To find duplicates without casesensitive

String ar[]=new String[]{"hello","hi","hello","Hi"};  
  
System.*out*.println(Arrays.*stream*(ar)  
 .collect(Collectors.*groupingBy*(s->s,Collectors.*counting*())));

To find duplicates with casesensitive

System.*out*.println(Arrays.*stream*(ar)  
 .collect(Collectors.*groupingBy*(String::toLowerCase,Collectors.*counting*())));

Arrays.sort(s);//sorts based on lexographical way

Arrays.sort(s,Comparator.reverseOrder()); //sorts based on lexographical way in reverse order

Arrays.sort(s,Comparator.comparing(String::length)); //sorts based length in ascending order

Arrays.sort(s,Comparator.comparing(String::length).reversed());//sorts based length in descending order

All above things will also apply to list List.sort(…

Comparator<Employee> c=Comparator.*comparing*(Employee::getName);

arrayList.sort(c);

Array -> String

If it is string array then

String.join(“”,array);

If it is int array then

**int** ar[]= {1,2,3,4,5};

String s=Arrays.*toString*(ar).replaceAll("[,\\s]", "");

List -> String

String.join(“”,list);

Array -> List

Arrays.asList(ar);

**String -> Arrays**

str.tocharArray();

String s=”Vikas”;

s.indexOf(“john”); // returns -1 as it is not present in string

s.substring(0,3); //returns Vi 0 is inclusive and 3 is exclusive

* replaceAll() and replaceFirst() uses regex to replace and replace() doesn’t use regex
* String.split(); //uses regex to split

//String st1="geeks";

* //System.out.println(st1.contains("g"));
* Pattern.
* Why String is immutable?

**Why not string concatenation or + ?**

**String** myString = "The " + "quick " + "brown " + "fox...";

**the + symbol translates to chains of StringBuilder.append() calls.** Due to this, **mixing the StringBuilder and + method** of concatenation is **considered bad practice**.

Additionally, String concatenation using the + operator within a loop should be avoided. Since the String object is immutable, each call for concatenation will result in a new String object being created.

When you use for loop of many iterations it creates more issue

Use String builder which appends all string in one object itself

**Which is best method to convert positive or negative int to String?**

int number=**537**;

String result = String.valueOf(number); //Method 1

String result = Integer.toString(number);//Method 2

System.out.println(number + "" );//Method 3

Method 2 (is best refer below page)

Method 1

public static String valueOf(Object obj) {

**return** (obj == null) ? "null" : obj.toString();

}

Method 3

String both = new StringBuilder().append(ab).append(cd).toString();

In Method 1 valueOf() internally calling toString() method , to convert integer to string. So best option to use is Method 2

**Ascii value 65 to 90 A to Z and 97 to 122 a to z (by str.codepointAt(a) will also get ascii value)**

**Streaming through list & map**

List<String> list= Arrays.asList(“hi ”,”joy ”,”bhanu”);

String s=list.stream().collect(Collectors.joining(“”));

**If list of characters is present then**

String s=list.stream().map(c->c+””). collect(Collectors.joining(“”));

If array of String

m is map

m.keySet().removeAll(m.entrySet().stream().filter(v->v.getValue()==1).map(e->e.getKey()).collect(Collectors.toList())); //collects in list

String s=mp.entrySet().stream().map(m->m.getValue().toString()).collect(Collectors.joining()); //collect values from map