

**RAJALAKSHMIENGINEERINGCOLLEGE**

**[AUTONOMOUS]**

**RAJALAKSHMI NAGAR,THANDALAM–602105**



**RAJALAKSHMI**  
**ENGINEERING COLLEGE**

**CS23333OBJECT ORIENTED PROGRAMING USING JAVA**

**Laboratory Record NoteBook**

Name:..SHREEKUMARAN .S.....

Year/Branch/Section :..II/IT/D.....

CollegeRollNo.: .....2116231001195.....

Semester:... ..III.....

AcademicYear:... ..2024-2025 .....

**RAJALAKSHMIENGINEERINGCOLLEGE**  
**[AUTONOMOUS]**

**RAJALAKSHMI NAGAR,THANDALAM-602105**

**BONAFIDE CERTIFICATE**

Name :... ..SHREEKUMARAN .S.....

AcademicYear:2024-2025

Semester:.3<sup>rd</sup>sem

Branch:IT-D

**RegisterNo.**

2116231001195

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**

## INDEX

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	5	
3	21.9.24	Arrays	11	
4	1.10.24	Classes And Objects	17	
5	1.10.24	Inheritance	23	
6	3.10.24	String, StringBuffer	29	
7	3.10.24	Interfaces	35	
8	6.10.24	Polymorphism, Abstract Classes, Final Keyword	41	
9	9.10.24	Exceptional Handling	47	
10	4.10.24	Collection-List	52	
11	10.11.24	Set, Map	57	
12	10.11.24	Introduction to I/O, I/O Operations, Object Serialization	63	
13	27.11.24	Java Project Report	72	



## **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());
}
}
```

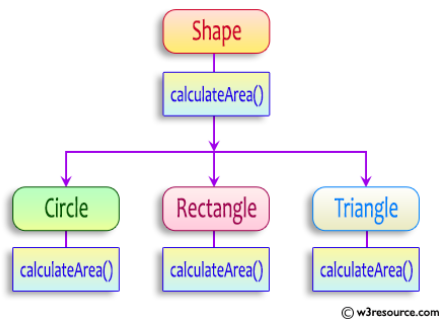
| Input                  | Expected         | Got              |   |
|------------------------|------------------|------------------|---|
| 3<br>oreo sirish apple | oreoapple        | oreoapple        | ✓ |
| 2<br>Mango banana      | no matches found | no matches found | ✓ |
| 3<br>Ate Ace Girl      | 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><br>First : 30, Last : 80 | ArrayList: [30, 20, 40, 50, 10, 80]<br><br>First : 30, Last : 80 |
|      | 2     | 4<br>5<br>15<br>25<br>35              | ArrayList: [5, 15, 25, 35]<br><br>First : 5, Last : 35           | ArrayList: [5, 15, 25, 35]<br><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();
}
}

```

[illegible]

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 ,seiGolohnhcT 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!

# **GYM MANAGEMENT SYSTEM**

**CS23333 – Object Oriented Programming using Java Project Report**

*Submitted by*

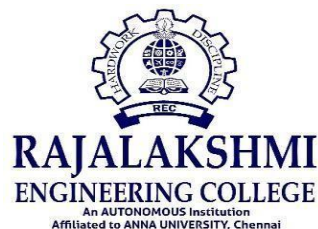
**SRIMATHI B.S -231001209  
SRINITHYA A -231001212  
SHREEKUMARAN S -231001195**

*Of*

**BACHELOR OF TECHNOLOGY**

*In*

**INFORMATION TECHNOLOGY**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**RAJALAKSHMI ENGINEERING COLLEGE**

**NOVEMBER-2024**

## **BONAFIDE CERTIFICATE**

Certified that this project titled “GYM MANAGEMENT SYSTEM” is the bonafide work of “**SRIMATHI B.S(231001209), SRINITHYA A (231001212), SHREEKUMARAN S (231001195)**”who carried out the project work under my supervision.

**SIGNATURE**

**Dr.P.Valarmathie**

**HEAD OF THE DEPARTMENT**

Department of Information Technology  
Rajalakshmi Engineering College

**SIGNATURE**

**Mr.K.E NARAYANAN**

**COURSE INCHARGE**

Department of Information Technology  
Rajalakshmi Engineering College

This project is submitted for CS23333 – Object Oriented Programming using Java held on \_\_\_\_\_

**INTERNAL EXAMINAR**

**EXTERNAL EXAMINAR**

## TABLE OF CONTENTS

| CHAPTER NO | TITLE                                  | PAGE NO |
|------------|----------------------------------------|---------|
|            | List of Figures                        | 4       |
|            | List of Tables                         | 5       |
| 1          | 1.1 Abstract                           | 6       |
|            | 1.1 Introduction                       | 6       |
|            | 1.3 Purpose                            | 6       |
|            | 1.4 Scope of Project                   | 7       |
|            | 1.5 Software Requirement Specification | 7       |
| 2          | <b>System flow Diagrams</b>            | 12      |
|            | 2.1 Use Case Diagram                   | 12      |
|            | 2.2 Entity-relationship Diagrams       | 13      |
|            | 2.3 Data Flow Diagram                  | 13      |
| 3          | <b>Module Description</b>              | 14      |
| 4          | 4.1 Design                             | 15      |
|            | 4.2 Database Design                    | 18      |
|            | 4.3 Code                               | 19      |
| 5          | <b>Conclusion</b>                      | 30      |
| 6          | <b>Reference</b>                       | 30      |

## **LIST OF FIGURES**

| <b>Figure Numbers</b> | <b>Figure Captions</b>  | <b>Pg.no</b> |
|-----------------------|-------------------------|--------------|
| 2.1                   | Use Case Diagram        | 12           |
| 2.2                   | Entity Relation diagram | 13           |
| 2.3                   | Date Flow Diagram       | 13           |
| 4.1.1                 | Login Page              | 15           |
| 4.1.2                 | Registration Page       | 15           |
| 4.1.3                 | Home Page               | 16           |
| 4.1.4                 | Cart Selection          | 16           |
| 4.1.5                 | Payment Page            | 17           |
| 4.1.6                 | Order Message           | 17           |



## **LIST OF Table**

| <b>Figure Numbers</b> | <b>Figure Captions</b> | <b>Pg.no</b> |
|-----------------------|------------------------|--------------|
| 4.2.1                 | Login Table            | 18           |

## **1. Abstract**

The Gym Management System is a Java-based desktop application developed to manage gym activities efficiently, such as member registration, payments, and trainer assignments. It uses Java Swing for the user interface and JDBC to connect with a MySQL database. By applying object-oriented principles like encapsulation and modularity, the system ensures scalability and maintainability. Features like date input using `jdatepicker` demonstrate Java's extensibility, making this project an effective solution for automating gym operations. It exemplifies Java's practicality in building small-scale, database-driven applications.

## **2. Introduction**

The Gym Management System is a Java-based application designed to automate gym operations, such as managing member data, tracking payments, and assigning trainers. Built with Java Swing for the user interface and JDBC for MySQL database integration, it allows efficient handling of gym-related information. The system follows object-oriented principles, ensuring modularity and scalability. Key features include a date selection component using the `jdatepicker` library. This solution simplifies manual tasks, improves data accuracy, and is suitable for small to medium-sized gyms.

## **3. Purpose**

The primary goal of the Gym Management System is to create an efficient and user-friendly platform that simplifies gym operations for administrators. The objectives of the project include:

- **Automating Gym Operations:** Streamlining tasks like member registration, payment tracking, and trainer assignments to reduce administrative workload.
- **Providing a User-Friendly Interface:** Creating an intuitive interface with Java Swing for easy use by gym staff, regardless of technical expertise.
- **Ensuring Data Accuracy:** Using JDBC for secure database access, ensuring consistent and reliable member, payment, and trainer data.
- **Improving Efficiency:** Minimizing errors and time spent on manual tasks like payment tracking and scheduling.
- **Providing Scalability:** Designing the system with object-oriented principles for easy expansion and maintenance as the gym grows.

#### **4. Scope of the Project:**

The Gym Management System aims to automate and streamline various operations within a gym, providing a comprehensive solution for member management, payment tracking, and trainer assignments. The system includes features such as membership registration, payment tracking, scheduling, and trainer assignments, ensuring efficient data handling and reducing administrative workload. By utilizing Java Swing for the user interface and JDBC for secure database connectivity, the system ensures a smooth and user-friendly experience for gym staff. The database integration with MySQL ensures secure and consistent data management. The system is scalable, allowing for future expansion, such as adding features like online booking or enhanced reporting tools, to meet evolving gym needs.

#### **5. Software Requirement Specification:**

##### Introduction

This section defines the hardware and software requirements for the Gym Management System, ensuring smooth operation, scalability, and efficient user interaction.

##### Product Scope

The system automates key gym operations such as member registration, payment tracking, and trainer assignments. Developed with Java for logic and MySQL for database management, it offers a seamless user experience for both gym staff and members. The system is designed to be scalable, supporting future features like online booking and advanced reporting.

##### References and Acknowledgements

- Java Development Kit (JDK) for application development.
- MySQL documentation for database integration.
- Java Swing for the user interface.
- JDBC API for secure database connectivity.

## Overall Description

The Gym Management System is a software solution developed using Java and MySQL to automate and streamline gym operations. It manages tasks such as member registration, payment tracking, trainer assignments, and scheduling. The system integrates a secure MySQL database via JDBC for reliable data storage, and the Java Swing interface ensures ease of use for gym staff. Scalable and maintainable, the system is designed to handle the evolving needs of small to medium-sized gyms, with the potential for future enhancements like online booking and advanced reporting.

### Product Perspective

The Gym Management System is a Java and MySQL-based solution that automates and streamlines gym operations. It allows administrators to manage member registrations, payments, trainer assignments, and class schedules efficiently. The system uses JDBC for secure database connectivity and features a Java Swing interface for ease of use. Scalable and cost-effective, it is designed for small to medium-sized gyms and can be expanded with future features like online booking or advanced reporting. This system can be easily integrated into existing gym management processes, offering a modern solution for operational efficiency.

### Product Functionality

- **Member Registration:** Allows members to register and update their profiles.
- **Payment Tracking:** Tracks member payments and dues in real-time.
- **Trainer Assignment:** Assigns trainers to members and manages schedules.
- **Scheduling:** Enables booking and managing gym classes or sessions.
- **Database Integration:** Uses MySQL for secure data storage and access.
- **User Interface:** Provides an intuitive Java Swing interface for easy navigation.
- **Scalability:** Supports future features like online booking and reporting.

### User and Characteristics

- **Qualification:** Users should have at least basic educational qualifications, such as matriculation, and be comfortable with English for understanding system instructions and managing data effectively.
- **Experience:** Familiarity with gym operations or membership management is beneficial, though not required. Administrators will benefit from prior experience in managing schedules, payments, and memberships.
- **Technical Experience:** Users are expected to have elementary knowledge of computers and should be comfortable with operating basic software tools (like navigating windows, entering data into forms, etc.) to interact with the system efficiently.

## Operating Environment

### *Hardware Requirements*

- Processor: Intel i3 or higher (or equivalent AMD processor)
- Operating System: Windows 8,10, 11
- Processor Speed: 2.0 GHz
- RAM: 4GB
- Hard Disk: 500GB

### *Software Requirements*

- System Requirements
- Software Requirements
- Database Requirements
- Development Tools
- Libraries and Frameworks
- Network Requirements

### *Constraints*

- *System Access: Access is limited to authorized personnel only, ensuring that sensitive data is protected and that the system is used by designated staff members only.*
- *Data Volume: The system is designed to handle a moderate volume of data, such as member details, payments, and schedules. However, for very large datasets (e.g., large gym chains), additional optimizations may be required to maintain system performance.*
- *Internet Connectivity: The system requires a stable internet connection for updates, remote access, and future integration of online features, which could be a limitation in regions with unreliable internet service.*
- *User Knowledge: Users should have basic computer literacy and some familiarity with gym operations. Those with limited technical expertise might require some training to navigate the system effectively.*
- *Hardware Limitations: The system is optimized for computers with at least 4 GB RAM and 2 GB disk space. Performance may degrade on systems with lower specifications.*

## User Interface

The Medical E-Commerce Store provides user-friendly, menu-driven interfaces for:

- **Register:** Allows new users to create an account by providing essential information like name, contact details, and membership preferences.
- **Login:** Enables existing users (administrators, trainers, or members) to log in using their credentials (username and password) for personalized access.
- **Member Dashboard:** Displays personalized details like membership status, upcoming sessions, payment history, and profile settings.
- **Trainer Assignment:** Admins can assign trainers to specific members and track their schedules.
- **Payment Tracking:** Admins can view and manage member payments, ensuring that dues are up to date.
- **Scheduling:** Allows users (administrators and members) to view and manage gym sessions, including booking and canceling classes or training sessions.
- **Order/Transaction History:** Members can view and track previous transactions, payments, and appointments, providing transparency and easy access to their gym-related activities.

## **Hardware Interface**

- Screen resolution of at least 640 x 480 or above.
- Compatible with any version of Windows 8, 10, 11.

## **Software Interface**

- Login and Registration Screens
- Admin Dashboard
- Member Dashboard
- Trainer Dashboard
- Scheduling Interface
- Payment Interface

## **Functional Requirements**

### User Registration and Authentication:

- Allow users (administrators, trainers, members) to register by providing details like name, email, password, and contact information.  
Provide secure login/logout functionality with encrypted password storage to ensure safe access.
- Gym Member Management:  
Enable administrators to view, update, and manage gym member profiles, including membership status, payment history, and session details.
- Trainer Assignment:  
Allow administrators to assign trainers to members and manage trainer schedules.  
Trainers should be able to view and update their assigned sessions and manage progress tracking.
- Session Scheduling:  
Allow members to book or cancel gym classes and personal training sessions through a scheduling interface.  
Admins can manage and update the class schedule.
- Payment Tracking:  
Enable tracking of member payments for membership and training services.  
Allow members to view their payment status and history.
- Admin Functions:  
Provide admins with tools to manage member, trainer, and session data. Allow for reports on payments, member activity, and session attendance.

## Non-functional Requirements

- **Performance:**  
The system should load within 2-3 seconds and handle a high volume of simultaneous users without delays or crashes.
- **Scalability:**  
The system must be able to accommodate increasing user data, transactions, and gym schedules without significant performance issues.
- **Security:**  
Ensure that all user data is securely stored and transmitted, using encryption methods like SSL for login and payment processes.
- **Availability:**  
The system should be available 99.9% of the time, with minimal downtime for scheduled maintenance.
- **Usability:**  
The system should offer a user-friendly interface that is intuitive for administrators, trainers, and members, ensuring ease of navigation and efficient task completion.
- **Reliability:**  
The system should be stable and consistently perform as expected with minimal downtime or errors.
- **Compatibility:**  
The system should be compatible with major browsers (Chrome, Firefox, Safari) and devices (desktops, tablets, smartphones) to ensure broad accessibility.



## 2.SYSTEM FLOW DIAGRAMS

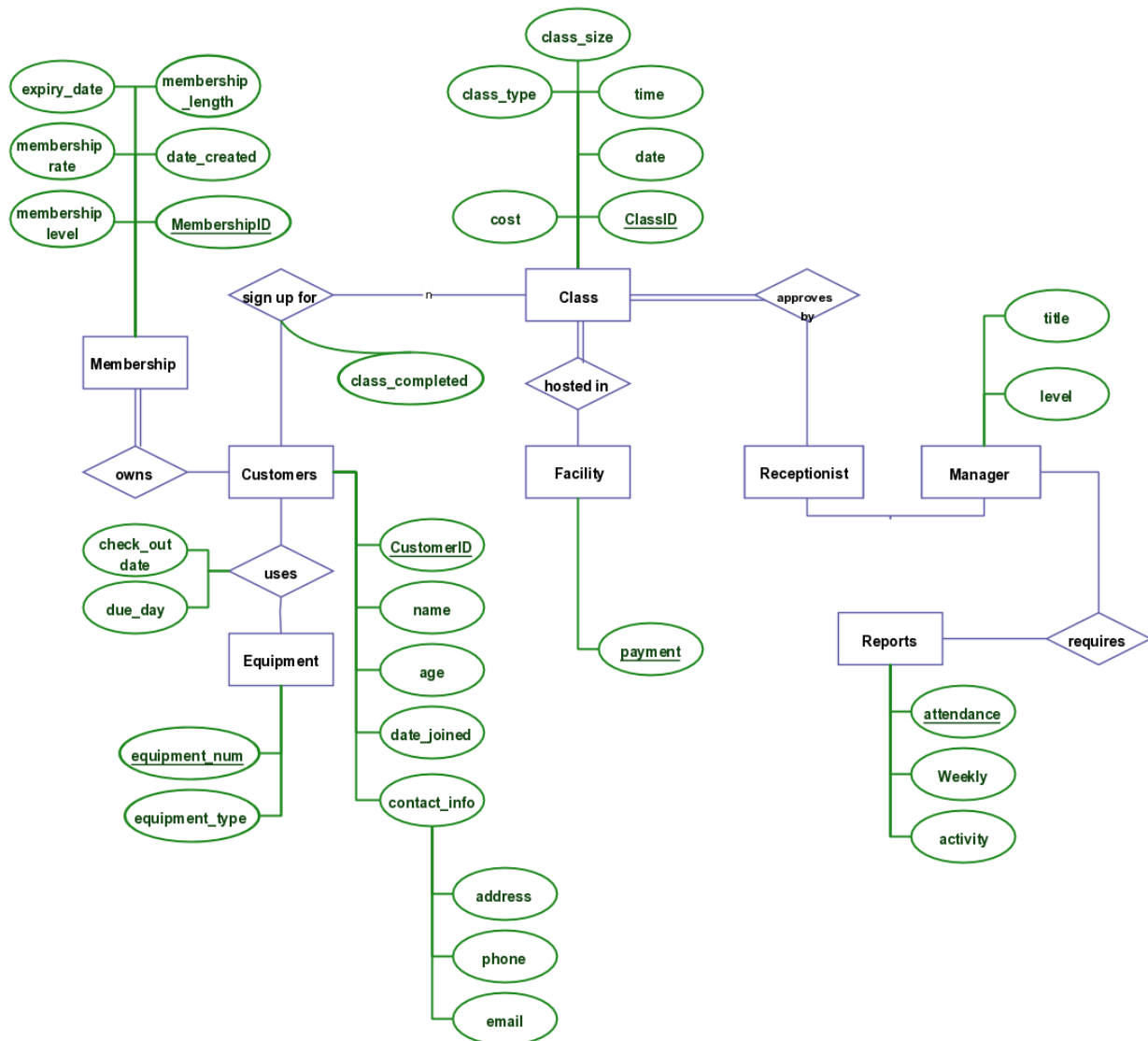


Figure 2.1 Use Case Diagrams

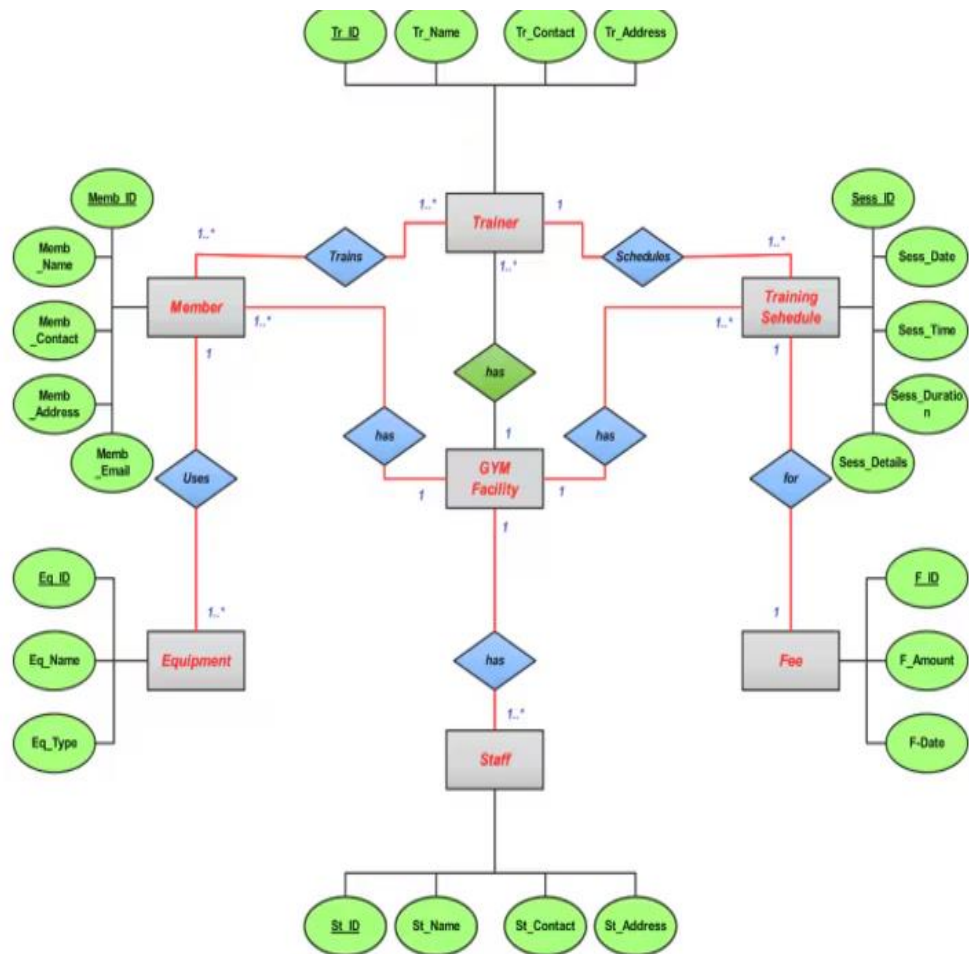


Figure 2.2 Entity Relationship Diagram

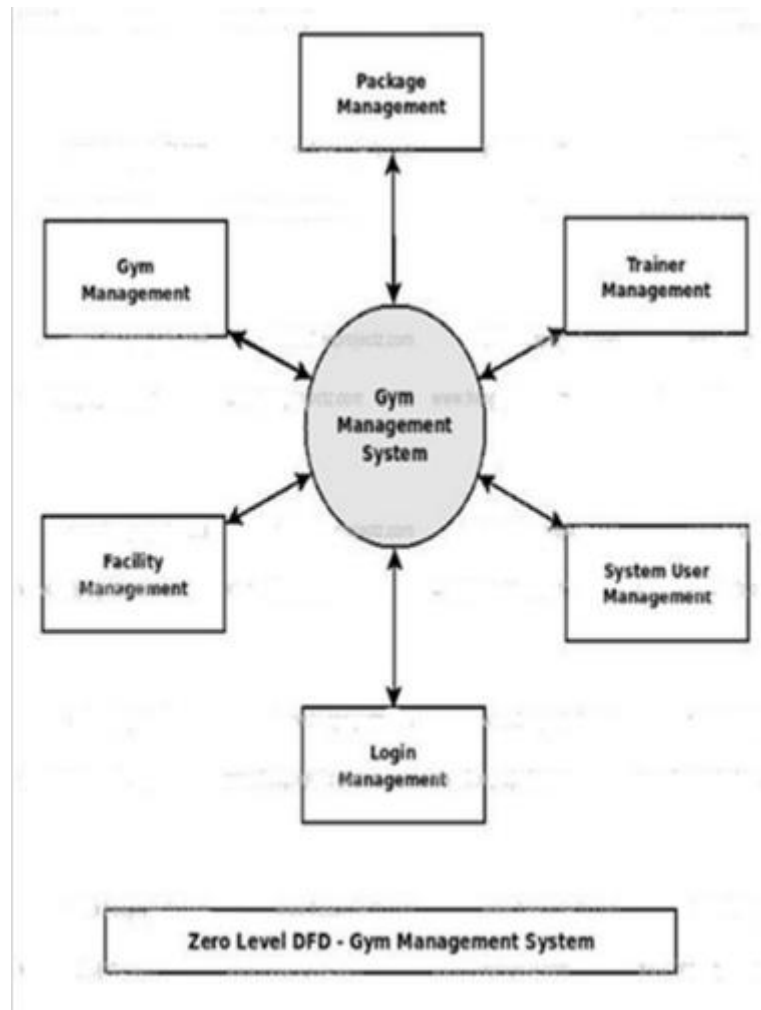


Figure 2.3 Data-flow diagram

### 3. MODULE DESCRIPTION

- **User Management:**  
Handles user registration, login, and profile management for both gym members and administrators. Ensures secure authentication and role-based access.
- **Member Management:**  
Allows admins to manage member profiles, including personal details, membership status, payment history, and session bookings.
- **Session Scheduling and Trainer Assignment:**  
Admins can assign trainers to members, manage schedules, and track session attendance. Members can book and view sessions.
- **Payment and Checkout:**  
Integrates secure payment gateways for processing transactions, including membership and session fees, using various payment methods.
- **Admin Dashboard:**  
Provides admins with an interface to manage sales, track inventory, monitor member activity, and generate reports.
- **Inventory and Equipment Management:**  
Tracks gym equipment and resources, allowing admins to manage stock levels and ensure necessary supplies are available for members.

### DESIGN:


Log In


# LOG IN

Enter username!

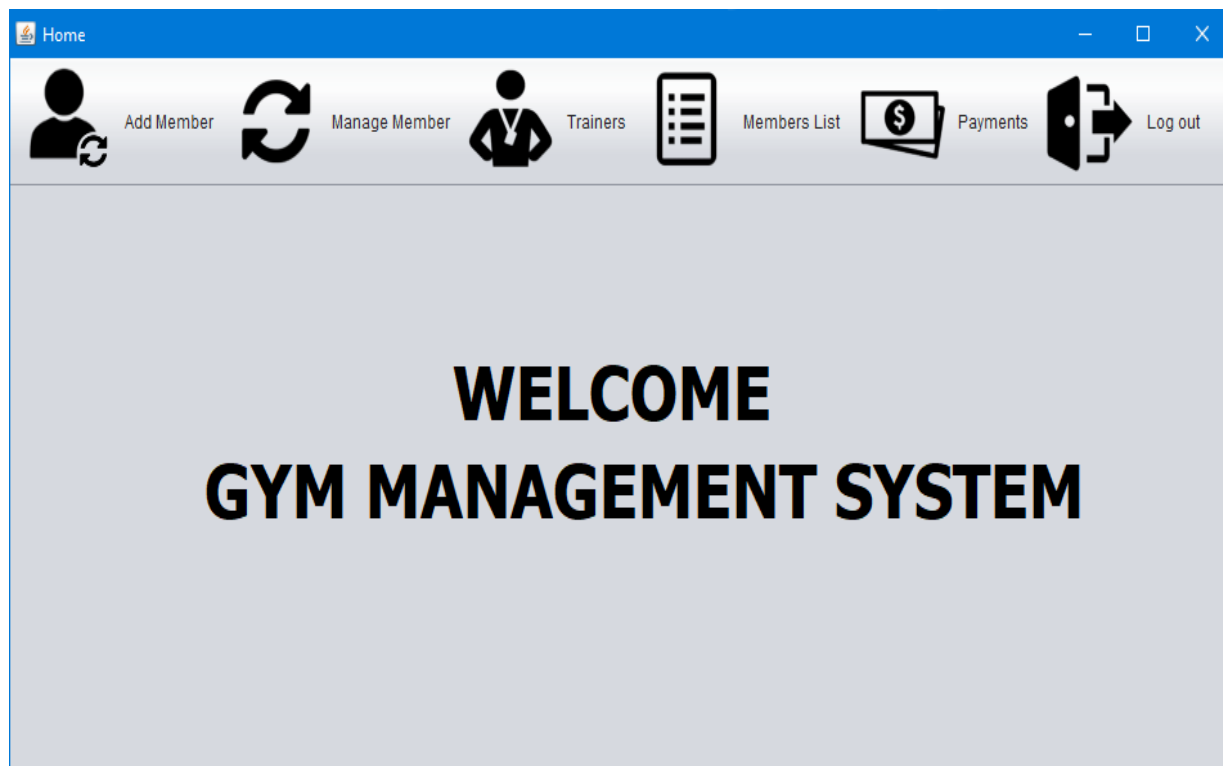
\*\*\*\*\*

☐ Show passwords

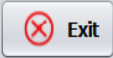
 Login

 Exit

**Figure 4.1.1 Login Page**



**Figure 4.1.3 Home page**



# ADD MEMBER

Member ID: **002**

Member Type: Plus

First Name

Phone Number

Last Name

Email

Address

Amount Pay

\$


Trainer


trainer 1


Gender

Male

Register Date



 Save

 Reset

**Figure 4.1.4 Add member page**

Payment

Exit

PAYMENT

Enter the valid ID to get the information

Member ID: 

Search

Member ID: Today: 2022-04-23

| ID | Member Name | Member Type | Amount Pay | Payment Date | Due Date   | Days Remaining | Status   |
|----|-------------|-------------|------------|--------------|------------|----------------|----------|
| 1  | Tai Quach   | Plus        | 19.99      | 2022-04-22   | 2022-05-22 | 29 Days        | Paid     |
| 2  | Kevin Ly    | Plus        | 19.99      | 2022-04-23   | 2022-05-23 | 30 Days        | Paid     |
| 3  | Hung Nguyen | Premium     | 24.99      | null         | null       | 0 Days         | Not Paid |
| 4  | Vy Trinh    | Basic       | 9.99       | null         | null       | 0 Days         | Not Paid |

Member Name:

Member Type:

Amount Pay:

Payment Date:

Due Date:

Days Remaining:

Status:

Reset

Pay

Figure 4.1.5 Payment page

20



## 4.1 Database Design

The database design for the **Gym Management System** includes several key tables that store and manage data across different modules:

- **Users Table:** Stores user details for members, trainers, and administrators, including personal information, roles, and login credentials.
- **Sessions Table:** Manages details about gym sessions, such as time, trainer assignments, and member bookings.
- **Payments Table:** Tracks member payments for memberships, sessions, and other services, ensuring financial data is accurately stored.
- **Inventory Table:** Tracks gym equipment and resources, including stock levels and updates, to ensure availability for members.

## 4.2 IMPLEMENTATIONS (CODE)

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools / Templates
 * and open the template in the editor.
 */
package com.mycompany.gymmanagementsystem;

import database.ConnectionProvider;
import java.awt.Color;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

/**
 *
 * @author quach
 */
public class benefitsPanel extends javax.swing.JPanel {

    /**
     * Creates new form NewJPanel
     */

    public benefitsPanel() {
        initComponents();

        basic1.setSelected(false);
        basic2.setSelected(false);
        plus1.setSelected(false);
        plus2.setSelected(false);
        plus3.setSelected(false);
        premium1.setSelected(false);
        premium2.setSelected(false);
        premium3.setSelected(false);
        premium4.setSelected(false);
    }
}
```

```

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code"> //GEN-BEGIN: initComponents
private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    plus1 = new javax.swing.JCheckBox();
    basic1 = new javax.swing.JCheckBox();
    basic2 = new javax.swing.JCheckBox();
    plus2 = new javax.swing.JCheckBox();
    plus3 = new javax.swing.JCheckBox();
    premium1 = new javax.swing.JCheckBox();
    premium2 = new javax.swing.JCheckBox();
    premium3 = new javax.swing.JCheckBox();
    premium4 = new javax.swing.JCheckBox();
    jComboBox1 = new javax.swing.JComboBox<>();

    setPreferredSize(new java.awt.Dimension(451, 571));

    jLabel1.setFont(new java.awt.Font("Tahoma", 1, 36)); // NOI18N
    jLabel1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/icons/benefiticon.png"))); // NOI18N
    jLabel1.setText("BENEFITS");

    plus1.setBackground(new java.awt.Color(204, 204, 204));
    plus1.setText("ALL GROUP EXERCISE CLASSES");

    basic1.setBackground(new java.awt.Color(204, 204, 204));
    basic1.setText("USE OF ALL STRENGTH EQUIPMENT");

    basic2.setBackground(new java.awt.Color(204, 204, 204));
    basic2.setText("USE OF ALL CARDIO EQUIPMENT");

    plus2.setBackground(new java.awt.Color(204, 204, 204));
    plus2.setText("1 PERSONAL TRAINING SESSION");

    plus3.setBackground(new java.awt.Color(204, 204, 204));
    plus3.setText("1 GROUP TRAINING SESSION");

```

```

premium1.setBackground(new java.awt.Color(204, 204, 204));
premium1.setText("USE OF BASKETBALL COURTS");

premium2.setBackground(new java.awt.Color(204, 204, 204));
premium2.setText("USE OF RACQUETBALL COURTS");

premium3.setBackground(new java.awt.Color(204, 204, 204));
premium3.setText("UNLIMITED STUDIO CYCLING");

premium4.setBackground(new java.awt.Color(204, 204, 204));
premium4.setText("UP TO TWO GUESTS PER VISIT");

jComboBox1.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Basic",
"Plus", "Premium" }));
jComboBox1.addItemListener(new java.awt.event.ItemListener() {
    public void itemStateChanged(java.awt.event.ItemEvent evt) {
        jComboBox1ItemStateChanged(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
this.setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
            layout.createSequentialGroup()
                .addGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
                    false)
                    .addComponent(basic1, javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(plus1, javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(basic2, javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(plus2, javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(29, 29, 29)
                    .addComponent(basic1)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                    .addComponent(basic2)
                    .addGap(18, 18, 18)

```

```
.addComponent(plus1)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(plus2)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(plus3)
.addGap(18, 18, 18)
.addComponent(premium1)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(premium2)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(premium3)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(premium4)
.addContainerGap(151, Short.MAX_VALUE))
);
}// </editor-fold>//GEN-END:initComponents
```





## 5. CONCLUSION

The Gym Management System is an efficient and comprehensive solution designed to automate and streamline the day-to-day operations of a gym. It simplifies tasks such as user management, session scheduling, trainer assignments, payment tracking, and inventory management. By leveraging Java and MySQL, the system ensures data integrity, secure transactions, and easy scalability to accommodate future growth.

With its user-friendly interface and powerful backend, this system improves operational efficiency, reduces administrative workload, and enhances the overall gym experience for both members and staff. Additionally, its flexible design allows for easy expansion and integration of new features, making it a reliable solution for managing small to medium-sized gyms. The system not only meets the current needs of gym operations but also provides a foundation for future advancements in the fitness industry.

## 6. REFERENCES

### Books:

**Head First Java by Kathy Sierra and Bert Bates – Beginner-friendly Java guide.**

**Java: The Complete Reference by Herbert Schildt – Comprehensive Java reference.**

**Database System Concepts by Abraham Silberschatz et al. – Covers database design.**

**User Interface Design and Evaluation by Debbie Stone et al. – Focuses on UI/UX principles.**

### Websites:

**Oracle Java Tutorials:** <https://docs.oracle.com/javase/tutorial/>

**SQL Basics (W3Schools):** <https://www.w3schools.com/sql/>

**Java Swing (Javatpoint):** <https://www.javatpoint.com/java-swing>

**Software Development Life Cycle (GeeksforGeeks):**

<https://www.geeksforgeeks.org/software-development-life-cycle-sdlc/>