## 2nd smallest element of an unsorted array.

|  |
| --- |
| **public** **class** second\_smallest\_elementArray {  **public** **static** **void** main(String[] args) {    **int** []arr = {12, 23,12,32,43,1,2,2,10};    **int** len = arr.length;  **int** temp;  **for**(**int** i =0 ; i<len;i++) {  **for**(**int** j=i; j<len;j++) {    **if**(arr[i]>arr[j]) {  temp=arr[i];  arr[i]=arr[j];  arr[j]=temp;  }    }      }  **for**(**int** c: arr) {  System.***out***.println(c);  }    System.***out***.println("second smallest element in an array is "+ arr[1]);    }  } |

Output:

second smallest element in an array is 2

## 2nd smallest element of a sorted rotated array. E.g. 5,6,1,2,3,4

|  |
| --- |
| **public** **class** second\_smallest\_sortRotatedArray {  **public** **static** **void** main(String[] args) {      **int** [] arr = {3,4,5,6,1,2};  **int** i=0, j=1;  **int** len = arr.length;    **while** (i<len&&j<len) {    **if**(arr[i]<arr[j])  j++;  **else** **if**(arr[i]>=arr[j]) {  i=j;  j++;  }    }  System.***out***.println(arr[i+1]);  }  } |

Output: 2

3) Set of anagrams of list of words. The output should be words which are anagrams from the list should be grouped together. cat mat god act atm dog tac tam ogd

|  |
| --- |
| **public** **class** sortGroupofAnagrams {      **public** **static** List<List<String>> groupAnagrams(String[] strs) {  List<List<String>> result = **new** ArrayList<List<String>>();  HashMap<String, ArrayList<String>> map = **new** HashMap<String, ArrayList<String>>();  **for**(String str: strs){  **char**[] arr = **new** **char**[26];  **for**(**int** i=0; i<str.length(); i++){  arr[str.charAt(i)-'a']++;    }  String ns = **new** String(arr);  **if**(map.containsKey(ns)){  map.get(ns).add(str);  }**else**{  ArrayList<String> al = **new** ArrayList<String>();  al.add(str);  map.put(ns, al);  }  }  result.addAll(map.values());  **return** result;  }  **public** **static** **void** main(String[] args) {    String sen10s = "cat mat god act atm dog tac tam ogd";      //List<Set<Character>> matches = new ArrayList<>();  String [] words = sen10s.split(" ");  System.***out***.println(sortGroupofAnagrams.*groupAnagrams*(words));  // for(int i = 0; i<words.length;i++) {  //  // char[] chars = words[i].toCharArray();  //  // Set<Character> group = new HashSet<>();  // for(char d: chars) {  // group.add(d);  // }  //  // }  // System.out.println(matches);      }  } |

1. Minimum distance between 2 words counting the number of characters from middle of both words. E.g. ABC is XYZ and ABC & XYZ are two distinct words. Minimum distance between ABC and XYZ

|  |
| --- |
|  |

1. First unique character of a string. E.g. aabdcce . Output : d

|  |
| --- |
| 1. **import** java.util.LinkedHashMap; 2. **import** java.util.Map; 3. **public** **class** firstUniqCharinString {  6. **public** **static** Character firstUniqChar(String s) { 7. **if** (s.length() == 0) { 8. System.***out***.println("Fail"); 9. **return** 0; 10. } **else** { 11. Map<Character, Integer> m = **new** LinkedHashMap<Character, Integer>(); 12. **for** (**int** i = 0; i < s.length(); i++) { 13. **if** (m.containsKey(s.charAt(i))) { 14. m.put(s.charAt(i), m.get(s.charAt(i)) + 1); 15. } **else** { 16. m.put(s.charAt(i), 1); 17. } 18. } 19. **for** (Map.Entry<Character, Integer> hm : m.entrySet()) { 20. **if** (hm.getValue() == 1) { 21. **return** hm.getKey(); 22. } 23. } 24. } 25. **return** 0; 26. } 27. **public** **static** **void** main(String args[]) { 29. String test = "blabllablasdad"; 31. System.***out***.println(firstUniqCharinString.*firstUniqChar*(test));  34. } 35. } |

6) Largest substring with unique characters e.g. aaabcbdeaf Output : cbdeaf

|  |
| --- |
| **public** **class** longestUniqchars {    **public** **static** **void** main(String[] args) {  String s = "aaabdasddlkjhgtyyhhjjaa";  // System.out.println(lengthOfLongestSubstring(s));  **int** uni = 0;  String largest = "";  Map<Character, Integer> map = **new** LinkedHashMap<>();  **int** len = s.length();  **for** (**int** i = 0; i < len; i++) {  **char** c = s.charAt(i);  **if** (!(map.containsKey(c))) {  map.put(c, i);  **if** (map.size() > uni) {  uni = map.size();  largest = map.keySet().toString();  }  } **else** {  i = map.get(c);  map.clear();  }  }  System.***out***.println("longest substring is "+ largest);  }  } |

16) Given Input,print the following input: “SSSSSTTPPQ” Output: “5S2T2P1Q”.

|  |
| --- |
| **package** sapient\_scrn;  **public** **class** countOfChars {      //imperative approach    **public** **static** **int** count(String s, **char** c) {  **int** count =0;  **for**(**int** i=0; i<s.length(); i++) {  **if**(s.charAt(i)==c) {  count++;  }  }  **return** count;  }  //////  **public** **static** **void** main(String[] args) {    String s = "SSSSSTTPPQ";      //imperative method call    String newWord = "";  **for**(**int** i=0; i<s.length(); i++) {    **if**(newWord.indexOf(s.charAt(i))<0) {  newWord=newWord+s.charAt(i)+countOfChars.*count*(s, s.charAt(i));  }  }  System.***out***.println(newWord);    ////////////////---- 2 nd approach  //using Java 8 approach  StringBuilder sb= **new** StringBuilder();    **for** (**char** c: s.toCharArray()) {    **if**(sb.toString().indexOf(c)<0) {  **long** count = s.chars().filter(ch -> ch == c).count();    sb=sb.append(c).append(count);  }  }  System.***out***.println(sb);            }  } |

Output: S5T2P2Q1

S5T2P2Q1

14) Missing characters to make a string Pangram.

Pangram is a sentence containing every letter in the English alphabet. Given a string, find all characters that are missing from the string, i.e., the characters that can make string a Pangram. We need to print output in alphabetic order.

Input : welcome to geeksforgeeks

Output : abdhijnpquvxyz

Input : The quick brown fox jumps

Output : adglvyz

|  |
| --- |
| **package** sapient\_scrn;  **import** java.util.ArrayList;  **public** **class** Pangram {        **public** **static** ArrayList<String> missingchar(String s) {    s= s.toLowerCase();  ArrayList<String> Al = **new** ArrayList<>();  **for**(**int** i=97; i<=122; i++){    **if**(!s.contains(Character.*toString*((**char**)i))) {    System.***out***.print(Character.*toString*((**char**)i));    Al.add(Character.*toString*((**char**)i));  }    }  **return** Al;  }  **public** **static** **void** main(String[] args) {      String s = "welcome to geeksforgeeks";    System.***out***.println(Pangram.*missingchar*(s));    s= s.toLowerCase();        **for**(**int** i=97; i<=122; i++){    **if**(!s.contains(Character.*toString*((**char**)i))) {    System.***out***.print(Character.*toString*((**char**)i));  }    }    }  } |