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

|  |
| --- |
| 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:**