Program:1

import java.util.HashSet;

import java.util.Set;

//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 product obtained by multiplying each combination of two values in an array.

**public** **class** Javapro2 {

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

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

*findMinMaxProduct*(arr);

}

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

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

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

**return**;

}

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

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

**int** pro = 0;

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

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

pro = arr[i] \* arr[j];

**if** (pro < minProduct) {

minProduct = pro;

}

**if** (pro > maxProduct) {

maxProduct = pro;

}

}

}

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

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

}

}

Program:3

**import** java.util.Arrays;

**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);

}

}

Program 4:

**import** java.util.Arrays;

**public** **class** JavaPro4 {

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

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

*findMinMaxSum*(arr);

}

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

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

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

**return**;

}

Arrays.*sort*(arr);

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

**int** minSum = 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 < minSum) {

minSum = pro;

}

**if** (pro > maxSum) {

maxSum = pro;

}

}

}

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

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

}

}

Program 5:

**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;

}

}

Program 6:

/Find maximum and minimum elements in an array.

**public** **class** JavaPro6 {

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

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

*minMaxElement*(arr);

}

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

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

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

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

**if** (min > arr[i]) {

min = arr[i];

}

**if** (max < arr[i]) {

max = arr[i];

}

}

System.***out***.println("minimum Element in Array:  " + min);

System.***out***.println("maximum Element in Array:  " + max);

}

}

Program 7:

//Write a program to find if a number is Palindrome or not.

**public** **class** JavaPro7 {

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

**int** n = 655;

**boolean** isPalindrome = *palindromeOrNot*(n);

System.***out***.println(n + " is a Palindrome: " + isPalindrome);

}

**private** **static** **boolean** palindromeOrNot(**int** n) {

**if** (n < 0) {

**return** **false**;

}

**int** reverseNum = 0;

**int** num = n;

**while** (n != 0) {

**int** digit = n % 10;

reverseNum = reverseNum \* 10 + digit;

n = n / 10;

}

**if** (num == reverseNum) {

**return** **true**;

}

**return** **false**;

}

}

Program 8:

**public** **class** JavaPro8 {

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

String str = "javava";

**boolean** ans = *isPalindrome*(str);

System.***out***.println(str + " is palindrome: " + ans);

}

**private** **static** **boolean** isPalindrome(String str) {

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

**return** **false**;

}

**int** left = 0;

**int** right = str.length() - 1;

**while** (left < right) {

**if** (str.charAt(left) != str.charAt(right)) {

**return** **false**;

}

left++;

right--;

}

**return** **true**;

}

}

Program 9:

//Write a program to find the count of occurrence of each character in a string.

**public** **class** JavaPro9 {

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

String str = "javaprogram";

*charOccurance*(str);

}

**private** **static** **void** charOccurance(String str) {

Map<Character, Integer> ans = **new** HashMap<Character, Integer>();

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

**if** (ans.containsKey(str.charAt(i))) {

ans.put(str.charAt(i), ans.get(str.charAt(i)) + 1);

} **else** {

ans.put(str.charAt(i), 1);

}

}

**for** (Entry<Character, Integer> entry : ans.entrySet()) {

System.***out***.println(entry.getKey() + " occurance count " + entry.getValue());

}

}

}

Program 10:

import java.util.HashMap;

import java.util.Map;

import java.util.Map.Entry;

public class JavaProgram10 {

public static void main(String[] args) {

String str = "hello java program hello world program";

String[] s = str.split(" ");

strOccuranceCount(s);

}

private static void strOccuranceCount(String[] s) {

Map<String, Integer> ans = new HashMap<String, Integer>();

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

if (ans.containsKey(s[i])) {

ans.put(s[i], ans.get(s[i]) + 1);

} else {

ans.put(s[i], 1);

}

}

for (Entry<String, Integer> entry : ans.entrySet()) {

System.out.println(entry.getKey() + " occurance count " + entry.getValue());

}

}

}

Program 11:

**public** **class** JavaPro11 {

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

**int**[] n = { 1, 4, 10, 7, 2 };

*highestMultiplicationValue*(n);

}

**private** **static** **void** highestMultiplicationValue(**int**[] n) {

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

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

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

**int** value = n[i] \* n[j];

**if** (highestmultiValue < value) {

highestmultiValue = value;

}

}

}

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

}

}

Program 12:

**public** **class** StringCompression {

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

String str = "aabaaaccdddd";

*compressString*(str);

}

**private** **static** **void** compressString(String str) {

StringBuilder ans = **new** StringBuilder();

**int** count = 0;

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

count++;

**if** (i + 1 >= str.length() || str.charAt(i) != str.charAt(i + 1)) {

ans.append(str.charAt(i));

ans.append(count);

count = 0;

}

}

System.***out***.println("Compressed String: " + ans);

}

}