RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

OOP USING JAVA RECORD

DILSHATH SHAFANA

IT A (II YEAR)

2116231001038

Table of Contents

Week 1:

Logic building

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.

```
import java.util.Scanner;
import java.util.Scanner;
import java.util.Scanner;
import java.util.Scanner;
public class Odd
{
    public static void main(String args[])
    {
        Scanner s=new Scanner(System.in);
        int x=s.nextInt();

        if(x%2==0)
        {System.out.println(1);}
        else
        {System.out.println(2);}
    }
}
```

2.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

```
import java.util.Scanner;
public class Last
{
    public static void main(String args[])
    {
        Scanner s= new Scanner(System.in);

        int x= s.nextInt();

        System.out.println(Math.abs(x%10));



    }
}
```

3.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

```
import java.util.Scanner;
public class Add
{
    public static void main(String args[])
    {
        Scanner a=new Scanner(System.in);
        int x= a.nextInt();
        int y= a.nextInt();
        int x1=Math.abs(x%10);
        int y1=Math.abs(y%10);
        System.out.println(x1+y1);
    }
}
```

Week 2:

You and your friend are movie fans and want to predict if the movie is going to be a hit!

The movie's success formula depends on 2 parameters:

the acting power of the actor (range 0 to 10)

the critic's rating of the movie (range 0 to 10)

The movie is a hit if the acting power is excellent (more than 8) or the rating is excellent (more than 8). This holds true except if either the acting power is poor (less than 2) or rating is poor (less than 2), then the movie is a flop. Otherwise the movie is average.

Write a program that takes 2 integers:

the first integer is the acting power

second integer is the critic's rating.

You have to print Yes if the movie is a hit, Maybe if the movie is average and No if the movie is flop.

Example input:

9 5

Output:

Yes

Example input:

1 9

Output:

No

Example input:

6 4

Output:

Maybe

```
import java.util.Scanner;

public class Solution {
    public static void Met(int actingPower, int criticRating) {
        if (actingPower < 2 || criticRating < 2) {
            System.out.println("No");
        }
        else if (actingPower > 8 || criticRating > 8) {
            System.out.println("Yes");
        }
        else {
            System.out.println("Maybe");
        }
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int actingPower = scanner.nextInt();
        int criticRating = scanner.nextInt();
        Met(actingPower, criticRating);
    }
}
```

2.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

```
import java.util.Scanner;

public class TrailingZeroes {
    public static int countTrailingZeroes(int n) {
        int count = 0;
        for (int i = 5; n / i >= 1; i *= 5) {
            count += n / i;
        }
        return count;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
           int n = scanner.nextInt();

        int result = countTrailingZeroes(n);
        System.out.println( result);
    }
}
```

2.Write a Java program to input a number from user and print it into words using for loop. How to display number in words using loop in Java programming.

Logic to print number in words in Java programming.

Example

Input

1234

Output

One Two Three Four

Input:

16

Output:

one six

```
import java.util.*;
public class Letter{
    public static void main(String args[])
    {
        Scanner s=new Scanner(System.in);
        String numberString= s.nextLine();
        StringBuilder words= new StringBuilder();
        for(char digit:numberString.toCharArray())
        {
            switch(digit)
            {
                case '0':
                    words.append("Zero ");
                    break;
                case '1':
                    words.append("One ");
                    break;
                case '2':
                    words.append("Two ");
                    break;
                case'3':
                    words.append("Three ");
                    break;
                case '4':
                    words.append("Four ");
                    break;
                case '5':
                    words.append("Five ");
                    break;
                case '6':
                    words.append("Six ");
                    break;
                case '7':
                    words.append("Seven ");
                    break;
                case '8':
```

```
                    words.append("Eight ");
                    break;
                case'9':
                    words.append("Nine ");
                    break;
                default:
                 words.append("Invalid");


            }
        }
        System.out.println(words.toString());

    }
}
```

Week 3:

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

```
import java.util.*;

public class Arrs {
    public static int[] arrs(int[] nums) {

        int max = Integer.MIN_VALUE;
        for (int num : nums) {
            if (num > max) {
                max = num;
            }
        }
        int[] result = new int[nums.length];
        for (int i = 0; i < nums.length; i++) {
            result[i] = (nums[i] - max) * max;
        }
        return result;
    }
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int size = s.nextInt();
        int[] number = new int[size];
```

```
        for (int i = 0; i < size; i++) {
            number[i] = s.nextInt();
        }
        int[] trans = arrs(number);
        for (int i = 0; i < trans.length; i++) {
            System.out.print(trans[i] + " ");
        }
    }
}
```

2.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.

```
import java.util.Scanner;

public class SumOfSpecificNumbers {
    public static int[] generateFirstDigits(int[] nums) {
        int[] firstDigits = new int[nums.length];
        for (int i = 0; i < nums.length; i++) {
            firstDigits[i] =
Character.getNumericValue(String.valueOf(nums[i]).charAt(0));
        }
        return firstDigits;
    }
    public static int[] squareArray(int[] nums) {
        int[] squaredArray = new int[nums.length];
        for (int i = 0; i < nums.length; i++) {
            squaredArray[i] = nums[i] * nums[i];
        }
        return squaredArray;
    }
    public static int calculateSum(int[] nums) {
        int sum = 0;
        for (int num : nums) {
            sum += num;
        }
        return sum;
    }
```

```
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int size = scanner.nextInt();
        int[] numbers = new int[size];

        for (int i = 0; i < size; i++) {
            numbers[i] = scanner.nextInt();
        }
        int[] firstDigits = generateFirstDigits(numbers);
        int[] squaredArray = squareArray(firstDigits);
        int finalSum = calculateSum(squaredArray);
        System.out.println(finalSum);
    }
}
```

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.

```
import java.util.Scanner;

public class LongestPositiveSequenceSum {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        int maxSum = -1;
        int currentSum = 0;
        for (int i = 0; i < n; i++) {
            if (arr[i] >= 0) {
                currentSum += arr[i];
                maxSum = Math.max(maxSum, currentSum);
            } else {
                currentSum = 0;
            }
        }
        System.out.println(maxSum);
    }
}
```

Week 4:

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;

    }
    public Student(String name)
    {
        this.name=name;
        this.roll=0;
```

```
    }
    public Student(String name,int roll)
    {
        this.name=name;
        this.roll=101;
    }
    public String getName()
    {
        return name;
    }
    public int getRoll()
    {
        return roll;
    }
    public static void main(String args[])
    {
        Student student1= new Student();
        System.out.println("No-arg constructor is invoked");
        Student student2= new Student("Rajalakshmi");
        System.out.println("1 arg constructor is invoked");
        Student student3=new Student("Lakshmi",101);
        System.out.println("2 arg constructor is invoked");
        System.out.println("Name ="+student1.getName()+" , "+"Roll
no = "+student1.getRoll());
        System.out.println("Name ="+student2.getName()+" , "+"Roll
no = "+student2.getRoll());
        System.out.println("Name ="+student3.getName()+" , "+"Roll
no = "+student3.getRoll());
    }
}
```

2.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.

Week 5:

Inheritance

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 {
    protected String collegeName;

    public College(String collegeName) {
        // Initialize the instance variable
        this.collegeName = collegeName;
    }

    public void admitted() {
        System.out.println("A student admitted in " + collegeName);
    }
}

class Student extends College {
    private String studentName;
    private String department;

    public Student(String collegeName, String studentName, String
department) {
        // Initialize the instance variables
        super(collegeName); // Call to the parent class constructor
        this.studentName = studentName;
        this.department = department;
    }

    @Override
    public String toString() {
        // Return the details of the student
        return "CollegeName : " + collegeName + "\n" +
               "StudentName : " + studentName + "\n" +
               "Department : " + department;
    }
}

public class Main {
    public static void main(String[] args) {
        Student s1 = new Student("REC", "Venkatesh", "CSE");

        // Invoke the admitted() method
        s1.admitted();

        // Print student details
        System.out.println(s1.toString());
    }
}
```

2.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.

```
// Parent class BankAccount
class BankAccount {
    // Private fields to store the account number and balance
    private String accountNumber;
    private double balance;

    // Constructor to initialize account number and balance
    public BankAccount(String accountNumber, double balance) {
        this.accountNumber = accountNumber;
        this.balance = balance;
    }

    // Method to deposit an amount into the account
    public void deposit(double amount) {
        balance += amount; // Increase the balance by the deposit
amount
    }

    // Method to withdraw an amount from the account
    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount; // Decrease the balance by the
withdrawal amount
        } else {
            System.out.println("Insufficient balance"); // Notify
insufficient balance
        }
    }

    // Method to get the current balance
    public double getBalance() {
        return balance; // Return the current balance
    }
}
// Child class SavingsAccount
 class SavingsAccount extends BankAccount {
    // Constructor to initialize account number and balance
    public SavingsAccount(String accountNumber, double balance) {
        super(accountNumber, balance); // Call the parent class
constructor
    }

    // Override the withdraw method from the parent class
    @Override
    public void withdraw(double amount) {
```

```
        if (getBalance() - amount < 100) {
            System.out.println("Minimum balance of $100 required!");
// Check minimum balance
        } else {
            super.withdraw(amount); // Call the parent class
withdraw method
        }
    }
}
// Main class to test the BankAccount and SavingsAccount classes
public class Main {
    public static void main(String[] args) {
        // Create a BankAccount object (A/c No. BA1234) with initial
balance of $500
        System.out.println("Create a Bank Account object (A/c No.
BA1234) with initial balance of $500:");
        BankAccount BA1234 = new BankAccount("BA1234", 500);

        // Deposit $1000 into account BA1234
        System.out.println("Deposit $1000 into account BA1234:");
        BA1234.deposit(1000);
        System.out.println("New balance after depositing $1000: $" +
BA1234.getBalance());

        // Withdraw $600 from account BA1234
        System.out.println("Withdraw $600 from account BA1234:");
        BA1234.withdraw(600);
        System.out.println("New balance after withdrawing $600: $" +
BA1234.getBalance());

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

        // Try to withdraw $250 from SA1000
        System.out.println("Try to withdraw $250 from SA1000!");
        SA1000.withdraw(250);
        System.out.println("Balance after trying to withdraw $250:
$" + SA1000.getBalance());
    }
}
```

3.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

// Base class Mobile

```
class Mobile {
    String brand;
    String model;
```

```
    // Constructor for Mobile
    public Mobile(String brand, String model) {
        this.brand = brand;
        this.model = model;
    }

    // Method to print basic mobile manufacturing message
    public void basicMobile() {
        System.out.println("Basic Mobile is Manufactured");
    }
}

// Subclass CameraMobile extending Mobile
class CameraMobile extends Mobile {
    int cameraQuality;

    // Constructor for CameraMobile
    public CameraMobile(String brand, String model, int
cameraQuality) {
        super(brand, model);  // Call the parent constructor
        this.cameraQuality = cameraQuality;
    }

    // Overriding the basicMobile method to print Camera Mobile is
Manufactured
    @Override
    public void basicMobile() {
        System.out.println("Camera Mobile is Manufactured");
    }

    // Additional method to print Camera Mobile with 5MG px
    public void cameraDetails() {
        System.out.println("Camera Mobile with 5MG px");
    }
}

// Subclass AndroidMobile extending CameraMobile
class AndroidMobile extends CameraMobile {
    String androidVersion;

    // Constructor for AndroidMobile
    public AndroidMobile(String brand, String model, int
cameraQuality, String androidVersion) {
        super(brand, model, cameraQuality);  // Call the parent
constructor
        this.androidVersion = androidVersion;
    }

    // Overriding the basicMobile method to print Android Mobile is
Manufactured
    @Override
```

```
    public void basicMobile() {
        System.out.println("Android Mobile is Manufactured");
    }

    // Method to print Touch Screen Mobile is Manufactured
    public void touchScreen() {
        System.out.println("Touch Screen Mobile is Manufactured");
    }
}

// Main class to test the functionality
public class Main {
    public static void main(String[] args) {
        // Creating a basic mobile object
        Mobile basicPhone = new Mobile("Nokia", "3310");
        basicPhone.basicMobile();  // Output: Basic Mobile is
Manufactured

        // Creating a camera mobile object
        CameraMobile cameraPhone = new CameraMobile("Sony",
"Xperia", 5);
        cameraPhone.basicMobile();  // Output: Camera Mobile is
Manufactured

        // Creating an Android mobile object
        AndroidMobile androidPhone = new AndroidMobile("Samsung",
"Galaxy S21", 108, "11");
        androidPhone.basicMobile();  // Output: Android Mobile is
Manufactured
        androidPhone.cameraDetails();  // Output: Camera Mobile with
5MG px
        androidPhone.touchScreen();  // Output: Touch Screen Mobile
is Manufactured
    }
}
```

Week 6:

6

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 WordProcessor {

    // Method to get the alphabet based on case 2
    public static char getAlphabetByPosition(int diff) {
        return (char) ('a' + (diff - 1));
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Taking input from the user
       // System.out.println("Enter the input string:");
        String input = sc.nextLine();

        // Split the input by ':'
        String[] words = input.split(":");
        StringBuilder result = new StringBuilder();

        // Loop through each pair
        for (String word : words) {
            char firstChar = word.charAt(0);
            char secondChar = word.charAt(1);
```

```
            if (firstChar == secondChar) {
                // Case 1: Same characters
                result.append(firstChar);
            } else {
                // Case 2: Different characters
                int pos1 = firstChar - 'a' + 1; // Get the position
of first char
                int pos2 = secondChar - 'a' + 1; // Get the position
of second char

                // Get the difference
                int diff = Math.abs(pos1 - pos2);

                // Get the corresponding character for the
difference
                result.append(getAlphabetByPosition(diff));
            }
        }

        // Output the result in uppercase
        System.out.println(result.toString().toUpperCase());

        sc.close();
    }
}
```

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 {

    // Method to process a word according to the given rule
    public static String processWord(String word) {
        int len = word.length();
        int mid = len / 2;

        // If the length is odd, include the middle character in
both parts
        if (len % 2 == 0) {
            // Middle-to-Begin part (reverse from middle to start)
            String middleToBegin = new
StringBuilder(word.substring(0, mid)).reverse().toString();
            // Middle-to-End part (from middle to end)
            String middleToEnd = word.substring(mid);
            return middleToBegin + middleToEnd;
        } else {
            // Middle-to-Begin part (reverse from middle to start,
including middle character)
            String middleToBegin = new
StringBuilder(word.substring(0, mid + 1)).reverse().toString();
            // Middle-to-End part (from middle to end, including
middle character)
            String middleToEnd = word.substring(mid);
```

```
            return middleToBegin + middleToEnd;
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Input string of words
        //System.out.println("Enter the sentence:");
        String input1 = sc.nextLine();

        // Input two-digit number
        //System.out.println("Enter the two-digit number:");
        String input2 = sc.nextLine();

        // Split the input string into words
        String[] words = input1.split(" ");

        // Extract the positions from the two-digit number (convert
to zero-based index)
        int firstPosition =
Character.getNumericValue(input2.charAt(1)) - 1;  // Second digit is
for the first word to process
        int secondPosition =
Character.getNumericValue(input2.charAt(0)) - 1; // First digit is
for the second word to process

        // Process both words based on the positions from input2
        String firstProcessedWord =
processWord(words[firstPosition]);
        String secondProcessedWord =
processWord(words[secondPosition]);

        // Output the result with processed words
        System.out.println(secondProcessedWord + " " +
firstProcessedWord);
    }
}
```

3.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.Scanner;

public class StringManipulation {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        String input1 = sc.nextLine();
        String input2 = sc.nextLine();

        String combinedString = input1 + input2;
        if (combinedString.trim().isEmpty()) {
            System.out.println("null");
            return;
        }
        StringBuilder uniqueChars = new StringBuilder();
        for (int i = 0; i < combinedString.length(); i++) {
            char c = combinedString.charAt(i);
            if (c != ' ' && uniqueChars.indexOf(String.valueOf(c))
== -1) {
                uniqueChars.append(c);
            }
        }
        char[] charArray = uniqueChars.toString().toCharArray();
        for (int i = 0; i < charArray.length - 1; i++) {
            for (int j = i + 1; j < charArray.length; j++) {
                if (charArray[i] < charArray[j]) {
                      char temp = charArray[i];
                    charArray[i] = charArray[j];
                    charArray[j] = temp;
                }
            }
        }
        System.out.println(new String(charArray));
    }
}
```

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.Scanner;

// Define the Playable interface
interface Playable {
    // Abstract method to play the respective sport
    void play();
}

// Football class implementing Playable interface
class Football implements Playable {
    String name;

    // Constructor
    public Football(String name) {
```

```
        this.name = name;
    }

    // Override the play method
    public void play() {
        System.out.println(name + " is Playing football");
    }
}

// Volleyball class implementing Playable interface
class Volleyball implements Playable {
    String name;

    // Constructor
    public Volleyball(String name) {
        this.name = name;
    }

    // Override the play method
    public void play() {
        System.out.println(name + " is Playing volleyball");
    }
}

// Basketball class implementing Playable interface
class Basketball implements Playable {
    String name;

    // Constructor
    public Basketball(String name) {
        this.name = name;
    }

    // Override the play method
    public void play() {
        System.out.println(name + " is Playing basketball");
    }
}

// Main class to test the functionality
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input for Football player

        String footballPlayerName = scanner.nextLine();
        Football footballPlayer = new Football(footballPlayerName);

        // Input for Volleyball player
```

```
        String volleyballPlayerName = scanner.nextLine();
        Volleyball volleyballPlayer = new
Volleyball(volleyballPlayerName);

        // Input for Basketball player

        String basketballPlayerName = scanner.nextLine();
        Basketball basketballPlayer = new
Basketball(basketballPlayerName);

        // Call the play method for each player
        footballPlayer.play();
        volleyballPlayer.play();
        basketballPlayer.play();

        scanner.close();
    }
}
```

2.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 {
    void setHomeTeam(String name);
    void setVisitingTeam(String name);
}

interface Football extends Sports {
    void homeTeamScored(int points);
    void visitingTeamScored(int points);
}

class College implements Football {
    private String homeTeam;
    private String visitingTeam;
    private int homeTeamPoints = 0;
    private int visitingTeamPoints = 0;

    public void setHomeTeam(String name) {
        this.homeTeam = name;
    }

    public void setVisitingTeam(String name) {
        this.visitingTeam = name;
    }

    public void homeTeamScored(int points) {
        homeTeamPoints += points;
        System.out.println(homeTeam + " " + points + " scored");
    }

    public void visitingTeamScored(int points) {
        visitingTeamPoints += points;
        System.out.println(visitingTeam + " " + points + " scored");
    }

    public void winningTeam() {
        if (homeTeamPoints > visitingTeamPoints) {
            System.out.println(homeTeam + " is the winner!");
        } else if (homeTeamPoints < visitingTeamPoints) {
```

```
            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) {
        Scanner sc = new Scanner(System.in);

        // Get home team name
        String hname = sc.nextLine();

        // Get visiting team name
        String vteam = sc.nextLine();

        // Create College object
        College match = new College();
        match.setHomeTeam(hname);
        match.setVisitingTeam(vteam);

        // Get points scored by home team
        int htpoints = sc.nextInt();
        match.homeTeamScored(htpoints);

        // Get points scored by visiting team
        int vtpoints = sc.nextInt();
        match.visitingTeamScored(vtpoints);

        // Determine and print the winning team
        match.winningTeam();

        sc.close();
    }
}
```

3.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.

```
// Define the RBI interface
interface RBI {
    // Variable declaration
    String parentBank = "RBI";

    // Abstract method
    double rateOfInterest();

    // Default method
    default void policyNote() {
        System.out.println("RBI has a new Policy issued in 2023");
    }
```

```
    // Static method
    static void regulations() {
        System.out.println("RBI has updated new regulations in
2024.");
    }
}

// SBI class implementing RBI interface
class SBI implements RBI {
    // Implementing the abstract method
    public double rateOfInterest() {
        return 7.6;
    }
}

// Karur class implementing RBI interface
class Karur implements RBI {
    // Implementing the abstract method
    public double rateOfInterest() {
        return 7.4;
    }
}

// Main class to test the functionality
public class Main {
    public static void main(String[] args) {
        // RBI policies and regulations
        RBI rbi = new SBI();  // Can be any class implementing RBI
        rbi.policyNote();     // Default method
        RBI.regulations();    // Static method

        // SBI bank details
        SBI sbi = new SBI();
        System.out.println("SBI rate of interest: " +
sbi.rateOfInterest() + " per annum.");

        // Karur bank details
        Karur karur = new Karur();
        System.out.println("Karur rate of interest: " +
karur.rateOfInterest() + " per annum.");
    }
}
```

Week 8:

8.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.

```
 import java.util.Scanner;

abstract class Shape {
    public abstract double calculateArea();
}

class Circle extends Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double calculateArea() {
        return Math.PI * radius * radius;
    }
}

class Rectangle extends Shape {
    private double length;
    private double breadth;

    public Rectangle(double length, double breadth) {
        this.length = length;
        this.breadth = breadth;
    }

    @Override
    public double calculateArea() {
        return length * breadth;
    }
}

class Triangle extends Shape {
    private double base;
    private double height;

    public Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    @Override
```

```
    public double calculateArea() {
        return 0.5 * base * height;
    }
}

public class ShapeAreaCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);



        double radius = scanner.nextDouble();
        Shape circle = new Circle(radius);

        double length = scanner.nextDouble();
        double breadth = scanner.nextDouble();
        Shape rectangle = new Rectangle(length, breadth);
        double base = scanner.nextDouble();
        double height = scanner.nextDouble();
        Shape triangle = new Triangle(base, height);
        System.out.printf("Area of a circle: %.2f%n",
circle.calculateArea());
        System.out.printf("Area of a Rectangle: %.2f%n",
rectangle.calculateArea());
        System.out.printf("Area of a Triangle: %.2f%n",
triangle.calculateArea());

        scanner.close();
    }
}
```

Week 9:

9.

1.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.InputMismatchException;
import java.util.Scanner;

public class ArrayInput {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            int size = scanner.nextInt();

            int[] numbers = new int[size];

            int sum = 0;

            for (int i = 0; i < size; i++) {
                numbers[i] = scanner.nextInt();
                sum += numbers[i];
            }

            System.out.println(sum);

        } catch (InputMismatchException e) {

            System.out.println("You entered bad data.");
        } finally {
            scanner.close();
        }
    }
}
```

2.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 OddNumberException extends Exception {
    public OddNumberException(String message) {
        super(message);
    }
}
public class prog {
    public static void main(String[] args) {

        try {
            int n = 82;
            tryNumber(n);
        } catch (OddNumberException e) {
            System.out.println("Error: " + e.getMessage());
        }
        try {
            int n = 37;
            tryNumber(n);
        } catch (OddNumberException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
    public static void tryNumber(int n) throws OddNumberException {
        if (n % 2 == 0) {
            System.out.println(n + " is even.");
        } else {
            throw new OddNumberException(n + " is odd.");
        }
    }
}
```

3.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

```
 import java.util.Scanner;

public class ArrayDivisionExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            int n = scanner.nextInt();
            int[] array = new int[n];
            for (int i = 0; i < n; i++) {
                array[i] = scanner.nextInt();
            }
            int result = array[0] / array[1];
            System.out.println("Trying to access index 3: " +
array[3]);

        } catch (ArithmeticException e) {
            System.out.println("java.lang.ArithmeticException: / by
```

```
zero");
        } catch (ArrayIndexOutOfBoundsException e) {

System.out.println("java.lang.ArrayIndexOutOfBoundsException: " +
e.getMessage());
        } finally {
            System.out.println("I am always executed");
        }

        scanner.close();
    }
}
```

Week 10:

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<>();
        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)); // Output: false
        System.out.println("Size Of ArrayList = " + list.size());
```

```
        list.add(1, 500);
        list.remove(3);
        System.out.print("ArrayList: " + list);

        sc.close();
    }
}
```

Week 11:

public class HashSet<E> extends AbstractSet<E> implements Set<E>, Cloneable, Serializable

Sample Input and Output:

5

90

56

45

78

25

78

Sample Output:

78 was found in the set.

Sample Input and output:

3

2

7

9

5

Sample Input and output:

5 was not found in the set.

```
import java.util.HashSet;
import java.util.Scanner;

public class prog {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        HashSet<Integer> numbers = new HashSet<>();
        for (int i = 0; i < n; i++) {
            numbers.add(sc.nextInt());
        }
        int skey = sc.nextInt();
        if (numbers.contains(skey)) {
            System.out.println(skey + " was found in the set.");
        } else {
            System.out.println(skey + " was not found in the set.");
```

```
        }

        sc.close();
    }
}
```

2.Write a Java program to compare two sets and retain elements that are the same.

Sample Input and Output:

5

Football

Hockey

Cricket

Volleyball

Basketball

7 // HashSet 2:

Golf

Cricket

Badminton

Football

Hockey

Volleyball

Handball

SAMPLE OUTPUT:

Football

Hockey

Cricket

Volleyball

Basketball

```
 import java.util.HashSet;
import java.util.Iterator;
import java.util.Scanner;

public class SetComparison {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        int n1 = scanner.nextInt();
        scanner.nextLine();
        HashSet<String> set1 = new HashSet<>();
        for (int i = 0; i < n1; i++) {
            set1.add(scanner.nextLine());
        }
        int n2 = scanner.nextInt();
        scanner.nextLine();
        HashSet<String> set2 = new HashSet<>();
        for (int i = 0; i < n2; i++) {
            set2.add(scanner.nextLine());
```

```
        }
        set1.retainAll(set2);
        Iterator<String> iterator = set1.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }

        scanner.close();
    }
}
```

3.Java HashMap Methods

containsKey() Indicate if an entry with the specified key exists in the map

containsValue() Indicate if an entry with the specified value exists in the map

putIfAbsent() Write an entry into the map but only if an entry with the same key does not already exist

remove() Remove an entry from the map

replace() Write to an entry in the map only if it exists

size() Return the number of entries in the map

Your task is to fill the incomplete code to get desired output

```
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)
    {
        //Creating HashMap with default initial capacity and load
factor
```

```
        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);
         }

        //Printing key-value pairs

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

        for (Entry<String, Integer> entry : entrySet)
        {
            System.out.println(entry.getKey()+" :
"+entry.getValue());
        }
         System.out.println("----------");
        //Creating another HashMap

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

        //Inserting key-value pairs to anotherMap using put() method

        anotherMap.put("SIX", 6);

        anotherMap.put("SEVEN", 7);

        //Inserting key-value pairs of map to anotherMap using
putAll() method

        anotherMap.      (    );  // code here

        //Printing key-value pairs of anotherMap

        entrySet = anotherMap.entrySet();

        for (Entry<String, Integer> entry : entrySet)
        {
            System.out.println(entry.getKey()+" :
"+entry.getValue());
        }

        //Adds key-value pair 'FIVE-5' only if it is not present in
```

```
map

        map.putIfAbsent("FIVE", 5);

        //Retrieving a value associated with key 'TWO'

        int value = map.get("TWO");
        System.out.println(value);

         //Checking whether key 'ONE' exist in map

        System.out.println(                          );

        //Checking whether value '3' exist in map

        System.out.println(                    );

        //Retrieving the number of key-value pairs present in map

        System.out.println(              );
    }
}
```

Week 12:

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.HashSet;

public class CommonCharsSum {

    // Function
    public static int getSingleDigitSum(char[] input1, char[]
input2) {
        HashSet<Character> set1 = new HashSet<>();
        for (char c : input1) {
            set1.add(c);
        }

        int sum1 = 0;
        for (char c : input2) {

            if (set1.contains(c)) {

                sum1 += (int) c;
            }
        }

        int result = sum1;
        while (result >= 10) {
            result = sumDigits(result);
        }

        return result;
    }


    private static int sumDigits(int num) {
        int sum = 0;
        while (num > 0) {
            sum += num % 10;
            num /= 10;
        }
        return sum;
    }

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

```
        char[] input1 = {'a', 'b', 'c'};
        char[] input2 = {'b', 'c'};

        // Call the function and print the result
        int result = getSingleDigitSum(input1, input2);
        System.out.println(result);
    }
}
```