Find duplicate elements from an array and print.

**public** **class** Javapro1

{

**public** **static** **void** main(String[] args) {

**int**[] arr = { 12, 19, 34, 12, 26, 34, 99, 81, 56, 64, 19, 54 };

*findDuplicate*(arr);

}

**private** **static** **void** findDuplicate(**int**[] arr) {

Set<Integer> uniqueSet = **new** HashSet<Integer>();

Set<Integer> duplicateSet = **new** HashSet<Integer>();

**for** (**int** n : arr) {

**if** (!uniqueSet.add(n)) {

duplicateSet.add(n);

}

}

System.***out***.println(duplicateSet);

}

}

Write a program that finds the highest/lowest difference obtained by subtraction of each combination of two values in an array.

**public** **class** JavaPro3 {

**public** **static** **void** main(String[] args) {

**int**[] arr = { 2, 5, 7, 12, 1, 3, 4 };

*findMinMaxDiff*(arr);

}

**private** **static** **void** findMinMaxDiff(**int**[] arr) {

**if** (arr.length <= 2) {

System.***out***.println("Only two elements in the array");

**return**;

}

Arrays.*sort*(arr);

**int** maxDiff = Integer.***MIN\_VALUE***;

**int** minDiff = Integer.***MAX\_VALUE***;

**for** (**int** i = 0; i < arr.length - 1; i++) {

**for** (**int** j = i + 1; j < arr.length; j++) {

**int** pro = arr[j] - arr[i];

**if** (pro < minDiff) {

minDiff = pro;

}

**if** (pro > maxDiff) {

maxDiff = pro;

}

}

}

System.***out***.println("Minimum product value:  " + minDiff);

System.***out***.println("Maximum product value:  " + maxDiff);

}

}

Write a program to find the longest substring from a given string that doesn’t contain any duplicate characters.

**public** **class** Javapro5 {

**public** **static** **void** main(String[] args) {

String str = "aabbcabd";

String ans = *findLongStringwithoutDuplicate*(str);

System.***out***.println(ans);

}

**private** **static** String findLongStringwithoutDuplicate(String str) {

**if** (str.length() == 0) {

**return** **null**;

}

**int** maxLength = 0;

String longestSubstring = "";

**int** start = 0;

Map<Character, Integer> charIndexMap = **new** LinkedHashMap<>();

**for** (**int** i = 0; i < str.length(); i++) {

**char** ch = str.charAt(i);

**if** (charIndexMap.containsKey(ch)) {

start = Math.*max*(start, charIndexMap.get(ch) + 1);

}

charIndexMap.put(ch, i);

**if** (charIndexMap.size() > maxLength) {

maxLength = charIndexMap.size();

longestSubstring = str.substring(start, i + 1);

}

}

**return** longestSubstring;

}

}