using StringBuilder (Best for Performance)
 private static String usingStringBuilder(String a, String b, String c) {
  StringBuilder sb = new StringBuilder();
  sb.append(a);
  sb.append(b);
  sb.append(c);
  return sb.toString();
 }

 //Using String.join() (Best for Joining Multiple Strings)
 private static String usingJoin(String a, String b, String c) {
  return String.join("",a, b, c);
 }

}
```

File: StringOperationsExample.java

```
package StringExample;
public class StringOperationsExample {
    public static void main(String[] args) {
     // 1. Creating Strings
        String str1 = "Hello";
        String str2 = new String("World");
        // 2. Concatenation
        String result = str1 + " " + str2;
        System.out.println("Concatenation: " + result);
        // Using concat() method
        System.out.println("Using concat(): " + str1.concat(" ").concat(str2));
        // 3. Getting Length
        System.out.println("Length of str1: " + str1.length());
        // 4. Character at Index
        System.out.println("Character at index 1 in str1: " + str1.charAt(1));
        // 5. Substring
        System.out.println("Substring (1,4) of str1: " + str1.substring(1, 4));
        // 6. Contains
        System.out.println("Does str1 contain 'lo'? " + str1.contains("lo"));
        // 7. Equals & Equals Ignore Case
        System.out.println("str1 equals 'hello': " + str1.equals("hello"));
        System.out.println("str1 equalsIgnoreCase 'hello': " + str1.equalsIgnoreCase("hello"));
        // 8. Convert to Upper and Lower Case
        System.out.println("Upper case: " + str1.toUpperCase());
        System.out.println("Lower case: " + str1.toLowerCase());
        // 9. Trim Spaces
        String str3 = "  Java Programming  ";
        System.out.println("Trimmed: '" + str3.trim() + "'");
        // 10. Replace Characters
        System.out.println("Replace 'l' with 'w' in str1: " + str1.replace("l", "w"));
        // 11. Split a String
        String str4 = "apple,banana,orange";
        String[] fruits = str4.split(",");
        System.out.println("Splitting str4:");
        for (String fruit : fruits) {
            System.out.println(fruit);
        }
        // 12. Convert String to Integer
        String numStr = "123";
        int num = Integer.parseInt(numStr);
        System.out.println("Integer + 1: " + (num + 1));
        // 13. Convert Integer to String
        int num2 = 456;
        String numStr2 = String.valueOf(num2);
        System.out.println("Integer converted to String: " + numStr2);
    }
}
```

File: SwapStringWithoutThirdVariable.java

```
package StringExample;
//Java Program to Swap two Strings without using Third Variable
public class SwapStringWithoutThirdVariable {
 public static void main(String[] args) {
    String s1 = "Love";
    String s2 = "You";
    System.out.println("Before swapping, s1 = " +s1+ ", s2 = "+s2 );
    withoutThirdVarible(s1,s2);

    withThirdVariable(s1,s2);
  }
 private static void withoutThirdVarible(String s1, String s2) {
  // Concatenate both the string s1 and s2 and store it in s1
     s1 = s1 + s2;
     // Extract s2 from updated string s1.
     s2 = s1.substring(0, (s1.length() - s2.length()));

  // Extract s1 from updated string s1
     s1 = s1.substring(s2.length());
     System.out.println("After swapping without thrid Variable, s1 = " +s1+ ", s2 = "+s2 );

 }

 private static void withThirdVariable(String s1, String s2) {
  String temp = s1;
  s1 = s2;
  s2 = temp;
  System.out.println("After swapping using thrid Variable, s1 = " +s1+ ", s2 = "+s2 );
 }
}
```

File: VowelsCount.java

```
package StringExample;
import java.util.Arrays;
import java.util.Map;
import java.util.stream.Collectors;
public class VowelsCount {
 public static void main(String[] args) {
  String str = "Find Out Vowels Count From String";

  int count = usingTraditionWay(str);
  System.out.println("Count Of Vowels using Traditional Way :"+count);

  Map<String,Integer> map= usingStream(str);
  System.out.println(map);
 }
 private static int usingTraditionWay(String str) {
  int count = 0;
  for(int i = 0; i < str.length()-1; i++){
   Character ch = str.charAt(i);
   if(ch == 'a' || ch == 'i' || ch == 'e' || ch == 'o' || ch == 'u' ||
      ch == 'A' || ch == 'I' || ch == 'E' || ch == 'O' || ch == 'U') {
    count ++;
   }
  }
  return count;
 }
 private static Map<String, Integer> usingStream(String str) {
   return str.chars()
               .mapToObj(c -> String.valueOf((char) c)) // Convert char to lowercase string
               .filter(ch -> "aeiouAEIOU".contains(ch))                    // Keep only vowels
               .collect(Collectors.groupingBy(                        // Group by vowel
                        ch -> ch,
                        Collectors.summingInt(ch -> 1)                // Count each vowel
               ));
 }
}
```