Ans 1 :

import java.util.\*;

import java.lang.Math;

public class Main{

public static void main(String[] args) {

Scanner ob = new Scanner(System.in);

int num;

int positive=0, negative=0, zero=0, odd=0, even=0;

for(int i=0; i< 20; ++i){

num = ob.nextInt();

if(num > 0)

positive++;

else if(num < 0)

negative++;

if(Math.abs(num) %2 == 0)

even++;

else

odd++;

if(num==0)

zero++;

}

System.out.println("Number of Positive Numbers : " + positive);

System.out.println("Number of Negative Numbers : " + negative);

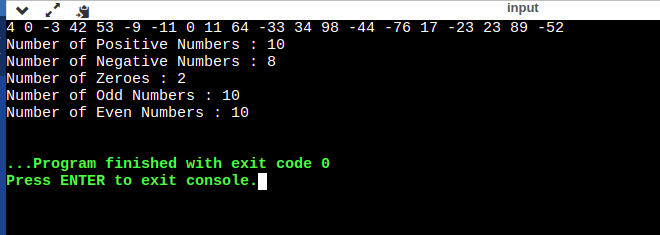
System.out.println("Number of Zeroes : " + zero);

System.out.println("Number of Odd Numbers : " + odd);

System.out.println("Number of Even Numbers : " + even);

}

}



Ans 2 :

import java.util.\*;

import java.lang.Math;

public class Main{

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int n = scan.nextInt();

int arr[] = new int[n];

int arr1[] = new int[n/2];

int arr2[] = new int[n/2];

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

arr[i] = scan.nextInt();

if(i<n/2)

arr1[i] = arr[i];

else

arr2[i -n/2] = arr[i];

}

System.out.print("First Array : " );

for(int i = 0; i < n/2; ++i)

System.out.print(arr1[i] + " ");

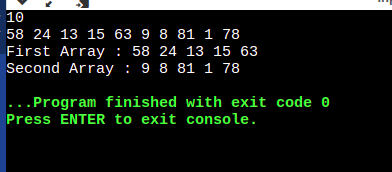
System.out.print("\nSecond Array : " );

for(int i = 0; i < n/2; ++i)

System.out.print(arr2[i] + " ");

}

}



Ans : 3

import java.util.\*;

import java.lang.Math;

class parent{

void parentmethod(){

System.out.println("This is parent method");

}

}

class child extends parent{

void childmethod(){

System.out.println("This is child method");

}

}

public class Main{

public static void main(String[] args) {

parent p = new parent();

child c = new child();

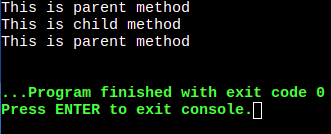
p.parentmethod();

c.childmethod();

c.parentmethod();

}

}



Ans 4 :

class Employee

{

private String name;

private int salary;

private String doj;

Employee (String name, int sal, String doj)

{

this.name = name;

this.salary = sal;

this.doj = doj;

}

void getEmployeeInfo ()

{

System.out.println ("Employee Name : " + this.name);

System.out.println ("Employee Salary : " + this.salary);

System.out.println ("Employee Date of joining : " + this.doj);

}

}

public class Main

{

public static void main (String[]args)

{

Employee arr[] = new Employee[10];

arr[0] = new Employee ("A", 20, "01-01-2023");

arr[1] = new Employee("B", 25, "01-01-2023");

arr[2] = new Employee("C", 22, "01-01-2023");

arr[0].getEmployeeInfo();

arr[1].getEmployeeInfo();

arr[2].getEmployeeInfo();

}

}

Ans 5 :

import java.lang.\*;

import java.util.\*;

public class Main

{

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

String str = scan.nextLine();

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

int len = words.length;

String ans = "";

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

if( i == len -1)

ans = ans + words[i];

else

ans = ans + (words[i].charAt(0) + ".");

}

System.out.println(ans);

}

}

Ans 6 :

We can use == operator for value as well as reference comparison while we can’t use equal method for reference comparison.

Ans 7 :

StringBuffer is thread safe as only one thread can use it one at a time while StringBuilder is not thread safe as it can be accessed from multiple threads at the same time.

StringBuffer is slower as compared to StringBuilder.

Ans 8 :

Final in variables : It makes the variable constant and once a value is assigned to the variable it can’t be changed.

Final in method : final methods can’t be overridden by subclasses.

Final in classes : final classes can’t be inherited

Ans 9 :

public class Main

{

public static void main(String[] args) {

try{

System.out.println("try block executed");

System.exit(0);

}

catch(Exception e){

System.out.println("Exception occurred!");

}

finally{

System.out.println("finally block executed!");

}

}

}

Ans 10 :

Shallow copy doesn’t make new object, instead it references to the original object.

Clone() method does shallow copy(on default).

On deletion of original object, shallow copy becomes useless.

Hard copy makes new object each time.

We need to override clone() method to make hard copy.

On deletion of original object, hard copy still works.

Ans 11 : Arithmetic Exception , as everything is non-divisible by 0.

Ans 12 : Because finally block will always be executed no matter what.

Ans 13 : ArrayIndexOutOfBounds Exception will be thrown.

Ans 14 :

A : 2

B : 1

C : 3

Ans 15 :

By using isEmpty() method, we can find if a string is empty or not.

str1.isEmpty();

Ans 16 :

import java.lang.\*;

import java.util.\*;

public class Main

{

public static void main(String[] args) {

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

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

ArrayList<Integer> list = new ArrayList<Integer>();

list1.add(5); list1.add(1);

list2.add(2); list2.add(9);

list.addAll(list1);

list.addAll(list2);

System.out.println(list);

}

}

Ans 17 : fill method can used as following :

import java.util.\*;

public class Main

{

public static void main (String[]args)

{

int arr[] = new int[8];

Arrays.fill(arr, 5);

for(int a:arr)

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

}

}

Ans 18 :

Generics offers type-safety in collections.

Ans 19)

comparable:-

1) It provide a single sorting sequence .For ex:- we can sort on the basis of single element such

as id,name or price.

2)Comparable provide campareTo() method to sort element.

3)we can sort the list element of comparable type by Collecions.sort(List)

comparator:-

1)It provide multiple sorting sequence .For ex:-we can sort on the basis of multiple element such as name,id and price.

2)comparator provide compare() method to sort element.

3)we sort the list element of Comparator type by Collections.sort(List,Comparator) method.

Ans 20 :

class userDefinedException extends Exception{

}

public class Main

{

public static void main(String[] args) {

try{

throw new userDefinedException();

}

catch(Exception e){

System.out.println("User Defined Exception has been caught.");

}

}

}

Ans 21 :

false true

Ans 22 :

class Member{

protected int age, salary;

protected String name, address, phoneno;

Member(int age, String phoneno, int salary, String name, String address){

this.age = age;

this.phoneno = phoneno;

this.salary = salary;

this.name = name;

this.address = address;

}

void printSalary(){

System.out.println("Salary : " + this.salary);

}

}

class Employee extends Member{

protected String specialization;

Employee(int age, String phoneno, int salary, String name, String address, String specialization){

super(age, phoneno, salary, name, address);

this.specialization = specialization;

}

void printInfo(){

System.out.println("Name : " + this.name );

System.out.println("Age : " + this.age);

System.out.println("Phone number : " + this.phoneno);

System.out.println("Address : " + this.address);

System.out.println("Salary : " + this.salary);

System.out.println("Specialization : " + this.specialization + "\n");

}

}

class Manager extends Member{

protected String department;

Manager(int age, String phoneno, int salary, String name, String address, String department){

super(age, phoneno, salary, name, address);

this.department = department;

}

void printInfo(){

System.out.println("Name : " + this.name );

System.out.println("Age : " + this.age);

System.out.println("Phone number : " + this.phoneno);

System.out.println("Address : " + this.address);

System.out.println("Salary : " + this.salary);

System.out.println("Department : " + this.department + "\n");

}

}

public class Main

{

public static void main(String[] args) {

Manager m = new Manager(35, "987865422", 200000, "Abc", "India", "Devops");

Employee e = new Employee(27, "9894678124", 80000, "Def", "India", "Frontend");

m.printInfo();

e.printInfo();

}

}