**Java Program to Count the Number of Digits in a Number.**

import java.util.\*;

class Main {

public static int countDigits(int n) {

if(n == 0) return 1;

//if a negative number is entered

if(n < 0) n = -n;

int res = 0;

while(n != 0) {

n = n/10;

res++;

}

return res;

}

public static void main(String args[]) {

// Your code goes here

Scanner scn = new Scanner(System.in);

int n = scn.nextInt(); //input the number

System.out.println("The number of digits in " + n + " are: " + countDigits(n));

}

}

**Write a program in Java to calculate the number of times a digit ‘D’ appears in a number N. You have to take N and D as inputs from the user.**

import java.util.\*;

class Main {

public static int countDigitFreq(int n,int D) {

if(n == 0 && D == 0) return 1; //number 0 has 1 frequency of 0

//if a negative number is entered

if(n < 0) n = -n;

int counter = 0;

while(n != 0) {

int digit = n % 10; //calculate the digit

if(digit == D) counter++;

n = n/10;

}

return counter;

}

public static void main(String args[]) {

// Your code goes here

Scanner scn = new Scanner(System.in);

int n = scn.nextInt(); //input the number

int d = scn.nextInt(); //input the digit

int x = countDigitFreq(n,d);

System.out.println("The digit " + d + " occurs " + x + " times in " + n);

}

}

### Write a Java Program to calculate xn (x to the power n) using Recursion.

import java.util.\*;

class Main {

public static double power(double x, int n) {

if(n == 0) return 1.0;

double xpnm1 = power(x,n-1); //x power n-1 (xpnm1)

return x \* xpnm1;

}

public static double pow(double x, int n) {

if(n < 0) {

return 1.0 / power(x,-n);

}

return power(x,n);

}

public static void main(String args[]) {

// Your code goes here

Scanner scn = new Scanner(System.in);

double x = scn.nextDouble();

int n = scn.nextInt();

System.out.println(pow(x,n));

}

}

### Write a program in Java to Toggle the case of every character of a string. For instance, if the input string is “ApPLe”, the output should be “aPplE”.

import java.util.\*;

class Main {

public static void main(String args[]) {

// Your code goes here

Scanner scn = new Scanner(System.in);

String str = scn.nextLine();

StringBuilder res = new StringBuilder("");

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

char ch = str.charAt(i); //current character

if(ch >='A' && ch <= 'Z') {

res.append((char)(ch + 32));

} else if(ch >='a' && ch<='z'){

res.append((char)(ch - 32));

} else {

res.append(ch);

}

}

String ans = res.toString();

System.out.println("The string after toggling becomes: " + ans);

}

}

### Write a program in Java to count the total number of vowels and consonants in a String.

import java.util.\*;

class Main {

public static boolean isVowel(char ch) {

if(ch == 'a' || ch =='e' || ch =='i' || ch =='o' || ch =='u')

return true;

return false;

}

public static void main(String args[]) {

// Your code goes here

Scanner scn = new Scanner(System.in);

String str = scn.nextLine();

int vowelCount = 0;

int consCount = 0;

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

char ch = str.charAt(i);

if(isVowel(ch) == true) vowelCount++;

else if(ch >='a' && ch<='z' && isVowel(ch) == false) consCount++;

}

System.out.println("Number of vowels are: " + vowelCount);

System.out.println("Number of consonants are: " + consCount);

System.out.println("Number of other characters are: " + (int)(str.length() - vowelCount -consCount));

}

}

**Java Code to Print All Unique Characters in a String.**

import java.util.\*;

class Main {

public static void main(String args[]) {

// Your code goes here

Scanner scn = new Scanner(System.in);

String str = scn.nextLine();

HashSet<Character> unique = new HashSet<>();

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

char ch = str.charAt(i);

if(unique.contains(ch) == true) {

//this character has already occured

unique.remove(ch);

} else {

unique.add(ch);

}

}

if(unique.size() == 0) {

System.out.println("There are no unique characters");

}

for(Character ch : unique) {

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

}

}

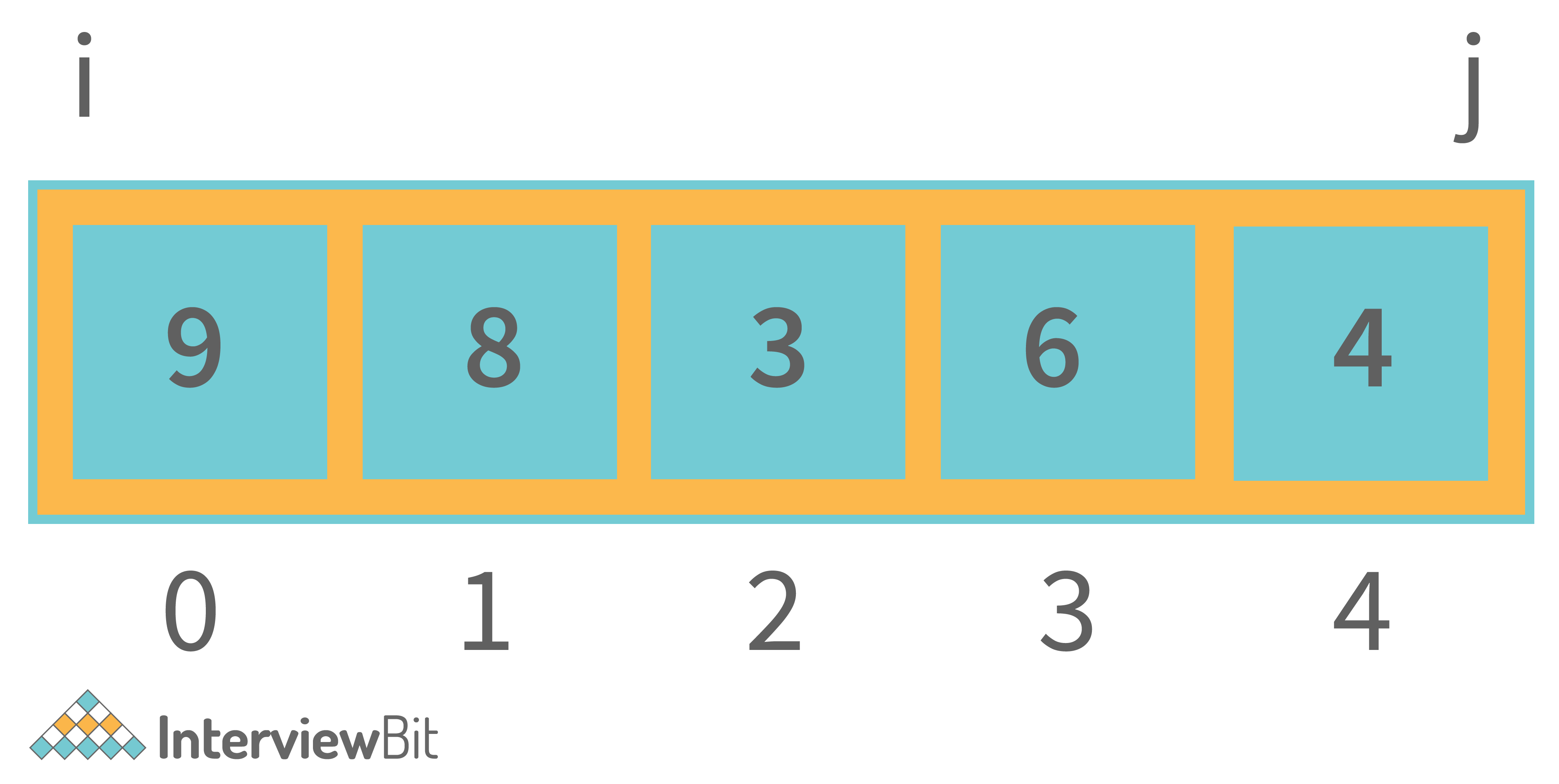
}

Input: abcab

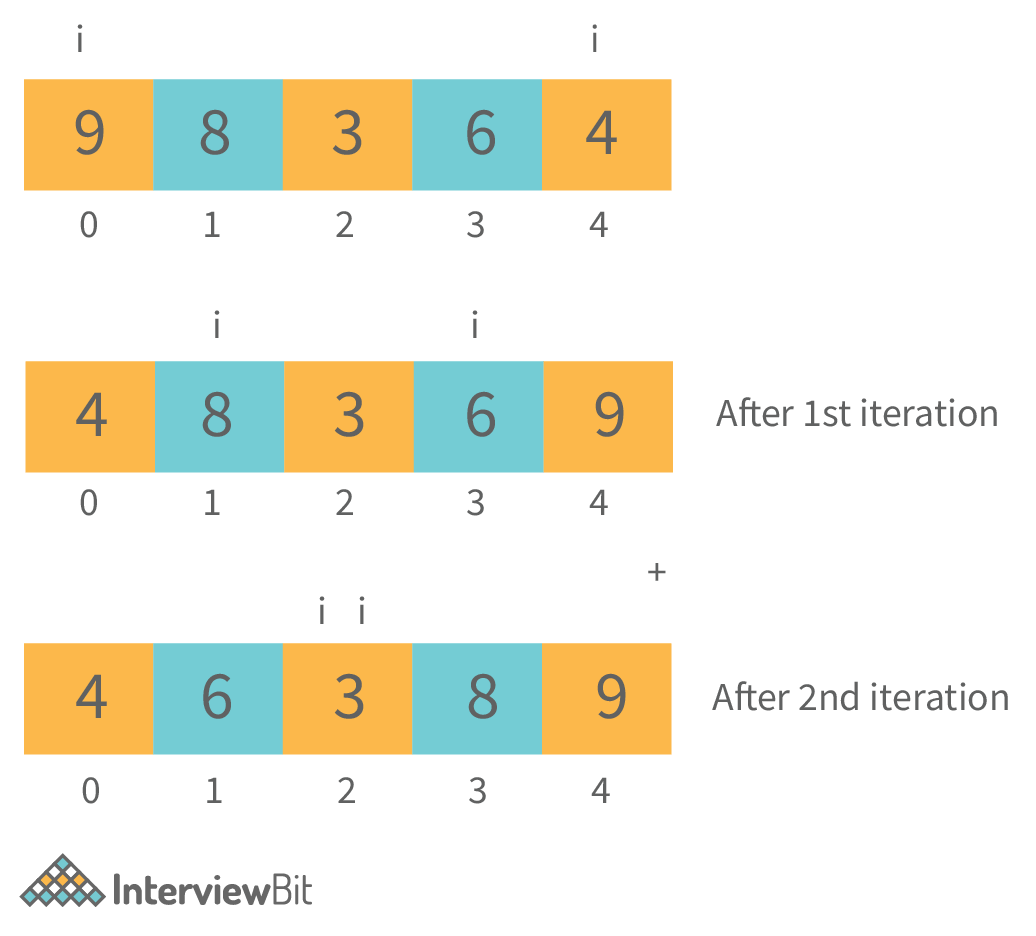
Output: c

**Write a program in Java to Reverse an Array without using extra space.**

We keep to variables i and j at both the ends of the array as shown below.



Now, swap the element at index i with the element at index j increment the i variable, and decrement the j variable. Keep doing this till i is less than j. This is shown in the image below.



import java.util.\*;

class Main {

public static void main(String args[]) {

// Your code goes here

Scanner scn = new Scanner(System.in);

int n = scn.nextInt();

int[] arr = new int[n];

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

arr[i] = scn.nextInt();

}

System.out.println("The reversed array is");

int i = 0;

int j = arr.length - 1;

while(i < j) {

//swapping ith and jth index elements

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

i++;

j--;

}

//displaying the array

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

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

}

}

}

### Write a program to find the index of first occurrence and last occurrence of an element in the array in a single iteration.

import java.util.\*;

class Main {

public static void main(String args[]) {

// Your code goes here

Scanner scn = new Scanner(System.in);

int n = scn.nextInt();

int[] arr = new int[n];

// input the array

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

arr[i] = scn.nextInt();

}

int target = scn.nextInt();

int fIndex = -1, lIndex = -1;

boolean found = false;

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

if(arr[i] == target) {

if(!found) {

fIndex = i;

lIndex = i;

found = true; //found for the first time

} else {

lIndex = i;

}

}

}

if(found == false) {

System.out.println("The element does not exist in the array");

} else {

System.out.println("First Index = " + fIndex + " Last Index = " + lIndex);

}

}

}

**Sample Output**

When element exists

Input:

5

1 2 3 2 5

2

Output: First Index = 1 Last Index = 3

### Write a program in Java to input an NxN matrix and display it row-wise and column-wise.

import java.util.\*;

class Main {

public static void main(String args[]) {

// Your code goes here

Scanner scn = new Scanner(System.in);

int N = scn.nextInt();

int[][] mat = new int[N][N];

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

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

mat[i][j] = scn.nextInt();

}

}

//Display Row wise

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

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

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

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

}

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

}

System.out.println();

//Display Col wise

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

System.out.print("Col " + j + " : ");

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

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

}

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

}

}

}

**Sample Output**

Input:

3

1 2 3

4 5 6

7 8 9

Output:

Row 0 : 1 2 3

Row 1 : 4 5 6

Row 2 : 7 8 9

Col 0 : 1 4 7

Col 1 : 2 5 8

Col 2 : 3 6 9

### Write a program in Java to show a basic “divide by 0 exception”.

Divide by zero exception occurs when we try to divide a number by 0 in Java. Following is the program showing divide by 0 exception.

import java.util.\*;

class Main {

public static void main(String args[]) {

// Your code goes here

Scanner scn = new Scanner(System.in);

int n = scn.nextInt();

System.out.println("Dividing this number by 0");

try {

System.out.println(n/0);

} catch(Exception e) {

System.out.println(e);

}

System.out.println("Program completed");

}

}

**Output**

Input: 3

Output:

Dividing this number by 0

java.lang.ArithmeticException: / by zero

Program completed

### Write a program to show a single thread in Java.

**Java Program to show Single Thread**

public class Main {

public static void main(String[] args) {

Thread t = Thread.currentThread();

t.setName("My Main Thread");

t.setPriority(7);

System.out.println(t);

System.out.println(t.getName());

System.out.println(t.getPriority());

}

}

**Output**

Thread[My Main Thread,7,main]

My Main Thread

7

### A sentence is said to be a palindrome if we convert all its alphabets to lowercase, include the numerics but exclude all the spaces, whitespaces, and other special characters and it reads the same from left to right and right to left.

**Java Code to check Palindromic Sentence**

import java.util.\*;

class Main {

public static boolean isStrPalindrome(String str) {

int lo = 0;

int hi = str.length()-1;

while(lo < hi) {

char ch1 = str.charAt(lo);

char ch2 = str.charAt(hi);

if(ch1 != ch2) return false;

lo++;

hi--;

}

return true;

}

public static boolean isSentencePalindrome(String s) {

String res = "";

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

char ch = s.charAt(i);

if((ch >='a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') || (ch >='0' && ch<='9')) {

if(ch >='A' && ch <= 'Z') res += (char)(ch + 32);

else res += ch;

} else continue;

}

if(isStrPalindrome(res)) return true;

return false;

}

public static void main(String args[]) {

// Your code goes here

Scanner scn = new Scanner(System.in);

String sentence = scn.nextLine();

if(isSentencePalindrome(sentence)) System.out.println(true);

else System.out.println(false);

}

}

### You are given 2 strings as input. You have to check whether they are anagrams or not.

### Anagrams are those strings that have the same characters occurring an equal number of times in both the strings. However, the order can be different. For example “anagram” and “nagrama” are Anagrams.

**Java Code to check Anagrams**

import java.util.\*;

class Main {

public static boolean isAnagram(String s1, String s2) {

if(s1.length() != s2.length()) return false;

HashMap<Character,Integer> fmap = new HashMap<>();

for(int i=0;i<s1.length();i++) {

int ofreq = fmap.getOrDefault(s1.charAt(i),0);

fmap.put(s1.charAt(i),ofreq+1);

}

for(int i=0;i<s2.length();i++) {

if(!fmap.containsKey(s2.charAt(i)) || fmap.get(s2.charAt(i)) == 0) {

return false;

} else {

int ofreq = fmap.get(s2.charAt(i));

fmap.put(s2.charAt(i),ofreq-1);

}

}

return true;

}

public static void main(String args[]) {

// Your code goes here

Scanner scn = new Scanner(System.in);

String str1 = scn.nextLine();

String str2 = scn.nextLine();

if(isAnagram(str1,str2)) System.out.println(true);

else System.out.println(false);

}

}

### Write a program in Java to create a user defined exception and also show it working means when it throws an exception.

Here, you have to explain and write a user-defined exception of your own. This code is just for reference purposes. So, we are going to create an exception called LowBalanceException for a bank. So, whenever a person comes to the bank to create a bank account, the minimum account balance should be 5000. So, if the balance is less than 5000, the exception will be thrown. Let us write the code for the same.

**Java Code for User-Defined Exception**

public class Main {

public static void main(String[] args) {

Account a1 = new Account(500);

Account a2 = new Account();

a2.setBalance(500);

Account a3 = new Account(10000);

System.out.println("a1 balance = " + a1.getBalance() + " a2 balance = " + a2.getBalance() + " a3 balance = " + a3.getBalance());

}

}

class Account {

private int balance;

Account() {

balance = 5000;

}

Account(int balance) {

try {

if(balance>=5000) {

this.balance = balance;

System.out.println("The account is created and the balance is set to: "+ balance);

} else {

this.balance=0;

System.out.println("Account can not be created");

throw new LowBalanceException();

}

} catch(LowBalanceException e) {

System.out.println(e);

}

}

void setBalance(int balance) {

try {

if(balance>=5000) {

this.balance = balance;

System.out.println("The account is created and the balance is set to: "+ balance);

} else {

this.balance=0;

System.out.println("Account can not be created");

throw new LowBalanceException();

}

} catch(LowBalanceException e) {

System.out.println(e);

}

}

int getBalance() {

return balance;

}

}

class LowBalanceException extends Exception {

public String toString() {

return "Low Balance: The balance cannot be less than Rs.5000/-";

}

}

**Output**

The account can not be created

Low Balance: The balance cannot be less than Rs.5000/-

The account can not be created

Low Balance: The balance cannot be less than Rs.5000/-

The account is created and the balance is set to 10000

a1 balance = 0 a2 balance = 0 a3 balance =10000

**Java Code for Multiple Inheritance**

public class Main {

public static void main(String[] args) {

SmartPhone sp1 = new SmartPhone();

Phone p1 = new SmartPhone();

ICamera c1 = new SmartPhone();

IMusicplayer m1 = new SmartPhone();

sp1.videocall();

p1.call();

p1.message();

c1.click();

c1.record();

m1.play();

m1.pause();

m1.stop();

}

}

class Phone {

void call() {

System.out.println("call");

}

void message() {

System.out.println("Message");

}

}

interface ICamera {

void click();

void record();

}

interface IMusicplayer {

void play();

void pause();

void stop();

}

class SmartPhone extends Phone implements ICamera, IMusicplayer {

void videocall() {

System.out.println("Video call");

}

@Override

public void click() {

System.out.println("Picture click");

}

@Override

public void record() {

System.out.println("Record video");

}

@Override

public void play() {

System.out.println("Play music");

}

@Override

public void pause() {

System.out.println("Pause Music");

}

@Override

public void stop() {

System.out.println("Stop music");

}

}

**Output**

Video call

Call

Message

Picture click

Record Video

Play music

Pause Music

Stop music

### Write a program in Java, to show nesting of classes.

Nesting of classes means writing a class inside another class. The inner classes in Java are usually static. This happens when we don’t want to use the variable of the outer class in the inner class. This helps to create an instance/object of the inner class without creating an instance of the Outer class. Following is a program to show the nesting of classes in Java.

**Java Code**

public class Main {

public static void main(String[] args) {

Outer obj1 = new Outer(10, 20);

Outer.Inner2 obj2 = new Outer.Inner2(40);

obj2.showData();

Outer.Inner3.z = 100;

System.out.println(Outer.Inner3.z);

}

}

class Outer {

private int x, y;

Outer() {

System.out.println("Outer class default constructor called");

}

Outer(int x, int y) {

this.x = x;

this.y = y;

}

void showData() {

System.out.println("the value of x is:" + x + " and the value of y is: " + y);

}

class Inner1 {

int z = 0;

Inner1() {

System.out.println("Inner class default constructor called");

}

Inner1(int z) {

this.z = z;

}

void showData() {

System.out.println("The value of x is: " + x + " the value of y is: " + y + " and z is: " + z);

}

}

static class Inner2 {

int z = 0;

Inner2() {

System.out.println("Inner class default constructor called");

}

Inner2(int z) {

this.z = z;

}

void showData() {

System.out.println("The value of z is: " + z);

}

}

static class Inner3 {

static int z = 0;

Inner3() {

System.out.println("Inner class default constructor called");

}

Inner3(int a) {

z = a;

}

void showData() {

System.out.println("The value of z is: " + z);

}

}

}

**Output**

The value of z is: 40

100

**Java Code to show isAlive() and join() operations**

class DemoThread extends Thread {

public DemoThread(String name) {

super(name);

setPriority(MAX\_PRIORITY);

}

}

class DemoThread2 extends Thread {

public void run() {

int count = 1;

while (true) {

System.out.println(count);

count++;

try {

Thread.sleep(100);

} catch (InterruptedException e) {

System.out.println(e);

}

}

}

}

public class Main {

public static void main(String[] args) {

DemoThread t = new DemoThread("Thread 1");

System.out.println("ID " + t.getId());

System.out.println("NAME " + t.getName());

System.out.println("PRIORITY " + t.getPriority());

t.start();

System.out.println("STATE " + t.getState());

System.out.println("ALIVE " + t.isAlive());

DemoThread2 t2 = new DemoThread2();

try {

Thread.sleep(100);

} catch (Exception e) {

}

t2.setDaemon(true);

t2.start();

// t2.interrupt();

Thread mainThread = Thread.currentThread();

try {

mainThread.join(); // Now main will not terminate till the daemon thread is terminated

} catch (Exception e) {

}

}

}

**Output**

ID 13

NAME Thread 1

PRIORITY 10

STATE RUNNABLE

ALIVE false

1

2

3

4

5

6

7

### Write a program in Java to show Thread Synchronization.

Here, you can create any 2 threads and synchronize them by using the **synchronized** keyword. An example program is given below.

**Java Program to show Thread Synchronization**

class Table {

public synchronized void display(int n) {

for (int i = 1; i <= 10; i++) {

System.out.println(n \* i);

}

}

}

class Thread1 extends Thread {

Table t;

public Thread1(Table t) {

this.t = t;

}

public void run() {

t.display(5);

}

}

class Thread2 extends Thread {

Table t;

public Thread2(Table t) {

this.t = t;

}

public void run() {

t.display(6);

}

}

public class Main {

public static void main(String[] args) {

Table table = new Table();

Thread1 th1 = new Thread1(table);

Thread2 th2 = new Thread2(table);

th1.start();

th2.start();

}

}

**Output**

5

10

15

20

25

30

35

40

45

50

6

12

18

24

30

36

42

48

54

60

## [How do you reverse a linked list in Java?](https://www.digitalocean.com/community/tutorials/java-programming-interview-questions#13-how-do-you-reverse-a-linked-list-in-java)

LinkedList descendingIterator() returns an iterator that iterates over the element in reverse order. The following example code shows how to use this iterator to create a new Linked List with elements listed in the reverse order:

LinkedList<Integer> ll = new LinkedList<>();

ll.add(1);

ll.add(2);

ll.add(3);

System.out.println(ll);

LinkedList<Integer> ll1 = new LinkedList<>();

ll.descendingIterator().forEachRemaining(ll1::add);

System.out.println(ll1);

### -----------------------------------------------------------------------------------------------

The array elements must be sorted to implement binary search. The binary search algorithm is based on the following conditions:

* If the key is less than the middle element, then you now need to search only in the first half of the array.
* If the key is greater than the middle element, then you need to search only in the second half of the array.
* If the key is equal to the middle element in the array, then the search ends.
* Finally, if the key is not found in the whole array, then it should return -1. This indicates that the element is not present.

The following example code implements a binary search:

public static int binarySearch(int arr[], int low, int high, int key) {

int mid = (low + high) / 2;

while (low <= high) {

if (arr[mid] < key) {

low = mid + 1;

} else if (arr[mid] == key) {

return mid;

} else {

high = mid - 1;

}

mid = (low + high) / 2;

}

if (low > high) {

return -1;

}

return -1;

}

### -------------------------------------------------------------------------------------

To check if two arrays contain the same elements, you need to first create a set of elements from both the arrays, and then compare the elements in these sets to find if there is an element that is not present in both sets. The following example code shows how to check if two arrays only contain common elements:

import java.util.Arrays;

import java.util.HashSet;

import java.util.Set;

public class ArraySameElements {

public static void main(String[] args) {

Integer[] a1 = {1,2,3,2,1};

Integer[] a2 = {1,2,3};

Integer[] a3 = {1,2,3,4};

System.out.println(sameElements(a1, a2));

System.out.println(sameElements(a1, a3));

}

static boolean sameElements(Object[] array1, Object[] array2) {

Set<Object> uniqueElements1 = new HashSet<>(Arrays.asList(array1));

Set<Object> uniqueElements2 = new HashSet<>(Arrays.asList(array2));

// if size is different, means there will be a mismatch

if (uniqueElements1.size() != uniqueElements2.size()) return false;

for (Object obj : uniqueElements1) {

// element not present in both?

if (!uniqueElements2.contains(obj)) return false;

}

return true;

}

}

Output

true

false

### ----------------------------------------------------------------------------------

The following example code shows how to use the Scanner class to read the file contents line by line and then use the String contains() method to check if the string is present in the file:

boolean findStringInFile(String filePath, String str) throws FileNotFoundException {

File file = new File(filePath);

Scanner scanner = new Scanner(file);

// read the file line by line

while (scanner.hasNextLine()) {

String line = scanner.nextLine();

if (line.contains(str)) {

scanner.close();

return true;

}

}

scanner.close();

return false;

}

### -------------------------------------------------------------------------------------

The following example code shows how to use the addAll() method to merge multiple lists in Java:

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

list1.add("1");

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

list2.add("2");

List<String> mergedList = new ArrayList<>(list1);

mergedList.addAll(list2);

System.out.println(mergedList); // [1, 2]

### --------------------------------------------------------------------------------------------

The String class doesn’t have a method to remove characters. The following example code shows how to use the replace() method to create a new string without the given character:

String str1 = "abcdABCDabcdABCD";

str1 = str1.replace("a", "");

System.out.println(str1); // bcdABCDbcdABCD

### String is immutable in Java. All the string manipulation methods return a new string, which is why you need to assign it to another variable.

### -----------------------------------------------------------------------------------------------

## [How do you get distinct characters and their count in a string in Java?](https://www.digitalocean.com/community/tutorials/java-programming-interview-questions#26-how-do-you-get-distinct-characters-and-their-count-in-a-string-in-java)

String str1 = "abcdABCDabcd";

char[] chars = str1.toCharArray();

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

for (char c : chars) {

if (charsCount.containsKey(c)) {

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

} else

charsCount.put(c, 1);

}

System.out.println(charsCount); // {a=2, A=1, b=2, B=1, c=2, C=1, d=2, D=1}

### ---------------------------------------------------------------------

## [Write a Java program to show a NullPointerException.](https://www.digitalocean.com/community/tutorials/java-programming-interview-questions#31-write-a-java-program-to-show-a-nullpointerexception)

public static void main(String[] args) {

printString(null, 3);

}

static void printString(String s, int count) {

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

System.out.println(s.toUpperCase()); // Exception in thread "main" java.lang.NullPointerException

}

}

### ---------------------------------------------------------------------------------------

An interface with exactly one abstract method is called a functional interface. The major benefit of functional interfaces is that you can use lambda expressions to instantiate them and avoid using bulky anonymous class implementation. The @FunctionalInterface annotation indicates a functional interface, as shown in the following example code:

@FunctionalInterface

interface Foo {

void test();

}

## [40. Show an example of using lambda expressions in Java.](https://www.digitalocean.com/community/tutorials/java-programming-interview-questions#40-show-an-example-of-using-lambda-expressions-in-java)

Runnable is an excellent example of a functional interface. You can use lambda expressions to create a runnable, as shown in the following example code:

Runnable r1 = () -> System.out.println("My Runnable");

## [41. Show examples of overloading and overriding in Java.](https://www.digitalocean.com/community/tutorials/java-programming-interview-questions#41-show-examples-of-overloading-and-overriding-in-java)

When a class has two or more methods with the same name, they are called overloaded methods. The following example code shows as overloaded method called print:

class Foo {

void print(String s) {

System.out.println(s);

}

void print(String s, int count) {

while (count > 0) {

System.out.println(s);

count--;

}

}

}

When a superclass method is also implemented in the child class, it’s called overriding. The following example code shows how to annotate the printname() method that’s implemented in both classes:

class Base {

void printName() {

System.out.println("Base Class");

}

}

class Child extends Base {

@Override

void printName() {

System.out.println("Child Class");

}

}