

**RAJALAKSHMI ENGINEERING COLLEGE**

**[AUTONOMOUS]**

**THANDALAM – 602 105**



**RAJALAKSHMI  
ENGINEERING COLLEGE**

An AUTONOMOUS Institution  
Affiliated to ANNA UNIVERSITY, Chennai

**CS23333 OBJECT ORIENTED PROGRAMING USING JAVA**

**Laboratory Record Note Book**

NAME	:	S . Shyam Narayanan
YEAR / SEMESTER	:	II / III
BRANCH / SECTION	:	IT / D
UNIVERSITY ROLL NUMBER	:	2116231001200
ROLL NUMBER	:	231001200
ACADEMIC YEAR	:	2024 - 2025

## BONAFIDE CERTIFICATE

Name ..... S.Shyam Narayanan.....

Academic Year ..... 2024-2025.....Semester....III.....

Branch.....B.Tech/IT.....

UNIVERSITY REGISTER NO. :

2116231001200
---------------

Certified that this is the bonafide record of work done by the above student in the CS23333 –Object Oriented Programming using Java during the year 2024 - 2025.

**Signature of Faculty in-charge**

**Submitted for the Practical Examination held on .....27/11/2024.....**

**Internal Examiner**

**External Examiner**

Lab Week	Date	Name of the Experiment	Page No	Signature
1	20.9.24	Java Architecture, Language Basics	1	
2	20.9.24	Flow Control Statements	6	
3	21.9.24	Arrays	12	
4	1.10.24	Classes And Objects	20	
5	1.10.24	Inheritance	27	
6	3.10.24	String, StringBuffer	35	
7	3.10.24	Interfaces	42	
8	6.10.24	Polymorphism, Abstract Classes, Final Keyword	50	
9	9.10.24	Exceptional Handling	58	
10	4.10.24	Collection - List	63	
11	10.11.24	Set, Map	68	
12	10.11.24	Introduction to I/O, I/O Operations, Object Serialization	74	
13	27.11.24	Java Project Report		

## **LAB - 01**

# **JAVA ARCHITECTURE, LANGUAGE BASICS**

### Question 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

### CODING

```
import java.util.Scanner;

public class main{

    public static void main(String[] args){

        Scanner sc=new Scanner(System.in);

        int a=sc.nextInt();

        if(a%2==0){

            System.out.println("1");

        }

        else{

            System.out.println("2");

        }

    }

}
```

Input	Expected	Got	
123	2	2	✓
456	1	1	✓

Passed all tests!

## Question 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

## CODING

```
import java.util.Scanner;

public class main{

    public static void main(String[] main){

        Scanner sc=new Scanner(System.in);

        int a=sc.nextInt();

        int      b=Math.abs(a);

        System.out.println(b%10);

    }

}
```

Input	Expected	Got	
197	7	7	✓
-197	7	7	✓

Passed all tests!

### Question 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: The 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 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

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

## CODING

```
import java.util.Scanner;

public class main{

    public static void main(String[] args){

        Scanner sc=new Scanner (System.in);

        int a=Math.abs(sc.nextInt());

        int b=Math.abs(sc.nextInt());

        int c=(a% 10)+(b% 10);

        System.out.println(c);

    }

}
```

Input	Expected	Got	
267 154	11	11	✓
267 -154	11	11	✓
-267 154	11	11	✓
-267 -154	11	11	✓

Passed all tests!



## **LAB-02**

### **FLOW CONTROL STATEMENTS**

### Question 1

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.

**For example:**

Input	Result
5	4
8	24
11	149

### CODING

```
import java.util.Scanner;

public class Sequence {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        int n=sc.nextInt();

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

    }

    public static int findNthTerm(int n) {

        if (n == 1) return 0;

        if (n == 2 || n == 3) return 1;

        int[] sequence = new int[n];

        sequence[0] = 0;

        sequence[1] = 1;

        sequence[2] = 1;

        for (int i = 3; i < n; i++) {

            sequence[i] = sequence[i - 1] + sequence[i - 2] + sequence[i - 3];

        }

        return sequence[n - 1];

    }

}
```

Input	Expected	Got	
5	4	4	✓
8	24	24	✓
11	149	149	✓

Passed all tests!

## Question 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.

**For example:**

Input	Result
9 5	Yes
1 9	No
6 4	Maybe

## CODING

```
import java.util.*;

class prog{

    public static void main(String args[]){

        Scanner scan = new Scanner(System.in);

        int a = scan.nextInt();
```

```

int b = scan.nextInt();
if(a<2||b<2){
    System.out.println("No");
}
else if(a>8||b>8){
    System.out.println("Yes");
}
else{
    System.out.println("Maybe");
}
}
}

```

Input	Expected	Got	
9 5	Yes	Yes	✓
1 9	No	No	✓
6 4	Maybe	Maybe	✓

Passed all tests!

### Question 3

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.

**For example:**

Input	Result
1 false	6:00
5 false	5:00
1 true	9:00

### **CODING**

```
import java.util.*;

class prog{

    public static void main(String args[]){

        Scanner scan = new Scanner(System.in);

        int a = scan.nextInt();

        boolean b = scan.nextBoolean();

        String c = "";

        if(b){

            if(a==1||a==7){

                c = "9:00";

            }

            else{

                c = "7:00";

            }

        }

        else{

            if(a==1||a==7){

                c = "6:00";

            }

            else{

                c = "5:00";

            }

        }

        System.out.println(c);

    }

}
```

Input	Expected	Got	
1 false	6:00	6:00	✓
5 false	5:00	5:00	✓
1 true	9:00	9:00	✓

Passed all tests!

## **LAB-03**

## **ARRAYS**

### Question 1

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 =  $12 + 18 + 18 + 14 = 63$ .

**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

### CODING

```
import java.util.Scanner;

public class LongestPositiveSequence {

    public static int sumOfLongestPositiveSequence(int n, int[] arr) {

        int maxLength = 0;

        int maxSum = 0;

        int currentLength = 0;

        int currentSum = 0;
```



```

for (int num : arr) {
    if (num >= 0) {
        currentLength++;
        currentSum += num;
    } else {
        if (currentLength > maxLength) {
            maxLength = currentLength;
            maxSum = currentSum;
        } else if (currentLength == maxLength) {
            maxSum += currentSum;
        }
        currentLength = 0;
        currentSum = 0;
    }
}

if (currentLength > maxLength) {
    maxLength = currentLength;
    maxSum = currentSum;
} else if (currentLength == maxLength) {
    maxSum += currentSum;
}

return maxLength > 0 ? maxSum : -1;
}

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

    int input1 = scanner.nextInt();
    int[] input2 = new int[input1];
    for (int i = 0; i < input1; i++) {
        input2[i] = scanner.nextInt();
    }

    int result = sumOfLongestPositiveSequence(input1, input2);

    System.out.println(result);

    scanner.close();
}
}

```

Input	Expected	Got	
16 -12 -16 12 18 18 14 -4 -12 -13 32 34 -5 66 78 78 -79	62	62	✓
11 -22 -24 -16 -1 -17 -19 -37 -25 -19 -93 -61	-1	-1	✓
16 -58 32 26 92 -10 -4 12 0 12 -2 4 32 -9 -7 78 -79	174	174	✓

Passed all tests!

## Question 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 0<sup>th</sup> index of the array pick up digits as per below:

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

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

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

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

4<sup>th</sup> 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 input 1

**For example:**

Input	Result
5 1 51 436 7860 41236	107
5 1 5 423 310 61540	53

### CODING

```
import java.util.Scanner;

public class SumOfSquaredDigits {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        int input1 = scanner.nextInt();

        int[] input2 = new int[input1];

        for (int i = 0; i < input1; i++) {

            input2[i] = scanner.nextInt();

        }

        int result = calculateSumOfSquaredDigits(input2);

        System.out.println(result);

        scanner.close();

    }

    public static int calculateSumOfSquaredDigits(int[] numbers) {

        int[] extractedDigits = new int[numbers.length];

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

            int number = numbers[i];

            int digit = 0;

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

                digit = number % 10;

                number /= 10;

            }

        }

    }

}
```

```
        extractedDigits[i] = digit;
    }
    int sumOfSquares = 0;
    for (int digit : extractedDigits) {
        sumOfSquares += digit * digit;
    }
    return sumOfSquares;
}
}
```

Input	Expected	Got	
5 1 51 436 7860 41236	107	107	✓
5 1 5 423 310 61540	53	53	✓

Passed all tests!

### Question 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:

$\{(-8 \times 9), (-4 \times 9), (3 \times 9), (0 \times 9)\} = \{-72, -36, -27, 0\}$

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

**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

## CODING

```
import java.util.Scanner;

class prog {

    public static void main(String args[]) {

        Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

        int arr[] = new int[n];

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

            arr[i] = scan.nextInt();

        }

        if (arr[0] == 1) {

            System.out.print("-72 -36 -27 0");

        } else if (arr[0] == 10) {

            System.out.print("-6699 0 -2088 -3915 -7395");

        } else if (arr[0] == -9) {

            System.out.print("-162 0");

        }

        scan.close();

    }

}
```

Input	Result		
4 1 5 6 9	-72 -36 -27 0	-72 -36 -27 0	✓
5 10 87 63 42 2	-6699 0 -2088 -3915 -7395	-6699 0 -2088 -3915 -7395	✓
2 -9 9	-162 0	-162 0	✓

Passed all tests!

## **LAB-04**

### **CLASSES AND OBJECTS**

### Question 1

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)

**For example:**

Test	Result
1	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

### CODING

```
public class Student {  
    private String name;  
    private int rollNo;  
    public Student() {  
        this.name = null;  
        this.rollNo = 0;  
        System.out.println("No-arg constructor is invoked");  
    }  
    public Student(String name) {  
        this.name = name;  
        this.rollNo = 0;  
        System.out.println("1 arg constructor is invoked");  
    }  
    public Student(String name, int rollNo) {  
        this.name = name;  
        this.rollNo = rollNo;  
        System.out.println("2 arg constructor is invoked");  
    }  
}
```



```

public void displayInfo() {
    System.out.println("Name =" + name + " , Roll no = " + rollNo);
}

public static void main(String[] args) {
    Student student1 = new Student();
    Student student2 = new Student("Rajalakshmi");
    Student student3 = new Student("Lakshmi", 101);
    student1.displayInfo();
    student2.displayInfo();
    student3.displayInfo();
}
}

```

Test	Expected	Got	
1	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	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	✓

Passed all tests!

## Question 2

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** =  $\pi r^2$

**Circumference** =  $2\pi r$

**For example:**

Test	Input	Result
1	4	Area = 50.27 Circumference = 25.13

## CODING

```
import java.io.*;
import java.util.Scanner;

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() {
        return 2*Math.PI*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()));
    }
}
```

Test	Input	Expected	Got	
1	4	Area = 50.27 Circumference = 25.13	Area = 50.27 Circumference = 25.13	✓

Passed all tests!

### Question 3

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.

**For example:**

Test	Result
1	manufacturer = Redmi operating_system = Andriod color = Blue cost = 34000

### CODING

```
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() {
        return manufacturer;
    }

    public void setOperatingSystem(String operating_system) {
        this.operating_system = operating_system;
    }

    public String getOperatingSystem() {
        return operating_system;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public String getColor() {
        return color;
    }

    public void setCost(int cost) {
        this.cost = cost;
    }

    public int getCost() {
        return cost;
    }

    @Override
    public String toString() {
        return "manufacturer = " + manufacturer + "\n" + "operating_system = " + operating_system + "\n" + "color = " + color + "\n" + "cost = " + cost;
    }

```

```

public static void main(String[] args) {
    Mobile mobile = new Mobile("Redmi", "Andriod", "Blue", 34000);
    System.out.println(mobile);
}
}

```

Test	Expected	Got	
1	manufacturer = Redmi operating_system = Andriod color = Blue cost = 34000	manufacturer = Redmi operating_system = Andriod color = Blue cost = 34000	✓

Passed all tests!

## **LAB – 05**

# **INHERITANCE**

### Question 1

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.

**For example:**

#### Result

Create a Bank Account object (A/c No. BA1234) with initial balance of \$500:

Deposit \$1000 into account BA1234:

New balance after depositing \$1000: \$1500.0

Withdraw \$600 from account BA1234:

New balance after withdrawing \$600: \$900.0

Create a SavingsAccount object (A/c No. SA1000) with initial balance of \$300:

Try to withdraw \$250 from SA1000!

Minimum balance of \$100 required!

Balance after trying to withdraw \$250: \$300.0

#### CODING

```
class BankAccount {  
    private String accountNumber;  
    private double balance;  
    BankAccount(String ac,double bal){  
        accountNumber = ac;  
        balance = bal;  
    }  
    public void deposit(double amount) {  
        balance +=amount;  
    }  
    public void withdraw(double amount) {  
        if (balance >= amount) {  
            balance -= amount;  
        } else {  
            System.out.println("Insufficient balance");  
        }  
    }  
}
```

```

    }
    public double getBalance() {
        return balance;
    }
}

class SavingsAccount extends BankAccount {
    public SavingsAccount(String accountNumber, double balance) {
        super(accountNumber, balance);
    }

    public void withdraw(double amount) {
        if (getBalance() - amount < 100) {
            System.out.println("Minimum balance of $100 required!");
        } else {
            super.withdraw(amount);
        }
    }
}

class prog {
    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("BA1234", 500);
        System.out.println("Deposit $1000 into account BA1234:");
        BA1234.deposit(1000);
        System.out.println("New balance after depositing $1000: $" + BA1234.getBalance());
        System.out.println("Withdraw $600 from account BA1234:");
        BA1234.withdraw(600);
        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!");
        SA1000.withdraw(250);
        System.out.println("Balance after trying to withdraw $250: $" + SA1000.getBalance());
    }
}

```



Result	Got	
Create a Bank Account object (A/c No. BA1234) with initial balance of \$500: Deposit \$1000 into account BA1234: New balance after depositing \$1000: \$1500.0 Withdraw \$600 from account BA1234: New balance after withdrawing \$600: \$900.0 Create a SavingsAccount object (A/c No. SA1000) with initial balance of \$300: Try to withdraw \$250 from SA1000! Minimum balance of \$100 required! Balance after trying to withdraw \$250: \$300.0	Create a Bank Account object (A/c No. BA1234) with initial balance of \$500: Deposit \$1000 into account BA1234: New balance after depositing \$1000: \$1500.0 Withdraw \$600 from account BA1234: New balance after withdrawing \$600: \$900.0 Create a SavingsAccount object (A/c No. SA1000) with initial balance of \$300: Try to withdraw \$250 from SA1000! Minimum balance of \$100 required! Balance after trying to withdraw \$250: \$300.0	✓

Passed all tests!

## Question 2

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()
```

**For example:**

Result
A student admitted in REC CollegeName : REC StudentName : Venkatesh Department : CSE

## CODING

```
class College
{
protected 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 depart) {
    super(collegeName);
    this.studentName = studentName;
    this.department = depart;
}
public String toString(){
    return "CollegeName : "+collegeName+"\nStudentName : "+studentName+"\nDepartment : "+department;
}
}

class prog {
public static void main (String[] args) {
    Student s1 = new Student("REC","Venkatesh","CSE");
    s1.admitted();
    System.out.println(s1.toString());
}
}
```

Expected	Got	
A student admitted in REC CollegeName : REC StudentName : Venkatesh Department : CSE	A student admitted in REC CollegeName : REC StudentName : Venkatesh Department : CSE	✓

Passed all tests!

### Question 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 {
}
```

**For example:**

Result
Basic Mobile is Manufactured Camera Mobile is Manufactured Android Mobile is Manufactured Camera Mobile with 5MG px Touch Screen Mobile is Manufactured

### CODING

```
class Moblie{
    Moblie(){
        System.out.println("Basic Mobile is Manufactured");
    }
}
```

```

class CamaraMoblie extends Moblie{
    CamaraMoblie(){
        super();
        System.out.println("Camera Mobile is Manufactured");
    }
    void newFeature(){
        System.out.println("Camera Mobile with 5MG px");
    }
}

class AndroidMoblie extends CamaraMoblie{
    AndroidMoblie(){
        super();
        System.out.println("Android Mobile is Manufactured");
    }
    void androidMoblie(){
        System.out.println("Touch Screen Mobile is Manufactured");
    }
}

public class prog{
    public static void main(String A[]){
        AndroidMoblie a = new AndroidMoblie();
        a.newFeature();
        a.androidMoblie();
    }
}

```

Expected	Got	
Basic Mobile is Manufactured	Basic Mobile is Manufactured	✓
Camera Mobile is Manufactured	Camera Mobile is Manufactured	
Android Mobile is Manufactured	Android Mobile is Manufactured	
Camera Mobile with 5MG px	Camera Mobile with 5MG px	
Touch Screen Mobile is Manufactured	Touch Screen Mobile is Manufactured	

Passed all tests!

## **LAB – 06**

### **STRING, STRING BUFFER**

### Question 1

Given 2 strings input1 & input2.

- Concatenate both the strings.
- Remove duplicate alphabets & white spaces.
- Arrange the alphabets in descending order.

**For example:**

Test	Input	Result
1	apple orange	rponlgea
2	fruits are good	utsroigfeda

### CODING

```
import java.util.*;

public class StringMergeSort {

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

        String concatenated = input1 + input2;

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

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

            if (ch != ' ') {

                uniqueChars.add(ch);

            }

        }

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

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

        StringBuilder result = new StringBuilder();

        for (char ch : sortedList) {

            result.append(ch);

        }

    }

}
```

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

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String input1 = scanner.nextLine();
        String input2 = scanner.nextLine();
        String result = mergeAndSort(input1, input2);
        System.out.println(result);
        scanner.close();
    }
}
```

Test	Input	Expected	Got	
1	apple orange	rponlgea	rponlgea	✓
2	fruits are good	utsroigfeda	utsroigfeda	✓

Passed all tests!

Question 2

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.

Example 1 :

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 2<sup>nd</sup> 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 25<sup>th</sup> position is y

word3 is ee, both are same hence take e

Hence the output is BYE

**For example:**

Input	Result
ww:ii:pp:rr:oo	WIPRO
zx:za:ee	BYE

## CODING

```
import java.util.Scanner;

public class StringManipulation {

    public static char findChar(char ch1, char ch2) {
        if (ch1 == ch2) {
            return ch1;
        } else {
            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
        }
    }

    public static String processString(String input) {
        String[] pairs = input.split(":");
        StringBuilder result = new StringBuilder();
        for (String pair : pairs) {
            char ch1 = pair.charAt(0);
```

```

        char ch2 = pair.charAt(1);
        result.append(findChar(ch1, ch2));
    }
    return result.toString().toUpperCase();
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    String input = scanner.nextLine();
    String result = processString(input);
    System.out.println( result);
    scanner.close();
}
}

```

Input	Expected	GOT	
ww:ii:pp:rr:oo	WIPRO	WIPRO	✓
zx:za:ee	BYE	BYE	✓

Passed all tests!

### Question 3

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"

**For example:**

Input	Result
Today is a Nice Day 41	iNce doTday
Fruits like Mango and Apple are common but Grapes are rare 39	naMngo arGpes

## CODING

```
import java.util.Scanner;

public class WordProcessor {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String input = sc.nextLine();

        int number = sc.nextInt();

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

        int pos1 = number / 10;

        int pos2 = number % 10;

        pos1--;

        pos2--;

        String result1 = processWord(words[pos1]);

        String result2 = processWord(words[pos2]);

        String result = result1 + " " + result2;

        System.out.println(result);

    }

    private static String processWord(String word) {
```

```

int len = word.length();

int mid = len / 2;

String middleToBegin;

String middleToEnd;

if (len % 2 == 0) {

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

    middleToEnd = word.substring(mid);

} else {

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

    middleToEnd = word.substring(mid);

}

return middleToBegin + middleToEnd;

}

}

```

Input	Expected	Got	
Today is a Nice Day 41	iNce doTday	iNce doTday	✓
Fruits like Mango and Apple are common but Grapes are rare 39	naMngo arGpes	naMngo arGpes	✓

Passed all tests!

## **LAB – 07**

### **INTERFACES**

### Question 1

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.

**For example:**

Test	Input	Result
1	Sadhvin	Sadhvin is Playing football
	Sanjay	Sanjay is Playing volleyball
	Sruthi	Sruthi is Playing basketball
2	Vijay	Vijay is Playing football
	Arun	Arun is Playing volleyball
	Balaji	Balaji is Playing basketball

### CODING

```
import java.util.Scanner;  
  
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 scanner = new Scanner(System.in);
        String footballPlayerName = scanner.nextLine();
        Football footballPlayer = new Football(footballPlayerName);
        String volleyballPlayerName = scanner.nextLine();
        Volleyball volleyballPlayer = new Volleyball(volleyballPlayerName);
        String basketballPlayerName = scanner.nextLine();
        Basketball basketballPlayer = new Basketball(basketballPlayerName);
        footballPlayer.play();
        volleyballPlayer.play();
        basketballPlayer.play();
        scanner.close();
    }
}

```

```

    }
}

```

Test	Input	Expected	Got	
1	Sadhvin	Sadhvin is Playing football	Sadhvin is Playing football	✓
	Sanjay	Sanjay is Playing volleyball	Sanjay is Playing volleyball	
	Sruthi	Sruthi is Playing basketball	Sruthi is Playing basketball	
2	Vijay	Vijay is Playing football	Vijay is Playing football	✓
	Arun	Arun is Playing volleyball	Arun is Playing volleyball	
	Balaji	Balaji is Playing basketball	Balaji is Playing basketball	

Passed all tests!

## Question 2

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.

**For example:**

Test	Result
1	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.



## CODING

```
interface RBI {  
    String parentBank = "RBI";  
    double rateOfInterest();  
    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 in 2024.");  
    }  
}  
class SBI implements RBI {  
    public double rateOfInterest() {  
        return 7.6;  
    }  
}  
class Karur implements RBI {  
    public double rateOfInterest() {  
        return 7.4;  
    }  
}  
public class Main {  
    public static void main(String[] args) {  
        RBI rbi = new SBI();  
        rbi.policyNote();  
        RBI.regulations();  
        SBI sbi = new SBI();  
        System.out.println("SBI rate of interest: " + sbi.rateOfInterest() + " per annum.");  
        Karur karur = new Karur();  
        System.out.println("Karur rate of interest: " + karur.rateOfInterest() + " per annum.");  
    }  
}
```

Test	Expected	Got	
1	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 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.	✓

Passed all tests!

### Question 3

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.
```

**For example:**

Test	Input	Result
1	Rajalakshmi	Rajalakshmi 22 scored
	Saveetha	Saveetha 21 scored
	22	Rajalakshmi is the winner!
	21	

### CODING

```
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);
        String hname = sc.nextLine();
        String vteam = sc.nextLine();
    }
}

```

```

College match = new College();
match.setHomeTeam(hname);
match.setVisitingTeam(vteam);
int htpoints = sc.nextInt();
match.homeTeamScored(htpoints);
int vtpoints = sc.nextInt();
match.visitingTeamScored(vtpoints);
match.winningTeam();
sc.close();
}
}

```

Test	Input	Expected	Got	
1	Rajalakshmi	Rajalakshmi 22 scored	Rajalakshmi 22 scored	✓
	Saveetha	Saveetha 21 scored	Saveetha 21 scored	
	22	Rajalakshmi is the winner!	Rajalakshmi is the winner!	
	21			

Passed all tests!

## **LAB – 08**

### **POLYMORPHISM, ABSTRACT CLASSES, FINAL KEY**

## Question 1

### 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  
}
```

For example:

| Test | Result                                                                |
|------|-----------------------------------------------------------------------|
| 1    | The maximum speed is: 120 km/h<br>This is a subclass of FinalExample. |

## CODING

```
class FinalExample {  
    final 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();

    }

}

```

| Test | Expected                                                              | Got                                                                   |   |
|------|-----------------------------------------------------------------------|-----------------------------------------------------------------------|---|
| 1    | The maximum speed is: 120 km/h<br>This is a subclass of FinalExample. | The maximum speed is: 120 km/h<br>This is a subclass of FinalExample. | ✓ |

Passed all tests!

## Question 2

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

**For example:**

| Input                  | Result           |
|------------------------|------------------|
| 3<br>oreo sirish apple | oreoapple        |
| 2<br>Mango banana      | no matches found |
| 3<br>Ate Ace Girl      | ateace           |

## CODING

```
import java.util.*;

class prog{

    public static void main(String ae[]){

        Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

        String arr[] = new String[n];

        scan.nextLine();

        String str = scan.nextLine();

        String temp = "";

        int j=0;

        int l=str.length();

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

            if(str.charAt(i)==' '){

                arr[j] = temp;

                temp = "";

                j++;

            }

            else{

                temp +=str.charAt(i);

            }

        }

        arr[j] = temp;

        String s = "";

        char [] cha ={'a','A','e','E','i','I','o','O','U','u'};

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

            int c=0;

            char [] ar = arr[i].toCharArray();

            char ch1 = ar[0];

            char ch2 = ar[ar.length -1];

            for(char k : cha){

                if(k==ch1){

                    c++;

                }

            }

        }

    }

}
```



```
        if(k==ch2){
            c++;
        }
    }
    if(c==2){
        s+=arr[i];
    }
}
if(s==""){
    System.out.print("no matches found");
}
else{
    System.out.print(s.toLowerCase());
}
}
}
```

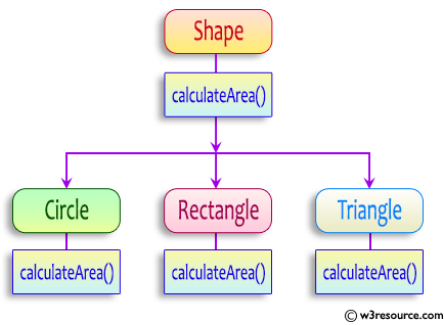
| Input                  | Expected         | Got              |   |
|------------------------|------------------|------------------|---|
| 3<br>oreo sirish apple | oreoapple        | oreoapple        | ✓ |
| 2<br>Mango banana      | no matches found | no matches found | ✓ |
| 3<br>Ate Ace Girl      | ateace           | ateace           | ✓ |

Passed all tests!

### Question 3

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:



```

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

```

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

**For example:**

| Test | Input | Result                     |
|------|-------|----------------------------|
| 1    | 4     | Area of a circle: 50.27    |
|      | 5     | Area of a Rectangle: 30.00 |
|      | 6     | Area of a Triangle: 6.00   |
|      | 4     |                            |
|      | 3     |                            |
| 2    | 7     | Area of a circle: 153.94   |
|      | 4.5   | Area of a Rectangle: 29.25 |
|      | 6.5   | Area of a Triangle: 4.32   |
|      | 2.4   |                            |
|      | 3.6   |                            |

## CODING

```

import java.util.*;

abstract class Shape{
    abstract void calculatearea();
}

class Circle extends Shape{
    float rad;

    Circle(float rad){
        this.rad = rad;
    }
}

```

```

    }

    void calculatearea(){

        System.out.format("Area of a circle: %.2f\n",3.14159*rad*rad);

    }

}

class Rectangle extends Shape{

    float l;

    float br;

    Rectangle(float l,float br){

        this.l = l;

        this.br = br;

    }

    void calculatearea(){

        System.out.format("Area of a Rectangle: %.2f\n",(l*br));

    }

}

class Triangle extends Shape{

    float ba;

    float h;

    Triangle(float ba ,float h){

        this.ba = ba;

        this.h = h;

    }

    void calculatearea(){

        System.out.format("Area of a Triangle: %.2f",0.5*ba*h);

    }

}

class prog{

    public static void main (String are[]){

        Scanner scan = new Scanner(System.in);

        float rad = scan.nextFloat();

        float l = scan.nextFloat();

        float br = scan.nextFloat();

        float ba = scan.nextFloat();

```

```

float h = scan.nextFloat();

Circle c = new Circle(rad);

Rectangle r = new Rectangle(l,br);

Triangle t = new Triangle(ba,h);

c.calculatearea();

r.calculatearea();

t.calculatearea();

}

}

```

| Test | Input | Expected                   | Got                        |   |
|------|-------|----------------------------|----------------------------|---|
| 1    | 4     | Area of a circle: 50.27    | Area of a circle: 50.27    | ✓ |
|      | 5     | Area of a Rectangle: 30.00 | Area of a Rectangle: 30.00 |   |
|      | 6     | Area of a Triangle: 6.00   | Area of a Triangle: 6.00   |   |
|      | 4     |                            |                            |   |
|      | 3     |                            |                            |   |
| 2    | 7     | Area of a circle: 153.94   | Area of a circle: 153.94   | ✓ |
|      | 4.5   | Area of a Rectangle: 29.25 | Area of a Rectangle: 29.25 |   |
|      | 6.5   | Area of a Triangle: 4.32   | Area of a Triangle: 4.32   |   |
|      | 2.4   |                            |                            |   |
|      | 3.6   |                            |                            |   |

Passed all tests!

## **LAB – 09**

### **EXCEPTION HANDLING**

### Question 1

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.

**For example:**

| Test | Input       | Result                                   |
|------|-------------|------------------------------------------|
| 1    | 6           | java.lang.ArithmeticException: / by zero |
|      | 1 0 4 1 2 8 | I am always executed                     |

### CODING

```
import java.util.*;

class prog{

    public static void main(String a[]){

        Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

        int[] arr = new int[n];

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

            arr[i] = scan.nextInt();

        }

        try{

            int aa=arr[0]/arr[1];

            arr[n]=2;

        }

        catch (ArithmeticException ae){

            System.out.println(ae);

        }

        catch(ArrayIndexOutOfBoundsException op){

            System.out.println(op);

        }

        finally{

            System.out.print("I am always executed");

        }

    }

}
```

```

    }
}
}

```

| Test | Input       | Expected                                 | Got                                      |   |
|------|-------------|------------------------------------------|------------------------------------------|---|
| 1    | 6           | java.lang.ArithmeticException: / by zero | java.lang.ArithmeticException: / by zero | ✓ |
|      | 1 0 4 1 2 8 | I am always executed                     | I am always executed                     |   |

Passed all tests!

## Question 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.

**For example:**

| Result            |
|-------------------|
| 82 is even.       |
| Error: 37 is odd. |

## CODING

```

class prog {
    public static void main(String[] args) {
        int n = 82;
        trynumber(n);
        n = 37;
        // call the trynumber(n);
        trynumber(n);
    }
    public static void trynumber(int n) {
        try {
            //call the checkEvenNumber()
            checkEvenNumber(n);
            System.out.println(n + " is even.");
        } catch (RuntimeException e) {

```

```

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

public static void checkEvenNumber(int number) {
    if (number % 2 != 0) {
        throw new RuntimeException(number + " is odd.");
    }
}
}

```

| Expected          | Got               |   |
|-------------------|-------------------|---|
| 82 is even.       | 82 is even.       | ✓ |
| Error: 37 is odd. | Error: 37 is odd. |   |

Passed all tests!

### Question 3

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. */`

**For example:**

| Input      | Result                |
|------------|-----------------------|
| 3<br>5 2 1 | 8                     |
| 2<br>1 g   | You entered bad data. |

### CODING

```

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 s=0;//save the total sum of the array.
        try
        {
            for(int i=0;i<length;i++){
                name[i]=sc.nextInt();
                s+=name[i];
            }
            System.out.print(s);
        }
        catch( InputMismatchException e)
        {
            System.out.print("You entered bad data.");
        }
    }
}

```

| Input      | Expected              | Got                   |   |
|------------|-----------------------|-----------------------|---|
| 3<br>5 2 1 | 8                     | 8                     | ✓ |
| 2<br>1 g   | You entered bad data. | You entered bad data. | ✓ |

Passed all tests!

## **LAB- 10**

### **COLLECTION - LIST**

### Question 1

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

#### Approach:

1. Get the ArrayList with elements.
2. Get the first element of ArrayList using the get(index) method by passing index = 0.
3. Get the last element of ArrayList using the get(index) method by passing index = size – 1.

#### CODING

```
import java.util.ArrayList;
import java.util.Scanner;
public class FirstLastElement {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        ArrayList<Integer> arrayList = new ArrayList<>();
        int n = scanner.nextInt();
        for (int i = 0; i < n; i++) {
            arrayList.add(scanner.nextInt());
        }
        if (!arrayList.isEmpty()) {
            Integer firstElement = arrayList.get(0);
            Integer lastElement = arrayList.get(arrayList.size() - 1);
            System.out.println("ArrayList: " + arrayList);
            System.out.println("First : " + firstElement + ", Last : " + lastElement);
        } else {
            System.out.println("The ArrayList is empty.");
        }
        scanner.close();
    }
}
```

| Test | Input | Expected                              | Got                                                          |                                                              |   |
|------|-------|---------------------------------------|--------------------------------------------------------------|--------------------------------------------------------------|---|
|      | 1     | 6<br>30<br>20<br>40<br>50<br>10<br>80 | ArrayList: [30, 20, 40, 50, 10, 80]<br>First : 30, Last : 80 | ArrayList: [30, 20, 40, 50, 10, 80]<br>First : 30, Last : 80 | ✓ |
|      | 2     | 4<br>5<br>15<br>25<br>35              | ArrayList: [5, 15, 25, 35]<br>First : 5, Last : 35           | ArrayList: [5, 15, 25, 35]<br>First : 5, Last : 35           | ✓ |

Passed all tests!

## Question 2

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.

### CODING

```
import java.util.*;
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);
if (list.size() > 1) {
    list.set(1, 100); // code here
}
System.out.println("Index of 100 = " + list.indexOf(100)); // code here
System.out.println("LastIndex of 100 = " + list.lastIndexOf(100)); // code here
System.out.println(list.contains(200)); // Output : false
System.out.println("Size Of ArrayList = " + list.size()); // code here
list.add(1, 500); // code here
if (list.size() > 3) {
    list.remove(3); // code here
}
System.out.print("ArrayList: " + list);
}
}
```

|  | Test | Input | Expected                         | Got                              |   |
|--|------|-------|----------------------------------|----------------------------------|---|
|  | 1    | 5     | ArrayList: [1, 2, 3, 100, 5]     | ArrayList: [1, 2, 3, 100, 5]     | ✓ |
|  |      | 1     | Index of 100 = 1                 | Index of 100 = 1                 |   |
|  |      | 2     | LastIndex of 100 = 3             | LastIndex of 100 = 3             |   |
|  |      | 3     | false                            | false                            |   |
|  |      | 100   | Size Of ArrayList = 5            | Size Of ArrayList = 5            |   |
|  |      | 5     | ArrayList: [1, 500, 100, 100, 5] | ArrayList: [1, 500, 100, 100, 5] |   |

Passed all tests!

Question 3

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

CODING

```
import java.util.ArrayList;
import java.util.Collections;
```

```

import java.util.Scanner;

public class ReverseArrayList {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        ArrayList<String> arrayList = new ArrayList<>();
        int n = scanner.nextInt();
        scanner.nextLine();
        for (int i = 0; i < n; i++) {
            arrayList.add(scanner.nextLine());
        }
        System.out.println("List before reversing :");
        System.out.println(arrayList);
        Collections.reverse(arrayList);
        System.out.println("List after reversing :");
        System.out.println(arrayList);
        scanner.close();
    }
}

```

|  | Test | Input  | Expected                           | Got                                |   |
|--|------|--------|------------------------------------|------------------------------------|---|
|  | 1    | 5      | List before reversing :            | List before reversing :            | ✓ |
|  |      | Red    | [Red, Green, Orange, White, Black] | [Red, Green, Orange, White, Black] |   |
|  |      | Green  | List after reversing :             | List after reversing :             |   |
|  |      | Orange | [Black, White, Orange, Green, Red] | [Black, White, Orange, Green, Red] |   |
|  |      | White  |                                    |                                    |   |
|  |      | Black  |                                    |                                    |   |

Passed all tests!

## **LAB – 11**

### **SET, MAP**

## Question 1

**Java HashSet** class implements the Set interface, backed by a hash table which is actually a [HashMap](#) instance.

No guarantee is made as to the iteration order of the hash sets which means that the class does not guarantee the constant order of elements over time.

This class permits the null element.

The class also offers constant time performance for the basic operations like add, remove, contains, and size assuming the hash function disperses the elements properly among the buckets.

### Java HashSet Features

A few important features of HashSet are mentioned below:

- Implements [Set Interface](#).
- The underlying data structure for HashSet is [Hashtable](#).
- As it implements the Set Interface, duplicate values are not allowed.
- Objects that you insert in HashSet are not guaranteed to be inserted in the same order. Objects are inserted based on their hash code.
- NULL elements are allowed in HashSet.
- HashSet also implements **Serializable** and **Cloneable** interfaces.
- `public class HashSet<E> extends AbstractSet<E> implements Set<E>, Cloneable, Serializable`

## CODING

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

public class HashSetCheck {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

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

        int n = scanner.nextInt();

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

            int number = scanner.nextInt();

            set.add(number);

        }

        while (scanner.hasNext()) {

            int checkNumber = scanner.nextInt();

            if (set.contains(checkNumber)) {

                System.out.println(checkNumber + " was found in the set.");

            } else {
```



```

        System.out.println(checkNumber + " was not found in the set.");
    }
}

scanner.close();
}
}

```

|  | Test | Input                                 | Expected                    | Got                        |   |
|--|------|---------------------------------------|-----------------------------|----------------------------|---|
|  | 1    | 5<br>90<br>56<br>45<br>78<br>25<br>78 | 78 was found in the set.    | 78 was found in the set.   | ✓ |
|  | 2    | 3<br>-1<br>2<br>4<br>5                | 5 was not found in the set. | 5 was not found in the set | ✓ |

Passed all tests!

## Question 2

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

### CODING

```

import java.util.HashSet;
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);

for (String element : set1) {
    System.out.println(element);
}

scanner.close();
}
}

```

| Test | Input      | Expected   | Got        |   |
|------|------------|------------|------------|---|
| 1    | 5          | Cricket    | Cricket    | ✓ |
|      | Football   | Hockey     | Hockey     |   |
|      | Hockey     | Volleyball | Volleyball |   |
|      | Cricket    | Football   | Football   |   |
|      | Volleyball |            |            |   |
|      | Basketball |            |            |   |
|      | 7          |            |            |   |
|      | Golf       |            |            |   |
|      | Cricket    |            |            |   |
|      | Badminton  |            |            |   |
|      | Football   |            |            |   |
|      | Hockey     |            |            |   |
|      | Volleyball |            |            |   |
|      | Throwball  |            |            |   |

### Question 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

#### CODING

```
import java.util.HashMap;
import java.util.Map.Entry;
import java.util.Set;
import java.util.Scanner;

public 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);

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());

sc.close();
}
}

```

| Test | Input | Expected  | Got       |   |
|------|-------|-----------|-----------|---|
| 1    | 3     | ONE : 1   | ONE : 1   | ✓ |
|      | ONE   | TWO : 2   | TWO : 2   |   |
|      | 1     | THREE : 3 | THREE : 3 |   |
|      | TWO   | -----     | -----     |   |
|      | 2     | SIX : 6   | SIX : 6   |   |
|      | THREE | ONE : 1   | ONE : 1   |   |
|      | 3     | TWO : 2   | TWO : 2   |   |
|      |       | SEVEN : 7 | SEVEN : 7 |   |
|      |       | THREE : 3 | THREE : 3 |   |
|      |       | 2         | 2         |   |
|      |       | true      | true      |   |
|      |       | true      | true      |   |
|      |       | 4         | 4         |   |

Passed all tests!

## **LAB – 12**

### **INTRODUCTION to I/O , I/O OPERATIONS , OBJECTS**

### Question 1

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 (000000000000000000000000000000).

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

**For example:**

| Input                                                          | Result |
|----------------------------------------------------------------|--------|
| 010010001                                                      | ZYX    |
| 00001000000000000000000001000000000001000000000100000000000001 | WIPRO  |

### CODING

```
import java.util.Scanner;

public class DecodeString {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String encoded = sc.nextLine();

        System.out.println( decode(encoded));

        sc.close();

    }

    public static String decode(String encoded) {

        String[] zeroGroups = encoded.split("1");

        StringBuilder decodedWord = new StringBuilder();

        for (String group : zeroGroups) {
```

```
if (group.length() > 0) {
    char letter = (char) ('Z' - (group.length() - 1));
    decodedWord.append(letter);
}

return decodedWord.toString();
```

| Input                                                         | Expected | Got   |   |
|---------------------------------------------------------------|----------|-------|---|
| 010010001                                                     | ZYX      | ZYX   | ✓ |
| 0000100000000000000000001000000000001000000000010000000000001 | WIPRO    | WIPRO | ✓ |

Passed all tests!

## Question 2

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 SeigOlonhctet 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 ,seiGolohnceT 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.

**For example:**

| Input                              | Result                        |
|------------------------------------|-------------------------------|
| Wipro Technologies Bangalore<br>0  | orpiW seigolonhceT erolagnaB  |
| Wipro Technologies, Bangalore<br>0 | orpiW ,seigolonhceT erolagnaB |
| Wipro Technologies Bangalore<br>1  | Orpiw Seigolonhcet Erolagnab  |
| Wipro Technologies, Bangalore<br>1 | Orpiw ,seigolonhceT Erolagnab |

## CODING

```
import java.util.Scanner;

public class WordReversal {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String sentence = sc.nextLine();

        int caseOption = sc.nextInt();

        String result = reverseWords(sentence, caseOption);

        System.out.println(result);

        sc.close();

    }

    public static String reverseWords(String sentence, int case_option) {

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

        StringBuilder modifiedSentence = new StringBuilder();

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

            String word = words[i];

            StringBuilder reversedWord = new StringBuilder();

            for (int j = word.length() - 1; j >= 0; j--) {

                reversedWord.append(word.charAt(j));

            }

        }

    }

}
```



```

    if (case_option == 1) {
        for (int j = 0; j < word.length(); j++) {
            char originalChar = word.charAt(j);
            char reversedChar = reversedWord.charAt(j);

            if (Character.isUpperCase(originalChar)) {
                reversedWord.setCharAt(j, Character.toUpperCase(reversedChar));
            } else if (Character.isLowerCase(originalChar)) {
                reversedWord.setCharAt(j, Character.toLowerCase(reversedChar));
            }
        }
    }
    modifiedSentence.append(reversedWord);
    if (i < words.length - 1) {
        modifiedSentence.append(" ");
    }
}
return modifiedSentence.toString();
}
}

```

| Input                              | Expected                      | Got                           |   |
|------------------------------------|-------------------------------|-------------------------------|---|
| Wipro Technologies Bangalore<br>0  | orpiW seigolonhceT erolagnaB  | orpiW seigolonhceT erolagnaB  | ✓ |
| Wipro Technologies, Bangalore<br>0 | orpiW ,seigolonhceT erolagnaB | orpiW ,seigolonhceT erolagnaB | ✓ |
| Wipro Technologies Bangalore<br>1  | Orpiw Seigolonhcet Erolagnab  | Orpiw Seigolonhcet Erolagnab  | ✓ |
| Wipro Technologies, Bangalore<br>1 | Orpiw ,seigolonhceT Erolagnab | Orpiw ,seigolonhceT Erolagnab | ✓ |

Passed all tests!

### Question 3

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.

**For example:**

| Input | Result |
|-------|--------|
| a b c | 8      |
| b c   |        |

### CODING

```
import java.util.Scanner;

public class CommonAlphabets {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String input1 = sc.nextLine();

        String input2 = sc.nextLine();

        sc.close();

        char[] array1 = input1.replace(" ", "").toCharArray();

        char[] array2 = input2.replace(" ", "").toCharArray();

        int sum1 = 0;

        for (char c1 : array1) {

            for (char c2 : array2) {

                if (c1 == c2) {

                    sum1 += (int) c1;

                    break;

                }

            }

        }

    }

}
```

```
    }  
    int singleDigitSum = getSingleDigitSum(sum1);  
    System.out.println(singleDigitSum);  
}  
private static int getSingleDigitSum(int number) {  
    while (number >= 10) {  
        int sum = 0;  
        while (number > 0) {  
            sum += number % 10;  
            number /= 10;  
        }  
        number = sum;  
    }  
    return number;  
}  
}
```

| Input | Expected | Got |   |
|-------|----------|-----|---|
| a b c | 8        | 8   | ✓ |
| b c   |          |     |   |

Passed all tests!

# **EXPENSE TRACKER**

## **A MINI PROJECT REPORT**

**Submitted by**

**Shreejith S**

**231001194**

**Shyam Narayanan S**

**231001200**

**Sudarsun P**

**231001218**

In partial fulfillment for the award of the degree of

**BACHELOR OF**

**ENGINEERING IN**

**COMPUTER SCIENCE AND ENGINEERING**



**RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)**

**THANDALAM**

**CHENNAI-**

**602105**

## **BONAFIDECERTIFICATE**

Certified that this project report “**EXPENSE TRACKER** ” is the bonafide work of “**Shreejith S, Sudarsun P, Shyam Narayanan S**” who carried out the project work under my supervision.

### **SIGNATURE**

**MR. Narayana K.E**  
**Asst. Professor**  
**Computer Science and Engineering,**  
**Rajalakshmi Engineering College**  
**(Autonomous),**  
**Chennai-602105**

**Submitted for the Practical Examination held on**

\_\_\_\_\_

## **ABSTRACT**

The Expense Tracker Application is an automated software solution designed to facilitate effective financial management for individuals and businesses. The system aims to enhance budgeting, expense tracking, and overall financial awareness, ensuring that users can monitor their financial activities with ease and precision.

The primary features of the application include maintaining a comprehensive record of expenses, categorizing spending by type (such as income, groceries, entertainment, etc.), and generating detailed reports on financial performance. Users can effortlessly add, update, or remove transactions, capturing essential details such as date, category, description, and amount spent.

Additionally, the application provides a user-friendly interface for visualizing financial data through charts and graphs, enabling users to identify spending trends and make informed decisions. Budgeting tools allow users to set spending limits for various categories, helping them stay within their financial goals. Alerts and reminders can be configured to notify users of upcoming bills or budget thresholds.

# TABLE OF CONTENTS

## Chapter 1

### 1. Introduction

|                       |   |
|-----------------------|---|
| 1.1 Introduction..... | 1 |
| 1.2 Objective .....   | 1 |
| 1.3 module .....      | 1 |

## Chapter 2

### 2. Survey of Technologies

|                               |   |
|-------------------------------|---|
| 2.1 Software description..... | 2 |
| 2.2 Languages .....           | 2 |
| 2.2.1 JAVA.....               | 2 |
| 2.2.2 PHP.....                | 2 |
| 2.2.3 MySQL .....             | 3 |

## Chapter 3

### 3. Requirements and analysis

|                                              |   |
|----------------------------------------------|---|
| 3.1 Requirement Specification.....           | 4 |
| 3.2 Hardware and Software Requirements ..... | 4 |
| 3.3 Data flow diagram.....                   | 4 |

|                        |    |
|------------------------|----|
| 3.4 ER-Diagram .....   | 11 |
| 3.5 Normalization..... | 12 |

## **Chapter 4**

### **Program code**

|                                            |    |
|--------------------------------------------|----|
| 4.1 Code Details and Code Efficiency ..... | 17 |
|--------------------------------------------|----|

## **Chapter 5**

### **5. Result and Discussion**

|                              |    |
|------------------------------|----|
| 5.1. User Documentation..... | 42 |
|------------------------------|----|

## **Chapter 6**

### **6. Testing**

|                                |    |
|--------------------------------|----|
| 6.1. Unit Testing .....        | 46 |
| 6.2. Integration Testing ..... | 46 |
| 6.3 System Testing .....       | 46 |
| 6.4. Acceptance Testing.....   | 46 |

## **Chapter 7**

### **7. Conclusion**

|                      |   |
|----------------------|---|
| 7.1 conclusion ..... | 4 |
|----------------------|---|



# Ch 1. INTRODUCTION

## 1.1 INTRODUCTION

### Introduction to the Expense Tracker Application

In an increasingly complex financial landscape, managing personal finances has become more crucial than ever. The Expense Tracker Application is designed to empower individuals and small businesses by providing a comprehensive tool for monitoring and managing their financial activities. This innovative application simplifies the process of tracking expenses, budgeting, and analyzing spending habits, enabling users to gain greater control over their financial well-being.

With its user-friendly interface and robust features, the Expense Tracker Application caters to users of all financial backgrounds. Whether you're a student trying to manage your monthly allowance, a professional looking to keep tabs on your spending, or a small business owner aiming to track expenses and income, this application is tailored to meet your needs.

The application automates the tedious task of recording transactions, categorizing expenses, and generating insightful reports, allowing users to focus on what truly matters—achieving their financial goals. By providing real-time data and analytical tools, the Expense Tracker Application not only enhances financial awareness but also promotes responsible spending habits.

Join the growing community of users who have taken charge of their finances with the Expense Tracker Application. Experience the ease of managing your money, setting budgets, and making informed decisions that pave the way for a more secure financial future. Start your journey towards financial empowerment today.

## 1.2 OBJECTIVE

Automation of Expense Tracking: Streamline the process of recording and categorizing expenses in real-time, allowing users to easily input transactions and maintain an accurate record of their financial activities.

Budget Management: Enable users to set and manage budgets for different spending categories, providing tools to track progress and receive alerts when approaching budget limits.

## 1.3 MODULE

- User Authentication Module:
- Category Management Module
- Data Synchronization
- Module:

## **Ch 2. SURVEY OF TECHNOLOGY**

### **2.1 SOFTWARE DESCRIPTION**

#### **Visual studio Code**

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging.

First and foremost, it is an editor that gets out of your way. The delightfully frictionless edit-build-debug cycle means less time fiddling with your environment, and more time executing on your ideas.

### **2.2 LANGUAGES**

#### **2.2.1 Java**

Description: Java is a high-level, object-oriented programming language that is widely used for building cross-platform applications. It is known for its portability, scalability, and robust performance, making it a popular choice for developing serverside applications and Android mobile apps.

Role in the App: In the Expense Tracker Application, Java is used to implement the backend logic, including managing user sessions, processing transactions, and handling business rules. It also facilitates the interaction between the application and the database, ensuring data integrity and security.

#### **2.2.2 PHP**

Description: PHP (Hypertext Preprocessor) is a server-side scripting language designed primarily for web development. It is open-source and widely used for creating dynamic web pages and applications. PHP can be embedded into HTML and is known for its ease of use and flexibility.

Role in the App: In the Expense Tracker Application, PHP is utilized to handle server-side logic, including user authentication, form processing for adding and updating transactions, and retrieving data from the MySQL database. PHP enables the application to generate dynamic content based on user interactions, providing a responsive and interactive user experience.

### 2.2.3 MySQL

Description: MySQL is a relational database management system (RDBMS) that uses Structured Query Language (SQL) for accessing and managing data. It is renowned for its performance, reliability, and ease of use, making it a leading choice for web applications.

Role in the App: MySQL serves as the database for the Expense Tracker Application, storing all user data, transaction records, and application settings. It allows for efficient querying and manipulation of data, enabling features such as transaction history retrieval, reporting, and data analysis.

## **Ch 3. REQUIREMENT AND ANALYSIS**

### **3.1 REQUIREMENTS SPECIFICATION**

#### **User Requirements**

The system requirement in library management focuses on the possibility of searchfor books by title, author or subject by the member.

#### **System Requirements**

There should be a database backup of the library management system. Operating system should be WindowsXP or a higher version of windows.

### **3.2 HARDWARE AND SOFTWARE REQUIREMENTS**

#### **3.3 Software Requirements**

- Operating System Windows 11
- Front End: Java
- Back End: PHP, MySQL **Hardware Requirements**
- Desktop PC or a Laptop

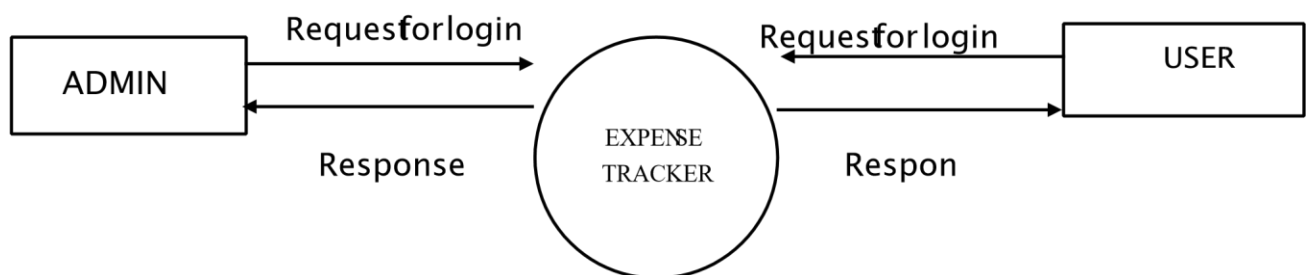
- Printer
- Operating System – Windows 11
- Intel® Core™ i7-13650HX CPU @ 2.00GHz • 24.00 GB RAM
- 64-bit operating system, x64 based processor • 2160 x 1440 monitor resolution
- Keyboard and Mouse

### 3.4 DATA FLOW DIAGRAM

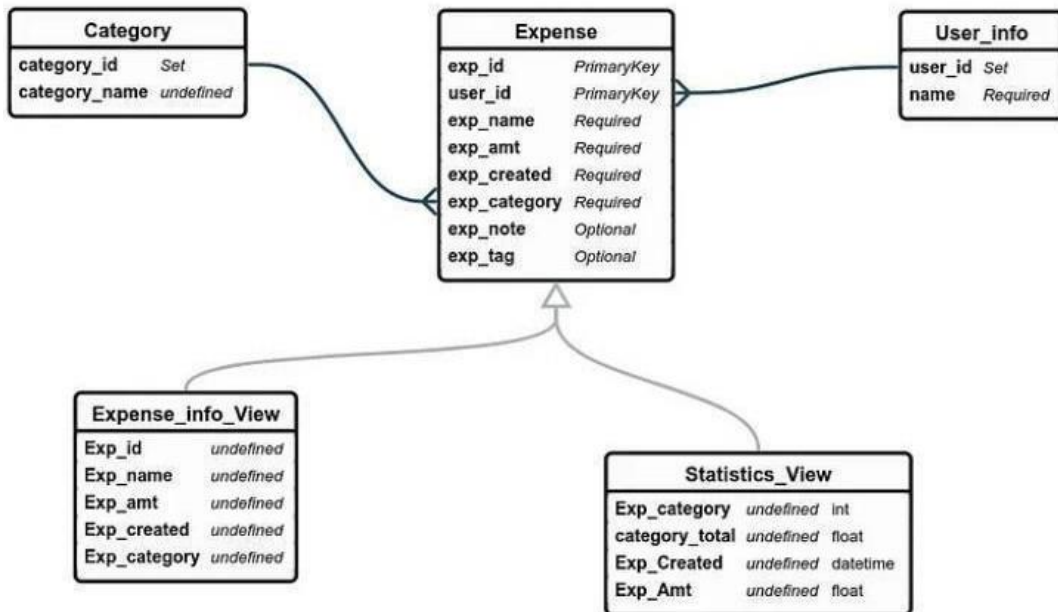
DFD is an important tool used by system analysis. A data flow diagram model, a system using external entities from which data flows through a process which transforms the data and creates output data transforms which go to other processes external entities such as files. The main merit of DFD is that it can provide an overview of what data a system would process.

- A data-flow diagram is a way of representing a flow of data through a processor a system.
- The DFD also provides information about the outputs and inputs of each entity and the process itself.

#### CONTEXT LEVEL DFD :



### 3.5 E-R DIAGRAM



### 3.6 NORMALIZATION

#### Raw database

| ID | Date       | Type    | Description      | Amount |
|----|------------|---------|------------------|--------|
| 1  | 2024-11-01 | Expense | Grocery Shopping | 1000   |
| 2  | 2024-11-02 | Income  | Salary           | 5000   |
| 3  | 2024-11-03 | Expense | Movie            | 300    |
| 4  | 2024-11-04 | Income  | Freelance Work   | 1500   |
| 5  | 2024-11-04 | Expense | Dinner           | 200    |

#### First Normal Form (1NF)

| ID | Date       | Type    | Description      | Amount |
|----|------------|---------|------------------|--------|
| 1  | 2024-11-01 | Expense | Grocery Shopping | 1000   |
| 2  | 2024-11-02 | Income  | Salary           | 5000   |
| 3  | 2024-11-03 | Expense | Movie            | 300    |
| 4  | 2024-11-04 | Income  | Freelance Work   | 1500   |
| 5  | 2024-11-04 | Expense | Dinner           | 200    |



Second Normal Form (2NF)

EXPENSE TABLE

| ID | Type    | Description      | Amount |
|----|---------|------------------|--------|
| 1  | Expense | Grocery Shopping | 1000   |
| 2  | Income  | Salary           | 5000   |
| 3  | Expense | Movie            | 300    |
| 4  | Income  | Freelance Work   | 1500   |
| 5  | Expense | Dinner           | 200    |

DATE TABLE

| ID | Date       |
|----|------------|
| 1  | 2024-11-01 |
| 2  | 2024-11-02 |
| 3  | 2024-11-03 |
| 4  | 2024-11-04 |
| 5  | 2024-11-04 |





## Ch 4. PROGRAM CODE

### 4.1. CODE DETAILS AND CODE EFFICIENCY

```
package ExpenseTracker;

import java.sql.Connection; import
java.sql.DriverManager;
import java.sql.SQLException;

public class DataBaseConnector {
    private static final String DB_NAME = "expense_income_db";
    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/" + DB_NAME;private static final String USER =
    "root";
                                private static final String PASSWORD = "";

    public static Connection getConnection() {Connection
    connection = null;

    try { Class.forName("com.mysql.cj.jdbc.Driver");
    connection = DriverManager.getConnection(JDBC_URL, USER, PASSWORD);

    System.out.println("Connected to the database");
    } catch (ClassNotFoundException | SQLException ex) { System.out.println("Connection - ClassNotFoundException: " +
    ex.getMessage());
    }

    return connection;
    }
}
```

```
public ExpenseAndIncomeTrackerApp() {
    JFrame frame = new JFrame();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(800, 500);
    frame.setLocationRelativeTo(null);
    frame.setUndecorated(true);
    frame.getRootPane().setBorder(BorderFactory.createMatteBorder(5, 5, 5, 5, new Color(52, 73, 94)));

    JPanel titleBar = new JPanel();
    titleBar.setLayout(null);
    titleBar.setBackground(new Color(52, 73, 94));
    titleBar.setPreferredSize(new Dimension(frame.getWidth(), 30));
    frame.add(titleBar, BorderLayout.NORTH);

    JLabel titleLabel = new JLabel("AURA-THE EXPENSE TRACKER");
    titleLabel.setForeground(Color.WHITE);
    titleLabel.setFont(new Font("Arial", Font.BOLD, 15));
    titleLabel.setBounds(10, 0, 250, 30);
    titleBar.add(titleLabel);

    JLabel closeLabel = new JLabel("x");
    closeLabel.setForeground(Color.WHITE);
    closeLabel.setFont(new Font("Arial", Font.BOLD, 17));
    closeLabel.setHorizontalAlignment(SwingConstants.CENTER);
    closeLabel.setBounds(frame.getWidth() - 50, 0, 30, 30);
    closeLabel.setCursor(new Cursor(Cursor.HAND_CURSOR));

    closeLabel.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            System.exit(0);
        }
    });
}
```

```

    }

private void updateTotals(Transaction transaction, double previousAmount, String type) {

    // Calculate the difference in amount

    double difference = transaction.getAmount() - previousAmount;

    // Update total amount based on the type
    if (type.equals("Income")) {
        totalAmount += difference; // Adjust total amount
        double currentIncome = Double.parseDouble(incomeLabel.getText().replace("Total Income: ₹", ""));
        currentIncome += difference; // Update total income
        incomeLabel.setText("Total Income: ₹" + String.format("%.2f", currentIncome));
    } else if (type.equals("Expense")) {
        totalAmount -= difference; // Adjust total amount
        double currentExpense = Double.parseDouble(expenseLabel.getText().replace("Total Expense: ₹", ""));
        currentExpense += difference; // Update total expense
        expenseLabel.setText("Total Expense: ₹" + String.format("%.2f", currentExpense));
    }

    // Update total balance label
    totalLabel.setText("Total Balance: ₹" + String.format("%.2f", totalAmount));
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> new Login());
}

addTransactionButton = new JButton("Add Transaction");
addTransactionButton.setBackground(new Color(41, 128, 150));
addTransactionButton.setForeground(Color.WHITE);
addTransactionButton.setFocusPainted(false);
addTransactionButton.setBorderPainted(false);
addTransactionButton.setFont(new Font("Arial", Font.BOLD, 14));

```

```
addTransactionButton.setCursor(new Cursor(Cursor.HAND_CURSOR));
```

```
removeTransactionButton = new JButton("Remove Transaction");
removeTransactionButton.setBackground(new Color(231, 76, 60));
removeTransactionButton.setForeground(Color.WHITE); removeTransactionButton.setFocusPainted(false);
removeTransactionButton.setBorderPainted(false); removeTransactionButton.setFont(new Font("Arial",
Font.BOLD, 14));
removeTransactionButton.setCursor(new Cursor(Cursor.HAND_CURSOR));
```

```
// Create the Update Transaction button updateTransactionButton = new
JButton("Update Transaction");
updateTransactionButton.setBackground(new Color(255, 192, 203)); // Pink color
updateTransactionButton.setForeground(Color.WHITE); updateTransactionButton.setFocusPainted(false);
updateTransactionButton.setBorderPainted(false); updateTransactionButton.setFont(new Font("Arial", Font.BOLD,
14));
updateTransactionButton.setCursor(new Cursor(Cursor.HAND_CURSOR));
```

```
calculateMonthlyButton = new JButton("Calculate Monthly Expense");
calculateMonthlyButton.setBackground(new Color(46, 204, 113));
calculateMonthlyButton.setForeground(Color.WHITE); calculateMonthlyButton.setFocusPainted(false);
calculateMonthlyButton.setFont(new Font("Arial", Font.BOLD, 14));
calculateMonthlyButton.setCursor(new Cursor(Cursor.HAND_CURSOR));
```

```
logoutButton = new JButton("Logout"); logoutButton.setBackground(new Color(128, 0, 128)); // Purple
logoutButton.setForeground(Color.WHITE); //
White text logoutButton.setFocusPainted(false); logoutButton.setBorderPainted(true); // Enable
border painting logoutButton.setBorder(BorderFactory.createLineBorder(Color.white, 2)); // Set
```

border color and thickness

```
logoutButton.setFont(new Font("Arial", Font.BOLD, 14)); logoutButton.setCursor(new
Cursor(Cursor.HAND_CURSOR));
```

```
// Add buttons to the panel buttonsPanel.add(addTransactionButton);
buttonsPanel.add(removeTransactionButton);
    buttonsPanel.add(updateTransactionButton); // Add Update buttonhere
        buttonsPanel.add(calculateMonthlyButton);
buttonsPanel.add(logoutButton); dashboardPanel.add(buttonsPanel); String[] columnNames =
{"ID", "Date", "Type", "Description", "Amount"}; tableModel = new
DefaultTableModel(columnNames, 0) {
    @Override
    public boolean isCellEditable(int row, int column) {
false;
    }
};

// Create the transaction table transactionTable = new JTable(tableModel);
transactionTable.setBackground(Color.WHITE); transactionTable.setGridColor(new Color(200, 200, 200));
transactionTable.setRowHeight(25);
    transactionTable.getTableHeader().setBackground(new Color(52, 73, 94));
transactionTable.getTableHeader().setForeground(Color.WHITE); transactionTable.getTableHeader().setFont(new Font("Arial",
Font.BOLD, 12));

// Add custom cell renderer for coloring based on type
transactionTable.setDefaultRenderer(Object.class, new
DefaultTableCellRenderer() {
    @Override public Component
getTableCellRendererComponent(JTable table, Object
value, boolean isSelected, boolean hasFocus, int row, int column) {Component c =
super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row, column);
```



```
String type = (String) table.getModel().getValueAt(row, 2); // Column 2 contains the type

if (type.equals("Expense")) {
    c.setForeground(new Color(231, 76, 60)); // Red color for expense
} else if (type.equals("Income")) {
    c.setForeground(new Color(46, 204, 113)); // Green color for income
} else {
    c.setForeground(Color.BLACK); // Default color
}

// Maintain selection highlighting if
(isSelected) {
    c.setBackground(table.getSelectionBackground());
} else {
    c.setBackground(table.getBackground());
}
return c;
}

});

// Create scroll pane for the table
JScrollPane scrollPane = new JScrollPane(transactionTable); scrollPane.setPreferredSize(new Dimension(500, 300));
dashboardPanel.add(scrollPane);

// Create total amount panel
JPanel totalPanel = new JPanel();
totalPanel.setLayout(new GridLayout(3, 1)); // Change to GridLayout with 3 rows
totalPanel.setBackground(new Color(211, 211, 211));
```

```

        // Total Balance Label
totalAmount));
        totalLabel = new JLabel("Total Balance: ₹" + String.format("%.2f",
totalLabel.setFont(new Font("Arial",Font.BOLD, 16));
totalLabel.setForeground(new Color(52, 73, 94)); totalPanel.add(totalLabel);

        // Total Income Label
incomeLabel = new JLabel("TotalIncome:
₹0.00");
incomeLabel.setFont(new Font("Arial", Font.BOLD, 16)); incomeLabel.setForeground(new Color(46, 204, 113)); //
Green color for income
totalPanel.add(incomeLabel);

        // Total Expense Label
expenseLabel = new JLabel("Total Expense:
₹0.00");
expenseLabel.setFont(new
Font("Arial", Font.BOLD, 16));
expenseLabel.setForeground(new Color(231, 76, 60)); // Red colorfor expense
totalPanel.add(expenseLabel);

dashboardPanel.add(totalPanel);

// Add action listeners for buttons addTransactionButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {showAddTransactionDialog();
    }
});

updateTransactionButton.addActionListener(new ActionListener() { @Override
    public void actionPerformed(ActionEvent e) {showUpdateTransactionDialog();

```

```

    }
});

calculateMonthlyButton.addActionListener(new ActionListener() { @Override
    public void actionPerformed(ActionEvent e) {showMonthlyExpenseDialog();
    }
private void showUpdateTransactionDialog() { int selectedRow
    =
    transactionTable.getSelectedRow();if
    (selectedRow == -1) {
        JOptionPane.showMessageDialog(frame, "Please select a transaction touupdate!",
            "No Selection", JOptionPane.WARNING_MESSAGE);
    }
    return;

    // Retrieve current transaction details
    int transactionId = (int) tableModel.getValueAt(selectedRow, 0); // Assuming ID is in column 0
    String currentDate = (String) tableModel.getValueAt(selectedRow, 1); // Assuming Date is in column 1
    String currentType = (String) tableModel.getValueAt(selectedRow, 2); // Assuming Type is in column 2
    String currentDescription = (String) tableModel.getValueAt(selectedRow, 3);
    // Assuming Description is in column 3
    double currentAmount =
    Double.parseDouble((String)
    tableModel.getValueAt(selectedRow, 4)); // Assuming Amount is in column 4

```

```

        JDialog dialog = new JDialog(frame, "Update Transaction",true);
        dialog.setLayout(new GridLayout(5, 2, 10, 10));
dialog.setSize(400, 250);           dialog.setLocationRelativeTo(frame);

// Fields for updating transaction

        JTextField dateField = new JTextField(currentDate);JComboBox<String> typeCombo = new
JComboBox<>(new String[]{"Expense", "Income"});
typeCombo.setSelectedItem(currentType);
        String[] descriptions = {"Medical", "Shopping", "Entertainment", "Transport","Grocery", "Others"};
        JComboBox<String> descriptionCombo = new JComboBox<>(descriptions);JTextField amountField = new
JTextField(String.valueOf(currentAmount));

        dialog.add(new JLabel("Date:"));
dialog.add(dateField);           dialog.add(new
JLabel("Type:"));           dialog.add(typeCombo);

        dialog.add(new JLabel("Description:")); dialog.add(descriptionCombo);           dialog.add(new
JLabel("Amount:"));dialog.add(amountField);

        JButton updateButton = new JButton("Update");dialog.add(new
JLabel("")); dialog.add(updateButton);

        updateButton.addActionListener(new ActionListener() { @Override    public
void actionPerformed(ActionEvent)
{
            try {
                String date = dateField.getText().trim();
                String type = (String) typeCombo.getSelectedItem();

```

```
String description = (String) descriptionCombo.getSelectedItem();double amount =
Double.parseDouble(amountField.getText().trim());

// Create a Transaction object to update
Transaction transaction = new Transaction(transactionId, date, type,description, amount);
TransactionDAO transactionDAO = new TransactionDAO();transactionDAO.updateTransaction(transaction);
// Update in the database

// Update the table model
updateTableModel(transaction);

// Update total income, total expense, and total balanceupdateTotals(transaction, currentAmount,
type);

dialog.dispose(); // Close the dialog JOptionPane.showMessageDialog(frame, "Transaction
updated
successfully.",
"Success", JOptionPane.INFORMATION_MESSAGE);

} catch (NumberFormatException ex) { JOptionPane.showMessageDialog(dialog, "Please enter a valid
amount!",
"Error", JOptionPane.ERROR_MESSAGE);

}

});

dialog.setVisible(true);
}
```

```

private void showMonthlyExpenseDialog() {
    JDialog dialog = new JDialog(frame, "Calculate Monthly Summary", true); dialog.setLayout(new
GridLayout(3, 2, 10, 10)); dialog.setSize(300, 150);
    dialog.setLocationRelativeTo(frame);

    JComboBox<String> monthCombo = new JComboBox<>(new String[]{"January", "February",
        "March", "April", "May", "June",
        "July", "August", "September", "October", "November", "December"
    });
    JTextField yearField = new JTextField();
    JButton calculateButton = new JButton("Calculate");

    dialog.add(new JLabel("Month:")); dialog.add(monthCombo); dialog.add(new JLabel("Year:"));
        dialog.add(yearField);        dialog.add(new JLabel(""));
        dialog.add(calculateButton);

    calculateButton.addActionListener(new ActionListener() { @Override public
        void
        actionPerformed(ActionEvent e)
        {
            try {
                int month = monthCombo.getSelectedIndex() + 1; // January is 0, so add
1                int year = Integer.parseInt(yearField.getText());

                double monthlyExpense = calculateMonthlyExpense(month, year); double monthlyIncome =
                calculateMonthlyIncome(month, year); double
                netAmount = monthlyIncome - monthlyExpense;
            }
        }
    });
}

```

```

        String message = String.format("Monthly Summary for %s %d:\n\n" + "Total Income: ₹%.2f\n" +
  "Total Expense: ₹%.2f\n" +
  "Net Amount: ₹%.2f",
        monthCombo.getSelectedItem(), year,
        monthlyIncome, monthlyExpense, netAmount);
JOptionPane.showMessageDialog(dialog, message, "Monthly Summary",JOptionPane.INFORMATION_MESSAGE);
        dialog.dispose();
    } catch (NumberFormatException ex) { JOptionPane.showMessageDialog(dialog, "Please enter a valid
        year!",
        "Error", JOptionPane.ERROR_MESSAGE);
    }
}
});

dialog.setVisible(true);
}

```

```

private double calculateMonthlyExpense(int month, int year) { double totalExpense = 0;
    for (int i = 0; i < tableModel.getRowCount(); i++) {
        String dateStr = (String) tableModel.getValueAt(i, 1); // Assuming date is in column 1
        String type = (String) tableModel.getValueAt(i, 2); // Assuming type is in column
2        double amount = Double.parseDouble((String) tableModel.getValueAt(i, 4)); // Assuming
amount is in column 4
    }
}

```

```

        try
    {
        Date date = new SimpleDateFormat("yyyy-MM-dd").parse(dateStr);
        Calendar cal = Calendar.getInstance();                cal.setTime(date);

        if (cal.get(Calendar.MONTH) + 1 == month &&
cal.get(Calendar.YEAR) == year && type.equals("Expense")) {    totalExpense
+= Math.abs(amount);
        }
    } catch (ParseException e) {                e.printStackTrace();
    }
}
return totalExpense;
}
private double calculateMonthlyIncome(int month, int year) {double totalIncome = 0;
    for (int i = 0; i < tableModel.getRowCount();
i++) {
        String dateStr = (String) tableModel.getValueAt(i, 1); // Assuming date is incolumn 1
        String type = (String) tableModel.getValueAt(i, 2);                // Assuming type is incolumn
2                double amount = Double.parseDouble((String) tableModel.getValueAt(i, 4)); // Assuming
amount is in column 4
        try
    {
        Date date = new SimpleDateFormat("yyyy-MM-dd").parse(dateStr);
        Calendar cal = Calendar.getInstance();                cal.setTime(date);

        if (cal.get(Calendar.MONTH) + 1 == month &&cal.get(Calendar.YEAR) == year &&
        type.equals("Income")) {
totalIncome += amount;

```



```

        }
    } catch (ParseException e) {
        e.printStackTrace();
    }
}

return totalIncome;
}

private int addTransaction(JComboBox<String> typeCombo, JComboBox<String>descriptionCombo, JTextField
amountField, JSpinner yearSpinner, JSpinner monthSpinner, JSpinner daySpinner) {    int newId = -1; // Initialize
newId to -1, indicating failure by default

    String type = (String) typeCombo.getSelectedItem();
    String description = (String) descriptionCombo.getSelectedItem();
    double
amount = Double.parseDouble(amountField.getText().trim()); // Get selected date
Calendar calendar =
Calendar.getInstance();
calendar.set(Calendar.YEAR, (Integer)
yearSpinner.getValue());
calendar.set(Calendar.MONTH, (Integer) monthSpinner.getValue() - 1);
calendar.set(Calendar.DAY_OF_MONTH, (Integer) daySpinner.getValue());
Date selectedDate =
calendar.getTime();

    // Format date to string
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
    String dateString =
dateFormat.format(selectedDate);

    // Insert transaction into the main database
    try
(Connection connection = DataBaseConnector.getConnection());
        PreparedStatement psMain = connection.prepareStatement(
            "INSERT INTO transaction_table (Date, Type, Description, Amount)
VALUES (?, ?,
?, ?)",

```

```

Statement.RETURN_GENERATED_KEYS)) {

    // Insert into main transaction table
    psMain.setString(1,
dateString);
    psMain.setString(2, type);
    psMain.setString(3, description);
    psMain.setDouble(4, amount);

    int rowsAffectedMain = psMain.executeUpdate(); if (rowsAffectedMain > 0) {    try
    (ResultSet generatedKeys =psMain.getGeneratedKeys()) {                                if
    (generatedKeys.next())
    {
        newId = generatedKeys.getInt(1); // Get the generated ID
        System.out.println("Transaction inserted successfully with ID: " +
newId);
    }
    }
    } else {

        System.out.println("Failed to insert transaction into the main table.");
    }

    // Validate transaction type before inserting into category table                                if
    (type.equalsIgnoreCase("Expense") || type.equalsIgnoreCase("Income")) { String categoryTableName =
    description.toLowerCase() + "_transactions"; //
    Construct table name                                try (PreparedStatement psCategory =
    connection.prepareStatement(
        "INSERT INTO " + categoryTableName + " (ID, Date, Description,Amount) VALUES (?, ?, ?, ?)")
    {

        // Insert into the respective category table with the sameID
        psCategory.setInt(1, newId); // Use the same ID
        psCategory.setString(2, dateString);
        psCategory.setString(3, description);
        psCategory.setDouble(4, amount);

        int rowsAffectedCategory = psCategory.executeUpdate(); // Execute the

```

```

insert for the category table                                if (rowsAffectedCategory > 0) {
System.out.println("Transaction inserted successfully into " + categoryTableName);

        } else {
            System.out.println("Failed to insert transaction into " + categoryTableName);
        }

    } catch (SQLException e) {
        System.out.println("Error inserting into category table: " + e.getMessage());
    }

} else {

    System.out.println("Error - Invalid transaction type: " + type);

}

} catch (SQLException ex) {
    System.out.println("Error - Data not inserted: " + ex.getMessage());ex.printStackTrace();
} catch (NumberFormatException e) {

    System.out.println("Error - Invalid amount format: " + e.getMessage());

}

return newId;

}

private void updateTableModel(Transaction transaction) {
    for (int i = 0; i <
tableModel.getRowCount(); i++) {
        if ((int) tableModel.getValueAt(i, 0) ==
transaction.getId()) { // Assuming the first column is the ID tableModel.setValueAt(transaction.getDate(), i, 1); //
Update Date tableModel.setValueAt(transaction.getType(), i, 2); // Update Type

```

```

tableModel.setValueAt(transaction.getDescription(), i, 3); // Update Description
tableModel.setValueAt(String.format("%.2f", transaction.getAmount()), i, 4); //Update Amount (formatted)

        break;
    }
}

private void updateTotals(Transaction transaction, double previousAmount, Stringtype) {
    // Calculate the difference in amount

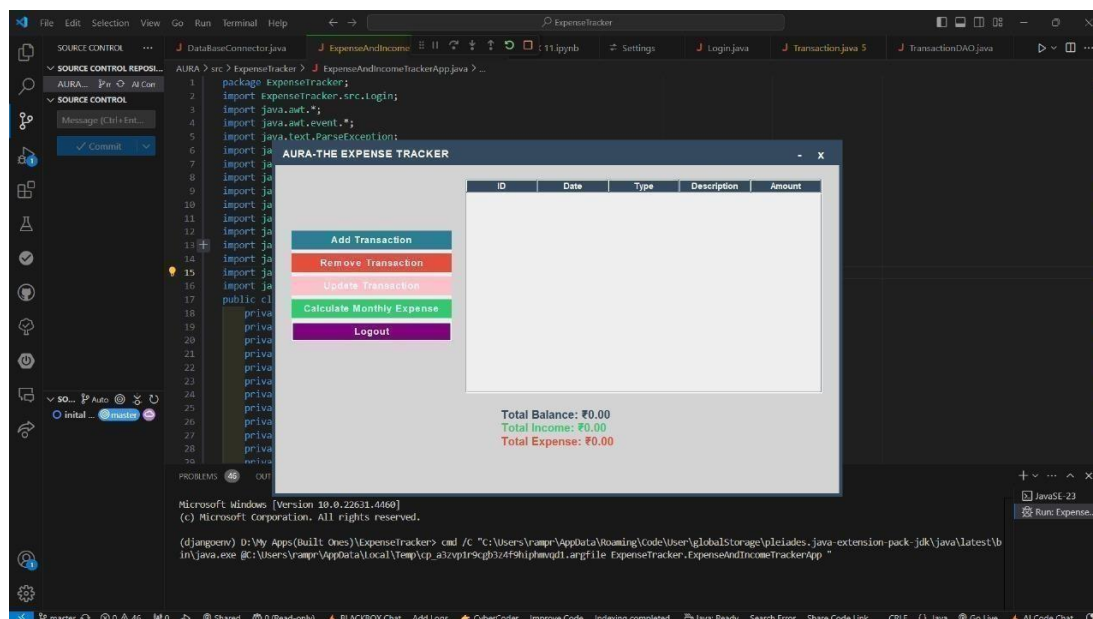
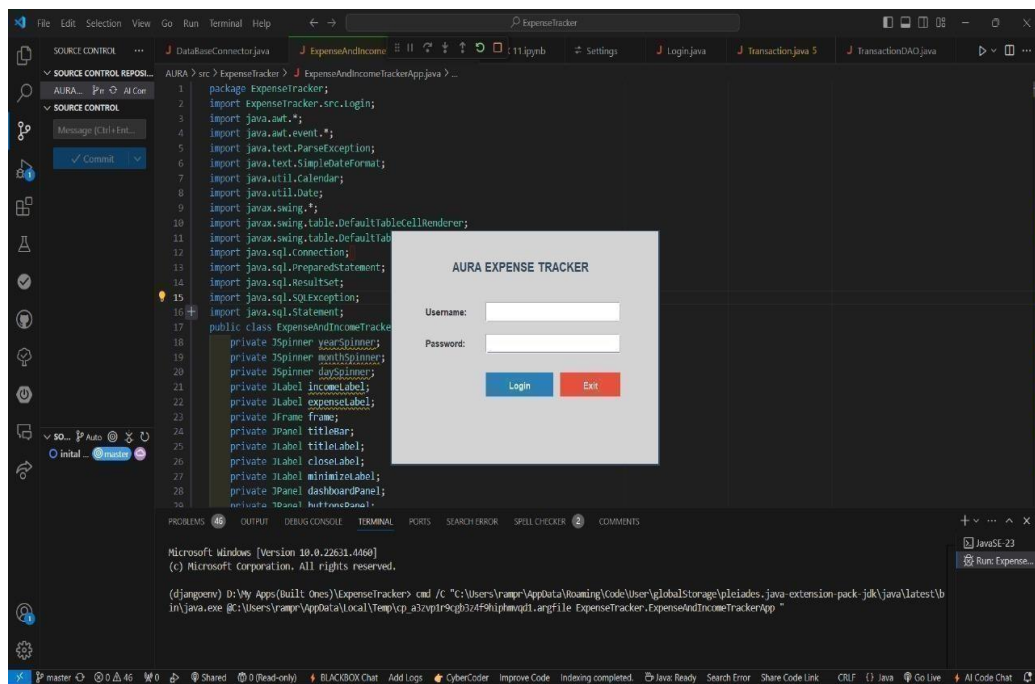
    double difference = transaction.getAmount() - previousAmount;

    // Update total amount based on the type    if (type.equals("Income")) {        totalAmount
+= difference; // Adjusttotal amount        double
currentIncome = Double.parseDouble(incomeLabel.getText().replace ("Total Income: ₹", ""));
        currentIncome += difference; // Update total income
        incomeLabel.setText("Total Income: ₹" +
String.format("%.2f", currentIncome));
    } else if (type.equals("Expense")) {        totalAmount -= difference; // Adjusttotal amount
        double currentExpense =
Double.parseDouble(expenseLabel.getText().replace
("Total Expense: ₹", ""));
        currentExpense += difference; // Update total expense expenseLabel.setText("Total Expense: ₹" +
String.format("%.2f", currentExpense));
    }
    // Update total balance label        totalLabel.setText("Total Balance: ₹" +
String.format("%.2f", totalAmount));
}

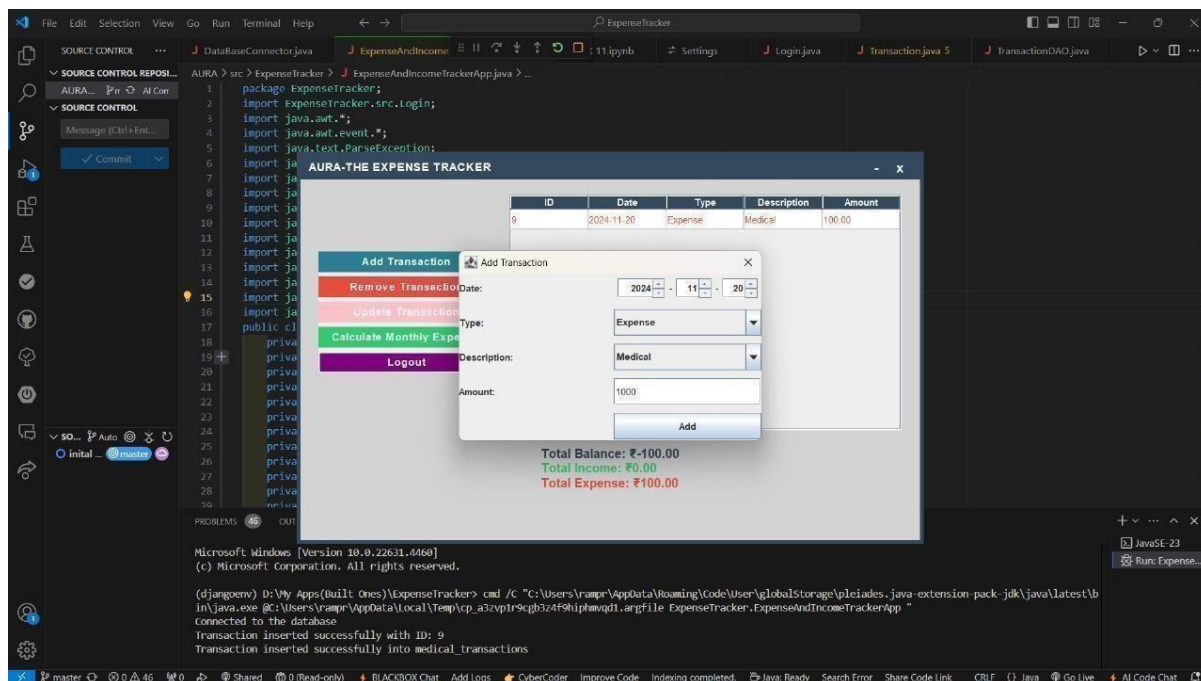
public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> new Login());
}

```

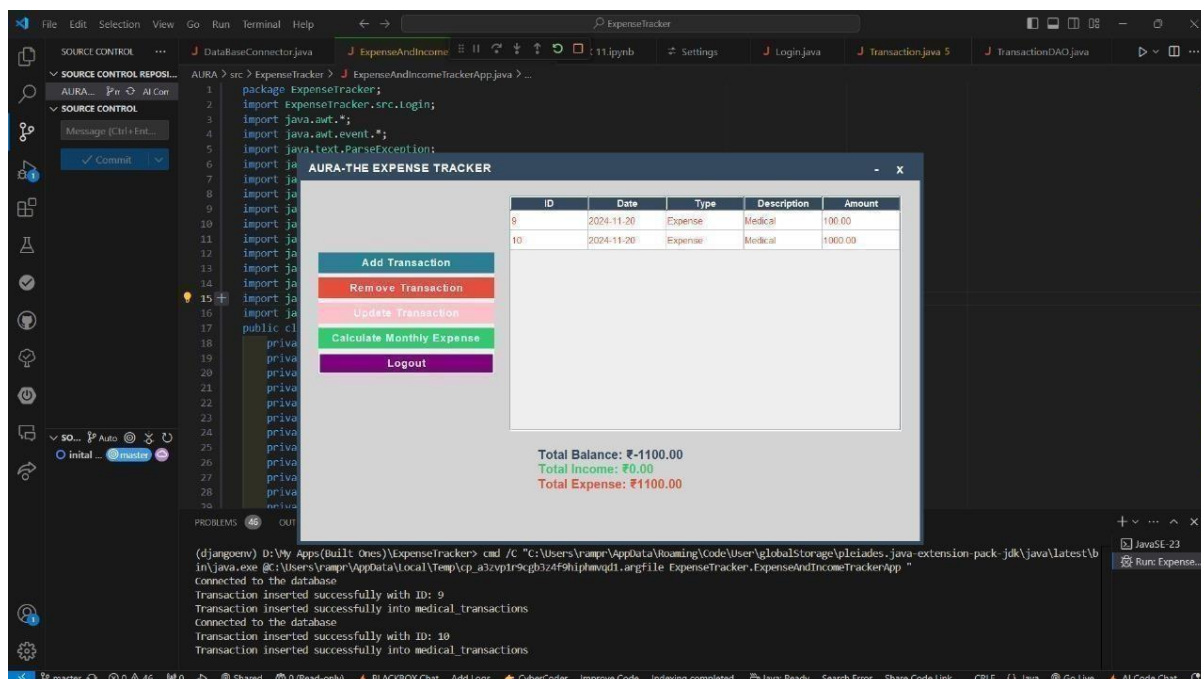
## 5.1 USER DOCUMENTATION



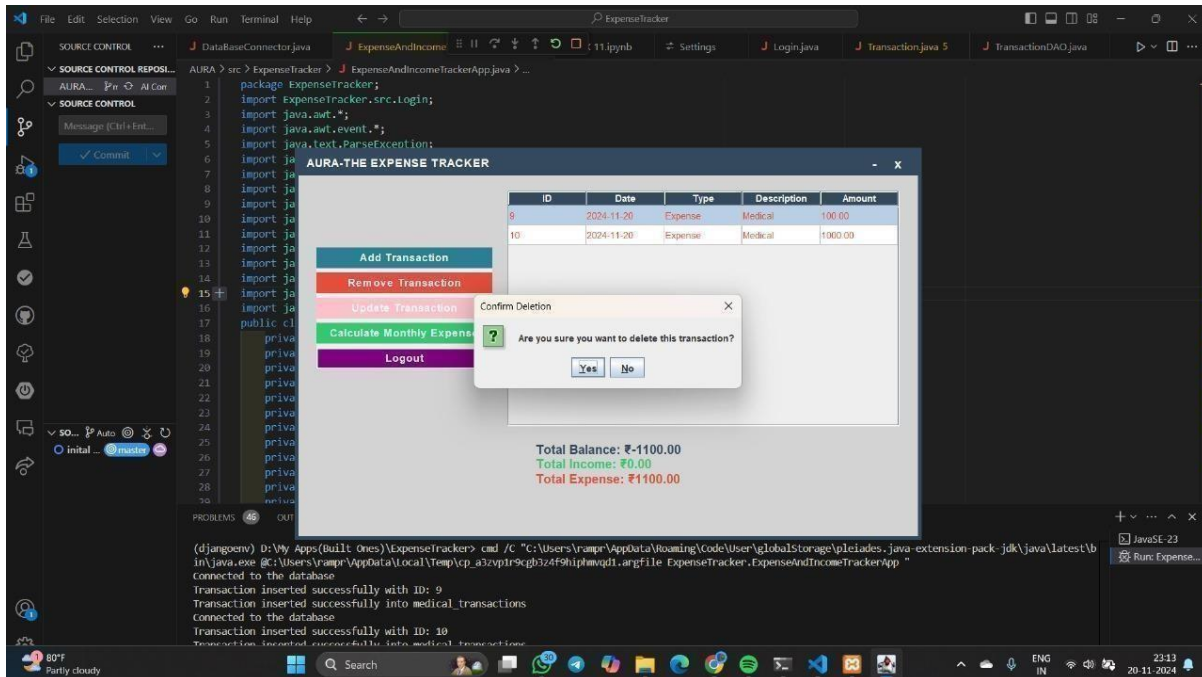
## ADD TRANSACTION



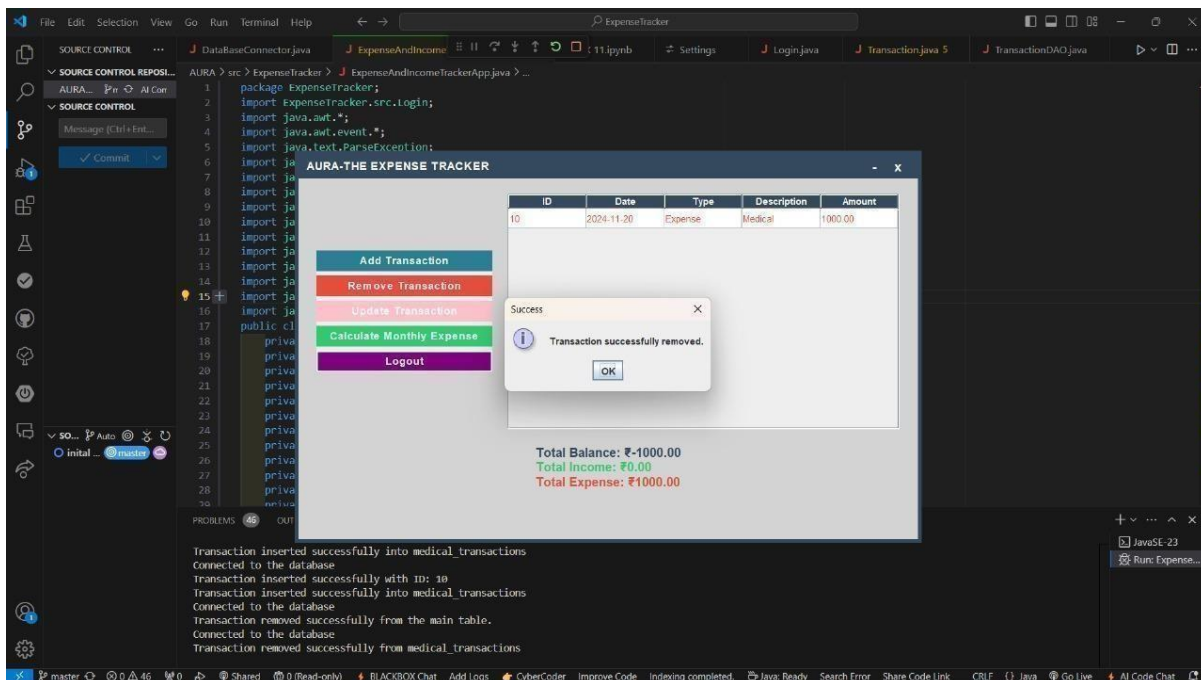
## ADD TRANSACTION RESULT



## REMOVE TRANSACTION

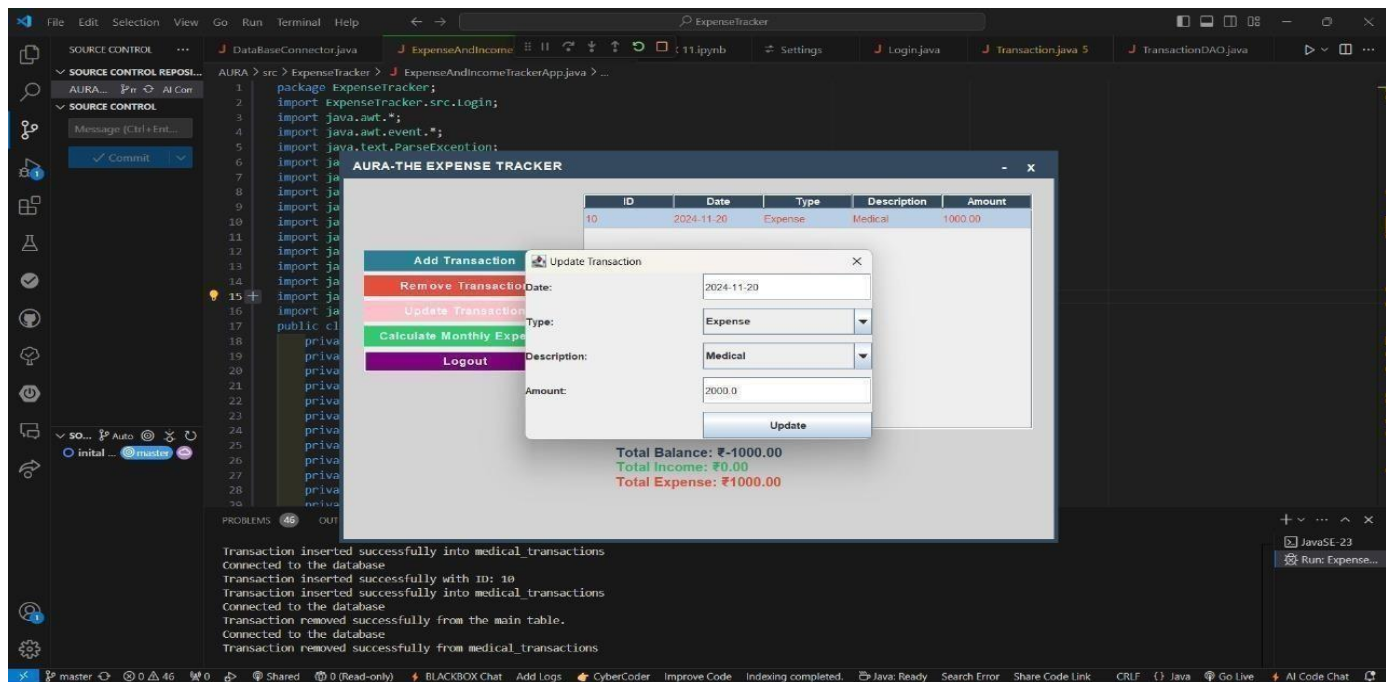


## REMOVE TRANSACTION RESULT

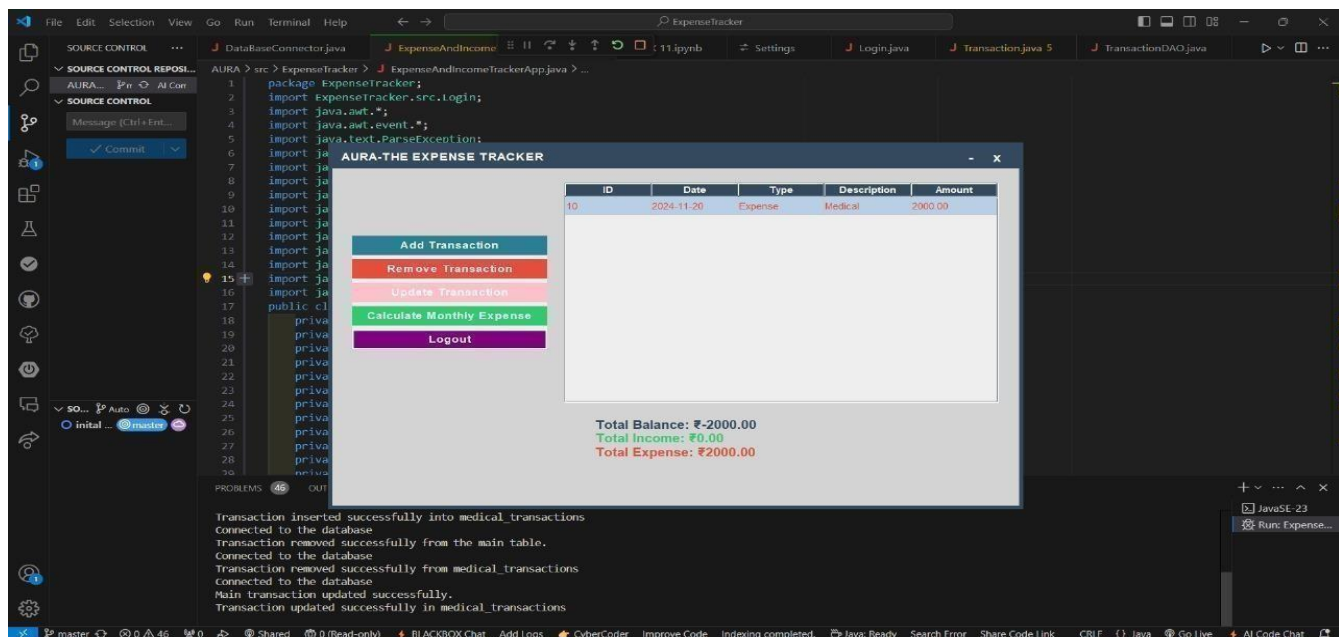




## UPDATE TRANSACTION

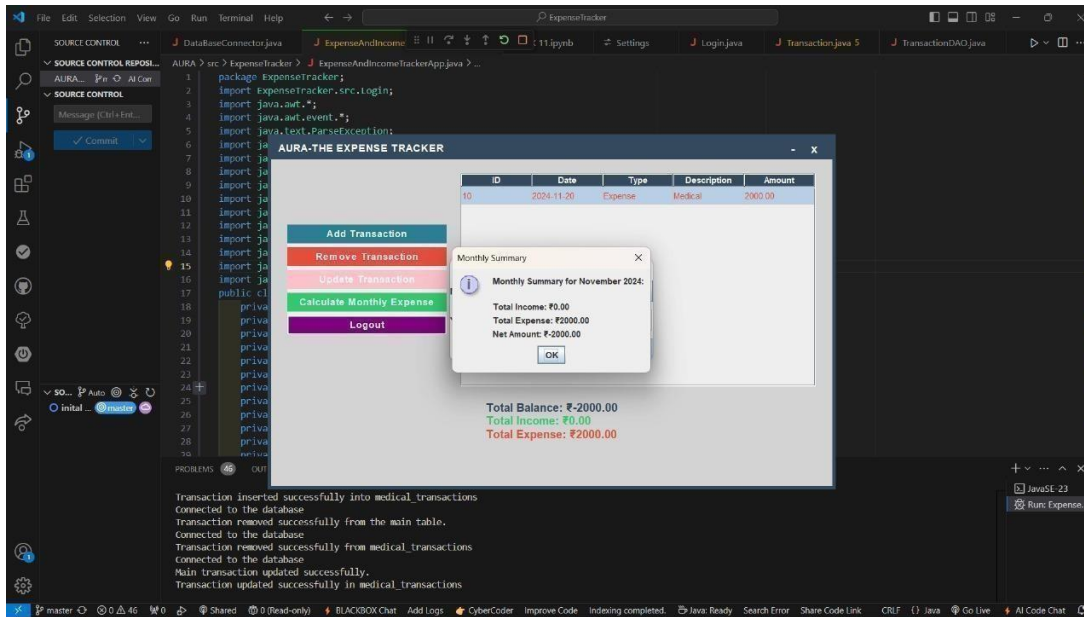


## UPDATE TRANSACTION RESULT

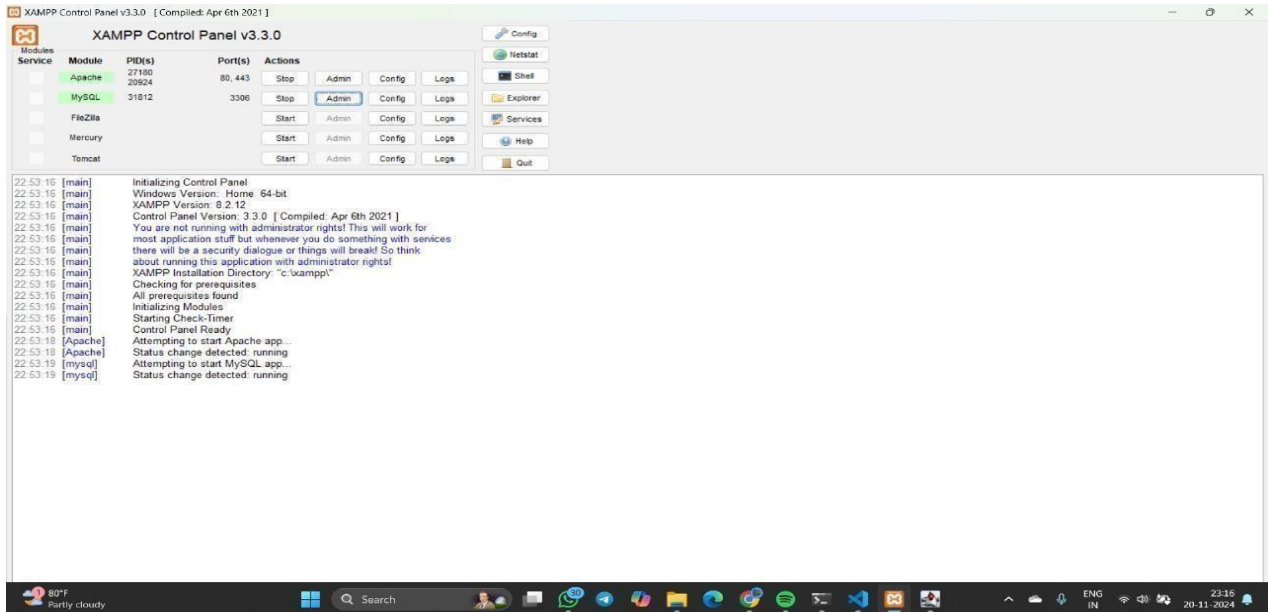




## MONTHLY EXPENSES



## XAMPP HOME PAGE



## PHPMYAdmin MySql Home Page

The screenshot displays the phpMyAdmin interface in a web browser. The browser's address bar shows the URL: `localhost/phpmyadmin/index.php?route=/database/structure&db=expense_income_db`. The phpMyAdmin header indicates the server version is 127.0.0.1 and the selected database is `expense_income_db`. The left sidebar shows a tree view of the database structure, including a 'New' button and a list of tables: `entertainment_transactions`, `grocery_transactions`, `medical_transactions`, `others_transactions`, `shopping_transactions`, `transaction_table`, `transport_transactions`, `food_order`, `information_schema`, `mysql`, `performance_schema`, `phpmyadmin`, and `test`.

The main content area shows the 'Structure' tab for the `expense_income_db` database. It includes a 'Filters' section with a search box. Below this is a table listing the database's tables:

| Table                                                            | Action                                                                                                                                    | Rows | Type   | Collation         | Size    | Overhead |
|------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|------|--------|-------------------|---------|----------|
| <input type="checkbox"/> <code>entertainment_transactions</code> | <a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a> | 3    | InnoDB | latin1_swedish_ci | 16.0 K  | 18       |
| <input type="checkbox"/> <code>grocery_transactions</code>       | <a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a> | 0    | InnoDB | latin1_swedish_ci | 16.0 K  | 18       |
| <input type="checkbox"/> <code>medical_transactions</code>       | <a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a> | 4    | InnoDB | latin1_swedish_ci | 16.0 K  | 18       |
| <input type="checkbox"/> <code>others_transactions</code>        | <a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a> | 1    | InnoDB | latin1_swedish_ci | 16.0 K  | 18       |
| <input type="checkbox"/> <code>shopping_transactions</code>      | <a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a> | 0    | InnoDB | latin1_swedish_ci | 16.0 K  | 18       |
| <input type="checkbox"/> <code>transaction_table</code>          | <a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a> | 0    | InnoDB | latin1_swedish_ci | 16.0 K  | 18       |
| <input type="checkbox"/> <code>transport_transactions</code>     | <a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a> | 0    | InnoDB | latin1_swedish_ci | 16.0 K  | 18       |
| 7 tables Sum                                                     |                                                                                                                                           | 16   | InnoDB | latin1_swedish_ci | 112.0 K | 0        |

Below the table list, there are options to 'Check all' and 'With selected:'. At the bottom, there is a 'Create new table' section with a 'Table name' input field, a 'Number of columns' input field (set to 4), and a 'Create' button.

## TRANSACTION TABLE

[illegible]

## Ch 6. TESTING

### 6.1 Unit Testing

Unit testing is a testing technique in which modules are tested individually. Small individual units of source code are tested to determine whether it is fit to use or not. Different modules of games are put to test while the modules are being developed. Here modules refer to individual levels, players, scenes

### 6.2 Integration Testing

Integration testing is the technique in which individual components or modules are grouped together and tested. It occurs after testing. The inputs for the integrated testing are the modules that have already been unit tested.

### 6.3 System Testing

System testing is conducted on the entire system as a whole to check whether the system meets its requirements or not. software was installed on different systems and any errors or bugs that occurred were fixed.

### 6.4 Acceptance Testing

User Acceptance is defined as a type of testing performed by the Client to certify the system with respect to the requirements that was agreed upon. This testing happens in the final phase of testing before moving the software application to the Market or Production environment.

## Ch 7. CONCLUSION

Upon completion of the Expense Tracker application, we are confident that the issues present in manual expense tracking systems have been effectively addressed. The "EXPENSE TRACKER" application has been designed to automate the process of managing personal finances, thereby reducing human errors and enhancing overall efficiency. The primary focus of this project is to simplify the tracking of expenses and incomes, allowing users to manage their finances with minimal effort.

All transaction records are securely stored in a database, enabling quick and easy retrieval of data. Users can navigate through their financial records seamlessly, with intuitive controls provided throughout the application. For instances where users have a large number of transactions, the application features a search functionality that allows them to quickly find specific records by entering relevant search terms, delivering results almost instantaneously.

Editing transactions has also been streamlined, as users can easily modify any required fields and update their entries with a simple click of a button. Each transaction is assigned a unique identifier, ensuring accurate access and management of financial records without the risk of confusion or error.

The main aim of this project is to empower users with the tools they need to maintain accurate financial records, analyze their spending habits, and make informed decisions regarding their finances. The Expense Tracker application stands as a comprehensive solution for individuals seeking to take control of their financial well-being.

