Java8 coding programs : <https://javaconceptoftheday.com/java-8-interview-sample-coding-questions/>

**Q) Given a list of integers, separate odd and even numbers?**

**Q) Input : 1, 6 , 3, 8 , 5, 7 , Sort by key , Sort by value when we give list of customer objects – key will be age and name will be value ?**

|  |
| --- |
| package com.javastreams.Jan12th;  import java.util.Arrays;  import java.util.Comparator;  import java.util.IntSummaryStatistics;  import java.util.LinkedHashMap;  import java.util.List;  import java.util.Map;  import java.util.Objects;  import java.util.Optional;  import java.util.function.BinaryOperator;  import java.util.function.Function;  import java.util.stream.Collector;  import java.util.stream.Collectors;  import com.java8.streams.collectors.Student;  public class InputKeyValueSort {  /\*\*  \* @param args  \*/  public static void main(String[] args) {  Customer s1 = new Customer(1, 2);  Customer s2 = new Customer(1, 2);  Customer s3 = new Customer(1, 2);  Customer s4 = new Customer(3, 4);  Customer s5 = new Customer(3, 4);  Customer s6 = new Customer(5, 7);    List<Customer> studentList = Arrays.asList(s1,s2,s3,s4,s5,s6);    BinaryOperator<Integer> reduce = (a, b) -> a+ b;    // studentList.stream()  // .filter(Objects::nonNull)  // .collect(Collectors.groupingBy(Customer::getId,  // Collectors.reducing((a,b) -> a.getAge() + b.getAge())));  Map<Integer, Integer> summingBy = studentList.stream()  .filter(Objects::nonNull)  .collect(Collectors.groupingBy(Customer::getId,  Collectors.summingInt(Customer::getAge)));    Map<Integer, Integer> sortBy = studentList.stream()  .filter(Objects::nonNull)  .collect(Collectors.groupingBy(Customer::getId,  Collectors.summingInt(Customer::getAge)));  //sortBy.entrySet().stream().sorted(Map.Entry::comparingKey);    Map sortBykeyIndes = sortBy.entrySet().stream()  .sorted(Map.Entry.comparingByKey(Comparator.reverseOrder()))  .collect(Collectors.toMap(Map.Entry::getKey, Map.Entry::getValue,  (oldValue, newValue) -> oldValue, LinkedHashMap::new));  System.out.println("sortBykeyIndes" + sortBykeyIndes);  Map sortByvalueIndes = sortBy.entrySet().stream()  .sorted(Map.Entry.comparingByValue(Comparator.reverseOrder()))  .collect(Collectors.toMap(Map.Entry::getKey, Map.Entry::getValue,  (oldValue, newValue) -> oldValue, LinkedHashMap::new));    System.out.println("sortByvalueIndes" +sortByvalueIndes);  }  }  class Customer {  private Integer id;  private Integer age;  public Integer getId() {  return id;  }  public void setId(Integer id) {  this.id = id;  }  public Integer getAge() {  return age;  }  public void setAge(Integer age) {  this.age = age;  }  public Customer(Integer id, Integer age) {  super();  this.id = id;  this.age = age;  }  } |

**15th Jan java8 stream method:**

5th Feb Java 8 Practice Questions:

**Q) Given a list of integers, find maximum and minimum of those numbers?**

|  |
| --- |
| package com.javastreams.satiee.feb5th;  import java.util.Arrays;  import java.util.Comparator;  import java.util.List;  import java.util.Optional;  public class MaximumAndMiniumNumberFromListUsingStreams {  public static void main(String[] args) {  List<Integer> listOfIntegers = Arrays.asList(45, 12, 56, 15, 24, 75, 31, 89);  // I am giving Natural sorting order - Asc  Optional<Integer> max = listOfIntegers.stream().max((i1, i2) -> i1 > i2 ? 1002 : i1 < i2 ? -27371 : 0);  System.out.println("Using natural sorting logic max " + max.get());  Optional<Integer> min = listOfIntegers.stream().min((i1, i2) -> i1 > i2 ? 828282 : i1 < i2 ? -18227 : 0);  System.out.println("Using natural sorting logic min " + min.get());  // I am giving Reverse sorting order - Desc  Optional<Integer> max1 = listOfIntegers.stream().max((i1, i2) -> i1 > i2 ? -1 : i1 < i2 ? 1 : 0);  System.out.println("Using Desc sorting logic max " + max1.get());  Optional<Integer> min1 = listOfIntegers.stream().min((i1, i2) -> i1 > i2 ? -1 : i1 < i2 ? 1 : 0);  System.out.println("Using Desc sorting logic min " + min1.get());  int max2 = listOfIntegers.stream().max(Comparator.naturalOrder()).get();  System.out.println("Maximum Element : " + max2);  int min2 = listOfIntegers.stream().min(Comparator.naturalOrder()).get();  System.out.println("Minimum Element : " + min2);  }  } |

**Q) How do you merge two unsorted arrays into single sorted array using Java 8 streams?**

|  |
| --- |
| package com.javastreams.satiee.feb5th;  import java.util.Arrays;  import java.util.stream.IntStream;  import java.util.stream.Stream;  public class MergeTwoArraysUsingJava8Streams {  public static void main(String[] args) {  int[] a = new int[] { 4, 2, 7, 1 };  int[] b = new int[] { 8, 3, 9, 5, 4, 1, 8, 2};    Object[] meredSortedArray = Stream.concat(Arrays.stream(a).boxed(), Arrays.stream(b).boxed())  .map(Integer::valueOf)  .sorted()  .distinct()  .toArray();    System.out.println("Merged sorted array: "+ Arrays.toString(meredSortedArray));      //Another way of doing it    int[] array = IntStream.concat(Arrays.stream(a), Arrays.stream(b)).sorted().toArray();    System.out.println("Merged sorted array: "+ Arrays.toString(array));    }  } |

**Q) How do you get three maximum numbers and three minimum numbers from the given list of integers?**

|  |
| --- |
| package com.javastreams.satiee.feb5th;  import java.util.Arrays;  import java.util.Comparator;  import java.util.List;  import java.util.stream.Collectors;  public class FirstThreemaxAndMinNumbersFromList {    public static void main(String[] args) {    List<Integer> list = Arrays.asList(1,5,3,7,4,2,6,10,9,8);  //First three minimum - [1,2,3]  //first three maximum - [8,9,10]    Integer max = list.stream().max(Comparator.naturalOrder()).get();  List<Integer> firstThreeMaximumNumbers = list.stream().sorted().skip(list.size() - 3).collect(Collectors.toList());  System.out.println("firstThreeMaximumNumbers" + firstThreeMaximumNumbers);    List<Integer> firstThreeMinumumNumbers = list.stream().sorted(Comparator.reverseOrder()).skip(list.size() - 3).collect(Collectors.toList());  System.out.println("firstThreeMinumumNumbers" + firstThreeMinumumNumbers);      List<Integer> firstThreeMaximumNumbersUsingLimit = list.stream().sorted().limit(3).collect(Collectors.toList());  System.out.println("firstThreeMaximumNumbersUsingLimit" + firstThreeMaximumNumbersUsingLimit);    List<Integer> firstThreeMinumumNumbersUsingLimit = list.stream().sorted(Comparator.reverseOrder()).limit(3).collect(Collectors.toList());  System.out.println("firstThreeMinumumNumbersUsingLimit" + firstThreeMinumumNumbersUsingLimit);        }  } |

**Q) Java 8 program to check if two strings are anagrams or not?**

|  |
| --- |
| package com.javastreams.satiee.feb17th;  import java.util.Arrays;  import java.util.stream.Collectors;  public class AngramDemo {  public static void main(String[] args) {    String s1 = "cat"  ;  String s2 = "act";    char[] charArray1 = s1.toCharArray();  char[] charArray2 = s2.toCharArray();    if(charArray1.length != charArray2.length) {  System.out.println("Both are differen");  return;  }    Arrays.sort(charArray1);  Arrays.sort(charArray2);    System.out.println("After sort arr1" + Arrays.toString(charArray1));  System.out.println("After sort arr2" + Arrays.toString(charArray2));    boolean equals = Arrays.equals(charArray1, charArray2);    System.out.println("Both arrays are equals " + equals);    String s1SortedString = Arrays.stream(s1.split("")).sorted().collect(Collectors.joining());    String s2SortedString = Arrays.stream(s2.split("")).sorted().collect(Collectors.joining());    System.err.println("Both are equal strings : " + s1SortedString.equals(s2SortedString));      }  } |

Given: List of Employees

|  |
| --- |
| List<Employee> list = Arrays.asList(  new Employee(101, "Ramesh","Bonalu", "User", 10000, "Male",51),  new Employee(102, "Prabhu", "Gunapalu","Admin", 20000,"Male", 29),  new Employee(108, "Bala", "Thopu", "User", 10000, "Male",31),  new Employee(111, "Rupesh", "Reddy", "Hardware", 13000, "Male",21),  new Employee(212, "Anirudh", "Mannuru" ,"Network", 18000,"Male", 20),  new Employee(213, "Uma", "Sirisella", "User", 50000, "Male", 31),  new Employee(213, "Devi", "Rani", "User", 50000, "Female", 26),  new Employee(213, "Divya", "Utasad", "User", 50000, "Female", 27),  new Employee(213, "Ashu", "Kongala", "User", 50000, "Female", 28)); |

1.Collect the first name of all employees?

|  |
| --- |
| String result1 = list.stream().map(Employee::getFirstName).collect(Collectors.joining(","));  String result2 = list.stream().map(Employee::getFirstName).collect(Collectors.joining());  String result3 = list.stream().map(Employee::getFirstName).collect(Collectors.joining("deleimter", "prefix", "suffix"));  System.out.println(result1);  System.out.println(result2);  System.out.println(result3); |

2. Partitioning the data

|  |
| --- |
| Map<Boolean, List<Employee>> partionResult = list.stream()  .collect(Collectors.partitioningBy(e -> e.getGender().equals("Male")));  System.out.println(partionResult);    Map<Boolean, Long> partionResult1 = list.stream()  .collect(Collectors.partitioningBy(e -> e.getGender().equals("Male"), Collectors.counting()));  System.out.println(partionResult1); |

3. grouping by

|  |
| --- |
| Map<String, List<Employee>> groupByDept = list.stream().collect(Collectors.groupingBy(Employee::getDeptName));  System.out.println(groupByDept);    Map<String, Long> groupByDept1 = list.stream().collect(Collectors.groupingBy(Employee::getDeptName, Collectors.counting()));  System.out.println(groupByDept1);    Map<String, Long> groupByDept2 = list.stream().collect(Collectors.groupingBy(Employee::getDeptName, LinkedHashMap::new, Collectors.counting()));  System.out.println(groupByDept2); |

4.List the employee’s names by their department

|  |
| --- |
| Map<String, List<String>> grupbyDepartandListNames = list.stream().collect(Collectors.groupingBy(Employee::getDeptName,  Collectors.mapping(Employee::getFirstName, Collectors.toList())));  System.out.println(grupbyDepartandListNames); |

5. Group the employees by their department - sum the slaries of emplyess by each department

|  |
| --- |
| Map<String, Integer> groupByDeaprtSalrySum= list.stream().collect(Collectors.groupingBy(Employee::getDeptName,  Collectors.mapping(Employee::getSalay, Collectors.reducing(0, (a,b) -> a+b))));  System.out.println(groupByDeaprtSalrySum); |

6. Group the employees by their department - sum the slaries of emplyess by each department – summingDouble

|  |
| --- |
| Map<String, Double> groupByDeaprtSalrySumDouble= list.stream().collect(Collectors.groupingBy(Employee::getDeptName,  Collectors.summingDouble(Employee::getSalay)));  System.out.println(groupByDeaprtSalrySumDouble); |

7. Group the emplyess by thier department - sumamry statistics of the salries of employess by each department

|  |
| --- |
| Map<String, DoubleSummaryStatistics> collect = list.stream().collect(Collectors.groupingBy(Employee::getDeptName,  Collectors.summarizingDouble(e ->e.getSalay()))); |

8. summary statistics of employees in "User" department only

|  |
| --- |
| DoubleSummaryStatistics collect2 = list.stream().collect(Collectors.filtering(e -> e.getDeptName().equals("User"), Collectors.summarizingDouble(Employee::getSalay)));  System.out.println(collect2); |

9. Employee with heighest age

Q. **Find sum of all digits of a number in Java 8?**

|  |
| --- |
| package com.java8.operators.unaryoperator;  import java.util.Arrays;  import java.util.stream.Collectors;  public class SumOfAllDigitsOnGivenNumber {    public static void main(String[] args) {    int num = 1238172;    Integer sumOfAllDigits = Arrays.stream(String.valueOf(num).split(""))  .map(Integer::valueOf).reduce((a,b) -> a+b).get();  System.out.println("sumOfAllDigits : "+ sumOfAllDigits);    Integer sumOfDigits = Arrays.stream(String.valueOf(num).split(""))  .collect(Collectors.summingInt(Integer::parseInt));  System.out.println("sumOfDigits: "+ sumOfDigits);    }  } |

Q. **Find second largest number in an integer array?**

|  |
| --- |
| package com.java8.secondHeighestValue;  import java.util.Arrays;  import java.util.Comparator;  import java.util.List;  public class SecondHeighestValueUsingJava8 {  public static void main(String[] args) {    List<Integer> listOfIntegers = Arrays.asList(45, 12, 56, 15, 24, 75, 31, 89);    Integer secondHeighestValue = listOfIntegers.stream().sorted(Comparator.reverseOrder())  .skip(1).findFirst().get();  System.out.println("secondHeighestValue: " + secondHeighestValue);  }  } |

Q. **Given a list of strings, sort them according to increasing order of their length?**

|  |
| --- |
| List<String> listOfStrings = Arrays.asList("Java", "Python", "C#", "HTML", "Kotlin", "C++", "COBOL", "C");  Comparator<String> stringlegthComparator = (s1, s2) -> s1.length() > s2.length() ? 1 : s1.length() < s2.length() ? -1: 0;  List<String> sortedStrings = listOfStrings.stream().sorted(stringlegthComparator).collect(Collectors.toList());  System.out.println("sorted Strings: " + sortedStrings);    List<String> sortedstringBasedOnLength = listOfStrings.stream().sorted(Comparator.comparing(String::length)).collect(Collectors.toList());    System.out.println("sortedstringBasedOnLength: "+ sortedstringBasedOnLength); |

Q. **How do you find common elements between two arrays?**

**Q. Reverse each word of a string using Java 8 streams?**

**Q.** **How do you find sum of first 10 natural numbers?**