Java Program to Concatenate Two List

// using addAll() method

import java.io.\*;

import java.util.ArrayList;

import java.util.List;

public class GFG {

public static void main(String[] args)

{

// given list 1

List<Integer> list1 = new ArrayList<Integer>();

list1.add(1);

list1.add(2);

list1.add(3);

list1.add(4);

// given list 2

List<Integer> list2 = new ArrayList<Integer>();

list2.add(5);

list2.add(6);

list2.add(7);

list2.add(8);

// creating new empty list

List<Integer> concatenated\_list = new ArrayList<Integer>();

// using addAll( ) method to concatenate the lists

concatenated\_list.addAll(list1);

concatenated\_list.addAll(list2);

System.out.println("list1: " + list1);

System.out.println("list2: " + list2);

System.out.println("Concatenated list: "+ concatenated\_list);

}

}

**Write a singleton class.**

Description:

Singleton class means you can create **only one object** for the given class. You can create a singleton class by making its **constructor as private,** so that you can restrict the creation of the object. Provide a **static method** to get instance of the object, wherein you can handle the object creation inside the class only. In this example we are creating object by using static block.

public class MySingleton {

private static MySingleton myObj;

static{

myObj = new MySingleton();

}

private MySingleton(){

}

public static MySingleton getInstance(){

return myObj;

}

public void testMe(){

System.out.println("Hey.... it is working!!!");

}

public static void main(String a[]){

MySingleton ms = getInstance();

ms.testMe();

}

}

**Sorting of an Array**

**class** MySingleton {

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

**int**[] a = { -12, 223, 34, 3124, 45, 45, 653 };

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

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

**int** temp = 0;

**if** (a[i] > a[j]) {

temp = a[i];

a[i] = a[j];

a[j] = temp;

}

}

System.*out*.println(a[i]);

}

}

}

**Find duplicate elements in an array**

**import** java.util.\*;

**class** ZZZ {

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

**int**[] array = { 1, 1, 2, 3, 4, 5, 6, 7, 8, 8 };

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

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

**if** (set.add(array[i]) == **false**) {

System.*out*.println("Duplicate element found:" + array[i]);

}

}

}

}

Duplicate element found:1

Duplicate element found:8

**Method Two**

**import** java.io.\*;

**class** MySingleTon {

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

**int** a[] = { 12, 12, 23, 34, 56, 34 };

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

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

**if** (a[i] == a[j]) {

System.*out*.println(a[i]);

}

}

}

}

}

**12**

**34**

**Print the Largest Number in an Array Using Java**

**class** Manager {

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

**int**[] number = { 12, 23, 45, 67, 78, 43, 12 };

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

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

**if** (number[i] > maximum) {

maximum = number[i];

}

}

System.*out*.println(maximum);

}

}

# [Find Smallest Number in an Array Using Java](http://www.quickprogrammingtips.com/java/find-smallest-number-in-an-array-using-java.html)

**class** Manager {

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

**int**[] number = { 12, 23, 45, 67, 78, 43, 12 };

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

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

**if** (smallest > number[i]) {

smallest = number[i];

}

}

System.*out*.println(smallest);

}

}

**Write a program to find top two maximum numbers in a array.**

**public** **class** MaxTwo {

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

**int** num[] = { 5, 34, 78, 2, 2, 45, 1, 99, 23 };

**int** maxOne = 0;

**int** maxTwo = 0;

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

**if** (maxOne < n) {

maxTwo = maxOne;

maxOne = n;

} **else** **if** (maxTwo < n) {

maxTwo = n;

}

}

System.***out***.println("First Max Number: " + maxOne);

System.***out***.println("Second Max Number: " + maxTwo);

}

}

**Method Two**

**import** java.util.\*;

**class** Linked {

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

**int** b[] = { 1, 2, 31, 22, 12, 12 };

Arrays.*sort*(b);

System.*out*.println(b[b.length - 1]);

System.*out*.println(b[b.length - 2]);

}

}

**Write a Program to find all Pairs in Array of Integers whose Sum is equal to a given Number**

**import** java.util.\*;

**class** Linked {

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

**int**[] a = { 2, 45, 7, 3, 5, 1, 8, 9 };

*printSumPairs*(a, 10);

}

**public** **static** **void** printSumPairs(**int**[] input, **int** k) {

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

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

**if** (pairs.containsKey(input[i]))

System.*out*.println(input[i] + ", " + pairs.get(input[i]));

**else**

pairs.put(k - input[i], input[i]);

}

}

}

**Find all substrings of a String in java.  
class** Linked {

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

String str = "abbc";

System.*out*.println("All substring of abbc are:");

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

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

System.*out*.println(str.substring(i, j));

}

}

}

}

**Read a File from a Drive.**

**import** java.io.\*;

**class** MySingleton {

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

String str = **null**;

**try** {

BufferedReader br = **new** BufferedReader(**new** FileReader("E:\\A.txt"));

**if** ((str = br.readLine()) != **null**) {

System.*out*.println(str);

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

**How to Read a File from a Drive.**

**import** java.io.\*;

**class** MySingleTon {

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

**try** {

FileInputStream fis = **new** FileInputStream("F:\\A.txt");

InputStreamReader isr = **new** InputStreamReader(fis);

BufferedReader br = **new** BufferedReader(isr);

**while** (**true**) {

String id = br.readLine();

**if** (id == **null**)

**break**;

System.*out*.println(id);

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

**Write the data of One File to another File.**

**import** java.io.\*;

**class** MySingleTon {

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

**try** {

FileInputStream fis = **new** FileInputStream("F:\\A.txt");

FileOutputStream fos = **new** FileOutputStream("F:\\B.txt");

**while** (**true**) {

**int** asc = fis.read();

**if** (asc == -1)

**break**;

fos.write(asc);

}

System.*out*.println("complete");

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

**Write from One File to another File.**

**import** java.io.\*;

**class** MySingleTon {

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

**try** {

FileReader fr = **new** FileReader("F:\\A.txt");

BufferedReader br = **new** BufferedReader(fr);

FileWriter fw = **new** FileWriter("F:\\B.txt");

BufferedWriter bwr = **new** BufferedWriter(fw);

**while** (**true**) {

String st = br.readLine();

**if** (st == **null**)

**break**;

bwr.write(st);

bwr.newLine();

}

bwr.close();

System.*out*.println("Write");

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

# [Using java to count occurrences of letters in string [duplicate]](http://stackoverflow.com/questions/15010706/using-java-to-count-occurrences-of-letters-in-string)

**import** java.util.\*;

**public** **class** PrgmPrac {

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

String str = "We are Better";

Map<Character, Integer> map = **new** TreeMap<Character, Integer>();

**for** (**char** c : str.toCharArray()) {

**if** (map.get(c) == **null**) {

map.put(c, 1);

} **else** {

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

}

}

System.*out*.println(map);

}

}

{ =2, B=1, W=1, a=1, e=4, r=2, t=2}

**How to find duplicate words in a string in java**

**import** java.util.\*;

**class** Faltu {

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

String str = "I am the Best am the Best";

List lst = Arrays.*asList*(str.split(" "));

Set<String> set = **new** HashSet<String>(lst);

**for** (String st : set) {

System.*out*.println(st + "\t\t" + Collections.*frequency*(lst, st));

}

}

}

Best 2

am 2

the 2

I 1

**How to Split a String (Without In-Built Method).**

**import** java.util.\*;

**public** **class** Lab2 {

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

String str="My name is Rehan Khan";

String s="";

List<String> list=**new** ArrayList<String>();

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

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

**if**(ch!=' '){

s=s+ch;

}**else**{

list.add(s);

s="";

}

}

**for**(String s1:list)

System.*out*.println(s1);

}

}

[**Remove duplicate chars from a string - Java**](http://myvedham.blogspot.com/2013/03/remove-duplicate-chars-from-string-java.html)

**import** java.util.\*;

**class** SortTest {

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

String str = "aassddrrqqwwre";

**char**[] ch = str.toCharArray();

Set<Character> st = **new** LinkedHashSet<Character>();

**for** (**char** c : ch) {

st.add(c);

}

StringBuilder sb = **new** StringBuilder();

**for** (Character c1 : st) {

sb.append(c1);

}

System.*out*.println(sb.toString());

}

}

Output

asdrqwe

**Program to reverse a string**

**import** java.util.\*;

**class** SortTest {

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

System.*out*.println("Enter a String ");

Scanner sc = **new** Scanner(System.*in*);

String org = sc.next();

String rev = "";

**int** len = org.length();

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

rev = rev + org.charAt(i);

}

System.*out*.println(rev);

}

}

**Reverse a Number**

**class** MySingleTon {

**public** **int** revNumber(**int** number) {

**int** reverse = 0;

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

reverse = (reverse \* 10) + (number % 10);

number = number / 10;

}

**return** reverse;

}

public static void main(String[] args) {

MySingleTon m1 = **new** MySingleTon();

System.*out*.println(m1.revNumber(123456));

}

}

**Write a program to display the multiples of 7 between 1and 100.**

**Add total no of elements.**

class Test {

public static void main(String args[]) {

int c = 0;

int number = 7;

int count = 99;

while (number <= count) {

System.*out*.println(number);

number = number + 7;

c++;

}

System.*out*.println(c);

// int count = 0;

// for (int i = 1; i <= 100; i++) {

// if (i % 7 == 0) {

// System.out.println(i);

// count++;

// }

// }

// System.out.println("Total Elements : " + count);

}

}

**Write a program that prints the numbers from 1 to 100. But for multiples of three print "Rehan" instead of the number and for the multiples of five print "Khan". For numbers which are multiples of both three and five print "RehanKhan"**

**class** Test {

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

StringBuilder sb = **new** StringBuilder(100);

**for** (**int** i = 1; i <= 100; i++) {

**if** (i % 15 == 0) {

sb.append("RehanKhan\n");

} **else** **if** (i % 5 == 0) {

sb.append("Khan\n");

} **else** **if** (i % 3 == 0) {

sb.append("Rehan\n");

} **else** {

sb.append(i);

sb.append("\n");

}

}

System.*out*.println(sb.toString());

}

}

**How to Remove Duplicate Values From Java ArrayList**

**import** java.util.\*;

**public** **class** Test {

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

List lst = **new** ArrayList();

lst.add("Rehan");

lst.add("Rehan");

lst.add("Khan");

lst.add("Khan");

lst.add("What");

List newlst = **new** ArrayList(**new** HashSet(lst));

Iterator it = newlst.iterator();

**while** (it.hasNext()) {

System.*out*.println(it.next());

}

}

}

**import** java.util.\*;

**import** java.util.Map.Entry;

**class** MySingleTon {

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

List<String> lst = **new** ArrayList<String>();

lst.add("Rehan");

lst.add("Rehan");

lst.add("Rehan");

lst.add("aaa");

lst.add("aaa");

List<String> temp = **new** ArrayList<String>();

**for** (String dup : lst) {

**if** (!temp.contains(dup)) {

temp.add(dup);

System.*out*.println(dup);

}

}

}

}

**TreeSet Sorting in ascending order & descending Order.**

**package** rev;

**import** java.util.\*;

**public** **class** Test {

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

TreeSet ts = **new** TreeSet();

ts.add(123);

ts.add(13);

ts.add(12);

ts.add(23);

Iterator it = ts.iterator();

**while** (it.hasNext()) {

System.*out*.println(it.next());

}

System.*out*.println("---------------");

Iterator it1 = ts.descendingIterator();

**while** (it1.hasNext()) {

System.*out*.println(it1.next());

}

}

}

**Conversion of Map to List**

**import** java.util.\*;

**import** java.util.Map.Entry;

**class** Lab2 {

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

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

map.put(1, "ABC");

map.put(2, "GFHG");

map.put(3, "EFE");

map.put(4, "DEF");

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

List<Entry<Integer, String>> list = **new** ArrayList<>(set);

System.*out*.println(list);

}

}

**How To Take Duplicate value in Set.**

**import** java.util.\*;

**public** **class** Test {

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

Set<Employee> st = **new** HashSet<Employee>();

Employee emp1 = **new** Employee(1, "Rehan");

Employee emp2 = **new** Employee(1, "Rehan");

st.add(emp1);

st.add(emp2);

Iterator<Employee> it = st.iterator();

**while** (it.hasNext()) {

System.*out*.println(it.next());

}

}

}

**class** Employee {

**int** eid;

String ename;

**public** Employee(**int** eid, String ename) {

**this**.eid = eid;

**this**.ename = ename;

}

@Override

**public** String toString() {

**return** "Employee [eid=" + eid + ", ename=" + ename + "]";

}

}

**Sorting In List ascending and descending Order.**

**import** java.util.\*;

**class** Test {

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

List lst = **new** ArrayList();

lst.add(1);

lst.add(1);

lst.add(23);

lst.add(1234);

lst.add(232);

lst.add(123);

Collections.*sort*(lst);

Iterator it = lst.iterator();

**while** (it.hasNext()) {

System.*out*.println(it.next());

}

System.*out*.println("-----------------------------------");

Collections.*reverse*(lst);

Iterator it1 = lst.iterator();

**while** (it1.hasNext()) {

System.*out*.println(it1.next());

}

}

}

**Sorting Using Comparable Interface**

**import** java.util.\*;

**class** Test {

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

TreeSet set = **new** TreeSet();

set.add(**new** Student(1, "Rehan"));

set.add(**new** Student(2, "Khan"));

set.add(**new** Student(3, "ABC"));

set.add(**new** Student(4, "AGDS"));

Iterator it = set.iterator();

**while** (it.hasNext()) {

System.*out*.println(it.next());

}

}

}

**class** Student **implements** Comparable {

**int** sid;

String sname;

**public** Student(**int** sid, String sname) {

**this**.sid = sid;

**this**.sname = sname;

}

@Override

**public** **int** compareTo(Object obj) {

**if** (obj **instanceof** Student) {

Student st = (Student) obj;

**return** **this**.sname.compareTo(st.sname);

}

// if (obj instanceof Student) {

// Student st = (Student) obj;

// return this.sid - st.sid;

// }

**return** 0;

}

@Override

**public** String toString() {

**return** "Student [sid=" + sid + ", sname=" + sname + "]";

}

}

**Sorting Using Comparator Interface based on Sid**

**(Descending Order)**

**import** java.util.\*;

**public** **class** Test {

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

TreeSet ts = **new** TreeSet(**new** SidComp());

ts.add(**new** Student(1, "Rehan"));

ts.add(**new** Student(2, "Khan"));

ts.add(**new** Student(3, "Adidas"));

ts.add(**new** Student(4, "ZZZ"));

Iterator it = ts.iterator();

**while** (it.hasNext()) {

System.*out*.println(it.next());

}

}

}

**class** Student {

**int** sid;

String sname;

**public** Student(**int** sid, String sname) {

**this**.sid = sid;

**this**.sname = sname;

}

@Override

**public** String toString() {

**return** "Student [sid=" + sid + ", sname=" + sname + "]";

}

}

**class** SidComp **implements** Comparator {

@Override

**public** **int** compare(Object obj1, Object obj2) {

**if** (obj1 **instanceof** Student && obj2 **instanceof** Student) {

Student s1 = (Student) obj1;

Student s2 = (Student) obj2;

**return** s2.sid - s1.sid;

}

**return** 0;

}

}

Output :

Student [sid=4, sname=ZZZ]

Student [sid=3, sname=Adidas]

Student [sid=2, sname=Khan]

Student [sid=1, sname=Rehan]

**Sorting Using Comparator Interface based on name.**

**(Descending Order)**

**import** java.util.\*;

**class** Test {

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

TreeSet ts = **new** TreeSet(**new** SidComp());

ts.add(**new** Student(1, "Rehan"));

ts.add(**new** Student(2, "Khan"));

ts.add(**new** Student(3, "Adidas"));

ts.add(**new** Student(4, "ZZZ"));

Iterator it = ts.iterator();

**while** (it.hasNext()) {

System.*out*.println(it.next());

}

}

}

**class** Student {

**int** sid;

String sname;

**public** Student(**int** sid, String sname) {

**this**.sid = sid;

**this**.sname = sname;

}

@Override

**public** String toString() {

**return** "Student [sid=" + sid + ", sname=" + sname + "]";

}

}

**class** SidComp **implements** Comparator {

@Override

**public** **int** compare(Object obj1, Object obj2) {

**if** (obj1 **instanceof** Student && obj2 **instanceof** Student) {

Student s1 = (Student) obj1;

Student s2 = (Student) obj2;

**return** s2.sname.compareTo(s1.sname);

}

**return** 0;

}

}

OutPut:

Student [sid=4, sname=ZZZ]

Student [sid=1, sname=Rehan]

Student [sid=2, sname=Khan]

Student [sid=3, sname=Adidas]

**Write a program to find top two maximum numbers**

**in a array.**

**class** MySingleTon {

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

**int** num[] = { 5, 34, 78, 2, 45, 1, 99, 23 };

**int** maxOne = 0;

**int** maxTwo = 0;

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

**if** (maxOne < n) {

maxTwo = maxOne;

maxOne = n;

} **else** **if** (maxTwo < n) {

maxTwo = n;

}

}

System.*out*.println(maxOne);

System.*out*.println(maxTwo);

}

}

**Java program to check armstrong number**

**import** java.util.\*;

**class** Test {

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

**int** sum = 0, rem, temp;

System.*out*.println("Enter a Number");

Scanner sc = **new** Scanner(System.*in*);

**int** n = sc.nextInt();

temp = n;

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

rem = temp % 10;

sum = sum + rem \* rem \* rem;

temp = temp / 10;

}

**if** (sum == n) {

System.*out*.println("Amrstrong");

} **else** {

System.*out*.println("Nor Armstrong");

}

}

}

153

Armstrong

**Pattern**

1

12

123

1234

12345

**class** ZZZ {

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

**for** (**int** i = 1; i <= 5; i++) {

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

System.*out*.print(j);

}

System.*out*.println();

}

}

}

-------------------------------------------

54321

5432

543

54

5

**class** ZZZ {

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

**for** (**int** i = 1; i <= 5; i++) {

**for** (**int** j = 5; j >= i; j--) {

System.*out*.print(j);

}

System.*out*.println();

}

}

}

-------------------------------------------

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

**class** ZZZ {

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

**for** (**int** i = 1; i <= 5; i++) {

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

System.*out*.print("\*");

}

System.*out*.println();

}

}

}

**Example of Fail Fast Iterator and Fail Safe Iterator**

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

public class FailFastExample

{

public static void main(String[] args)

{

Map<String,String> premiumPhone = new HashMap<String,String>();

premiumPhone.put("Apple", "iPhone");

premiumPhone.put("HTC", "HTC one");

premiumPhone.put("Samsung","S5");

Iterator iterator = premiumPhone.keySet().iterator();

while (iterator.hasNext())

{

System.out.println(premiumPhone.get(iterator.next()));

premiumPhone.put("Sony", "Xperia Z");

}

}

}

**Output :**

iPhone

Exception in thread "main" java.util.ConcurrentModificationException

at java.util.HashMap$HashIterator.nextEntry(Unknown Source)

at java.util.HashMap$KeyIterator.next(Unknown Source)

at FailFastExample.main(FailFastExample.java:20)

**Fail Safe Iterator Example :**

import java.util.concurrent.ConcurrentHashMap;

import java.util.Iterator;

public class FailSafeExample

{

public static void main(String[] args)

{

ConcurrentHashMap<String,String> premiumPhone =

new ConcurrentHashMap<String,String>();

premiumPhone.put("Apple", "iPhone");

premiumPhone.put("HTC", "HTC one");

premiumPhone.put("Samsung","S5");

Iterator iterator = premiumPhone.keySet().iterator();

while (iterator.hasNext())

{

System.out.println(premiumPhone.get(iterator.next()));

premiumPhone.put("Sony", "Xperia Z");

}

}

}

**Output :**

S5

HTC one

iPhone

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

**class** ZZZ {

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

**for** (**int** i = 1; i <= 5; i++) {

**for** (**int** j = 5; j >= i; j--) {

System.*out*.print("\*");

}

System.*out*.println();

}

}

}

# [Read text file and store in hashmap. Then sort in order](http://stackoverflow.com/questions/13513667/read-text-file-and-store-in-hashmap-then-sort-in-order)

import java.io.\*;

import java.util.\*;

class Test

{

public static void main(String[] args) throws FileNotFoundException {

Scanner scanner = new Scanner(new FileReader("score.txt"));

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

while (scanner.hasNextLine()) {

String[] columns = scanner.nextLine().split("\t\t");

map.put(columns[0], columns[1]);

}

System.out.println(map);

}

}

Program: Find out duplicate number between 1 to N numbers.

package com.java2novice.algos;

import java.util.ArrayList;

import java.util.List;

public class DuplicateNumber {

public int findDuplicateNumber(List<Integer> numbers){

int highestNumber = numbers.size() - 1;

int total = getSum(numbers);

int duplicate = total - (highestNumber\*(highestNumber+1)/2);

return duplicate;

}

public int getSum(List<Integer> numbers){

int sum = 0;

for(int num:numbers){

sum += num;

}

return sum;

}

public static void main(String a[]){

List<Integer> numbers = new ArrayList<Integer>();

for(int i=1;i<30;i++){

numbers.add(i);

}

//add duplicate number into the list

numbers.add(22);

DuplicateNumber dn = new DuplicateNumber();

System.out.println("Duplicate Number: "+dn.findDuplicateNumber(numbers));

}

}

Output:

Duplicate Number: 22

**Program: Find out middle index where sum of both ends are equal.**

package com.java2novice.algos;

public class FindMiddleIndex {

public static int findMiddleIndex(int[] numbers) throws Exception {

int endIndex = numbers.length - 1;

int startIndex = 0;

int sumLeft = 0;

int sumRight = 0;

while (true) {

if (sumLeft > sumRight) {

sumRight += numbers[endIndex--];

} else {

sumLeft += numbers[startIndex++];

}

if (startIndex > endIndex) {

if (sumLeft == sumRight) {

break;

} else {

throw new Exception(

"Please pass proper array to match the requirement");

}

}

}

return endIndex;

}

public static void main(String a[]) {

int[] num = { 2, 4, 4, 5, 4, 1 };

try {

System.out.println("Starting from index 0, adding numbers till index "+ findMiddleIndex(num) + " and");

System.out.println("adding rest of the numbers can be equal");

} catch (Exception ex) {

System.out.println(ex.getMessage());

}

}

}

Output:

Starting from index 0, adding numbers till index 2 and

adding rest of the numbers can be equal

**Write a program to create deadlock between two threads.**

Description:

Deadlock describes a situation where two or more threads are blocked forever, waiting for each other. Deadlocks can occur in Java when the synchronized keyword causes the executing thread to block while waiting to get the lock, associated with the specified object. Since the thread might already hold locks associated with other objects, two threads could each be waiting for the other to release a lock. In such case, they will end up waiting forever.

package com.java2novice.algos;

public class MyDeadlock {

String str1 = "Java";

String str2 = "UNIX";

Thread trd1 = new Thread("My Thread 1"){

public void run(){

while(true){

synchronized(str1){

synchronized(str2){

System.out.println(str1 + str2);

}

}

}

}

};

Thread trd2 = new Thread("My Thread 2"){

public void run(){

while(true){

synchronized(str2){

synchronized(str1){

System.out.println(str2 + str1);

}

}

}

}

};

public static void main(String a[]){

MyDeadlock mdl = new MyDeadlock();

mdl.trd1.start();

mdl.trd2.start();

}

}

**Write a program to reverse a string using recursive algorithm.**

Description:

Write a program to reverse a string using recursive methods.

You should not use any string reverse methods to do this.

package com.java2novice.algos;

public class StringRecursiveReversal {

String reverse = "";

public String reverseString(String str){

if(str.length() == 1){

return str;

} else {

reverse += str.charAt(str.length()-1) +reverseString(str.substring(0,str.length()-1));

return reverse;

}

}

public static void main(String a[]){

StringRecursiveReversal srr = new StringRecursiveReversal();

System.out.println("Result: "+srr.reverseString("Java2novice"));

}

}

Output:

Result: ecivon2avaJ

**Write a program to reverse a number.**

Description:

Write a program to reverse a number using numeric operations. Below example shows how to reverse a number using numeric operations.

package com.java2novice.algos;

public class NumberReverse {

public int reverseNumber(int number){

int reverse = 0;

while(number != 0){

reverse = (reverse\*10)+(number%10);

number = number/10;

}

return reverse;

}

public static void main(String a[]){

NumberReverse nr = new NumberReverse();

System.out.println("Result: "+nr.reverseNumber(17868));

}

}

Output:

Result: 86871

**Program: Write a program to find perfect number or not.**

Description:

A perfect number is a positive integer that is equal to the sum

of its proper positive divisors, that is, the sum of its positive

divisors excluding the number itself. Equivalently, a perfect number

is a number that is half the sum of all of its positive divisors.

The first perfect number is 6, because 1, 2 and 3 are its proper

positive divisors, and 1 + 2 + 3 = 6. Equivalently, the number 6

is equal to half the sum of all its positive divisors:

( 1 + 2 + 3 + 6 ) / 2 = 6.

package com.java2novice.algos;

public class IsPerfectNumber {

public boolean isPerfectNumber(int number){

int temp = 0;

for(int i=1;i<=number/2;i++){

if(number%i == 0){

temp += i;

}

}

if(temp == number){

System.out.println("It is a perfect number");

return true;

} else {

System.out.println("It is not a perfect number");

return false;

}

}

public static void main(String a[]){

IsPerfectNumber ipn = new IsPerfectNumber();

System.out.println("Is perfect number: "+ipn.isPerfectNumber(28));

}

}

Output:

28

It is a perfect number

Is perfect number: true

**Program: Write a program to implement ArrayList.**

Description:

Write a program to implement your own ArrayList class. It should

contain add(), get(), remove(), size() methods. Use dynamic array logic.

It should increase its size when it reaches threshold.

package com.java2novice.algos;

import java.util.Arrays;

public class MyArrayList {

private Object[] myStore;

private int actSize = 0;

public MyArrayList(){

myStore = new Object[10];

}

public Object get(int index){

if(index < actSize){

return myStore[index];

} else {

throw new ArrayIndexOutOfBoundsException();

}

}

public void add(Object obj){

if(myStore.length-actSize <= 5){

increaseListSize();

}

myStore[actSize++] = obj;

}

public Object remove(int index){

if(index < actSize){

Object obj = myStore[index];

myStore[index] = null;

int tmp = index;

while(tmp < actSize){

myStore[tmp] = myStore[tmp+1];

myStore[tmp+1] = null;

tmp++;

}

actSize--;

return obj;

} else {

throw new ArrayIndexOutOfBoundsException();

}

}

public int size(){

return actSize;

}

private void increaseListSize(){

myStore = Arrays.copyOf(myStore, myStore.length\*2);

System.out.println("\nNew length: "+myStore.length);

}

public static void main(String a[]){

MyArrayList mal = new MyArrayList();

mal.add(new Integer(2));

mal.add(new Integer(5));

mal.add(new Integer(1));

mal.add(new Integer(23));

mal.add(new Integer(14));

for(int i=0;i<mal.size();i++){

System.out.print(mal.get(i)+" ");

}

mal.add(new Integer(29));

System.out.println("Element at Index 5:"+mal.get(5));

System.out.println("List size: "+mal.size());

System.out.println("Removing element at index 2: "+mal.remove(2));

for(int i=0;i<mal.size();i++){

System.out.print(mal.get(i)+" ");

}

}

}

Output:

2 5 1 23 14

New length: 20

Element at Index 5:29

List size: 6

Removing element at index 2: 1

2 5 23 14 29

**Write a program to find maximum repeated words from a file.**

Description:

Write a program to read words from a file. Count the

repeated or duplicated words. Sort it by maximum repeated or

duplicated word count.

package com.java2novice.algos;

import java.io.\*;

import java.util.\*;

import java.util.Map.Entry;

public class MaxDuplicateWordCount {

public Map<String, Integer> getWordCount(String fileName){

FileInputStream fis = null;

DataInputStream dis = null;

BufferedReader br = null;

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

try {

fis = new FileInputStream(fileName);

dis = new DataInputStream(fis);

br = new BufferedReader(new InputStreamReader(dis));

String line = null;

while((line = br.readLine()) != null){

StringTokenizer st = new StringTokenizer(line, " ");

while(st.hasMoreTokens()){

String tmp = st.nextToken().toLowerCase();

if(wordMap.containsKey(tmp)){

wordMap.put(tmp, wordMap.get(tmp)+1);

} else {

wordMap.put(tmp, 1);

}

}

}

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} finally{

try{

if(br != null) br.close();}catch(Exception ex){}

}

return wordMap;

}

public List<Entry<String, Integer>> sortByValue(Map<String, Integer> wordMap){

Set<Entry<String, Integer>> set = wordMap.entrySet();

List<Entry<String, Integer>> list = new ArrayList<Entry<String, Integer>>(set);

Collections.sort( list, new Comparator<Map.Entry<String, Integer>>()

{

public int compare( Map.Entry<String, Integer> o1, Map.Entry<String, Integer> o2 )

{

return (o2.getValue()).compareTo( o1.getValue() );

}

} );

return list;

}

public static void main(String a[]){

MaxDuplicateWordCount mdc = new MaxDuplicateWordCount();

Map<String, Integer> wordMap =mdc.getWordCount("C:/MyTestFile.txt");

List<Entry<String, Integer>> list = mdc.sortByValue(wordMap);

for(Map.Entry<String, Integer> entry:list){

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

}

}

}

Output:

one ==== 3

the ==== 3

that ==== 3

of ==== 2

in ==== 2

some ==== 2

to ==== 1

summary ==== 1

but ==== 1

have ==== 1

common ==== 1

least ==== 1

simplest ==== 1

**Write a program to find out duplicate characters in a string.**

Description:

Write a program to find out duplicate or repeated characters in a

string, and calculate the count of repeatation.

package com.java2novice.algos;

import java.util.\*;

public class DuplicateCharsInString {

public void findDuplicateChars(String str){

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

char[] chrs = str.toCharArray();

for(Character ch:chrs){

if(dupMap.containsKey(ch)){

dupMap.put(ch, dupMap.get(ch)+1);

} else {

dupMap.put(ch, 1);

}

}

Set<Character> keys = dupMap.keySet();

for(Character ch:keys){

if(dupMap.get(ch) > 1){

System.out.println(ch+"--->"+dupMap.get(ch));

}

}

}

public static void main(String a[]){

DuplicateCharsInString dcs = new DuplicateCharsInString();

dcs.findDuplicateChars("Java2Novice");

}

}

Output:

v--->2

a--->2

**Write a program to find top two maximum numbers in a array.**

Description:

Write a program to find top two maximum numbers in the

given array. You should not use any sorting functions. You

should iterate the array only once. You should not use any

kind of collections in java.

package com.java2novice.algos;

public class TwoMaxNumbers {

public void printTwoMaxNumbers(int[] nums){

int maxOne = 0;

int maxTwo = 0;

for(int n:nums){

if(maxOne < n){

maxTwo = maxOne;

maxOne =n;

} else if(maxTwo < n){

maxTwo = n;

}

}

System.out.println("First Max Number: "+maxOne);

System.out.println("Second Max Number: "+maxTwo);

}

public static void main(String a[]){

int num[] = {5,34,78,2,45,1,99,23};

TwoMaxNumbers tmn = new TwoMaxNumbers();

tmn.printTwoMaxNumbers(num);

}

}

Output:

First Max Number: 99

Second Max Number: 78

**Write a program to sort a map by value.**

package com.java2novice.algos;

import java.util.\*;

import java.util.Map.Entry;

class OrderByValue {

public static void main(String a[]){

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

map.put("java", 20);

map.put("C++", 45);

map.put("Java2Novice", 2);

map.put("Unix", 67);

map.put("MAC", 26);

map.put("Why this kolavari", 93);

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

List<Entry<String, Integer>> list = new ArrayList<Entry<String, Integer>>(set);

Collections.sort( list, new Comparator<Map.Entry<String, Integer>>()

{

public int compare( Map.Entry<String, Integer> o1, Map.Entry<String, Integer> o2 )

{

return (o2.getValue()).compareTo( o1.getValue() );

}

} );

for(Map.Entry<String, Integer> entry:list){

System.out.println(entry.getKey()+" ==== "+entry.getValue()); }

}

}

Output:

Why this kolavari ==== 93

Unix ==== 67

C++ ==== 45

MAC ==== 26

java ==== 20

Java2Novice ==== 2

**Write a program to find common elements between two arrays.**

Description:

Write a program to identify common elements or numbers between

two given arrays. You should not use any inbuilt methods are list to

find common values.

Code:

package com.java2novice.algos;

public class CommonElementsInArray {

public static void main(String a[]){

int[] arr1 = {4,7,3,9,2};

int[] arr2 = {3,2,12,9,40,32,4};

for(int i=0;i<arr1.length;i++){

for(int j=0;j<arr2.length;j++){

if(arr1[i]==arr2[j]){

System.out.println(arr1[i]);

}

}

}

}

}

Output:

4

3

9

2

**How to swap two numbers without using temporary variable?**

Description:

Write a program to swap or exchange two numbers. You should

not use any temporary or third variable to swap.

Code:

package com.java2novice.algos;

public class MySwapingTwoNumbers {

public static void main(String a[]){

int x = 10;

int y = 20;

System.out.println("Before swap:");

System.out.println("x value: "+x);

System.out.println("y value: "+y);

x = x+y;

y=x-y;

x=x-y;

System.out.println("After swap:");

System.out.println("x value: "+x);

System.out.println("y value: "+y);

}

}

Output:

Before swap:

x value: 10

y value: 20

After swap:

x value: 20

y value: 10

Program: Write a program to print fibonacci series.

Description:

In mathematics, the Fibonacci numbers or Fibonacci series or Fibonacci sequence are the numbers in the following integer sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144... By definition, the first two numbers in the Fibonacci sequence are 0 and 1, and each subsequent number is the sum of the previous two. Below example shows how to create fibonacci series.

package com.java2novice.algos;

public class MyFibonacci {

public static void main(String a[]){

int febCount = 15;

int[] feb = new int[febCount];

feb[0] = 0;

feb[1] = 1;

for(int i=2; i < febCount; i++){

feb[i] = feb[i-1] + feb[i-2];

}

for(int i=0; i< febCount; i++){

System.out.print(feb[i] + " ");

}

}

}

Output:

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

Program: Write a program to convert decimal number to binary format.

Description:

Write a program to convert decimal number to binary format using numeric operations. Below example shows how to convert decimal number to binary format using numeric operations.

Code:

?

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21 package com.java2novice.algos;

public class DecimalToBinary {

public void printBinaryFormat(int number){

int binary[] = new int[25];

int index = 0;

while(number > 0){

binary[index++] = number%2;

number = number/2;

}

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

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

}

}

public static void main(String a[]){

DecimalToBinary dtb = new DecimalToBinary();

dtb.printBinaryFormat(25);

}

}

Output:

11001

Program: Write a program to find sum of each digit in the given number using recursion.

Description:

Below example shows how to find out sum of each digit in the given number using recursion logic. For example, if the number is 259, then the sum should be 2+5+9 = 16.

package com.java2novice.algos;

public class MyNumberSumRec {

int sum = 0;

public int getNumberSum(int number){

if(number == 0){

return sum;

} else {

sum += (number%10);

getNumberSum(number/10);

}

return sum;

}

public static void main(String a[]){

MyNumberSumRec mns = new MyNumberSumRec();

System.out.println("Sum is: "+mns.getNumberSum(223));

}

}

Output:

Sum is: 7

Program: Write a program to check the given number is a prime number or not?

Description:

A prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself. A natural number greater than 1 that is not a prime number is called a composite number. For example, 5 is prime, as only 1 and 5 divide it, whereas 6 is composite, since it has the divisors 2 and 3 in addition to 1 and 6. The fundamental theorem of arithmetic establishes the central role of primes in number theory: any integer greater than 1 can be expressed as a product of primes that is unique up to ordering. This theorem requires excluding 1 as a prime.

package com.java2novice.algos;

public class MyPrimeNumCheck {

public boolean isPrimeNumber(int number){

for(int i=2; i<=number/2; i++){

if(number % i == 0){

return false;

}

}

return true;

}

public static void main(String a[]){

MyPrimeNumCheck mpc = new MyPrimeNumCheck();

System.out.println("Is 17 prime number? "+mpc.isPrimeNumber(17));

System.out.println("Is 19 prime number? "+mpc.isPrimeNumber(19));

System.out.println("Is 15 prime number? "+mpc.isPrimeNumber(15));

}

}

Output:

Is 17 prime number? true

Is 19 prime number? true

Is 15 prime number? false

Program: Write a program to find the given number is Armstrong number or not?

Description:

Armstrong numbers are the sum of their own digits to the power of

the number of digits. It is also known as narcissistic numbers.

package com.java2novice.algos;

public class MyArmstrongNumber {

public boolean isArmstrongNumber(int number){

int tmp = number;

int noOfDigits = String.valueOf(number).length();

int sum = 0;

int div = 0;

while(tmp > 0)

{

div = tmp % 10;

int temp = 1;

for(int i=0;i<noOfDigits;i++){

temp \*= div;

}

sum += temp;

tmp = tmp/10;

}

if(number == sum) {

return true;

} else {

return false;

}

}

public static void main(String a[]){

MyArmstrongNumber man = new MyArmstrongNumber();

System.out.println("Is 371 Armstrong number? "+man.isArmstrongNumber(371));

System.out.println("Is 523 Armstrong number? "+man.isArmstrongNumber(523));

System.out.println("Is 153 Armstrong number? "+man.isArmstrongNumber(153));

}

}

Output:

Is 371 Armstrong number? true

Is 523 Armstrong number? false

Is 153 Armstrong number? true

Program: Write a program to convert binary to decimal number.

Description:

Write a program to convert binary format to decimal number using numeric operations. Below example shows how to convert binary to decimal format using numeric operations.

package com.java2novice.algos;

public class BinaryToDecimal {

public int getDecimalFromBinary(int binary){

int decimal = 0;

int power = 0;

while(true){

if(binary == 0){

break;

} else {

int tmp = binary%10;

decimal += tmp\*Math.pow(2, power);

binary = binary/10;

power++;

}

}

return decimal;

}

public static void main(String a[]){

BinaryToDecimal bd = new BinaryToDecimal();

System.out.println("11 ===> "+bd.getDecimalFromBinary(11));

System.out.println("110 ===> "+bd.getDecimalFromBinary(110));

System.out.println("100110 ===> "+bd.getDecimalFromBinary(100110));

}

}

Output:

11 ===> 3

110 ===> 6

100110 ===> 38

**Write a program to check the given number is binary number or not?**

Description:

The binary numeral system, or base-2 number system, represents numeric values using two symbols: 0 and 1. More specifically, the usual base-2 system is a positional notation with a radix of 2. Because of its straightforward implementation in digital electronic circuitry using logic gates, the binary system is used internally by almost all modern computers.

package com.java2novice.algos;

public class MyBinaryCheck {

public boolean isBinaryNumber(int binary){

boolean status = true;

while(true){

if(binary == 0){

break;

} else {

int tmp = binary%10;

if(tmp > 1){

status = false;

break;

}

binary = binary/10;

}

}

return status;

}

public static void main(String a[]){

MyBinaryCheck mbc = new MyBinaryCheck();

System.out.println("Is 1000111 binary? :"+mbc.isBinaryNumber(1000111));

System.out.println("Is 10300111 binary? :"+mbc.isBinaryNumber(10300111));

}

}

Output:

Is 1000111 binary? :true

Is 10300111 binary? :false

**Write a program for Bubble Sort in java.**

Description:

Bubble sort is a simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm gets its name from the way smaller elements bubble to the top of the list. Because it only uses comparisons to operate on elements, it is a comparison sort. You can see the code implementation below:

package com.java2novice.algos;

public class MyBubbleSort {

// logic to sort the elements

public static void bubble\_srt(int array[]) {

int n = array.length;

int k;

for (int m = n; m >= 0; m--) {

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

k = i + 1;

if (array[i] > array[k]) {

swapNumbers(i, k, array);

}

}

printNumbers(array);

}

}

private static void swapNumbers(int i, int j, int[] array) {

int temp;

temp = array[i];

array[i] = array[j];

array[j] = temp;

}

private static void printNumbers(int[] input) {

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

System.out.print(input[i] + ", ");

}

System.out.println("\n");

}

public static void main(String[] args) {

int[] input = { 4, 2, 9, 6, 23, 12, 34, 0, 1 };

bubble\_srt(input);

}

}

Output:

2, 4, 6, 9, 12, 23, 0, 1, 34,

2, 4, 6, 9, 12, 0, 1, 23, 34,

2, 4, 6, 9, 0, 1, 12, 23, 34,

2, 4, 6, 0, 1, 9, 12, 23, 34,

2, 4, 0, 1, 6, 9, 12, 23, 34,

2, 0, 1, 4, 6, 9, 12, 23, 34,

0, 1, 2, 4, 6, 9, 12, 23, 34,

0, 1, 2, 4, 6, 9, 12, 23, 34,

0, 1, 2, 4, 6, 9, 12, 23, 34,

0, 1, 2, 4, 6, 9, 12, 23, 34,

**Write a program for Insertion Sort in java.**

Description:

Insertion sort is a simple sorting algorithm that builds the final sorted array one item at a time. It is much less efficient on large lists than more advanced algorithms such as quicksort, heapsort, or merge sort. Every repetition of insertion sort removes an element from the input data, inserting it into the correct position in the already-sorted list, until no input elements remain. The choice of which element to remove from the input is arbitrary, and can be made using almost any choice algorithm. You can see the code implementation below:

package com.java2novice.algos;

public class MyInsertionSort {

public static void main(String[] args) {

int[] input = { 4, 2, 9, 6, 23, 12, 34, 0, 1 };

insertionSort(input);

}

private static void printNumbers(int[] input) {

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

System.out.print(input[i] + ", ");

}

System.out.println("\n");

}

public static void insertionSort(int array[]) {

int n = array.length;

for (int j = 1; j < n; j++) {

int key = array[j];

int i = j-1;

while ( (i > -1) && ( array [i] > key ) ) {

array [i+1] = array [i];

i--;

}

array[i+1] = key;

printNumbers(array);

}

}

}

Output:

2, 4, 9, 6, 23, 12, 34, 0, 1,

2, 4, 9, 6, 23, 12, 34, 0, 1,

2, 4, 6, 9, 23, 12, 34, 0, 1,

2, 4, 6, 9, 23, 12, 34, 0, 1,

2, 4, 6, 9, 12, 23, 34, 0, 1,

2, 4, 6, 9, 12, 23, 34, 0, 1,

0, 2, 4, 6, 9, 12, 23, 34, 1,

0, 1, 2, 4, 6, 9, 12, 23, 34,

**Write a program to implement hashcode and equals.**

Description:

The hashcode of a Java Object is simply a number, it is 32-bit signed int, that allows an object to be managed by a hash-based data structure. We know that hash code is an unique id number allocated to an object by JVM. But actually speaking, Hash code is not an unique number for an object. If two objects are equals then these two objects should return same hash code. So we have to implement hashcode() method of a class in such way that if two objects are equals, ie compared by equal() method of that class, then those two objects must return same hash code. If you are overriding hashCode you need to override equals method also.

The below example shows how to override equals and hashcode methods. The class Price overrides equals and hashcode. If you notice the hashcode implementation, it always generates unique hashcode for each object based on their state, ie if the object state is same, then you will get same hashcode. A HashMap is used in the example to store Price objects as keys. It shows though we generate different objects, but if state is same, still we can use this as key.

package com.java2novice.algos;

import java.util.HashMap;

public class MyHashcodeImpl {

public static void main(String a[]){

HashMap<Price, String> hm = new HashMap<Price, String>();

hm.put(new Price("Banana", 20), "Banana");

hm.put(new Price("Apple", 40), "Apple");

hm.put(new Price("Orange", 30), "Orange");

//creating new object to use as key to get value

Price key = new Price("Banana", 20);

System.out.println("Hashcode of the key: "+key.hashCode());

System.out.println("Value from map: "+hm.get(key));

}

}

class Price{

private String item;

private int price;

public Price(String itm, int pr){

this.item = itm;

this.price = pr;

}

public int hashCode(){

System.out.println("In hashcode");

int hashcode = 0;

hashcode = price\*20;

hashcode += item.hashCode();

return hashcode;

}

public boolean equals(Object obj){

System.out.println("In equals");

if (obj instanceof Price) {

Price pp = (Price) obj;

return (pp.item.equals(this.item) && pp.price == this.price);

} else {

return false;

}

}

public String getItem() {

return item;

}

public void setItem(String item) {

this.item = item;

}

public int getPrice() {

return price;

}

public void setPrice(int price) {

this.price = price;

}

public String toString(){

return "item: "+item+" price: "+price;

}

}

Output:

In hashcode

In hashcode

In hashcode

In hashcode

Hashcode of the key: 1982479637

In hashcode

In equals

Value from map: Banana

**How to get distinct elements from an array by avoiding duplicate elements?**

The below example shows how to avoid duplicate elements from an array and disply only distinct elements. Please use only arrays to process it.

package com.java2novice.algos;

public class MyDisticntElements {

public static void printDistinctElements(int[] arr){

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

boolean isDistinct = false;

for(int j=0;j<i;j++){

if(arr[i] == arr[j]){

isDistinct = true;

break;

}

}

if(!isDistinct){

System.out.print(arr[i]+" ");

}

}

}

public static void main(String a[]){

int[] nums = {5,2,7,2,4,7,8,2,3};

MyDisticntElements.printDistinctElements(nums);

}

}

Output:

5 2 7 4 8 3

**Write a program to get distinct word list from the given file.**

Description:

Write a program to find all distinct words from the given file. Remove special chars like ".,;:" etc. Ignore case sensitivity.

package com.java2novice.algos;

import java.io.\*;

import java.util.\*;

public class MyDistinctFileWords {

public List<String> getDistinctWordList(String fileName){

FileInputStream fis = null;

DataInputStream dis = null;

BufferedReader br = null;

List<String> wordList = new ArrayList<String>();

try {

fis = new FileInputStream(fileName);

dis = new DataInputStream(fis);

br = new BufferedReader(new InputStreamReader(dis));

String line = null;

while((line = br.readLine()) != null){

StringTokenizer st = new StringTokenizer(line, " ,.;:\"");

while(st.hasMoreTokens()){

String tmp = st.nextToken().toLowerCase();

if(!wordList.contains(tmp)){

wordList.add(tmp);

}

}

}

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} finally{

try{if(br != null) br.close();}catch(Exception ex){}

}

return wordList;

}

public static void main(String a[]){

MyDistinctFileWords distFw = new MyDistinctFileWords();

List<String> wordList = distFw.getDistinctWordList("C:/sample.txt");

for(String str:wordList){

System.out.println(str);

}

}

}

Output:

the

while

statement

verifies

condition

before

entering

into

loop

to

see

whether

next

iteration

should

occur

or

not

do-while

executes

first

without

checking

it

after

finishing

each

will

always

execute

body

of

a

at

least

once

**Write a program to get a line with max word count from the given file.**

Below example shows how to find out the line with maximum number of word count in the given file. In case if it has multiple lines with max number of words, then it has to list all those lines.

package com.java2novice.algos;

import java.io.\*;

import java.util.\*;

public class MaxWordCountInLine {

private int currentMaxCount = 0;

private List<String> lines = new ArrayList<String>();

public void readMaxLineCount(String fileName){

FileInputStream fis = null;

DataInputStream dis = null;

BufferedReader br = null;

try {

fis = new FileInputStream(fileName);

dis = new DataInputStream(fis);

br = new BufferedReader(new InputStreamReader(dis));

String line = null;

while((line = br.readLine()) != null){

int count = (line.split("\\s+")).length;

if(count > currentMaxCount){

lines.clear();

lines.add(line);

currentMaxCount = count;

} else if(count == currentMaxCount){

lines.add(line);

}

}

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} finally{

try{

if(br != null) br.close();

}catch(Exception ex){}

}

}

public int getCurrentMaxCount() {

return currentMaxCount;

}

public void setCurrentMaxCount(int currentMaxCount) {

this.currentMaxCount = currentMaxCount;

}

public List<String> getLines() {

return lines;

}

public void setLines(List<String> lines) {

this.lines = lines;

}

public static void main(String a[]){

MaxWordCountInLine mdc = new MaxWordCountInLine();

mdc.readMaxLineCount("/Users/ngootooru/MyTestFile.txt");

System.out.println("Max number of words in a line is: "+mdc.getCurrentMaxCount());

System.out.println("Line with max word count:");

List<String> lines = mdc.getLines();

for(String l:lines){

System.out.println(l);

}

}

}

MyTestFile.txt:

true, false, and null might seem like keywords, but they are actually literals.

You cannot use them as identifiers in your programs. The servlet context

is an interface which helps to communicate with other servlets. It contains

information about the Web application and container. It is kind of

application environment. Using the context, a servlet can obtain URL

references to resources, and store attributes that other servlets in the

context can use.

Output:

Max number of words in a line is: 13

Line with max word count:

true, false, and null might seem like keywords, but they are actually literals.

**Write a program to convert string to number without using Integer.parseInt() method.**

Description:

Below example shows how to convert string format of a number to number without calling Integer.parseInt() method. We can do this by converting each character into ascii format and form the number.

package com.java2novice.algos;

public class MyStringToNumber {

public static int convert\_String\_To\_Number(String numStr){

char ch[] = numStr.toCharArray();

int sum = 0;

//get ascii value for zero

int zeroAscii = (int)'0';

for(char c:ch){

int tmpAscii = (int)c;

sum = (sum\*10)+(tmpAscii-zeroAscii);

}

return sum;

}

public static void main(String a[]){

System.out.println("\"3256\" == "+convert\_String\_To\_Number("3256"));

System.out.println("\"76289\" == "+convert\_String\_To\_Number("76289"));

System.out.println("\"90087\" == "+convert\_String\_To\_Number("90087"));

}

}

Output:

"3256" == 3256

"76289" == 76289

"90087" == 90087

**Write a program to find two lines with max characters in descending order.**

Description:

Write a program to read a multiple line text file and write the 'N' longest lines to the output console, where the file to be read is specified as command line aruguments. The program should read an input file. The first line should contain the value of the number 'N' followed by multiple lines. 'N' should be a valid positive integer.

package com.longest.lines;

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

BufferedReader br = null;

String filePath = args[0];

int topList = 0;

Set<Entries> liSet = new TreeSet<Entries>(new MyComp());

try {

br = new BufferedReader(new FileReader(new File(filePath)));

String line = br.readLine();

topList = Integer.parseInt(line.trim());

while((line = br.readLine()) != null){

line = line.trim();

if(!"".equals(line)){

liSet.add(new Entries(line.length(), line));

}

}

int count = 0;

for(Entries ent:liSet){

System.out.println(ent.line);

if(++count == topList){

break;

}

}

} catch (FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public static class Entries{

Integer length;

String line;

public Entries(Integer l,String line){

length = l;

this.line = line;

}

}

public static class MyComp implements Comparator<Entries>{

@Override

public int compare(Entries e1, Entries e2) {

if(e2.length > e1.length){

return 1;

} else {

return -1;

}

}

}

}

Sample input file:

3

Java2novice

My Test line 123

Java world

I know java language

This is a test program

java is simple

Output:

This is a test program

I know java language

My Test line 123

Program: Write a program to find the sum of the first 1000 prime numbers.

Description:

**Write a program to find the sum of the first 1000 prime numbers.**

package com.primesum;

public class Main {

public static void main(String args[]){

int number = 2;

int count = 0;

long sum = 0;

while(count < 1000){

if(isPrimeNumber(number)){

sum += number;

count++;

}

number++;

}

System.out.println(sum);

}

private static boolean isPrimeNumber(int number){

for(int i=2; i<=number/2; i++){

if(number % i == 0){

return false;

}

}

return true;

}

}

Output:

3682913

**Find longest substring without repeating characters.**

**Description:**

Given a string, find the longest substrings without repeating characters. Iterate through the given string, find the longest maximum substrings.

package com.java2novice.algos;

import java.util.\*;

public class MyLongestSubstr {

private Set<String> subStrList = new HashSet<String>();

private int finalSubStrSize = 0;

public Set<String> getLongestSubstr(String input){

//reset instance variables

subStrList.clear();

finalSubStrSize = 0;

// have a boolean flag on each character ascii value

boolean[] flag = new boolean[256];

int j = 0;

char[] inputCharArr = input.toCharArray();

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

char c = inputCharArr[i];

if (flag[c]) {

extractSubString(inputCharArr,j,i);

for (int k = j; k < i; k++) {

if (inputCharArr[k] == c) {

j = k + 1;

break;

}

flag[inputCharArr[k]] = false;

}

} else {

flag[c] = true;

}

}

extractSubString(inputCharArr,j,inputCharArr.length);

return subStrList;

}

private String extractSubString(char[] inputArr, int start, int end){

StringBuilder sb = new StringBuilder();

for(int i=start;i<end;i++){

sb.append(inputArr[i]);

}

String subStr = sb.toString();

if(subStr.length() > finalSubStrSize){

finalSubStrSize = subStr.length();

subStrList.clear();

subStrList.add(subStr);

} else if(subStr.length() == finalSubStrSize){

subStrList.add(subStr);

}

return sb.toString();

}

public static void main(String a[]){

MyLongestSubstr mls = new MyLongestSubstr();

System.out.println(mls.getLongestSubstr("java2novice")); System.out.println(mls.getLongestSubstr("java\_language\_is\_sweet"));

System.out.println(mls.getLongestSubstr("java\_java\_java\_java"));

System.out.println(mls.getLongestSubstr("abcabcbb"));

}

}

Output:

[a2novice]

[uage\_is]

[\_jav, va\_j]

[cab, abc, bca]

**Write a program to remove duplicates from sorted array.**

Description:

Given array is already sorted, and it has duplicate elements. Write a program to remove duplicate elements and return new array without any duplicate elements. The array should contain only unique elements.

package com.java2novice.algos;

public class MyDuplicateElements {

public static int[] removeDuplicates(int[] input){

int j = 0;

int i = 1;

//return if the array length is less than 2

if(input.length < 2){

return input;

}

while(i < input.length){

if(input[i] == input[j]){

i++;

}else{

input[++j] = input[i++];

}

}

int[] output = new int[j+1];

for(int k=0; k<output.length; k++){

output[k] = input[k];

}

return output;

}

public static void main(String a[]){

int[] input1 = {2,3,6,6,8,9,10,10,10,12,12};

int[] output = removeDuplicates(input1);

for(int i:output){

System.out.print(i+" ");

}

}

}

Output:

2 3 6 8 9 10 12