//Binary format using Array

public static void toBinaryArray(int decimal) {

int[] binary = new int[25];

int index = 0;

while (decimal != 0) {

binary[index++] = decimal % 2;

decimal = decimal / 2;

}

for (int i = index - 1; i >= 0; i--) {

System.out.print(binary[i]);

}

}

public static void toBinary(int decimal) {

int binaryResult = 0;

while (decimal != 0) {

binaryResult = binaryResult \* 10 + decimal % 2;

decimal = decimal / 2;

}

public static boolean isAnagram(String first, String second)

{

// remove all whitespaces and convert strings to lowercase

first = first.replaceAll("\\s", "").toLowerCase();

second = second.replaceAll("\\s", "").toLowerCase();

/\* check whether string lengths are equal or not,

if unequal then not anagram \*/

if (first.length() != second.length())

return false;

// convert string to char array

char[] firstArray = first.toCharArray();

char[] secondArray = second.toCharArray();

// sort both the arrays

Arrays.sort(firstArray);

Arrays.sort(secondArray);

// checking whether both strings are equal or not

return Arrays.equals(firstArray,secondArray);

}

Or

public static boolean isAnagram(String first, String second)

{

// remove all whitespaces and convert strings to lowercase

first = first.replaceAll("\\s", "").toLowerCase();

second = second.replaceAll("\\s", "").toLowerCase();

/\* check whether string lengths are equal or not,

if unequal then not anagram \*/

if (first.length() != second.length())

return false;

// convert first string to char array

char[] firstArray = first.toCharArray();

// check whether each character of firstArray is present in second string

for (char c : firstArray)

{

int index = second.indexOf(c);

// indexOf function returns -1 if the character is not found

if (index == -1)

return false;

// if character is present in second string, remove that character from second string

second = second.substring(0,index) + second.substring(index+1, second.length());

}

return true;

}

//find maximum

public static void max(int[] input) {

// TODO Auto-generated method stub

int max=input[0];

for(int i=1;i<input.length;i++) {

if(input[i]>max)

max=input[i];

}

System.out.println(max);

}

//SingleTon

public class JavaHungrySingleton

{

private static volatile JavaHungrySingleton uniqueInstance;

private JavaHungrySingleton(){}

public static JavaHungrySingleton getInstance()

{

if (uniqueInstance ==null )

{ synchronized(JavaHungrySingleton.class)

{

if (uniqueInstance ==null )

{

uniqueInstance=new JavaHungrySingleton();

}

}

}

return uniqueInstance ;

}

//Find Duplicate in Array

private static void findDuplicatesUsingHashSet(int[] inputArray)

{

    HashSet<Integer> set = new HashSet<Integer>();

    for (int element : inputArray)

    {

        if( ! set.add(element))

        {

            System.out.println("Duplicate Element : "+element);

        }

     How To Find Duplicates In Array Using Java 8 Streams?

private static void findDuplicatesUsingJava8(int[] inputArray)

{

    Set<Integer> uniqueElements = new HashSet<>();

    Set<Integer> duplicateElements =  Arrays.stream(inputArray)

                                            .filter(i -> !uniqueElements.add(i))

                                            .boxed()

                                            .collect(Collectors.toSet());

    System.out.println(duplicateElements);

}

private static void findDuplicatesUsingHashMap(int[] inputArray)

{

HashMap<Integer, Integer> map = new HashMap<>();

for (int element : inputArray)

{

if(map.get(element) == null)

{

map.put(element, 1);

}

else

{

map.put(element, map.get(element)+1);

}

}

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

for (Entry<Integer, Integer> entry : entrySet)

{

if(entry.getValue() > 1)

{

System.out.println("Duplicate Element : "+entry.getKey()+" - found "+entry.getValue()+" times.");

}

}

}