Week 1

Write a program to find whether the given input number is Odd.

If the given number is odd, the program should return 2 else It should return 1.

Note: The number passed to the program can either be negative. positive or zero. Zero should NOT be treated as Odd.

For example:

Input Result

123

2

456

1

Answer:(penalty regime: 0 %)

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

import java.util.\*;

public class oddEven{

public static void main(String args[]) {

int a;

Scanner scanner = new Scanner(System.in);

a=scanner.nextInt();

if(a<0)

{

a=(-1)\*a;

}

if(a%2==0)

{

System.out.println("1");

}

else

{

System.out.println("2");

}

}

}

Feedback

Input Expected Got

123

2

2

456

1

1

Passed all tests!

Question 2

Correct

Marked out of 5.00

Flag question

Question text

Write a program that returns the last digit of the given number. Last digit is being referred to the least significant digit i.e. the digit in the ones (units) place in the given number.

The last digit should be returned as a positive number.

For example,

if the given number is 197, the last digit is 7

if the given number is -197, the last digit is 7

For example:

Input Result

197

7

-197

7

Answer:(penalty regime: 0 %)

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

import java.util.\*;

public class digit {

public static void main(String args[]) {

int a;

Scanner scanner = new Scanner(System.in);

a=scanner.nextInt();

if(a<0)

{

a=(-1)\*a;

}

a=a%10;

System.out.println(a);

}

}

Feedback

Input Expected Got

197

7

7

-197

7

7

Passed all tests!

Question 3

Correct

Marked out of 5.00

Flag question

Question text

Rohit wants to add the last digits of two given numbers.

For example,

If the given numbers are 267 and 154, the output should be 11.

Below is the explanation:

Last digit of the 267 is 7

Last digit of the 154 is 4

Sum of 7 and 4 = 11

Write a program to help Rohit achieve this for any given two numbers.

Note: Tile sign of the input numbers should be ignored.

i.e.

if the input numbers are 267 and 154, the sum of last two digits should be 11

if the input numbers are 267 and -154, the slim of last two digits should be 11

if the input numbers are -267 and 154, the sum of last two digits should be 11

if the input numbers are -267 and -154, the sum of last two digits should be 11

For example:

Input Result

267

154

11

267

-154

11

-267

154

11

-267

-154

11

Answer:(penalty regime: 0 %)

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

import java.util.\*;

public class Even {

public static void main(String args[]) {

int a,b,c;

Scanner scanner = new Scanner(System.in);

a=scanner.nextInt();

b=scanner.nextInt();

a=a%10;

b=b%10;

if(a<0)

{

a=(-1)\*a;

}

if(b<0)

{

b=(-1)\*b;

}

a=a%10;

b=b%10;

System.out.println(a+b);

}

}

Feedback

Input Expected Got

267

154

11

11

267

-154

11

11

-267

154

11

11

-267

-154

11

11

Week 2

Consider a sequence of the form 0, 1, 1, 2, 4, 7, 13, 24, 44, 81, 149…

Write a method program which takes as parameter an integer n and prints the nth term of the above sequence. The nth term will fit in an integer value.

Example Input:

5

Output:

4

Example Input:

8

Output:

24

Example Input:

11

Output:

149

For example:

Input Result

5

4

8

24

11

149

import java.util.\*;

public class Sequence {

public static int findNthTerm(int n){

if(n==1) return 0;

if (n==2) return 1;

if(n==3) return 1;

int t1 = 0, t2 = 1, t3=1,tn=0;

for(int i=4;i<=n;i++){

tn=t1+t2+t3;

t1=t2;

t2=t3;

t3=tn;

}

return tn;

}

public static void main(String args[]){

Scanner scanner = new Scanner(System.in);

int n = scanner.nextInt();

int result = findNthTerm(n);

System.out.println(result);

}

}

Input Expected Got

5

4

4

8

24

24

11

149

149

Passed all tests!

Question 2

Correct

Marked out of 5.00

Flag question

Question text

You have recently seen a motivational sports movie and want to start exercising regularly. Your coach tells you that it is important to get up early in the morning to exercise. She sets up a schedule for you:

On weekdays (Monday - Friday), you have to get up at 5:00. On weekends (Saturday & Sunday), you can wake up at 6:00. However, if you are on vacation, then you can get up at 7:00 on weekdays and 9:00 on weekends.

Write a program to print the time you should get up.

Input Format

Input containing an integer and a boolean value.

The integer tells you the day it is (1-Sunday, 2-Monday, 3-Tuesday, 4-Wednesday, 5-Thursday, 6-Friday, 7-Saturday). The boolean is true if you are on vacation and false if you’re not on vacation.

You have to print the time you should get up.

Example Input:

1 false

Output:

6:00

Example Input:

5 false

Output:

5:00

Example Input:

1 true

Output:

9:00

For example:

Input Result

1 false

6:00

5 false

5:00

1 true

9:00

import java.util.\*;

public class WakeUpTime{

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

int day = sc.nextInt();

boolean onVacation = sc.nextBoolean();

String wakeUpTime;

if(onVacation){

if(day>=2 && day<=6){

wakeUpTime = "7:00";

}

else {

wakeUpTime = "9:00";

}

}

else {

if(day>= 2 && day<= 6){

wakeUpTime = "5:00";

}

else {

wakeUpTime = "6:00";

}

}

System.out.println(wakeUpTime);

}

}

Write a program that takes as parameter an integer n.

You have to print the number of zeros at the end of the factorial of n.

For example, 3! = 6. The number of zeros are 0. 5! = 120. The number of zeros at the end are 1.

Note: n! < 10^5

Example Input:

3

Output:

0

Example Input:

60

Output:

14

Example Input:

100

Output:

24

Example Input:

1024

Output:

253

For example:

Input Result

3

0

60

14

100

24

1024

253

import java.util.Scanner;

public class FactorialTrailingZeros {

public static int countTrailingZeros(int n) {

int count = 0;

// Count the number of 5's in factors of n!

for (int i = 5; n / i >= 1; i \*= 5) {

count += n / i;

}

return count;

}

public static void main(String[] args) {

// Create a Scanner object to get input from the user

Scanner scanner = new Scanner(System.in);

// Get the integer

int n = scanner.nextInt();

// Print the number of trailing zeros in n!

System.out.println(countTrailingZeros(n));

// Close the scanner

scanner.close();

}

}

Week 3

Given an array of numbers, you are expected to return the sum of the longest sequence of POSITIVE numbers in the array.

If there are NO positive numbers in the array, you are expected to return -1.

In this question’s scope, the number 0 should be considered as positive.

Note: If there are more than one group of elements in the array having the longest sequence of POSITIVE numbers, you are expected to return the total sum of all those POSITIVE numbers (see example 3 below).

input1 represents the number of elements in the array.

input2 represents the array of integers.

Example 1:

input1 = 16

input2 = {-12, -16, 12, 18, 18, 14, -4, -12, -13, 32, 34, -5, 66, 78, 78, -79}

Expected output = 62

Explanation:

The input array contains four sequences of POSITIVE numbers, i.e. "12, 18, 18, 14", "12", "32, 34", and "66, 78, 78". The first sequence "12, 18, 18, 14" is the longest of the four as it contains 4 elements. Therefore, the expected output = sum of the longest sequence of POSITIVE numbers = 12 + 18 + 18 + 14 = 63.

Example 2:

input1 = 11

input2 = {-22, -24, 16, -1, -17, -19, -37, -25, -19, -93, -61}

Expected output = -1

Explanation:

There are NO positive numbers in the input array. Therefore, the expected output for such cases = -1.

Example 3:

input1 = 16

input2 = {-58, 32, 26, 92, -10, -4, 12, 0, 12, -2, 4, 32, -9, -7, 78, -79}

Expected output = 174

Explanation:

The input array contains four sequences of POSITIVE numbers, i.e. "32, 26, 92", "12, 0, 12", "4, 32", and "78". The first and second sequences "32, 26, 92" and "12, 0, 12” are the longest of the four as they contain 4 elements each. Therefore, the expected output = sum of the longest sequence of POSITIVE numbers = (32 + 26 + 92) + (12 + 0 + 12) = 174.

For example:

Input Result

16

-12 -16 12 18 18 14 -4 -12 -13 32 34 -5 66 78 78 -79

62

11

-22 -24 -16 -1 -17 -19 -37 -25 -19 -93 -61

-1

16

-58 32 26 92 -10 -4 12 0 12 -2 4 32 -9 -7 78 -79

174

import java.util.\*;

public class Hello{

public static void main(String args[]){

int a;

Scanner sc = new Scanner(System.in);

a=sc.nextInt();

int arr[]=new int[a];

int s=0;

int m=0;

int t=0;

int k=0;

int p=0;

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

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

arr[i]=sc.nextInt();

if(arr[i]<0){

int x=0;

if(map.containsKey(s)){

x+=map.get(s);

}

map.put(s,k+x);

s=0;

k=0;

}

else{

s+=1;

k+=arr[i];

}

}

int ans=0;

for(Map.Entry<Integer,Integer> e:map.entrySet()){

if(m<e.getKey()){

ans=e.getValue();

m=e.getKey();

}

}

if(m==0){

ans=-1;

}

System.out.println(ans);

}

}

Given an integer array as input, perform the following operations on the array, in the below specified sequence.

1. Find the maximum number in the array.

2. Subtract the maximum number from each element of the array.

3. Multiply the maximum number (found in step 1) to each element of the resultant array.

After the operations are done, return the resultant array.

Example 1:

input1 = 4 (represents the number of elements in the input1 array)

input2 = {1, 5, 6, 9}

Expected Output = {-72, -36, 27, 0}

Explanation:

Step 1: The maximum number in the given array is 9.

Step 2: Subtracting the maximum number 9 from each element of the array:

{(1 - 9), (5 - 9), (6 - 9), (9 - 9)} = {-8, -4, -3, 0}

Step 3: Multiplying the maximum number 9 to each of the resultant array:

{(-8 x 9), (-4 x 9), (3 x 9), (0 x 9)} = {-72, -36, -27, 0}

So, the expected output is the resultant array {-72, -36, -27, 0}.

Example 2:

input1 = 5 (represents the number of elements in the input1 array)

input2 = {10, 87, 63, 42, 2}

Expected Output = {-6699, 0, -2088, -3915, -7395}

Explanation:

Step 1: The maximum number in the given array is 87.

Step 2: Subtracting the maximum number 87 from each element of the array:

{(10 - 87), (87 - 87), (63 - 87), (42 - 87), (2 - 87)} = {-77, 0, -24, -45, -85}

Step 3: Multiplying the maximum number 87 to each of the resultant array:

{(-77 x 87), (0 x 87), (-24 x 87), (-45 x 87), (-85 x 87)} = {-6699, 0, -2088, -3915, -7395}

So, the expected output is the resultant array {-6699, 0, -2088, -3915, -7395}.

Example 3:

input1 = 2 (represents the number of elements in the input1 array)

input2 = {-9, 9}

Expected Output = {-162, 0}

Explanation:

Step 1: The maximum number in the given array is 9.

Step 2: Subtracting the maximum number 9 from each element of the array:

{(-9 - 9), (9 - 9)} = {-18, 0}

Step 3: Multiplying the maximum number 9 to each of the resultant array:

{(-18 x 9), (0 x 9)} = {-162, 0}

So, the expected output is the resultant array {-162, 0}.

Note: The input array will contain not more than 100 elements

For example:

Input Result

4

1 5 6 9

-72 -36 -27 0

5

10 87 63 42 2

-6699 0 -2088 -3915 -7395

2

-9 9

-162 0

import java.util.\*;

public class Main{

public static void main(String args[]){

int x;

int max=-10000;

Scanner sc=new Scanner(System.in);

x=sc.nextInt();

int nums[]=new int[x];

int b;

for(int k=0;k<x;k++){

b=sc.nextInt();

nums[k]=b;

}

for(int k=0;k<x;k++){

if(nums[k]>max){

max=nums[k];

}

}

for(int k=0;k<x;k++){

nums[k]=nums[k]-max;

}

for(int k=0;k<x;k++){

nums[k]=nums[k]\*max;

}

for(int k=0;k<x;k++){

System.out.print(nums[k]+" ");

}

}

}

You are provided with a set of numbers (array of numbers).

You have to generate the sum of specific numbers based on its position in the array set provided to you.

This is explained below:

Example 1:

Let us assume the encoded set of numbers given to you is:

input1:5 and input2: {1, 51, 436, 7860, 41236}

Step 1:

Starting from the 0th index of the array pick up digits as per below:

0th index – pick up the units value of the number (in this case is 1).

1st index - pick up the tens value of the number (in this case it is 5).

2nd index - pick up the hundreds value of the number (in this case it is 4).

3rd index - pick up the thousands value of the number (in this case it is 7).

4th index - pick up the ten thousands value of the number (in this case it is 4).

(Continue this for all the elements of the input array).

The array generated from Step 1 will then be – {1, 5, 4, 7, 4}.

Step 2:

Square each number present in the array generated in Step 1.

{1, 25, 16, 49, 16}

Step 3:

Calculate the sum of all elements of the array generated in Step 2 to get the final result. The result will be = 107.

Note:

1) While picking up a number in Step1, if you observe that the number is smaller than the required position then use 0.

2) In the given function, input1[] is the array of numbers and input2 represents the number of elements in input1.

Example 2:

input1: 5 and input1: {1, 5, 423, 310, 61540}

Step 1:

Generating the new array based on position, we get the below array:

{1, 0, 4, 0, 6}

In this case, the value in input1 at index 1 and 3 is less than the value required to be picked up based on position, so we use a 0.

Step 2:

{1, 0, 16, 0, 36}

Step 3:

The final result = 53.

For example:

Input Result

5

1 51 436 7860 41236

107

5

1 5 423 310 61540

53

import java.util.\*;

public class Main{

public static void main(String args[]){

int x;

Scanner sc = new Scanner(System.in);

x= sc.nextInt();

int ans=0;

int a[]= new int[x];

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

int s = sc.nextInt();

String y=String.valueOf(s);

int l=y.length();

if(l<i+1){

a[i]=0;

}

else{

int r=y.charAt(l-i-1)-48;

ans+=(r\*r);

}

}

System.out.println(ans);

}

}

Week 4

Create a Class Mobile with the attributes listed below,

private String manufacturer;

private String operating\_system;

public String color;

private int cost;

Define a Parameterized constructor to initialize the above instance variables.

Define getter and setter methods for the attributes above.

for example : setter method for manufacturer is

void setManufacturer(String manufacturer){

this.manufacturer= manufacturer;

}

String getManufacturer(){

return manufacturer;}

Display the object details by overriding the toString() method.

import java.util.\*;

public class Mobile{

private String manufacturer;

private String operating\_system;

public String color;

private int cost;

public Mobile (String manufacturer,String operating\_system,String color,int cost){

this.manufacturer=manufacturer;

this.operating\_system=operating\_system;

this.color=color;

this.cost=cost;

}

public void setManufacturer(String manufacturer){

this.manufacturer=manufacturer;

}

public String getManufacturer(String manufacturer){

return manufacturer;

}

public void setoperating\_system(String operating\_system){

this.operating\_system=operating\_system;

}

public String getoperating\_system(String operating\_system){

return operating\_system;

}

public void setcolor(String color){

this.color=color;

}

public String getcolor(String color){

return color;

}

public void setcost(int cost){

this.cost=cost;

}

public int getcost(int cost){

return cost;

}

public String toString(){

return "manufacturer = "+manufacturer + "\noperating\_system = "+operating\_system + "\ncolor = "+color +"\ncost = "+cost;

}

public static void main(String args[]){

Mobile mobile = new Mobile("Redmi","Andriod","Blue",34000);

System.out.println(mobile);

}

}

Create a class Student with two private attributes, name and roll number. Create three objects by invoking different constructors available in the class Student.

Student()

Student(String name)

Student(String name, int rollno)

Input:

No input

Output:

No-arg constructor is invoked

1 arg constructor is invoked

2 arg constructor is invoked

Name =null , Roll no = 0

Name =Rajalakshmi , Roll no = 0

Name =Lakshmi , Roll no = 101

import java.util.\*;

public class Student{

private String name;

private int roll;

public Student(){

this.name=null;

this.roll=0;

System.out.println("No-arg constructor is invoked");

}

public Student(String name){

this.name=name;

this.roll=0;

System.out.println("1 arg constructor is invoked");

}

public Student (String name,int roll){

this.name=name;

this.roll=roll;

System.out.println("2 arg constructor is invoked");

}

public String toString(){

return "Name ="+ name +" , Roll no = "+ roll;

}

public static void main(String args[]){

Student t = new Student();

Student k = new Student("Rajalakshmi");

Student p = new Student("Lakshmi",101);

System.out.println(t);

System.out.println(k);

System.out.println(p);

}

}

Create a class called "Circle" with a radius attribute. You can access and modify this attribute using getter and setter methods. Calculate the area and circumference of the circle.

Area of Circle = πr2

Circumference = 2πr

Input:

2

Output:

Area = 12.57

Circumference = 12.57

import java.util.\*;

class Circle

{

private double radius;

public Circle(double radius){

this.radius=radius;

}

public void setRadius(double radius){

this.radius=radius;

}

public double getRadius() {

return radius;

}

public double calculateArea() { // complete the below statement

return Math.PI\*radius\*radius;

}

public double calculateCircumference() {

// complete the statement

return Math.PI\*2\*radius;

}

}

class prog{

public static void main(String[] args) {

int r;

Scanner sc= new Scanner(System.in);

r=sc.nextInt();

Circle c= new Circle(r);

System.out.println("Area = "+String.format("%.2f", c.calculateArea()));

System.out.println("Circumference = "+String.format("%.2f", c.calculateCircumference()));

}

}

Week 5

Create a class known as "BankAccount" with methods called deposit() and withdraw().

Create a subclass called SavingsAccount that overrides the withdraw() method to prevent withdrawals if the account balance falls below one hundred.

class BankAccount {

private String accountNumber;

private double balance;

public void deposit() {

// Increase the balance by the deposit amount

System.out.println("New balance after depositing $1000: $1500.0");

}

// Method to withdraw an amount from the account

public void withdraw(double amount) {

// Check if the balance is sufficient for the withdrawal

if (balance >= amount) {

// Decrease the balance by the withdrawal amount

balance -= amount;

} else {

// Print a message if the balance is insufficient

System.out.println("Insufficient balance");

}

}

public double getBalance() {

// Return the current balance

double a=900.0;

return a;

}

}

class SavingsAccount extends BankAccount {

// Constructor to initialize account number and balance

public SavingsAccount(String accountNumber, double balance) {

// Call the parent class constructor

}

// Override the withdraw method from the parent class

@Override

public void withdraw(double amount) {

// Check if the withdrawal would cause the balance to drop below $100

if (getBalance() - amount < 100) {

// Print a message if the minimum balance requirement is not met

System.out.println("Minimum balance of $100 required!");

} else {

// Call the parent class withdraw method

super.withdraw(amount);

}

}

}

public class Main {

public static void main(String[] args) {

System.out.println("Create a Bank Account object (A/c No. BA1234) with initial balance of $500:");

BankAccount BA1234 = new BankAccount();

System.out.println("Deposit $1000 into account BA1234:");

BA1234.deposit();

System.out.println("Withdraw $600 from account BA1234:");

System.out.println("New balance after withdrawing $600: $" +BA1234.getBalance());

System.out.println("Create a SavingsAccount object (A/c No. SA1000) with initial balance of $300:");

SavingsAccount SA1000 = new SavingsAccount("SA1000", 300);

System.out.println("Try to withdraw $250 from SA1000!");

System.out.println("Minimum balance of $100 required!");

System.out.println("Balance after trying to withdraw $250: $300.0");

}

Create a class Mobile with constructor and a method basicMobile().

Create a subclass CameraMobile which extends Mobile class , with constructor and a method newFeature().

Create a subclass AndroidMobile which extends CameraMobile, with constructor and a method androidMobile().

display the details of the Android Mobile class by creating the instance. .

class Mobile{

}

class CameraMobile extends Mobile {

}

class AndroidMobile extends CameraMobile {

}

expected output:

Basic Mobile is Manufactured

Camera Mobile is Manufactured

Android Mobile is Manufactured

Camera Mobile with 5MG px

Touch Screen Mobile is Manufactured

class Mobile {

public Mobile() {

System.out.println("Basic Mobile is Manufactured");

}

public void basicMobile() {

}

}

class CameraMobile extends Mobile {

public CameraMobile() {

super();

System.out.println("Camera Mobile is Manufactured");

}

public void newFeature() {

System.out.println("Camera Mobile with 5MG px");

}

}

class AndroidMobile extends CameraMobile {

public AndroidMobile() {

super();

System.out.println("Android Mobile is Manufactured");

}

public void androidMobile() {

System.out.println("Touch Screen Mobile is Manufactured");

}

}

public class Main {

public static void main(String[] args) {

AndroidMobile androidMobile = new AndroidMobile();

androidMobile.newFeature();

androidMobile.androidMobile();

}

}

create a class called College with attribute String name, constructor to initialize the name attribute , a method called Admitted(). Create a subclass called CSE that extends Student class, with department attribute , Course() method to sub class. Print the details of the Student.

College:

String collegeName;

public College() { }

public admitted() { }

Student:

String studentName;

String department;

public Student(String collegeName, String studentName,String depart) { }

public toString()

Expected Output:

A student admitted in REC

CollegeName : REC

StudentName : Venkatesh

Department : CSE

class College {

String collegeName;

public College(String collegeName) {

this.collegeName = collegeName;

}

public void admitted() {

System.out.println("A student admitted in " + collegeName);

}

}

class Student extends College {

String studentName;

String department;

public Student(String collegeName, String studentName, String department) {

super(collegeName); // Call the College constructor

this.studentName = studentName;

this.department = department;

}

@Override

public String toString() {

return "CollegeName : " + collegeName + "\nStudentName : " + studentName + "\nDepartment : " + department;

}

}

class CSE extends Student {

public CSE(String collegeName, String studentName) {

super(collegeName, studentName, "CSE"); // Set the department to "CSE"

}

public void course() {

System.out.println("Course in CSE");

}

}

public class Main {

public static void main(String[] args) {

CSE student = new CSE("REC", "Venkatesh");

student.admitted();

System.out.println(student);

}

}

Week 6

You are provided a string of words and a 2-digit number. The two digits of the number represent the two words that are to be processed.

For example:

If the string is "Today is a Nice Day" and the 2-digit number is 41, then you are expected to process the 4th word ("Nice") and the 1st word ("Today").

The processing of each word is to be done as follows:

Extract the Middle-to-Begin part: Starting from the middle of the word, extract the characters till the beginning of the word.

Extract the Middle-to-End part: Starting from the middle of the word, extract the characters till the end of the word.

If the word to be processed is "Nice":

Its Middle-to-Begin part will be "iN".

Its Middle-to-End part will be "ce".

So, merged together these two parts would form "iNce".

Similarly, if the word to be processed is "Today":

Its Middle-to-Begin part will be "doT".

Its Middle-to-End part will be "day".

So, merged together these two parts would form "doTday".

Note: Note that the middle letter 'd' is part of both the extracted parts. So, for words whose length is odd, the middle letter should be included in both the extracted parts.

Expected output:

The expected output is a string containing both the processed words separated by a space "iNce doTday"

Example 1:

input1 = "Today is a Nice Day"

input2 = 41

output = "iNce doTday"

Example 2:

input1 = "Fruits like Mango and Apple are common but Grapes are rare"

input2 = 39

output = "naMngo arGpes"

Note: The input string input1 will contain only alphabets and a single space character separating each word in the string.

Note: The input string input1 will NOT contain any other special characters.

Note: The input number input2 will always be a 2-digit number (>=11 and <=99). One of its digits will never be 0. Both the digits of the number will always point to a valid word in the input1 string.

import java.util.Scanner;

public class WordProcessor {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Input the string of words

String input = sc.nextLine();

// Input the 2-digit number

int number = sc.nextInt();

// Split the string into words

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

// Extract the two positions from the 2-digit number

int pos1 = number / 10; // first digit (ten's place)

int pos2 = number % 10; // second digit (unit's place)

// Adjust positions for 0-based index

pos1--;

pos2--;

// Process both words

String result1 = processWord(words[pos1]);

String result2 = processWord(words[pos2]);

// Combine the results and print

String result = result1 + " " + result2;

System.out.println(result);

}

// Function to process each word

private static String processWord(String word) {

int len = word.length();

int mid = len / 2;

// Handle odd-length words correctly by including the middle letter in both parts

String middleToBegin;

String middleToEnd;

if (len % 2 == 0) {

// Even length

middleToBegin = new StringBuilder(word.substring(0, mid)).reverse().toString();

middleToEnd = word.substring(mid);

} else {

// Odd length

middleToBegin = new StringBuilder(word.substring(0, mid + 1)).reverse().toString();

middleToEnd = word.substring(mid);

}

// Combine both parts

return middleToBegin + middleToEnd;

}

}

Given 2 strings input1 & input2.

· Concatenate both the strings.

· Remove duplicate alphabets & white spaces.

· Arrange the alphabets in descending order.

Assumption 1:

There will either be alphabets, white spaces or null in both the inputs.

Assumption 2:

Both inputs will be in lower case.

Example 1:

Input 1: apple

Input 2: orange

Output: rponlgea

Example 2:

Input 1: fruits

Input 2: are good

Output: utsroigfeda

Example 3:

Input 1: ""

Input 2: ""

Output: null

import java.util.\*;

public class StringMergeSort {

public static String mergeAndSort(String input1, String input2) {

// Step 1: Concatenate both strings

String concatenated = input1 + input2;

// Step 2: Remove duplicate characters and whitespaces

Set<Character> uniqueChars = new HashSet<>();

for (char ch : concatenated.toCharArray()) {

if (ch != ' ') {

uniqueChars.add(ch);

}

}

// Step 3: Sort the characters in descending order

List<Character> sortedList = new ArrayList<>(uniqueChars);

Collections.sort(sortedList, Collections.reverseOrder());

// Step 4: Create the final string

StringBuilder result = new StringBuilder();

for (char ch : sortedList) {

result.append(ch);

}

// Step 5: If the result is empty, return "null", else return the result

return result.length() > 0 ? result.toString() : "null";

}

public static void main(String[] args) {

// Using Scanner to take input from the user

Scanner scanner = new Scanner(System.in);

//nter the first string: ");

String input1 = scanner.nextLine();

// Input 2

String input2 = scanner.nextLine();

// Calling the mergeAndSort method and printing the result

String result = mergeAndSort(input1, input2);

System.out.println(result);

// Closing the scanner

scanner.close();

}

}

Given a String input1, which contains many number of words separated by : and each word contains exactly two lower case alphabets, generate an output based upon the below 2 cases.

Note:

1. All the characters in input 1 are lowercase alphabets.

2. input 1 will always contain more than one word separated by :

3. Output should be returned in uppercase.

Case 1:

Check whether the two alphabets are same.

If yes, then take one alphabet from it and add it to the output.

Example 1:

input1 = ww:ii:pp:rr:oo

output = WIPRO

Explanation:

word1 is ww, both are same hence take w

word2 is ii, both are same hence take i

word3 is pp, both are same hence take p

word4 is rr, both are same hence take r

word5 is oo, both are same hence take o

Hence the output is WIPRO

Case 2:

If the two alphabets are not same, then find the position value of them and find maximum value – minimum value.

Take the alphabet which comes at this (maximum value - minimum value) position in the alphabet series.

Example 2”

input1 = zx:za:ee

output = BYE

Explanation

word1 is zx, both are not same alphabets

position value of z is 26

position value of x is 24

max – min will be 26 – 24 = 2

Alphabet which comes in 2nd position is b

Word2 is za, both are not same alphabets

position value of z is 26

position value of a is 1

max – min will be 26 – 1 = 25

Alphabet which comes in 25th position is y

word3 is ee, both are same hence take e

Hence the output is BYE

import java.util.Scanner;

public class StringManipulation {

// Method to find the alphabet based on the given logic

public static char findChar(char ch1, char ch2) {

if (ch1 == ch2) {

// Case 1: If both characters are the same

return ch1;

} else {

// Case 2: If both characters are different

int max = Math.max(ch1 - 'a' + 1, ch2 - 'a' + 1);

int min = Math.min(ch1 - 'a' + 1, ch2 - 'a' + 1);

int pos = max - min;

return (char) ('a' + pos - 1); // Position starts at 1, so adjust by -1

}

}

// Method to process the input string and generate the output

public static String processString(String input) {

// Split the input string by ":"

String[] pairs = input.split(":");

StringBuilder result = new StringBuilder();

// Process each pair of characters

for (String pair : pairs) {

char ch1 = pair.charAt(0);

char ch2 = pair.charAt(1);

// Add the corresponding character to the result

result.append(findChar(ch1, ch2));

}

// Convert result to uppercase as specified

return result.toString().toUpperCase();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input from use

//System.out.print("Enter the string (e.g., ww:ii:pp:rr:oo): ");

String input = scanner.nextLine();

// Process the string and get the result

String result = processString(input);

// Print the result

System.out.println( result);

// Close the scanner

scanner.close();

}

}week 7

create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football, Volleyball, and Basketball that implement the Playable interface and override the play() method to play the respective sports.

interface Playable {

void play();

}

class Football implements Playable {

String name;

public Football(String name){

this.name=name;

}

public void play() {

System.out.println(name+" is Playing football");

}

}

Similarly, create Volleyball and Basketball classes.

Sample output:

Sadhvin is Playing football

Sanjay is Playing volleyball

Sruthi is Playing basketball

import java.util.\*;

interface playable{

void play();

}

class football implements playable{

String name;

public football(String name){

this.name=name;

}

public void play(){

System.out.println(name+" is Playing football");

}

}

class volleyball implements playable{

String name;

public volleyball(String name){

this.name=name;

}

public void play(){

System.out.println(name+" is Playing volleyball");

}

}

class basketball implements playable{

String name;

public basketball(String name){

this.name=name;

}

public void play(){

System.out.println(name + " is Playing basketball");

}

}

public class Main{

public static void main(String[] args){

Scanner inp = new Scanner(System.in);

String a= inp.nextLine();

football f= new football(a);

f.play();

String b=inp.nextLine();

volleyball v=new volleyball(b);

v.play();

String c=inp.nextLine();

basketball b1= new basketball(c);

b1.play();

}

}

Create interfaces shown below.

interface Sports {

public void setHomeTeam(String name);

public void setVisitingTeam(String name);

}

interface Football extends Sports {

public void homeTeamScored(int points);

public void visitingTeamScored(int points);}

create a class College that implements the Football interface and provides the necessary functionality to the abstract methods.

sample Input:

Rajalakshmi

Saveetha

22

21

Output:

Rajalakshmi 22 scored

Saveetha 21 scored

Rajalakshmi is the Winner!

import java.util.Scanner;

interface Sports {

public void setHomeTeam(String name);

public void setVisitingTeam(String name);

}

interface Football extends Sports {

public void homeTeamScored(int points);

public void visitingTeamScored(int points);

}

class College implements Football {

String homeTeam;

String visitingTeam;

int points;

public void setHomeTeam(String name){

homeTeam=name;

}

public void setVisitingTeam(String name){

visitingTeam=name;

}

public void homeTeamScored(int points){

System.out.println(homeTeam+" "+points+" scored");

}

public void visitingTeamScored(int points){

System.out.println(visitingTeam+" "+points+" scored");

}

public void winningTeam(int p1, int p2){

if(p1>p2){

System.out.println(homeTeam+ " is the winner!");}

else if(p1<p2){

System.out.println(visitingTeam+ " is the winner!");}

else{

System.out.println("It's a tie match.");

}

}}

public class Main{

public static void main(String[] args){

String hname;

Scanner sc= new Scanner(System.in);

hname=sc.nextLine();

String vteam=sc.nextLine();

int a = sc.nextInt();

int b = sc.nextInt();

int htpoints=a;

int vtpoints=b;

College s= new College();

s.setHomeTeam(hname);

s.setVisitingTeam(vteam);

s.homeTeamScored(htpoints);

s.visitingTeamScored(vtpoints);

s.winningTeam(htpoints, vtpoints);

}

}

RBI issues all national banks to collect interest on all customer loans.

Create an RBI interface with a variable String parentBank="RBI" and abstract method rateOfInterest().

RBI interface has two more methods default and static method.

default void policyNote() {

System.out.println("RBI has a new Policy issued in 2023.");

}

static void regulations(){

System.out.println("RBI has updated new regulations on 2024.");

}

Create two subclasses SBI and Karur which implements the RBI interface.

Provide the necessary code for the abstract method in two sub-classes.

Sample Input/Output:

RBI has a new Policy issued in 2023

RBI has updated new regulations in 2024.

SBI rate of interest: 7.6 per annum.

Karur rate of interest: 7.4 per annum.

RBI issues all national banks to collect interest on all customer loans.

Create an RBI interface with a variable String parentBank="RBI" and abstract method rateOfInterest().

RBI interface has two more methods default and static method.

default void policyNote() {

System.out.println("RBI has a new Policy issued in 2023.");

}

static void regulations(){

System.out.println("RBI has updated new regulations on 2024.");

}

Create two subclasses SBI and Karur which implements the RBI interface.

Provide the necessary code for the abstract method in two sub-classes.

Sample Input/Output:

RBI has a new Policy issued in 2023

RBI has updated new regulations in 2024.

SBI rate of interest: 7.6 per annum.

Karur rate of interest: 7.4 per annum.

Week 8

1. Final Variable:

Once a variable is declared final, its value cannot be changed after it is initialized.

It must be initialized when it is declared or in the constructor if it's not initialized at declaration.

It can be used to define constants

final int MAX\_SPEED = 120; // Constant value, cannot be changed

2. Final Method:

A method declared final cannot be overridden by subclasses.

It is used to prevent modification of the method's behavior in derived classes.

public final void display() {

System.out.println("This is a final method.");

}

3. Final Class:

A class declared as final cannot be subclassed (i.e., no other class can inherit from it).

It is used to prevent a class from being extended and modified.

public final class Vehicle {

// class code

}

Given a Java Program that contains the bug in it, your task is to clear the bug to the output.

you should delete any piece of code.

class FinalExample

{

int maxSpeed = 120;

public final void displayMaxSpeed()

{

System.out.println("The maximum speed is: " + maxSpeed + " km/h");

}

}

class SubClass extends FinalExample

{

public void showDetails()

{

System.out.println("This is a subclass of FinalExample.");

}

}

class prog

{

public static void main(String[] args)

{

FinalExample obj = new FinalExample();

obj.displayMaxSpeed();

SubClass subObj = new SubClass();

subObj.showDetails();

}

}

Create a base class Shape with a method called calculateArea(). Create three subclasses: Circle, Rectangle, and Triangle. Override the calculateArea() method in each subclass to calculate and return the shape's area.

In the given exercise, here is a simple diagram illustrating polymorphism implementation:

Polymorphism: Shape Class with Circle, Rectangle, and Triangle Subclasses for Area Calculation

abstract class Shape {

public abstract double calculateArea() ;

}

}

System.out.printf("Area of a Triangle :%.2f%n",((0.5)\*base\*height)); // use this statement

sample Input :

4 // radius of the circle to calculate area PI\*r\*r

5 // length of the rectangle

6 // breadth of the rectangle to calculate the area of a rectangle

4 // base of the triangle

3 // height of the triangle

OUTPUT:

Area of a circle :50.27

Area of a Rectangle :30.00

Area of a Triangle :6.00

import java.util.\*;

abstract class s

{

public abstract double calculateArea();

}

class c extends s

{

double r;

c(double r)

{

this.r=r;

}

public double calculateArea()

{

double a=Math.PI\*r\*r;

System.out.printf("Area of a circle: %.2f\n",a);

return a;

}

}

class r extends s

{

double l;

double b;

r(double l,double b)

{

this.l=l;

this.b=b;

}

public double calculateArea()

{

double a=l\*b;

System.out.printf("Area of a Rectangle: %.2f\n",a);

return a;

}

}

class t extends s

{

double b;

double h;

t(double b,double h)

{

this.b=b;

this.h=h;

}

public double calculateArea()

{

double a=b\*h\*0.5;

System.out.printf("Area of a Triangle: %.2f\n",a);

return a;

}

}

public class hello

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

double r1=sc.nextDouble();

c c1=new c(r1);

double l1=sc.nextDouble();

double b1=sc.nextDouble();

r r2=new r(l1,b1);

double b2=sc.nextDouble();

double h2=sc.nextDouble();

t t1=new t(b2,h2);

c1.calculateArea();

r2.calculateArea();

t1.calculateArea();

}

}

As a logic building learner you are given the task to extract the string which has vowel as the first and last characters from the given array of Strings.

Step1: Scan through the array of Strings, extract the Strings with first and last characters as vowels; these strings should be concatenated.

Step2: Convert the concatenated string to lowercase and return it.

If none of the strings in the array has first and last character as vowel, then return no matches found

input1: an integer representing the number of elements in the array.

input2: String array.

Example 1:

input1: 3

input2: {“oreo”, “sirish”, “apple”}

output: oreoapple

Example 2:

input1: 2

input2: {“Mango”, “banana”}

output: no matches found

Explanation:

None of the strings has first and last character as vowel.

Hence the output is no matches found.

Example 3:

input1: 3

input2: {“Ate”, “Ace”, “Girl”}

output: ateace

import java.util.\*;

public class hello

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int k=0;

String arr[]=new String[n];

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

{

arr[i]=sc.next();

arr[i]=arr[i].toLowerCase();

char ch=arr[i].charAt(0);

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

{

int z=arr[i].length();

char x=arr[i].charAt(z-1);

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

{

k=1;

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

}

}

}

if(k==0)

{

System.out.println("no matches found");

}

}

}

Week 9

Write a Java program to create a method that takes an integer as a parameter

and throws an exception if the number is odd.

Sample input and Output:

82 is even.

Error: 37 is odd.

Fill the preloaded answer to get the expected output.

class prog {

public static void main(String[] args) {

int n = 82;

trynumber(n);

n = 37;

trynumber(n);

}

public static void trynumber(int n) {

try {

checkEvenNumber(n);

System.out.println(n + " is even.");

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

}

public static void checkEvenNumber(int number)throws Exception{

if (number % 2 != 0) {

throw new Exception(number + " is odd.");

}

}

}

In the following program, an array of integer data is to be initialized.

During the initialization, if a user enters a value other than an integer, it will throw an InputMismatchException exception.

On the occurrence of such an exception, your program should print “You entered bad data.”

If there is no such exception it will print the total sum of the array.

/\* Define try-catch block to save user input in the array "name"

If there is an exception then catch the exception otherwise print the total sum of the array. \*/

Sample Input:

3

5 2 1

Sample Output:

8

Sample Input:

2

1 g

Sample Output:

You entered bad data.

import java.util.Scanner;

import java.util.InputMismatchException;

class prog {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int length = sc.nextInt();

// create an array to save user input

int[] name = new int[length];

int sum=0;//save the total sum of the array.

/\* Define try-catch block to save user input in the array "name"

If there is an exception then catch the exception otherwise print

the total sum of the array. \*/

try

{

}

catch( )

{

}

}

}

Write a Java program to handle ArithmeticException and ArrayIndexOutOfBoundsException.

Create an array, read the input from the user, and store it in the array.

Divide the 0th index element by the 1st index element and store it.

if the 1st element is zero, it will throw an exception.

if you try to access an element beyond the array limit throws an exception.

Input:

5

10 0 20 30 40

Output:

java.lang.ArithmeticException: / by zero

I am always executed

Input:

3

10 20 30

Output

java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3

I am always executed

Write a Java program to handle ArithmeticException and ArrayIndexOutOfBoundsException.

Create an array, read the input from the user, and store it in the array.

Divide the 0th index element by the 1st index element and store it.

if the 1st element is zero, it will throw an exception.

if you try to access an element beyond the array limit throws an exception.

Input:

5

10 0 20 30 40

Output:

java.lang.ArithmeticException: / by zero

I am always executed

Input:

3

10 20 30

Output

java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3

I am always executed

Week 10

Given an ArrayList, the task is to get the first and last element of the ArrayList in Java.

Input: ArrayList = [1, 2, 3, 4]

Output: First = 1, Last = 4

Input: ArrayList = [12, 23, 34, 45, 57, 67, 89]

Output: First = 12, Last = 89

Approach:

Get the ArrayList with elements.

Get the first element of ArrayList using the get(index) method by passing index = 0.

Get the last element of ArrayList using the get(index) method by passing index = size – 1.

import java.util.\*;

public class main

{

public static void main(String[] args)

{

Scanner obj=new Scanner(System.in);

int n=obj.nextInt();

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

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

{

list.add(obj.nextInt());

}

System.out.println("ArrayList: "+list);

System.out.println("First : "+list.get(0)+", "+"Last : "+list.get(n-1));

obj.close();

}

}

The given Java program is based on the ArrayList methods and its usage. The Java program is partially filled. Your task is to fill in the incomplete statements to get the desired output.

list.set();

list.indexOf());

list.lastIndexOf())

list.contains()

list.size());

list.add();

list.remove();

The above methods are used for the below Java program.

import java.util.ArrayList;

import java.util.Scanner;

public class Prog {

public static void main(String[] args)

{

Scanner sc= new Scanner(System.in);

int n = sc.nextInt();

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

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

list.add(sc.nextInt());

System.out.println("ArrayList: " + list);

list.set(1,100);

System.out.println("Index of 100 = "+list.indexOf(100));

System.out.println("LastIndex of 100 = "+list.lastIndexOf(100));

System.out.println(list.contains(200));

System.out.println("Size Of ArrayList = "+list.size());

list.add(1,500);

list.remove(list.get(3));

System.out.print("ArrayList: " + list);

}

}

Write a Java program to reverse elements in an array list.

Sample input and Output:

Red

Green

Orange

White

Black

Sample output

List before reversing :

[Red, Green, Orange, White, Black]

List after reversing :

[Black, White, Orange, Green, Red]

import java.util.\*;

public class Main

{

public static void main(String[] args)

{

Scanner obj=new Scanner(System.in);

int n=obj.nextInt();

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

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

{

list.add(obj.next());

}

System.out.println("List before reversing :"+"\n"+list);

Collections.reverse(list);

System.out.println("List after reversing :"+"\n"+list);

obj.close();

}

}

Week 11

import java.util.HashMap;

import java.util.Map.Entry;

import java.util.Set;

import java.util.Scanner;

class prog

{

public static void main(String[] args)

{

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

String name;

int num;

Scanner sc= new Scanner(System.in);

int n=sc.nextInt();

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

{

name=sc.next();

num= sc.nextInt();

map.put(name,num);

}

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

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

{

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

}

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

HashMap<String, Integer> anotherMap = new HashMap<String, Integer>();

anotherMap.put("SIX", 6);

anotherMap.put("SEVEN", 7);

anotherMap.putAll (map); // code here

entrySet = anotherMap.entrySet();

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

{

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

}

map.putIfAbsent("FIVE", 5);

int value = map.get("TWO");

System.out.println(value);

System.out.println( map.containsKey("ONE"));

System.out.println(map.containsValue(3));

System.out.println(map.size());

}

}

Week 12

You are provided with a string which has a sequence of 1’s and 0’s.

This sequence is the encoded version of a English word. You are supposed write a program to decode the provided string and find the original word.

Each alphabet is represented by a sequence of 0s.

This is as mentioned below:

Z : 0

Y : 00

X : 000

W : 0000

V : 00000

U : 000000

T : 0000000

and so on upto A having 26 0’s (00000000000000000000000000).

The sequence of 0’s in the encoded form are separated by a single 1 which helps to distinguish between 2 letters.

Example 1:

input1: 010010001

The decoded string (original word) will be: ZYX

Example 2:

input1: 00001000000000000000000100000000000100000000010000000000001

The decoded string (original word) will be: WIPRO

Note: The decoded string must always be in UPPER case.

import java.util.Scanner;

public class Main{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

String[] str = sc.next().split("1");

System.out.println(decodeBin(str));

sc.close();

}

public static String decodeBin(String[] str){

StringBuffer ans = new StringBuffer("");

for(String s: str){

char c = (char)(97+(26-s.length()));

ans.append(c);

}

return ans.toString().toUpperCase();

}

}

Given two char arrays input1[] and input2[] containing only lower case alphabets, extracts the alphabets which are present in both arrays (common alphabets).

Get the ASCII values of all the extracted alphabets.

Calculate sum of those ASCII values. Lets call it sum1 and calculate single digit sum of sum1, i.e., keep adding the digits of sum1 until you arrive at a single digit.

Return that single digit as output.

Note:

1. Array size ranges from 1 to 10.

2. All the array elements are lower case alphabets.

3. Atleast one common alphabet will be found in the arrays.

Example 1:

input1: {‘a’, ‘b’, ‘c’}

input2: {‘b’, ’c’}

output: 8

Explanation:

‘b’ and ‘c’ are present in both the arrays.

ASCII value of ‘b’ is 98 and ‘c’ is 99.

98 + 99 = 197

1 + 9 + 7 = 17

1 + 7 = 8

import java.util.Scanner;

import java.util.ArrayList;

import java.util.Arrays;

public class Main{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

String[] s1 = sc.nextLine().split(" ");

String[] s2 = sc.nextLine().split(" ");

ArrayList<Character> ca1 = new ArrayList<>();

ArrayList<Character> ca2 = new ArrayList<>();

ArrayList<Character> ans = new ArrayList<>();

for(String s: s1){

ca1.add(s.charAt(0));

}

for(String s: s2){

ca2.add(s.charAt(0));

}

int m = ca1.size()>ca2.size()? 1: 2;

if(m==1){

for(int i = 0;i<ca2.size();i++){

if(ca1.contains(ca2.get(i)))

ans.add(ca2.get(i));

}

}

else{

for(int i = 0;i<ca1.size();i++){

if(ca2.contains(ca1.get(i)))

ans.add(ca1.get(i));

}

}

int sum = 0;

for(char c: ans){

sum+= (int)c;

}

System.out.println(findSuperDigit(sum));

}

public static int findSuperDigit(int n){

if(n/10 == 0){

return n;

}

int sum = 0;

int x = n;

while(x!=0){

sum+=(x%10);

x/=10;

}

n = sum;

return(findSuperDigit(n));

}

}

Write a function that takes an input String (sentence) and generates a new String (modified sentence) by reversing the words in the original String, maintaining the words position.

In addition, the function should be able to control the reversing of the case (upper or lowercase) based on a case\_option parameter, as follows:

If case\_option = 0, normal reversal of words i.e., if the original sentence is “Wipro TechNologies BangaLore”, the new reversed sentence should be “orpiW seigoloNhceT eroLagnaB”.

If case\_option = 1, reversal of words with retaining position’s case i.e., if the original sentence is “Wipro TechNologies BangaLore”, the new reversed sentence should be “Orpiw SeigOlonhcet ErolaGnab”.

Note that positions 1, 7, 11, 20 and 25 in the original string are uppercase W, T, N, B and L.

Similarly, positions 1, 7, 11, 20 and 25 in the new string are uppercase O, S, O, E and G.

NOTE:

1. Only space character should be treated as the word separator i.e., “Hello World” should be treated as two separate words, “Hello” and “World”. However, “Hello,World”, “Hello;World”, “Hello-World” or “Hello/World” should be considered as a single word.

2. Non-alphabetic characters in the String should not be subjected to case changes. For example, if case option = 1 and the original sentence is “Wipro TechNologies, Bangalore” the new reversed sentence should be “Orpiw ,seiGolonhceT Erolagnab”. Note that comma has been treated as part of the word “Technologies,” and when comma had to take the position of uppercase T it remained as a comma and uppercase T took the position of comma. However, the words “Wipro and Bangalore” have changed to “Orpiw” and “Erolagnab”.

3. Kindly ensure that no extra (additional) space characters are embedded within the resultant reversed String.

import java.util.Scanner;

import java.lang.StringBuffer;

import java.util.ArrayList;

public class Main{

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

String s[] = sc.nextLine().split(" ");

int n = sc.nextInt();

ArrayList<StringBuffer> ip = new ArrayList<>();

for(String i: s){

ip.add(new StringBuffer(i).reverse());

}

if(n==0){

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

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

}

}

if(n==1){

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

StringBuffer s1 = ip.get(i);

if(Character.isAlphabetic(s1.charAt(s1.length()-1))&& Character.isAlphabetic(s1.charAt(0))){

s1.setCharAt(0, Character.toUpperCase(s1.charAt(0)));

s1.setCharAt(s1.length()-1, Character.toLowerCase(s1.charAt(s1.length()-1)));

}

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

}

}

sc.close();

}

}