

Internship Report

Internship report submitted in fulfillment of the requirement for the
certificate of the internship

By

Your Name: ARPAN KUNDU

**College Name: SANAKA EDUCATIONAL
TRUST GROUP OF INSTITUTIONS**

**Domain Name: JAVA DEVELOPER
Batch (JUNE-JULY)**



Date of Submission: 30th JULY

2024

DECLARATION

I hereby declare that the projects entitled "**SIMPLE CALCULATOR**" and submitted for the completion of my internship at InternJunction, are my original work. These projects have not been submitted for any degree, diploma, or other internship programs at any other organizations.

Name: Arpan Kundu

Place: Durgapur

Date: 20th july

ACKNOWLEDGEMENT

It is my proud privilege to express my heartfelt gratitude to several individuals who have assisted me directly or indirectly in completing this project. First and foremost, I would like to extend my deepest appreciation to my internship guides at InternJunction. Their sincere guidance, unwavering support, and invaluable inspiration have been instrumental in the successful completion of this internship.

The insights and knowledge I gained during this period have been profound and enlightening, significantly broadening my understanding of the topics at hand. This experience has not only deepened my academic and professional knowledge but also prepared me for future challenges and opportunities.

I am genuinely grateful for the opportunities provided by InternJunction, which have allowed me to explore and engage with new and exciting avenues of knowledge. This internship has been an enriching journey, and I am confident that the skills and insights I have gained will be of immense benefit in my future endeavours.

To all those who have contributed to this project, whether directly or indirectly, I extend my sincere thanks. Your support and encouragement have been invaluable, and I am truly indebted to you all.

TABLE OF CONTENTS

DECLARATION	2
ACKNOWLEDGEMENTS	3
TABLE OF CONTENTS	4
INTRODUCTION	5
METHODOLOGY	7
RESULTS	14
CONCLUSION	15
FUTURE WORK	16

INTRODUCTION

Project Report: Simple Calculator (Console-Based)

Project Overview

Project Name:

Simple Calculator (Console-Based)

Description:

The Simple Calculator is a console-based application designed to perform basic arithmetic operations. It supports addition, subtraction, multiplication, and division for various numeric types including integers, floats, and doubles. This project aims to provide a user-friendly tool for everyday calculations while serving as an educational resource for individuals interested in programming and understanding the core principles of object-oriented programming (OOP).

Scope of the Project

Functional Scope: *Basic Calculations:*

1. Perform addition, subtraction, multiplication, and division.
2. Support for different numeric types (integers, floats, doubles).

Educational Scope: *Learning Resource:*

1. Ideal for beginners interested in coding and software development.
2. Provides practical experience in implementing arithmetic operations.

Demonstrates the four pillars of OOP:

1. Encapsulation

2.Abstraction

3.Inheritance

4.Polymorphism

Objectives

Primary Objective:

To create a simple, user-friendly calculator application for daily use that performs accurate arithmetic operations based on user input.

Secondary Objectives:

1. Educational Goals:

- Teach beginners the basics of programming through a practical project.
- Enhance understanding of OOP concepts through hands-on implementation.

2. Industry-Level Learning:

- Gain experience in developing a console-based application.
- Understand the process of creating, testing, and refining software.

Features

User Input:

- Accepts two numbers and an operation choice (addition, subtraction, multiplication, division) from the user.

Output:

- Displays the accurate result of the chosen arithmetic operation.

Error Handling:

- Manages invalid inputs and division by zero scenarios gracefully.

METHODOLOGY

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

        try {
            System.out.println("Please Enter your choice: ");
            System.out.println("PRESS '1' FOR ADDITION\tPRESS '2' FOR
SUBTRACTION\tPRESS '3' FOR MULTIPLICATION\tPRESS '4' FOR DIVISION: ");
            int sch=sc.nextInt();
            switch (sch) {
                case 1:new TakeInput(sch).input();
                    break;
                case 2:new TakeInput(sch).input();
                    break;
                case 3:new TakeInput(sch).input();
                    break;
                case 4:new TakeInput(sch).input();
                    break;
                default:System.out.println("Please enter valid choice.");
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
            System.out.println("Please enter the right option which is
provided above... ");
        }
        finally
        {
            sc.close();
        }
    }
}
```

Main Class: SimpleCalculator

The SimpleCalculator class serves as the entry point for our console-based calculator application. It includes the main method, where the application flow begins. Exception handling is implemented using a try-catch block to manage any unexpected errors that may arise during the execution.

Functionality

1.User Interaction:

- The application prompts the user to choose the desired operation by entering a number corresponding to the operation:
 - 1 for addition
 - 2 for subtraction
 - 3 for multiplication
 - 4 for division
- After selecting an operation, the program calls the `input()` method from the `TakeInput` class.

2.Operation Handling:

- The user's choice is managed using a switch-case statement, which determines the appropriate arithmetic operation.
- The chosen operation is then passed to the `TakeInput` class constructor for further processing.

In this implementation, the `SimpleCalculator` class handles user input and directs the flow of the program based on the selected operation. By utilizing a try-catch block, it ensures that any unexpected exceptions are caught and managed gracefully, providing a robust and user-friendly experience.

```
class TakeInput
{
    int op=0;
    TakeInput(){
    public TakeInput(int op)
    {
        this.op=op;
    }
    int aI=0;
    int bI=0;
    double aD=0.0;
    double bD=0.0;
    int ch1=0;
    int ch2=0;
    Scanner sc=new Scanner(System.in);
    public void input()
    {
        try {
            System.out.print("Please enter a number: (IF THE NUMBER HAVE NO
DECIMAL POINT DIGIT THEN PRESS 1 AND IF THE NUMBER HAVE DECIMAL POINT DIGIT
THEN PRESS 2): ");
            ch1=sc.nextInt();
            System.out.println();
```



```

        if(ch1==1)
            aI=sc.nextInt();
        else
            aD=sc.nextDouble();

        System.out.println("Enter another number: (IF THE NUMBER HAVE NO
DECIMAL POINT DIGIT THEN PRESS 1 AND IF THE NUMBER HAVE DECIMAL POINT DIGIT
THEN PRESS 2): ");
        ch2=sc.nextInt();
        System.out.println();
        if(ch2==1)
            bI=sc.nextInt();
        else
            bD=sc.nextDouble();
    } catch (Exception e) {
        System.out.println(e.getMessage());
        System.out.println("Please enter the right option which is
provided above... ");
    }

    //Redirect to that method.
    if(op==1){
        Addition ob=new Addition();
        if(ch1==1 && ch2==1)
            ob.add(aI,bI);
        else if(ch1==2 && ch2==2)
            ob.add(aD, bD);
        else if(ch1==1 && ch2==2)
            ob.add(aI, bD);
        else if(ch1==2 && ch2==1)
            ob.add(aD,bI);
    }
    else if(op==2){
        Subtraction ob=new Subtraction();
        if(ch1==1 && ch2==1)
            ob.sub(aI,bI);
        else if(ch1==2 && ch2==2)
            ob.sub(aD, bD);
        else if(ch1==1 && ch2==2)
            ob.sub(aI, bD);
        else if(ch1==2 && ch2==1)
            ob.sub(aD,bI);
    }
    else if(op==3){
        Multiplication ob=new Multiplication();
        if(ch1==1 && ch2==1)
            ob.mul(aI,bI);
        else if(ch1==2 && ch2==2)

```

```

        ob.mul(aD, bD);
    else if(ch1==1 && ch2==2)
        ob.mul(aI, bD);
    else if(ch1==2 && ch2==1)
        ob.mul(aD, bI);
    }
    else if(op==4){
        Division ob=new Division();
        if(ch1==1 && ch2==1)
            ob.div(aI, bI);
        else if(ch1==2 && ch2==2)
            ob.div(aD, bD);
        else if(ch1==1 && ch2==2)
            ob.div(aI, bD);
        else if(ch1==2 && ch2==1)
            ob.div(aD, bI);
        }
    }
}

```

TakeInput Class

The TakeInput class is designed to handle user inputs and perform the chosen arithmetic operation. It stores the user's operation choice and processes the operands accordingly. This class ensures that the input is correctly interpreted and the appropriate calculation is performed.

Functionality

1. Variable Declaration:

- op: Stores the user's choice of operation, passed through the TakeInput class constructor.
- aI and bI: Integer variables to store user-provided integer values.
- aD and bD: Double variables to store user-provided double values.
- choice1 and choice2: Integer variables to determine whether the user-provided values are integers or doubles.

2. User Input Handling:

- The class prompts the user to input two operands.
- It uses if-else statements to check the user's operation choice and then calls the corresponding method to perform the calculation.

In this implementation, the TakeInput class is responsible for collecting and validating the user's inputs. It checks whether the inputs are integers or doubles

and processes them accordingly. Based on the user's choice of operation, the class redirects the flow to the appropriate method to perform the calculation and display the result. This approach ensures flexibility in handling different types of numeric inputs and provides a clear structure for performing arithmetic operations.

```
class Addition{
    public void add(int a , int b)
    {
        System.out.println("THE RESULT OF "+a+" + "+b+" IS "+"="+a+b));
    }
    public void add(int a, double b)
    {
        System.out.println("THE RESULT OF "+a+" + "+b+" IS "+"="+a+b));
    }
    public void add(double a, int b)
    {
        System.out.println("THE RESULT OF "+a+" + "+b+" IS "+"="+a+b));
    }
    public void add(double a , double b)
    {
        System.out.println("THE RESULT OF "+a+" + "+b+" IS "+"="+a+b));
    }
}
class Subtraction{
    public void sub(int a , int b)
    {
        System.out.println("THE RESULT OF "+a+" - "+b+" IS "+"="+a-b));
    }
    public void sub(int a, double b)
    {
        System.out.println("THE RESULT OF "+a+" - "+b+" IS "+"="+a-b));
    }
    public void sub(double a, int b)
    {
        System.out.println("THE RESULT OF "+a+" - "+b+" IS "+"="+a-b));
    }
    public void sub(double a , double b)
    {
        System.out.println("THE RESULT OF "+a+" - "+b+" IS "+"="+a-b));
    }
}
class Division{
    public void div(int a , int b)
    {
        System.out.println("THE RESULT OF "+a+" / "+b+" IS "+"="+a/b));
    }
    public void div(int a, double b)
```

```

    {
        System.out.println("THE RESULT OF "+a+" / "+b+" IS "+"="+a/b));
    }
    public void div(double a, int b)
    {
        System.out.println("THE RESULT OF "+a+" / "+b+" IS "+"="+a/b));
    }
    public void div(double a , double b)
    {
        System.out.println("THE RESULT OF "+a+" / "+b+" IS "+"="+a/b));
    }
}
class Multiplication{
    public void mul(int a , int b)
    {
        System.out.println("THE RESULT OF "+a+" * "+b+" IS "+"="+a*b));
    }
    public void mul(int a, double b)
    {
        System.out.println("THE RESULT OF "+a+" * "+b+" IS "+"="+a*b));
    }
    public void mul(double a, int b)
    {
        System.out.println("THE RESULT OF "+a+" * "+b+" IS "+"="+a*b));
    }
    public void mul(double a,double b)
    {
        System.out.println("THE RESULT OF "+a+" * "+b+" IS "+"="+a*b));
    }
}

```

Overview of Arithmetic Operation Classes: Class Structure

1. Addition Class
2. Subtraction Class
3. Multiplication Class
4. Division Class

These classes represent the four basic arithmetic operations: addition, subtraction, multiplication, and division. Each class contains methods to perform operations on both integer and double inputs.

Key Concepts: *Polymorphism*

The implementation of these classes demonstrates the concept of polymorphism. Depending on the type of user-provided inputs (integer or double), the appropriate method is executed to perform the calculation. This allows the same operation to be performed on different data types, showcasing the flexibility and power of polymorphism in object-oriented programming.

Execution Flow:

1. User Input:

- The user selects an arithmetic operation (addition, subtraction, multiplication, or division) and provides two operands.

2. Method Selection:

- Based on the user input, the corresponding method for the chosen operation and operand types (integer or double) is selected.
- Polymorphism ensures that the correct method is called, matching the data type of the operands.

3. Calculation and Output:

- The selected method is executed, performing the calculation with the provided operands.
- The result is displayed to the user.

4. Program Termination:

- After displaying the result, the program concludes its execution.

Summary

The design of the `Addition`, `Subtraction`, `Multiplication`, and `Division` classes effectively utilizes polymorphism to handle different types of numeric inputs seamlessly. The flow of the program—from user input to method execution and result display—is streamlined, ensuring a user-friendly experience. This approach not only simplifies the code structure but also enhances the flexibility and maintainability of the application.

RESULTS

```
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUCTION_PROJECTS> cd "d:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUCTION_PROJECTS\SIMPLE_CALCULATOR\" ; if ($?) { javac SimpleCalculator.java } ; if ($?) { java SimpleCalculator }
Please Enter your choice:
PRESS '1' FOR ADDITION PRESS '2' FOR SUBTRACTION PRESS '3' FOR MULTIPLICATION PRESS '4' FOR DIVISION:
1
Please enter a number: (IF THE NUMBER HAVE NO DECIMAL POINT DIGIT THEN PRESS 1 AND IF THE NUMBER HAVE DECIMAL POINT DIGIT THEN PRESS 2): 1
10
Enter another number: (IF THE NUMBER HAVE NO DECIMAL POINT DIGIT THEN PRESS 1 AND IF THE NUMBER HAVE DECIMAL POINT DIGIT THEN PRESS 2):
2
10.5
THE RESULT OF 10 + 10.5 IS =20.5
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUCTION_PROJECTS\SIMPLE_CALCULATOR> cd "d:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUCTION_PROJECTS\SIMPLE_CALCULATOR\" ; if ($?) { javac SimpleCalculator.java } ; if ($?) { java SimpleCalcu
lator }
Please Enter your choice:
PRESS '1' FOR ADDITION PRESS '2' FOR SUBTRACTION PRESS '3' FOR MULTIPLICATION PRESS '4' FOR DIVISION:
5
Please enter valid choice.
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUCTION_PROJECTS\SIMPLE_CALCULATOR> cd "d:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUCTION_PROJECTS\SIMPLE_CALCULATOR\" ; if ($?) { javac SimpleCalculator.java } ; if ($?) { java SimpleCalcu
lator }
Please Enter your choice:
PRESS '1' FOR ADDITION PRESS '2' FOR SUBTRACTION PRESS '3' FOR MULTIPLICATION PRESS '4' FOR DIVISION:
4
Please enter a number: (IF THE NUMBER HAVE NO DECIMAL POINT DIGIT THEN PRESS 1 AND IF THE NUMBER HAVE DECIMAL POINT DIGIT THEN PRESS 2): 2
6.7
Enter another number: (IF THE NUMBER HAVE NO DECIMAL POINT DIGIT THEN PRESS 1 AND IF THE NUMBER HAVE DECIMAL POINT DIGIT THEN PRESS 2):
```

```
THE RESULT OF 10 + 10.5 IS =20.5
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUCTION_PROJECTS\SIMPLE_CALCULATOR> cd "d:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUCTION_PROJECTS\SIMPLE_CALCULATOR\" ; if ($?) { javac SimpleCalculator.java } ; if ($?) { java SimpleCalcu
lator }
Please Enter your choice:
PRESS '1' FOR ADDITION PRESS '2' FOR SUBTRACTION PRESS '3' FOR MULTIPLICATION PRESS '4' FOR DIVISION:
5
Please enter valid choice.
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUCTION_PROJECTS\SIMPLE_CALCULATOR> cd "d:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUCTION_PROJECTS\SIMPLE_CALCULATOR\" ; if ($?) { javac SimpleCalculator.java } ; if ($?) { java SimpleCalcu
lator }
Please Enter your choice:
PRESS '1' FOR ADDITION PRESS '2' FOR SUBTRACTION PRESS '3' FOR MULTIPLICATION PRESS '4' FOR DIVISION:
4
Please enter a number: (IF THE NUMBER HAVE NO DECIMAL POINT DIGIT THEN PRESS 1 AND IF THE NUMBER HAVE DECIMAL POINT DIGIT THEN PRESS 2): 2
6.7
Enter another number: (IF THE NUMBER HAVE NO DECIMAL POINT DIGIT THEN PRESS 1 AND IF THE NUMBER HAVE DECIMAL POINT DIGIT THEN PRESS 2):
2
THE RESULT OF 6.7 / 2 IS =3.35
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUCTION_PROJECTS\SIMPLE_CALCULATOR> cd "d:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUCTION_PROJECTS\SIMPLE_CALCULATOR\" ; if ($?) { javac SimpleCalculator.java } ; if ($?) { java SimpleCalcu
lator }
Please Enter your choice:
PRESS '1' FOR ADDITION PRESS '2' FOR SUBTRACTION PRESS '3' FOR MULTIPLICATION PRESS '4' FOR DIVISION:
r
null
Please enter the right option which is provided above...
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUCTION_PROJECTS\SIMPLE_CALCULATOR>
```

CONCLUTION

Successful Completion of the Simple Calculator Project

The Simple Calculator (Console-Based) project provided by InternJunction has been successfully completed, marking a significant achievement in practical programming.

Practical Application of Knowledge

We applied theoretical knowledge effectively, implementing basic arithmetic operations and object-oriented programming concepts to develop a functional and user-friendly application.

Achievement of Objectives

The project met its primary objective of creating a user-friendly calculator and its educational goal of serving as a valuable learning resource for beginners in coding.

Skill Enhancement

Our programming skills were significantly enhanced through hands-on experience with Java, user input handling, error management, and polymorphism.

Positive Impact on Professional Growth

The project positively impacted our professional growth by improving problem-solving abilities, strengthening our understanding of OOP principles, and providing insights into industry-level software development.

Overall Experience

The project was enriching and fulfilling, solidifying our technical skills and boosting our confidence in real-world programming tasks. The knowledge and experience gained will contribute to our future success in software development.

We are grateful to InternJunction for this valuable opportunity and look forward to applying these skills in future endeavors.

FUTURE WORK

Advanced Technologies Integration

Graphical User Interface (GUI): Develop a GUI version of the calculator using frameworks like JavaFX or Swing for a more user-friendly experience.

Mobile Application: Create a mobile app version of the calculator for Android and iOS platforms.

New Applications Exploration

Scientific Calculations: Expand functionality to include scientific operations like trigonometric functions, logarithms, and exponentials.

Statistical Calculations: Add features for statistical analysis, including mean, median, mode, and standard deviation calculations.

Long-Term Evaluations

Performance Optimization: Conduct long-term evaluations to optimize performance and ensure the calculator remains efficient with larger datasets and more complex calculations.

User Feedback: Implement a system for collecting user feedback to continually improve the calculator's functionality and user experience.

These future work directions aim to enhance and expand the Simple Calculator project, ensuring its continued effectiveness and adaptability in various contexts.

Internship Report

Internship report submitted in fulfillment of the requirement for the
certificate of the internship

By

Your Name: ARPAN KUNDU

**College Name: SANAKA EDUCATIONAL
TRUST GROUP OF INSTITUTIONS**

**Domain Name: JAVA DEVELOPER
Batch (JUNE-JULY)**



Date of Submission: 30th JULY

2024

DECLARATION

I hereby declare that the projects entitled "**TIC-TAC-TOE GAME**" and submitted for the completion of my internship at InternJunction, are my original work. These projects have not been submitted for any degree, diploma, or other internship programs at any other organizations.

Name: Arpan Kundu

Place: Durgapur

Date: 20th july

ACKNOWLEDGEMENT

It is my proud privilege to express my heartfelt gratitude to several individuals who have assisted me directly or indirectly in completing this project. First and foremost, I would like to extend my deepest appreciation to my internship guides at InternJunction. Their sincere guidance, unwavering support, and invaluable inspiration have been instrumental in the successful completion of this internship.

The insights and knowledge I gained during this period have been profound and enlightening, significantly broadening my understanding of the topics at hand. This experience has not only deepened my academic and professional knowledge but also prepared me for future challenges and opportunities.

I am genuinely grateful for the opportunities provided by InternJunction, which have allowed me to explore and engage with new and exciting avenues of knowledge. This internship has been an enriching journey, and I am confident that the skills and insights I have gained will be of immense benefit in my future endeavours.

To all those who have contributed to this project, whether directly or indirectly, I extend my sincere thanks. Your support and encouragement have been invaluable, and I am truly indebted to you all.

TABLE OF CONTENTS

DECLARATION	2
ACKNOWLEDGEMENTS	3
TABLE OF CONTENTS	4
INTRODUCTION	5
METHODOLOGY	7
RESULTS	14
CONCLUSION	16
FUTURE WORK	17

INTRODUCTION

Description:

Tic-Tac-Toe, also known as Noughts and Crosses, is a classic game that has been enjoyed by people of all ages for generations. This project aims to recreate this simple yet engaging game in a console-based format using basic programming principles. By developing this game, we aim to provide a fun and interactive way to practice and enhance programming skills, particularly in the areas of game logic, user input handling, and state management.

Objective:

The primary objective of this project is to develop a console-based Tic-Tac-Toe game that provides an engaging and interactive experience for users. The game will facilitate a two-player mode where users can play against each other on the same console. This project aims to enhance programming skills, specifically in understanding game logic, implementing user input, and handling game states.

Scope:

This project covers the following areas:

Game Board Implementation: Creating a 3x3 grid that represents the Tic-Tac-Toe board.

Player Input: Allowing two players to enter their moves alternately.

Game Logic: Implementing the logic to check for win conditions, draw conditions, and invalid moves.

User Interface: Designing a simple text-based user interface to display the game board and status messages.

Game Flow Control: Ensuring the game progresses smoothly from start to end, including restarting or ending the game based on player choices.

The game will display the updated board after each move and notify the players of the game's status. Once a win or draw condition is reached, the game will announce the result and offer the players the option to restart or exit.

Result:

Upon completion, the project will result in a fully functional console-based Tic-Tac-Toe game. Players will be able to:

1. Enter their moves alternately.
2. See real-time updates of the game board after each move.
3. Receive feedback on the game status (win, lose, or draw).

The successful implementation of this project will demonstrate a foundational understanding of game development principles in a console environment and provide a basis for more complex game development projects in the future. This project not only serves as a fun exercise in programming but also as a stepping stone for learning more advanced concepts in game development.

METHODOLOGY

The TicTacToe class and the GameAlgorithms class are designed to showcase the principles of modularity in programming. The GameAlgorithms class is particularly structured with six distinct methods, each serving a specific function:

1. *showBoard()*: Displays the current state of the game board.
2. *operationOnTable()*: Handles operations on the game board, such as updating it based on user moves.
3. *decideUser()*: Determines which user's turn it is to play.
4. *winChecker()*: Checks the game board for a winning condition.
5. *takeInput()*: Manages the input from the users.
6. *moveOfUsers()*: Executes the moves of the users based on the input received.

This modular design demonstrates the usefulness of modularity in programming languages, allowing for better organization, readability, maintainability, and scalability of the code. Each method encapsulates a specific functionality, making the overall system easier to understand and modify.

Main Class: TicTacToe

```
public class TicTacToe {  
    public static void main(String[] args) {  
        System.out.println(".....LET'S PLAY THE ONE AND ONLY TIC-TAC-TOE  
GAME.....");  
        GameAlgorithm drawTable=new GameAlgorithm();  
        drawTable.showBoard();  
        drawTable.decideUser();  
    }  
}
```

The main class, 'TicTacToe', serves as the entry point for the game. Within the 'main' method of this class, several important steps are taken to set up and start the game:

1. *Create an Object of GameAlgorithms*: An instance of the 'GameAlgorithms' class is created. This object will be used to access various game-related methods.

2. *Display the Initial Game Board:* The `showBoard()` method is called using the `GameAlgorithms` object. This method displays the initial state of the TicTacToe table, allowing players to see the starting configuration.

3. *Conduct a Toss to Decide the First Player:* The `decideUser()` method is invoked next. This method performs a toss to determine which player will make the first move and assigns the symbols X and O to the players.

These steps ensure that the game is properly initialized, the board is displayed, and the first player and their symbols are decided, providing a clear and organized start to the TicTacToe game.

GameAlgorithm class:

```
class GameAlgorithm{
    char board[][]={{'A','B','C'},
                    {'D','E','F'},
                    {'G','H','I'}};

    static int conut=0;
    String Xuser;
    String Ouser;
    public void showBoard()
    {
        System.out.print(board[0][0]+" | "+board[0][1] +" | "+ board[0][2]);
        System.out.println();
        System.err.print(board[1][0]+" | "+board[1][1] +" | "+ board[1][2]);
        System.out.println();
        System.out.print(board[2][0]+" | "+board[2][1] +" | "+ board[2][2]);
        System.out.println();
    }
    public void operationOnTable(char pos, char symbol)
    {
        conut++;
        char table[][]=board;
        for(int i=0;i<table.length;i++)
        {
            for(int j=0;j<table[i].length;j++)
            {
                if(table[i][j]==pos)
                    table[i][j]=symbol;
            }
        }
        board=table;
        showBoard();
        if(conut==9)
        {
```



```

        System.out.println("So, the final TIC-TAC-TOE board is: ");
        showBoard();
        winChecker();
    }
}
public void takeInput(char input,String name)
{
    char symbol=input;
    Scanner sc=new Scanner(System.in);
    System.out.println(name+" Please enter your position: ");
    char pos=sc.next().toUpperCase().charAt(0);
    if(pos=='X' || pos=='O')
    {
        System.out.println("This are not applicable... Please enter the
provided named position to continue the game!!!!");
        takeInput(input, name);
    }
    if(pos=='A' || pos=='B' || pos=='C' || pos=='D' || pos=='E' ||
pos=='F' || pos=='G' || pos=='H' || pos=='I')
    {
        operationOnTable(pos,symbol);
    }
    else
    {
        System.out.println("Please enter the valid position not anything
else....");
        takeInput(input, name);
    }
    sc.close();
}
public void decideUser()
{
    String Xuser;
    String Ouser;
    Scanner sc=new Scanner(System.in);
    System.out.println("Please enter your name: ");
    System.out.println("Who will choose 'X' as symbol: ");
    Xuser=sc.next();
    System.out.println("Who will choose 'O' as symbol: ");
    Ouser=sc.next();
    Random random=new Random();
    int toss=2;
    int win=random.nextInt(toss);
    if(win==0)
        System.out.println(Xuser+" is Win the toss.....");
    else
        System.out.println(Ouser+" is win the toss.....");
    GameAlgorithm drawTable=new GameAlgorithm();

```

```

        drawTable.moveOfUsers(Xuser,Ouser,win);
        sc.close();
    }
    public void moveOfUsers(String user1, String user2, int win)
    {
        Xuser=user1;
        Ouser=user2;
        int firstMove=win;
        if (firstMove==0){
            takeInput('X',Xuser);
            firstMove=1;
            moveOfUsers(Xuser, Ouser, firstMove);
        }
        else{
            takeInput('O',Ouser);
            firstMove=0;
            moveOfUsers(Xuser, Ouser, firstMove);
        }
    }

    public void winChecker()
    {
        if((board[0][0]=='X' && board[1][0]=='X' && board[2][0]=='X') ||
(board[0][0]=='X' && board[0][1]=='X' && board[0][2]=='X') ||
(board[0][1]=='X' && board[1][1]=='X' && board[2][1]=='X') ||
(board[1][0]=='X' && board[1][1]=='X' && board[1][2]=='X') ||
(board[0][2]=='X' && board[1][2]=='X' && board[2][2]=='X') ||
(board[2][0]=='X' && board[2][1]=='X' && board[2][2]=='X')){
            System.out.println(Xuser+" won the match");
            System.out.println(Ouser+" loss the match");
            System.exit(0);
        }
        else if((board[0][0]=='O' && board[1][0]=='O' && board[2][0]=='O') ||
(board[0][0]=='O' && board[0][1]=='O' && board[0][2]=='O') ||
(board[0][1]=='O' && board[1][1]=='O' && board[2][1]=='O') ||
(board[1][0]=='O' && board[1][1]=='O' && board[1][2]=='O') ||
(board[0][2]=='O' && board[1][2]=='O' && board[2][2]=='O') ||
(board[2][0]=='O' && board[2][1]=='O' && board[2][2]=='O')){
            System.out.println(Ouser+" won the match.");
            System.out.println(Xuser+" loss the match");
            System.exit(0);
        }
        else{
            System.out.println("draw-match");
            System.exit(0);
        }
    }
}

```

The GameAlgorithm class is the backbone of the TicTacToe game, managing the core functionalities and ensuring a smooth gameplay experience. Here's an organized and styled breakdown of its key components:

Methods in the `GameAlgorithm` Class:

1. showBoard():

Description:

Displays the current state of the game board.

Functionality:

1.This method iterates through the `board` 2D array and prints each cell, providing a visual representation of the game board to the players after each move.

2. operationOnTable():

Description:

Handles the logic for placing a player's symbol on the board and updates the display.

Functionality:

1.Takes the player's chosen position, updates the `board` array with the player's symbol, and increments the `count` variable.
2.Calls the `showBoard()` method to display the updated board.
3.Checks if the `count` has reached 9, indicating a full board, and then calls `winChecker()` to check for a winner or declare a draw.

3. takeInput():

Description:

Manages and validates the player's input.

Functionality:

1.Receives the player's symbol (X or O) and their name.

- 2.Prompts the player to choose a position on the board to place their symbol.
- 3.Validates the chosen position to ensure it is not already occupied and is within the valid range.
- 4.If the position is valid, updates the board with the player's symbol; otherwise, prompts the player to select a different position.

4. decideUser():

Description:

Decides which player will use X and which will use O, and who makes the first move.

Functionality:

- 1.Conducts a toss using the 'Random()' method to determine which player will make the first move.
- 2.Assigns symbols (X and O) to the players based on the toss result.
- 3.Calls the 'moveOfUsers()' method with the players' names and the toss result to start the game.

5.moveOfUsers():

Description:

Manages player turns and updates the game state.

Functionality:

- 1.Alternates between players for their turns, ensuring each player gets to move.
- 2.Takes the player's name and an integer to store the winning result.
- 3.Calls 'takeInput()' for the current player's move and updates the 'winResult' based on the game status.
- 4.Repeats the process until a win condition is met or the board is full.

6. winChecker():

Description:

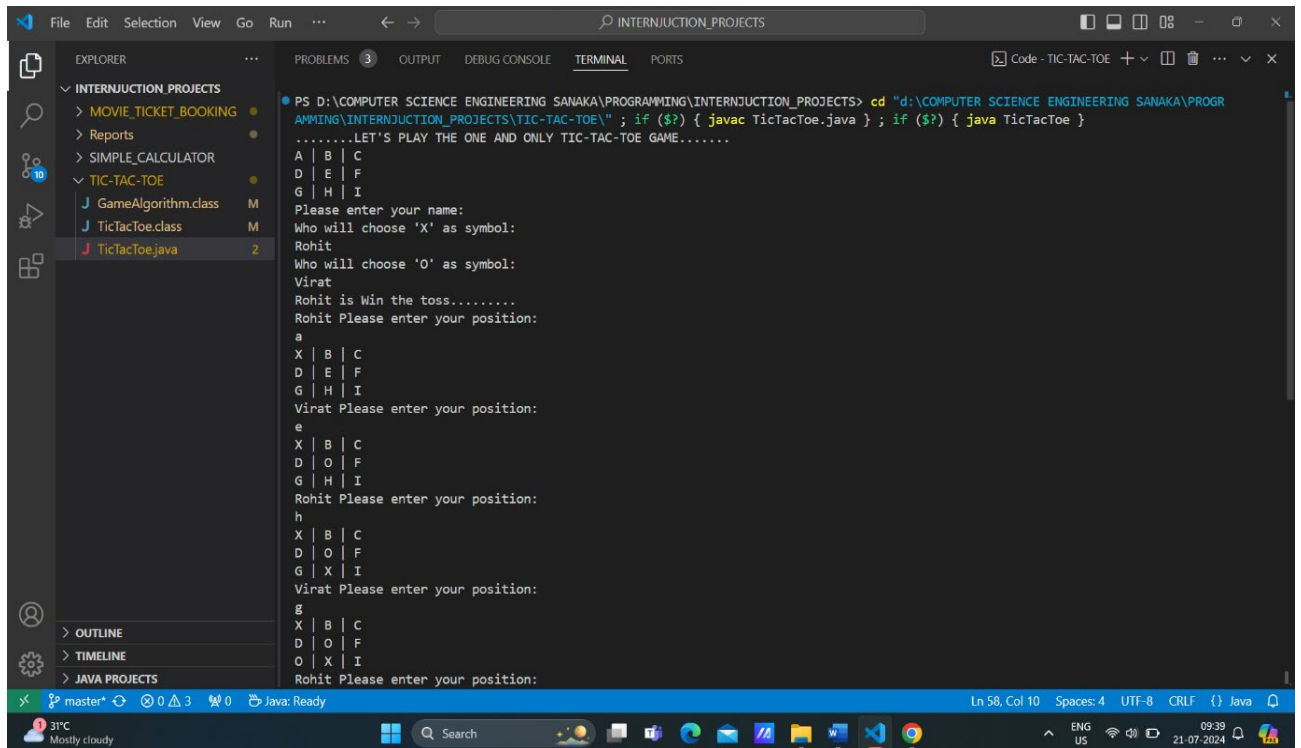
Checks for a winning condition or a draw and declares the game outcome.

Functionality:

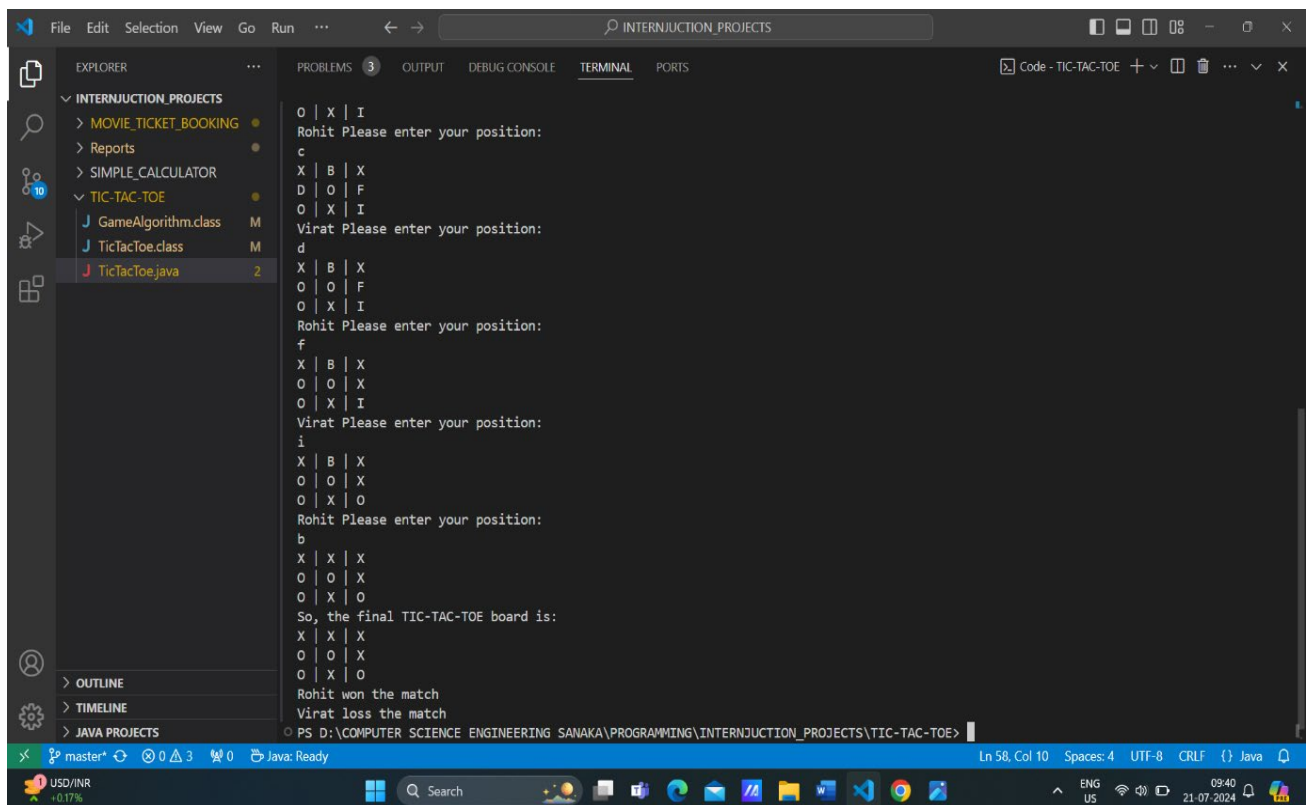
1. Analyzes the `board` array to check for a winning combination of symbols (three in a row, column, or diagonal).
2. Declares the winner if a winning condition is met.
3. If no winning condition is met and the board is full (`count` is 9), declares the game as a draw.
4. Terminates the game and displays the final result to the players.

These methods collectively ensure the smooth execution of the TicTacToe game, from displaying the board and handling player input to determining the winner and managing game turns. The modular design enhances the clarity and maintainability of the code.

RESULTS



```
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUCTION_PROJECTS> cd "d:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGR
AMMING\INTERNUCTION_PROJECTS\TIC-TAC-TOE\" ; if ($?) { javac TicTacToe.java } ; if ($?) { java TicTacToe }
.....LET'S PLAY THE ONE AND ONLY TIC-TAC-TOE GAME.....
A | B | C
D | E | F
G | H | I
Please enter your name:
Who will choose 'X' as symbol:
Rohit
Who will choose 'O' as symbol:
Virat
Rohit is Win the toss.....
Rohit Please enter your position:
a
X | B | C
D | E | F
G | H | I
Virat Please enter your position:
e
X | B | C
D | O | F
G | H | I
Rohit Please enter your position:
h
X | B | C
D | O | F
G | X | I
Virat Please enter your position:
g
X | B | C
D | O | F
O | X | I
Rohit Please enter your position:
```



```
O | X | I
Rohit Please enter your position:
c
X | B | X
D | O | F
O | X | I
Virat Please enter your position:
d
X | B | X
O | O | F
O | X | I
Rohit Please enter your position:
f
X | B | X
O | O | X
O | X | I
Virat Please enter your position:
i
X | B | X
O | O | X
O | X | O
Rohit Please enter your position:
b
X | X | X
O | O | X
O | X | O
So, the final TIC-TAC-TOE board is:
X | X | X
O | O | X
O | X | O
Rohit won the match
Virat loss the match
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUCTION_PROJECTS\TIC-TAC-TOE>
```

```
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUJCTION_PROJECTS\TIC-TAC-TOE> cd "d:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUJCTION_PROJECTS\TIC-TAC-TOE" ; if ($?) { javac TicTacToe.java } ; if ($?) { java TicTacToe }
.....LET'S PLAY THE ONE AND ONLY TIC-TAC-TOE GAME.....
A | B | C
D | E | F
G | H | I
Please enter your name:
Who will choose 'X' as symbol:
Rohit
Who will choose 'O' as symbol:
Virat
Virat is win the toss.....
Virat Please enter your position:
a
O | B | C
D | E | F
G | H | I
Rohit Please enter your position:
b
O | X | C
D | E | F
G | H | I
Virat Please enter your position:
c
O | X | O
D | E | F
G | H | I
Rohit Please enter your position:
d
O | X | O
X | E | F
G | H | I
Virat Please enter your position:
```

```
X | E | F
G | H | I
Virat Please enter your position:
e
O | X | O
X | O | F
G | H | I
Rohit Please enter your position:
f
O | X | O
X | O | X
G | H | I
Virat Please enter your position:
g
O | X | O
X | O | X
O | H | I
Rohit Please enter your position:
h
O | X | O
X | O | X
O | X | I
Virat Please enter your position:
i
O | X | O
X | O | X
O | X | O
So, the final TIC-TAC-TOE board is:
O | X | O
X | O | X
O | X | O
draw-match
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUJCTION_PROJECTS\TIC-TAC-TOE>
```

CONCLUTION

Successful Completion of the Simple Calculator Project

The Tic-Tac-Toe Game (Console-Based) project provided by InternJunction has been successfully completed, marking a significant achievement in practical programming.

Practical Application of Knowledge

We applied theoretical knowledge effectively, implementing modularity and object-oriented programming concepts to develop a functional and user-friendly application.

Achievement of Objectives

The project met its primary objective of creating a user-friendly tic-tac-toe game and its educational goal of serving as a valuable learning resource for beginners in coding.

Skill Enhancement

Our programming skills were significantly enhanced through hands-on experience with Java, user input handling, error management, and modularity programming.

Positive Impact on Professional Growth

The project positively impacted our professional growth by improving problem-solving abilities, strengthening our understanding of OOP principles, and providing insights into industry-level software development.

Overall Experience

The project was enriching and fulfilling, solidifying our technical skills and boosting our confidence in real-world programming tasks. The knowledge and experience gained will contribute to our future success in software development.

We are grateful to InternJunction for this valuable opportunity and look forward to applying these skills in future endeavors.

FUTURE WORK

1. Graphical User Interface (GUI):

- Develop a graphical interface to replace the console-based output, enhancing user interaction and visual appeal. Tools like Swing (Java) or Tkinter (Python) can be used to create an intuitive and engaging game window.

2. AI Opponent:

- Introduce an artificial intelligence opponent for single-player mode. Implement algorithms such as Minimax to enable the AI to make strategic moves and provide a challenging gameplay experience.

3. Network Play:

- Enable multiplayer functionality over a network, allowing players to compete with others remotely. This involves setting up server-client architecture to facilitate online gameplay.

4. Score Tracking:

- Add a feature to track and display scores across multiple games. This could include maintaining statistics for wins, losses, and draws, and presenting them in a leaderboard format.

5. Enhanced Game Modes:

- Expand the game with additional modes, such as tournament play or various difficulty levels. This would provide players with a range of options and increase the game's replay value.

These enhancements aim to improve the overall gaming experience, adding complexity and interactivity to the TicTacToe project.

Internship Report

Internship report submitted in fulfillment of the requirement for the
certificate of the internship

By

Your Name: ARPAN KUNDU

**College Name: SANAKA EDUCATIONAL
TRUST GROUP OF INSTITUTIONS**

**Domain Name: JAVA DEVELOPER
Batch (JUNE-JULY)**



Date of Submission: 30th JULY

2024

DECLARATION

I hereby declare that the projects entitled "**MOVIE TICKET BOOKING SYSTEM**" and submitted for the completion of my internship at InternJunction, are my original work. These projects have not been submitted for any degree, diploma, or other internship programs at any other organizations.

Name: Arpan Kundu

Place: Durgapur

Date: 20th july

ACKNOWLEDGEMENT

It is my proud privilege to express my heartfelt gratitude to several individuals who have assisted me directly or indirectly in completing this project. First and foremost, I would like to extend my deepest appreciation to my internship guides at InternJunction. Their sincere guidance, unwavering support, and invaluable inspiration have been instrumental in the successful completion of this internship.

The insights and knowledge I gained during this period have been profound and enlightening, significantly broadening my understanding of the topics at hand. This experience has not only deepened my academic and professional knowledge but also prepared me for future challenges and opportunities.

I am genuinely grateful for the opportunities provided by InternJunction, which have allowed me to explore and engage with new and exciting avenues of knowledge. This internship has been an enriching journey, and I am confident that the skills and insights I have gained will be of immense benefit in my future endeavours.

To all those who have contributed to this project, whether directly or indirectly, I extend my sincere thanks. Your support and encouragement have been invaluable, and I am truly indebted to you all.

TABLE OF CONTENTS

DECLARATION	2
ACKNOWLEDGEMENTS	3
TABLE OF CONTENTS	4
INTRODUCTION	5
METHODOLOGY	7
RESULTS	16
CONCLUSION	18
FUTURE WORK	19

INTRODUCTION

Overview:

The Movie Ticket Booking System is a console-based application designed to streamline the process of booking and managing movie tickets. The system consists of two main classes: `MovieTicketBookingSystem` and `Customer`. The `Customer` class includes various methods for interacting with the system, such as displaying movie lists, booking tickets, canceling reservations, and allowing user selections.

Scope of the Project:

The project aims to develop a robust and user-friendly ticket booking system that operates within a console environment. It focuses on providing essential functionalities for managing movie bookings, including:

- Viewing available movies and showtimes.
- Booking tickets for selected movies.
- Canceling existing bookings.
- Facilitating user interactions through a simple text-based interface.

Objectives:

- 1. Develop a Functional Booking System:* Create a working application that allows users to book and manage movie tickets through a console interface.
- 2. Implement Core Features:* Provide functionalities such as displaying movie lists, booking tickets, canceling reservations, and handling user selections.
- 3. Ensure Usability:* Design the system to be intuitive and easy to navigate, even within a console-based environment.
- 4. Enhance User Experience:* Improve user interaction by offering clear prompts and managing inputs effectively.

Result:

The completed Movie Ticket Booking System successfully provides a console-based solution for booking and managing movie tickets. The system enables users to:

- Browse available movies and their showtimes.
- Reserve tickets for desired shows.
- Cancel or modify existing reservations.
- Enjoy a straightforward and functional interface for all booking-related tasks.

The Movie Ticket Booking System project employs two key classes: `MovieTicketBookingSystem`, which handles the overall functionality and operations of the booking system, and `Customer`, which manages user interactions and ticket management processes. The `Customer` class is equipped with methods for displaying movie options, facilitating bookings, and handling cancellations. The project showcases practical application of object-oriented programming principles and provides a solid foundation for understanding and developing console-based applications.

METHODOLOGY

Movie Ticket Booking System

The Movie Ticket Booking System is designed to streamline the process of booking and canceling movie tickets. This system consists of two primary classes: `MovieTicketBookingSystem` and `Customer`.

Class Structure

1. MovieTicketBookingSystem

- This class serves as the main entry point for the system. It contains the main method, which displays options to the user and redirects the flow of control based on the user's selection.

2. Customer

- This class encapsulates the logic for handling customer interactions, such as adding movies and canceling bookings. It includes methods for displaying movie lists, handling bookings, and processing cancellations.

Flow of Control

When the application starts, the main method in the `MovieTicketBookingSystem` class presents the user with options to either book or cancel a ticket. Based on the user's choice:

Booking: If the user selects the booking option, control is passed to the `addMovies()` method in the `Customer` class. This method facilitates the process of selecting and booking a movie.

Canceling: If the user selects the cancel option, control is passed to the `cancel()` method in the `Customer` class. This method manages the cancellation of existing bookings.

The Movie Ticket Booking System is designed to be user-friendly, providing straightforward options for booking and canceling movie tickets. The separation of concerns between the `MovieTicketBookingSystem` and `Customer` classes ensures that the system is modular and easy to maintain. By following this structured approach, the system can be easily extended to include additional features in the future.

MovieTicketBookingSystem Class

The 'MovieTicketBookingSystem' class serves as the main entry point for the system. It contains the main method, which displays options to the user and redirects the flow of control based on the user's selection. The key responsibilities and flow of the 'MovieTicketBookingSystem' class are as follows:

1. Displaying Options:

- When the application starts, the main method in the 'MovieTicketBookingSystem' class presents the user with options to either book a ticket or cancel a ticket.

2. Handling User Choice:

Based on the user's choice, the system redirects control to the appropriate method in the 'Customer' class:

Booking: If the user selects the booking option, control is passed to the 'addMovies()' method in the 'Customer' class.

Canceling: If the user selects the cancel option, control is passed to the 'cancel()' method in the 'Customer' class.

3. Error Handling:

The system employs error handling mechanisms to manage any unexpected errors gracefully, ensuring a smooth user experience.

```
public class MoveTicketBookingSystem {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        Customer customer=new Customer();
        System.out.println("Welcome to the one stop solution of movies!!");
        System.out.println("press 1 for booking\tPress 2 for canceling: ");
        int ch=sc.nextInt();
        if (ch==1) {
            customer.addMovies();
        }
        else
            customer.cancel();
        sc.close();
    }
}
```

By following this structured approach, the 'MovieTicketBookingSystem' class effectively manages the primary user interactions and ensures a seamless flow within the movie ticket booking system. This design allows for easy maintenance and future extensions of the system.

Customer Class

The 'Customer' class is responsible for handling the logic related to customer interactions in the Movie Ticket Booking System. It includes methods for adding movies, displaying available movie genres, processing movie selections, handling ticket collections, making payments, and canceling tickets. The class uses six 'LinkedList' objects to store movies categorized by genre: action, comedy, detective, thrillers, suspense, and horror.

Key Methods and Their Functionality

1. LinkedLists for Movie Storage

- The 'Customer' class contains six 'LinkedList' objects to store movies of different genres: action, comedy, detective, thrillers, suspense, and horror.

2. addMovies()

- This void method is used to add new movies to a particular section. It performs the necessary logic to add movies and then calls the 'section()' method for further processing.

3. section()

- This method displays the types of movies available in the theatre, such as action, comedy, detective, thrillers, suspense, and horror. After displaying the options, it prompts the user to choose a genre and then calls the 'movieList(char choice)' method, passing the user's choice as an argument.

4. movieList()

- This method takes a character parameter representing the selected movie genre. It uses a switch-case statement to display the movies in the selected genre. For each case, it calls the `movieSelection(LinkedList<String> genreList)` method, passing the appropriate `LinkedList` for the selected genre. Error handling is managed using a try-catch block.

5. *movieSelection()*

- This method takes a `LinkedList` parameter representing the selected genre list. It allows the user to select a movie from the list, stores the selected movie name in a string variable, and then calls the `ticketCollection()` method for further processing. Error handling is managed using a try-catch block.

6. *ticketCollection()*

- This method manages the ticket collection process. It prompts the user to select the type of theatre and ticket class (first class, second class, third class) along with their respective prices. It also asks for the time slot (10:00 AM, 2:00 PM, 7:00 PM) and the number of tickets the user wants to buy. Finally, it calls the `makePayment(int ticketCount, String ticketType)` method, passing the ticket count and ticket type as parameters.

7. *makePayment()*

- This method calculates the total amount the user needs to pay based on the ticket count and ticket type. It provides options for payment modes (online and offline) and handles the payment process. The user is asked to enter the exact amount, and upon successful payment, the tickets are confirmed. If the payment fails, the user is prompted to try again. The method also provides an option to cancel the tickets by pressing the 'c' button, which redirects to the `cancel()` method.

8. *cancel()*

- This method handles the cancellation of tickets. If no tickets are available, it displays an appropriate message. Otherwise, it sets the ticket count to zero and declares that the tickets are canceled.

```

class Customer
{
    LinkedList<String> action=new LinkedList<String>();
    LinkedList<String> comedy=new LinkedList<String>();
    LinkedList<String> detective=new LinkedList<String>();
    LinkedList<String> thrillers=new LinkedList<String>();
    LinkedList<String> suspense=new LinkedList<String>();
    LinkedList<String> horror=new LinkedList<String>();
    public void addMovies()
    {
        action.add("Avenger");action.add("Attack"); action.add("Wolverine");
        action.add("Moon_Kinght"); action.add("Kalki");

        comedy.add("Hera
Phire");comedy.add("Golmal");comedy.add("Welcome");comedy.add("Welcome_back");
        comedy.add("HosueFull-4");

        detective.add("Bomkeysh");detective.add("Agent_Roy");detective.add("Fe
luDa");

        thrillers.add("Simle");thrillers.add("Katputli");thrillers.add("Hitman
");thrillers.add("Yodha");

        suspense.add("Feluda");suspence.add("chup");suspence.add("365_Days");

        horror.add("Conjuring");horror.add("Anabela");horror.add("Exorcist");
        section();
    }
    public void movieList(char ch)
    {
        switch (ch) {
            case 'A':for(int i=0;i<action.size();i++)
                System.out.println((i+1)+" : "+action.get(i));
                movieSelection(action);
                break;
            case 'C':for(int i=0;i<comedy.size();i++)
                System.out.println((i+1)+" : "+comedy.get(i));
                movieSelection(comedy);
                break;
            case 'D':for(int i=0;i<detective.size();i++)
                System.out.println((i+1)+" : "+thrillers.get(i));
                movieSelection(detective);
                break;
            case 'T':for(int i=0;i<thrillers.size();i++)
                System.out.println((i+1)+" : "+thrillers.get(i));
                movieSelection(thrillers);
                break;
            case 'S':for(int i=0;i<suspence.size();i++)

```

```

        System.out.println((i+1)+" : "+suspence.get(i));
        movieSelection(suspence);
        break;
    case 'H':for(int i=0;i<horror.size();i++)
        System.out.println((i+1)+" : "+horror.get(i));
        movieSelection(horror);
        break;
    default:
        break;
    }
}
}
public void movieSelection(LinkedList<String> choice)
{
    Scanner sc=new Scanner(System.in);
    try {
        int fev=0;
        System.out.println("Select your movie: ");
        fev=sc.nextInt();
        if(fev<=choice.size()+1){
            String movieName=choice.get(fev-1);
            System.out.println("You choose: "+movieName+" movie.");
            ticketCollection();
        }
        else
            movieSelection(choice);
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    finally
    {
        sc.close();
    }
}
int ticketcount=0;
public void ticketCollection()
{
    int fc=250,sc=200,tc=150;
    Scanner scan=new Scanner(System.in);
    System.out.println("Now choose type of the theater: \nPress '1' for
NORMAL\tPress '2' for IMAX 3D");
    int theater=scan.nextInt();
    if(theater==1)
        System.out.println("You select NORMAL");
    else
        System.out.println("You select IMAX 3D");
    System.out.println("What type of ticket are you looking for: ");
    System.out.println("Press '1' for FIRST CLASS(Price:"+fc+")\tPress '2'
for SECOND CLASS(Price:"+sc+)\tPress '3' for THIRD CLASS(Price:"+tc+)");

```

```

        int c=scan.nextInt();
        if(c==1)
            System.out.println("You choose FIRST CLASS");
        else if(c==2)
            System.out.println("You choose SECOND CLASS");
        else
            System.out.println("You choose THIRD CLASS");
        System.out.println("We have three slot: \n1 : 10.00AM\n2 : 02.00PM\n3
: 07.00PM");
        int slot=scan.nextInt();
        if(slot==1)
            System.out.println("You choose 10.00AM slot.");
        else if(slot==2)
            System.out.println("You choose 02.00PM slot.");
        else
            System.out.println("You choose 07.00PM slot");
        System.out.println("How many ticket do you want to book? ");
        ticketcount=scan.nextInt();
        makePayment(ticketcount,c);
        scan.close();
    }
    public void makePayment(int ticketcount,int c)
    {
        Scanner scan=new Scanner(System.in);
        int total=0;
        if(c==1)
            total=ticketcount*250;
        else if(c==2)
            total=ticketcount*200;
        else
            total=ticketcount*150;
        System.out.println("So, you total amount is: "+total);
        System.out.println("Press '1' for Online\tPress '2' for Cash: ");
        String mode=scan.next();
        if(mode.equals("1")){
            System.out.println("Enter your amount: ");
            int amount=scan.nextInt();
            if(amount==total)
                System.out.println("You money is accepted.");
            else
                System.out.println("Please enter actual amount to confirm.");
        }
        else
        {
            System.out.println("Enter your amount: ");
            int amount=scan.nextInt();
            if(amount==total)

```

```

        System.out.println("You money is accepted.");
    else
        System.out.println("Please enter actual amount to confirm.");
    }
    System.out.println("You "+ticketcount+"tickets are confirmed!!");
    System.out.println("If you want to cancel then press 'C' or press
anything: ");
    int can=scan.next().toUpperCase().charAt(0);
    if(can=='C')
        cancel();
    else
        System.out.println("Thank you!!\tVisit Agian!!");
    scan.close();
}

public void section()
{
    Scanner sc=new Scanner(System.in);
    char ch=' ';
    try{
        System.out.println("Hey, Welcome to the Customer Desk.....\nAt that
time our collections
are:\nComedy,Thrillers,Action,Detective,Suspence,Horror,etc");
        System.out.println("So,What type of movie are you looking for:
\nPlease Select from the below list: ");
        System.out.println("Press 'A' for Action\tPress 'C' for Comedy\tPress
'T' for Thrillers\tPress 'D' for Detective\tPress 'S' for Suspence\tPress 'H'
for Horror: ");
        ch=sc.next().toUpperCase().charAt(0);
    }
    catch(Exception e){
        e.printStackTrace();
    }
    switch (ch) {
        case 'A':movieList(ch);
            break;
        case 'C':movieList(ch);
            break;
        case 'T':movieList(ch);
            break;
        case 'D':movieList(ch);
            break;
        case 'S':movieList(ch);
            break;
        case 'H':movieList(ch);
            break;
        default:System.out.println("Please enter the actual one!!!");
            section();
            break;
    }
}

```

```

    }
    sc.close();
}
public void cancel()
{
    Scanner sc=new Scanner(System.in);
    String confirm=null;
    if(ticketcount==0)
        System.out.println("You do not book any ticket from any show!!");
    else
    {
        System.out.println("Are you sure yo want to cancel your booking:
yes/no");
        confirm=sc.next();
        if(confirm.equals("yes"))
        {
            ticketcount=0;
            System.out.println("Your tickets are successfully canceled.");
        }
        else
            System.out.println("Ok!!");
    }
    sc.close();
}
}

```

Summary of the Working Procedure

Movie Addition and Selection: The process starts with adding movies to different genres using the `addMovies()` method. The `section()` method displays the available genres, and the user selects a genre. The `movieList()` method processes the user's selection and calls the `movieSelection()` method to select a specific movie.

Ticket Collection and Payment: The `ticketCollection()` method manages the selection of theatre type, ticket class, and time slot, and collects the number of tickets. The `makePayment()` method handles the payment process, confirming the tickets upon successful payment or prompting the user to try again in case of failure.

Ticket Cancellation: If the user decides to cancel the tickets, the `cancel()` method is invoked to set the ticket count to zero and confirm the cancellation.

By following this structured approach, the `Customer` class effectively manages movie selection, ticket collection, payment, and cancellation processes, ensuring a user-friendly and maintainable system.

RESULTS

```
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUJCTION_PROJECTS> cd "d:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUJCTION_PROJECTS\MOVIE_TICKET_BOOKING\" ; if ($?) { javac MoveTicketBookingSystem.java } ; if ($?) { java MoveTicketBookingSystem }
Welcome to the one stop solution of movies!!
press 1 for booking Press 2 for canceling:
1
Hey, Welcome to the Customer Desk.....
At that time our collections are:
Comedy,Thrillers,Action,Detective,Suspence,Horrer,etc
So,What type of movie are you looking for:
Please Select from the below list:
Press 'A' for Action Press 'C' for Comedy Press 'T' for Thrillers Press 'D' for Detective Press 'S' for Suspence Press 'H' for Horrre:
a
1 : Avenger
2 : Attack
3 : Wolvarine
4 : Moon_Kinght
5 : Kalki
Select your movie:
4
You choose: Moon_Kinght movie.
Now choose type of the theater:
Press '1' for NORMAL Press '2' for IMAX 3D
2
You select IMAX 3D
What type of ticket are you looking for:
Press '1' for FIRST CLASS(Price:250) Press '2' for SECOND CLASS(Price:200) Press '3' for THIRD CLASS(Price:150)
1
You choose FIRST CLASS
We have three slot:
1 : 10.00AM
2 : 02.00PM
```

```
3 : Wolvarine
4 : Moon_Kinght
5 : Kalki
Select your movie:
4
You choose: Moon_Kinght movie.
Now choose type of the theater:
Press '1' for NORMAL Press '2' for IMAX 3D
2
You select IMAX 3D
What type of ticket are you looking for:
Press '1' for FIRST CLASS(Price:250) Press '2' for SECOND CLASS(Price:200) Press '3' for THIRD CLASS(Price:150)
1
You choose FIRST CLASS
We have three slot:
1 : 10.00AM
2 : 02.00PM
3 : 07.00PM
2
You choose 02.00PM slot.
How many ticket do you want to book?
4
So, you total amount is: 1000
Press '1' for Online Press '2' for Cash:
2
Enter your amount:
1000
You money is accepted.
You 4tickets are confirmed!!
If you want to cancel then press 'C' or press anything:
no
Thank you!! Visit Agian!!
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUJCTION_PROJECTS\MOVIE_TICKET_BOOKING>
```

```
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUJCTION_PROJECTS> cd "d:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUJCTION_PROJECTS\MOVIE_TICKET_BOOKING\" ; if ($?) { javac MoveTicketBookingSystem.java } ; if ($?) { java MoveTicketBookingSystem }
Welcome to the one stop solution of movies!!
press 1 for booking Press 2 for canceling:
1
Hey, Welcome to the Customer Desk.....
At that time our collections are:
Comedy,Thrillers,Action,Detective,Suspence,Horrer,etc
So,What type of movie are you looking for:
Please Select from the below list:
Press 'A' for Action Press 'C' for Comedy Press 'T' for Thrillers Press 'D' for Detective Press 'S' for Suspence Press 'H' for Horrre:
t
1 : Simle
2 : Katputli
3 : Hitman
4 : Yodha
Select your movie:
1
You choose: Simle movie.
Now choose type of the theater:
Press '1' for NORMAL Press '2' for IMAX 3D
1
You select NORMAL
What type of ticket are you looking for:
Press '1' for FIRST CLASS(Price:250) Press '2' for SECOND CLASS(Price:200) Press '3' for THIRD CLASS(Price:150)
2
You choose SECOND CLASS
We have three slot:
1 : 10.00AM
2 : 02.00PM
3 : 07.00PM
```

```
4 : Yodha
Select your movie:
1
You choose: Simle movie.
Now choose type of the theater:
Press '1' for NORMAL Press '2' for IMAX 3D
1
You select NORMAL
What type of ticket are you looking for:
Press '1' for FIRST CLASS(Price:250) Press '2' for SECOND CLASS(Price:200) Press '3' for THIRD CLASS(Price:150)
2
You choose SECOND CLASS
We have three slot:
1 : 10.00AM
2 : 02.00PM
3 : 07.00PM
3
You choose 07.00PM slot
How many ticket do you want to book?
4
So, you total amount is: 800
Press '1' for Online Press '2' for Cash:
1
Enter your amount:
800
You money is accepted.
You 4tickets are confirmed!!
If you want to cancel then press 'c' or press anything:
c
Are you sure yo want to cancel your booking: yes/no
yes
Your tickets are successfully canceled.
```

```
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUJCTION_PROJECTS\MOVIE_TICKET_BOOKING> cd "d:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUJCTION_PROJECTS\MOVIE_TICKET_BOOKING\" ; if ($?) { javac MoveTicketBookingSystem.java } ; if ($?) { java MoveTicketBookingSystem }
Welcome to the one stop solution of movies!!
press 1 for booking Press 2 for canceling:
2
You do not book any ticket from any show!!
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNUJCTION_PROJECTS\MOVIE_TICKET_BOOKING>
```

CONCLUSION

Successful Completion of the Movie Ticket Booking System Project

The successful completion of the Movie Ticket Booking System project marks a significant achievement in our practical programming journey.

Practical Application of Knowledge

We effectively applied our theoretical knowledge to this project, implementing modularity and object-oriented programming concepts to develop a functional and user-friendly movie ticket booking system. The project allowed us to manage user input, handle data, and integrate various functionalities in a real-world context.

Achievement of Objectives

The project met its primary objective of creating a robust platform for managing and booking movie tickets. It also served as a valuable learning resource, demonstrating practical applications of coding principles and providing a comprehensive understanding of system design and data management.

Skill Enhancement

Our programming skills were significantly enhanced through hands-on experience with Java. We gained proficiency in user input handling, error management, and modularity programming. The complexity of the Movie Ticket Booking System further strengthened our ability to develop scalable and maintainable software solutions.

Positive Impact on Professional Growth

This project positively impacted our professional growth by improving our problem-solving abilities, deepening our understanding of OOP principles, and offering insights into industry-level software development. It provided a practical experience in developing and managing complex systems, preparing us for future challenges in the field.

Overall Experience

Completing the Movie Ticket Booking System was both enriching and fulfilling. It solidified our technical skills and boosted our confidence in tackling real-world programming tasks. The knowledge and experience gained from this project will be invaluable for our future endeavors in software development. We are grateful for this opportunity and look forward to applying these skills in future projects.

FUTURE WORK

1. Enhanced User Interface:

- Develop a more intuitive and interactive user interface to improve the overall user experience. Consider incorporating features like color coding, better formatting of output, and more detailed prompts.

2. Advanced Search and Filtering:

- Implement advanced search and filtering options to help users find movies based on various criteria such as genre, rating, or showtimes more easily.

3. Reservation Management:

- Add functionality for users to view, modify, or cancel their existing reservations. This could include updating ticket quantities, changing showtimes, or processing refunds.

4. Error Handling and Validation:

- Enhance error handling and input validation to ensure robustness and reliability. Implement more comprehensive checks for invalid input, such as incorrect movie IDs or invalid dates.

5. Database Integration:

- Integrate a database system to manage movie listings, showtimes, and bookings persistently. This will allow for data storage and retrieval even after the console application is closed.

6. User Authentication:

- Add user authentication features to allow for personalized experiences, such as saving user profiles, tracking booking history, and managing multiple accounts.

7. Multi-User Support:

- Implement multi-user capabilities to allow multiple users to interact with the system simultaneously. This would include concurrent booking handling and session management.

8. Automated Reports and Analytics:

- Develop features to generate reports and analytics on ticket sales, popular movies, and user preferences. This could help in analyzing system performance and making informed decisions.

9. Integration with External APIs:

- Explore integrating with external APIs to fetch real-time movie data, such as current showtimes, ratings, and reviews. This can enhance the system's functionality and provide up-to-date information.

10. Testing and Optimization:

- Conduct thorough testing to identify and fix bugs or performance issues. Optimize the code for better efficiency and responsiveness, especially when handling large volumes of data or concurrent requests.

11. Documentation and User Guide:

- Create comprehensive documentation and a user guide to assist users in navigating and utilizing the system effectively. This will also be valuable for future maintenance and development.

12. Scalability and Future Enhancements:

- Plan for scalability to accommodate future growth and additional features. Consider how the system can be expanded to support more complex scenarios or integrated with other applications.