

# Practical 1 : Basic Program

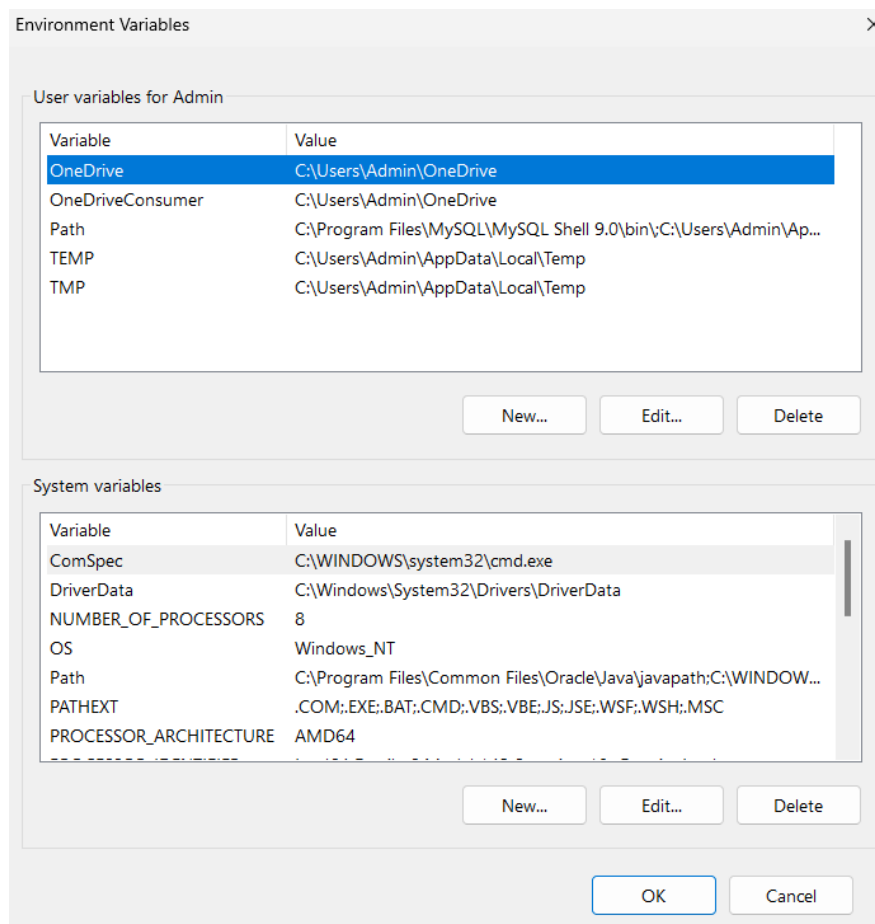
## 1. Study of Class Path and Java Runtime Environment

To set up the path variable for Java, there are two main approaches:

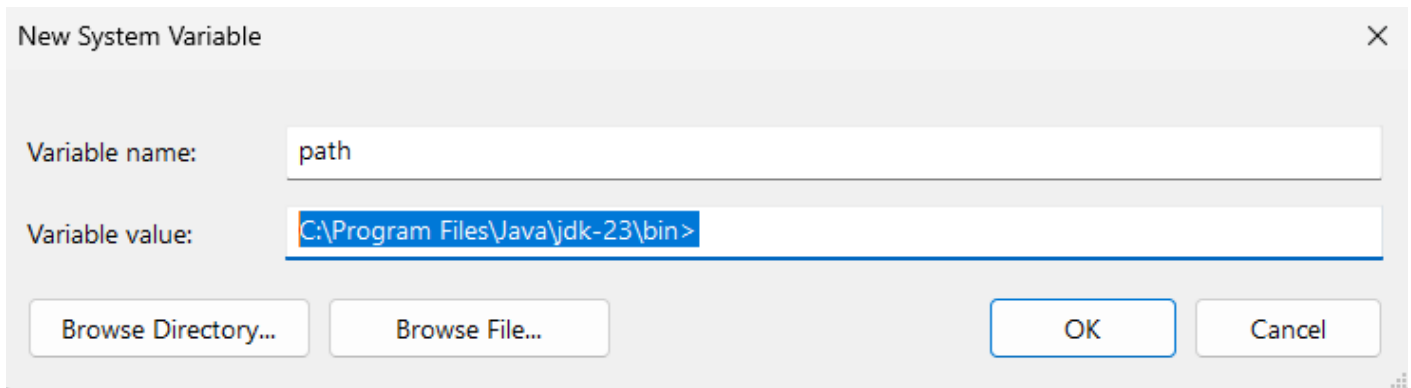
- 1) Using the Graphical User Interface (GUI)
- 2) Using the Command Prompt

=> Using the GUI Method

Step 1: Open a Environment Variables Dialog box by search box



Step 2: Click on New Button-> Enter PATH as a variable name and copy the path of bin file of JDK and paste to the variable value-> click on OK.



=> Using the Command Prompt

Prompt Syntax: `path[[:][:...];;%PATH%]`

```
C:\>cd C:\Program Files\Java\jdk-23\bin
C:\Program Files\Java\jdk-23\bin>
```

## Java Runtime Environment (JRE):

The JRE is a part of the Java Development Kit (JDK) that includes the libraries, Java Virtual Machine (JVM), and other components required to run Java applications. It provides the runtime environment to execute Java byte code.

- Key Components:

- o JVM: Executes Java bytecode and provides a platform-independent runtime environment.

- o Java Libraries: Predefined classes and APIs like `java.lang`, `java.util`, etc.

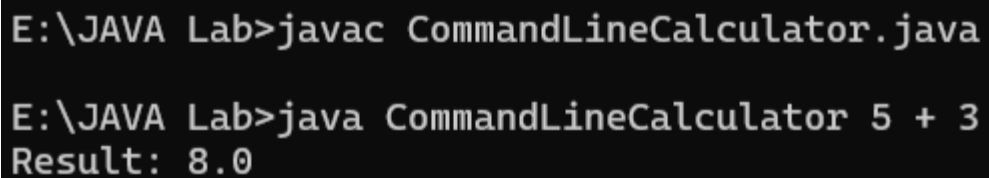
## 2. Write a program to

- Implement command line calculator

```
public class CommandLineCalculator {  
    public static void main(String[] args) {  
        if (args.length != 3) {  
            System.out.println("Usage: java CommandLineCalculator <num1> <operator>  
<num2>");  
            return;  
        }  
  
        double num1 = Double.parseDouble(args[0]);  
        String operator = args[1];  
        double num2 = Double.parseDouble(args[2]);  
        double result = 0;  
  
        switch (operator) {  
            case "+":  
                result = num1 + num2;  
                break;  
            case "-":  
                result = num1 - num2;  
                break;  
            case "*":
```

```
        result = num1 * num2;
        break;
    case "/":
        if (num2 != 0) {
            result = num1 / num2;
        } else {
            System.out.println("Error: Division by zero.");
            return;
        }
        break;
    default:
        System.out.println("Error: Invalid operator.");
        return;
}

System.out.println("Result: " + result);
}
}
```



```
E:\JAVA Lab>javac CommandLineCalculator.java
E:\JAVA Lab>java CommandLineCalculator 5 + 3
Result: 8.0
```

- Write To prints Fibonacci series.

```
class Fibonacci {  
    public static void main(String... args) {  
        int n = Integer.parseInt(args[0]);  
        int first = 0, second = 1;  
        System.out.print(first + " " + second);  
        for (int index = 2; index < n; index++) {  
            int next = first + second;  
            first = second;  
            second = next;  
            System.out.print(" " + next);  
        }  
    }  
}
```

```
E:\JAVA Lab>javac Fibonacci.java
```

```
E:\JAVA Lab>java Fibonacci 7
```

```
0 1 1 2 3 5 8
```

## Practical 2 : Array

1. Define a class Array with following member Field:

int data[];

Function:

Array( ) //create array data of size 10

Array(int size) // create array of size size

Array(int data[]) // initialize array with parameter array

void Reverse \_an \_array () //reverse element of an array

int Maximum \_of \_array () // find maximum element of array

int Average\_of \_array() //find average of element of array void Sorting ()

//sort element of array

void display() //display element of array

int search(int no) //search element and return index else return -1

int size(); //return size of an array

Use all the function in main method. Create different objects with different constructors.

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

```
class Array {  
    private int[] elements;  
    private int size;  
  
    public Array() {  
        this.size = 10;  
        this.elements = new int[size];  
    }  
  
    public Array(int size) {  
        this.size = size;  
        this.elements = new int[size];  
    }  
}
```

```
}

public Array(int[] inputData) {
    this.size = inputData.length;
    this.elements = new int[size];
    System.arraycopy(inputData, 0, elements, 0, size);
}

public void reverse() {
    for (int i = 0; i < size / 2; i++) {
        int temp = elements[i];
        elements[i] = elements[size - 1 - i];
        elements[size - 1 - i] = temp;
    }
}

public int getMax() {
    int max = elements[0];
    for (int i = 1; i < size; i++) {
        if (elements[i] > max) {
            max = elements[i];
        }
    }
    return max;
}

public int getAverage() {
    int sum = 0;
    for (int num : elements) {
        sum += num;
    }
    return sum / size;
}

public void sort() {
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - 1 - i; j++) {
```

```
        if (elements[j] > elements[j + 1]) {
            int temp = elements[j];
            elements[j] = elements[j + 1];
            elements[j + 1] = temp;
        }
    }
}

public void display() {
    for (int num : elements) {
        System.out.print(num + " ");
    }
    System.out.println();
}

public int search(int value) {
    for (int i = 0; i < size; i++) {
        if (elements[i] == value) {
            return i;
        }
    }
    return -1;
}

public int getSize() {
    return size;
}

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

    System.out.print("Enter the size of the array: ");
    int arraySize = scanner.nextInt();

    int[] userData = new int[arraySize];
    System.out.println("Enter " + arraySize + " elements:");
```



```
    for (int i = 0; i < arraySize; i++) {  
        userData[i] = scanner.nextInt();  
    }  
  
    Array userArray = new Array(userData);  
  
    System.out.println("\nOriginal Array:");  
    userArray.display();  
  
    System.out.println("\nReversing the Array:");  
    userArray.reverse();  
    userArray.display();  
  
    System.out.println("\nMaximum Element: " + userArray.getMax());  
    System.out.println("Average: " + userArray.getAverage());  
  
    System.out.println("\nSorting the Array:");  
    userArray.sort();  
    userArray.display();  
  
    System.out.print("\nEnter an element to search: ");  
    int searchValue = scanner.nextInt();  
    int foundIndex = userArray.search(searchValue);  
    if (foundIndex != -1) {  
        System.out.println("Value " + searchValue + " found at index: " + foundIndex);  
    } else {  
        System.out.println("Value " + searchValue + " not found.");  
    }  
  
    scanner.close();  
}  
}
```

```
PS E:\JAVA Lab> javac Array.java
PS E:\JAVA Lab> java Array
Enter the size of the array: 8
Enter 8 elements:
1
2
3
4
5
6
7
8

Original Array:
1 2 3 4 5 6 7 8

Reversing the Array:
8 7 6 5 4 3 2 1

Maximum Element: 8
Average: 4

Sorting the Array:
1 2 3 4 5 6 7 8

Enter an element to search: 5
Value 5 found at index: 4
```

2. Define a class Matrix with following Field:

int row, column;

float mat[][]

Function: Matrix(int a[][])

Matrix()

Matrix(int row, int col)

void readMatrix() //read element of array

float [][] transpose( ) //find transpose of first matrix

float [][] matrixMultiplication(Matrix second ) //multiply two matrices and return result

void displayMatrix(float [][]a) //display content of argument array

void displayMatrix() //display content

float maximum\_of\_array() // return maximum element of first array

float average\_of\_array( ) // return average of first array

create three object of Matrix class with different constructors in main and test all the functions in main

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

```
class Matrix {
```

```
    int row, column;
```

```
    float mat[][];
```

```
    public Matrix() {
```

```
        this.row = 2;
```

```
        this.column = 2;
```

```
        this.mat = new float[row][column];
```

```
    }
```

```
    public Matrix(int row, int column) {
```

```
        this.row = row;
```

```
        this.column = column;
```

```
        this.mat = new float[row][column];
```

```
}

public void readMatrix() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter elements for a " + row + "x" + column + " matrix:");
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < column; j++) {
            System.out.print("Enter element at position (" + (i + 1) + ", " + (j + 1) + "): ");
            mat[i][j] = scanner.nextFloat();
        }
    }
}

public float[][] transpose() {
    float[][] transposed = new float[column][row];
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < column; j++) {
            transposed[j][i] = mat[i][j];
        }
    }
    return transposed;
}

public float[][] matrixMultiplication(Matrix second) {
    if (this.column != second.row) {
        throw new IllegalArgumentException("Matrix multiplication is not possible");
    }

    float[][] result = new float[this.row][second.column];
    for (int i = 0; i < this.row; i++) {
        for (int j = 0; j < second.column; j++) {
            for (int k = 0; k < this.column; k++) {
                result[i][j] += this.mat[i][k] * second.mat[k][j];
            }
        }
    }
    return result;
}
```

```
}
```

```
public void displayMatrix(float[][] a) {  
    for (int i = 0; i < a.length; i++) {  
        for (int j = 0; j < a[i].length; j++) {  
            System.out.print(a[i][j] + " ");  
        }  
        System.out.println();  
    }  
}
```

```
public void displayMatrix() {  
    displayMatrix(this.mat);  
}
```

```
public float maximum_of_array() {  
    float max = mat[0][0];  
    for (int i = 0; i < row; i++) {  
        for (int j = 0; j < column; j++) {  
            if (mat[i][j] > max) {  
                max = mat[i][j];  
            }  
        }  
    }  
    return max;  
}
```

```
public float average_of_array() {  
    float sum = 0;  
    int totalElements = row * column;  
    for (int i = 0; i < row; i++) {  
        for (int j = 0; j < column; j++) {  
            sum += mat[i][j];  
        }  
    }  
    return sum / totalElements;
```

```
}

public static void main(String[] args) {

    Matrix matrix1 = new Matrix();
    matrix1.readMatrix();
    System.out.println("\nMatrix 1:");
    matrix1.displayMatrix();

    Matrix matrix3 = new Matrix(2, 2);
    matrix3.readMatrix();
    System.out.println("\nMatrix 3:");
    matrix3.displayMatrix();

    System.out.println("\nTranspose of Matrix 1:");
    float[][] transposed = matrix1.transpose();
    matrix1.displayMatrix(transposed);

    System.out.println("\nMultiplication of Matrix 1 and Matrix 3:");
    float[][] product = matrix1.matrixMultiplication(matrix3);
    matrix1.displayMatrix(product);

    System.out.println("\nMaximum value in Matrix 1: " +
matrix1.maximum_of_array());
    System.out.println("Average value in Matrix 1: " + matrix1.average_of_array());
}
}
```

```
PS E:\JAVA Lab> javac Matrix.java
PS E:\JAVA Lab> java Matrix
Enter elements for a 2x2 matrix:
Enter element at position (1,1): 12
Enter element at position (1,2): 2
Enter element at position (2,1): 34
Enter element at position (2,2): 8

Matrix 1:
12.0 2.0
34.0 8.0
Enter elements for a 2x2 matrix:
Enter element at position (1,1): 7
Enter element at position (1,2): 5
Enter element at position (2,1): 43
Enter element at position (2,2): 2

Matrix 3:
7.0 5.0
43.0 2.0

Transpose of Matrix 1:
12.0 34.0
2.0 8.0

Multiplication of Matrix 1 and Matrix 3:
170.0 64.0
582.0 186.0

Maximum value in Matrix 1: 34.0
Average value in Matrix 1: 14.0
```

### 3. Write a program to demonstrate usage of different methods of Wrapper class

```
public class Wrapper {  
    public static void main(String[] args) {  
        Integer integerValue = Integer.valueOf(150);  
        Double doubleVal = Double.valueOf("78.95");  
        Boolean booleanVal = Boolean.valueOf("false");  
  
        int primitiveInteger = integerValue.intValue();  
        double primitiveDouble = doubleVal.doubleValue();  
  
        int parsedInteger = Integer.parseInt("300");  
        double parsedDouble = Double.parseDouble("65.43");  
  
        System.out.println("Integer object: " + integerValue);  
        System.out.println("Double object: " + doubleVal);  
        System.out.println("Boolean object: " + booleanVal);  
  
        System.out.println("Primitive int: " + primitiveInteger);  
        System.out.println("Primitive double: " + primitiveDouble);  
  
        System.out.println("Parsed int: " + parsedInteger);  
        System.out.println("Parsed double: " + parsedDouble);  
  
        System.out.println("Maximum of 15 and 30: " + Integer.max(15, 30));  
        System.out.println("Minimum of 15 and 30: " + Integer.min(15, 30));  
        System.out.println("Binary format of 50: " + Integer.toBinaryString(50));  
    }  
}
```



```
PS E:\JAVA Lab> javac Wrapper.java
PS E:\JAVA Lab> java Wrapper
Integer object: 150
Double object: 78.95
Boolean object: false
Primitive int: 150
Primitive double: 78.95
Parsed int: 300
Parsed double: 65.43
Maximum of 15 and 30: 30
Minimum of 15 and 30: 15
Binary format of 50: 110010
```

#### 4. Write a program to demonstrate usage of String and StringBuffer class

```
public class StringManipulation {
    public static void main(String[] args) {
        String text = "Welcome";
        System.out.println("String: " + text);
        System.out.println("Length of String: " + text.length());
        System.out.println("Character at index 3: " + text.charAt(3));
        System.out.println("Substring (2, 5): " + text.substring(2, 5));
        System.out.println("String in uppercase: " + text.toUpperCase());
        System.out.println("String in lowercase: " + text.toLowerCase());
        System.out.println("Concatenation with ' Home': " + text.concat(" Home"));
        System.out.println("Index of 'o': " + text.indexOf('o'));

        StringBuffer sb = new StringBuffer("Welcome");
        System.out.println("StringBuffer: " + sb);
        sb.append(" Home");
        System.out.println("After append: " + sb);
        sb.insert(8, "Java ");
        System.out.println("After insert: " + sb);
        sb.replace(8, 12, "Wonderful");
        System.out.println("After replace: " + sb);
        sb.delete(8, 17);
        System.out.println("After delete: " + sb);
        sb.reverse();
    }
}
```

```
        System.out.println("After reverse: " + sb);  
    }  
}
```

```
PS E:\JAVA Lab> javac StringManipulation.java  
PS E:\JAVA Lab> java StringManipulation  
String: Welcome  
Length of String: 7  
Character at index 3: c  
Substring (2, 5): lco  
String in uppercase: WELCOME  
String in lowercase: welcome  
Concatenation with ' Home': Welcome Home  
Index of 'o': 4  
StringBuffer: Welcome  
After append: Welcome Home  
After insert: Welcome Java Home  
After replace: Welcome Wonderful Home  
After delete: Welcome Home  
After reverse: emoH emocleW
```

##### 5. Define a class Cipher with following data

Field:

String plainText;

int key

Functions:

Cipher(String plaintext,int key)

String Encryption( )

String Decryption( )

Read string and key from command prompt and replace every character of string with character which is key place down from current character.

Example

plainText = "GCET"

Key = 3

Encryption function written following String "JFHW"

Decryption function will convert encrypted string to original form "GCET"

```
import java.util.Scanner;

public class ShiftCipher {
    String message;
    int key;

    public ShiftCipher(String message, int key) {
        this.message = message;
        this.key = key;
    }

    public String encrypt() {
        StringBuilder encryptedText = new StringBuilder();
        for (int i = 0; i < message.length(); i++) {
            char ch = message.charAt(i);
            if (Character.isLetter(ch)) {
                char shiftedChar = (char) (ch + key);
                if (Character.isUpperCase(ch) && shiftedChar > 'Z') {
                    shiftedChar -= 26;
                } else if (Character.isLowerCase(ch) && shiftedChar > 'z') {
                    shiftedChar -= 26;
                }
                encryptedText.append(shiftedChar);
            } else {
                encryptedText.append(ch);
            }
        }
        return encryptedText.toString();
    }

    public String decrypt(String encryptedText) {
        StringBuilder decryptedText = new StringBuilder();
        for (int i = 0; i < encryptedText.length(); i++) {
            char ch = encryptedText.charAt(i);
            if (Character.isLetter(ch)) {
                char shiftedChar = (char) (ch - key);
            }
        }
    }
}
```

```
if (Character.isUpperCase(ch) && shiftedChar < 'A') {

    shiftedChar += 26;
} else if (Character.isLowerCase(ch) && shiftedChar < 'a') {
    shiftedChar += 26;
}
decryptedText.append(shiftedChar);
} else {
    decryptedText.append(ch);
}
}
return decryptedText.toString();
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter the message: ");
    String message = scanner.nextLine();
    System.out.print("Enter the shift key: ");
    int key = scanner.nextInt();

    ShiftCipher cipher = new ShiftCipher(message, key);
    String encryptedText = cipher.encrypt();
    System.out.println("Encrypted Text: " + encryptedText);
    String decryptedText = cipher.decrypt(encryptedText);
    System.out.println("Decrypted Text: " + decryptedText);

    scanner.close();
}
}
```

```
PS E:\JAVA Lab> javac ShiftCipher.java
PS E:\JAVA Lab> java ShiftCipher
Enter the message: GCET
Enter the shift key: 3
Encrypted Text: JFHW
Decrypted Text: GCET
```

## Practical 3 : Basic Program using Class

1. Create a class BankAccount that has Depositor name , Acc\_no, Acc\_type, Balance as Data Members and void createAcc() . void Deposit(), void withdraw() and void BalanceInquiry as Member Function. When a new Account is created assign next serial no as account number. Account number starts from 1

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

```
class CustomerAccount {  
    private static int nextAccountNo = 1;  
    private String customerName;  
    private int accountNo;  
    private String accountType;  
    private double currentBalance;  
  
    public CustomerAccount() {  
        this.accountNo = nextAccountNo++;  
        this.currentBalance = 0.0;  
    }  
  
    public void createAccount() {  
        Scanner scanner = new Scanner(System.in);  
        System.out.print("Enter Customer Name: ");  
        this.customerName = scanner.nextLine();  
        System.out.print("Enter Account Type (Current/Savings): ");  
        this.accountType = scanner.nextLine();  
        System.out.println("Account Created Successfully! Account Number: " +  
this.accountNo);  
    }  
  
    public void depositAmount() {  
        Scanner scanner = new Scanner(System.in);
```

```
        System.out.print("Enter amount to deposit: ");
        double amount = scanner.nextDouble();
        if (amount > 0) {
            this.currentBalance += amount;
            System.out.println("Amount Deposited Successfully! New Balance: " +
this.currentBalance);
        } else {
            System.out.println("Invalid amount. Please enter a positive value.");
        }
    }

    public void withdrawAmount() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter amount to withdraw: ");
        double amount = scanner.nextDouble();
        if (amount > 0 && amount <= this.currentBalance) {
            this.currentBalance -= amount;
            System.out.println("Amount Withdrawn Successfully! Remaining Balance: " +
this.currentBalance);
        } else {
            System.out.println("Invalid amount or insufficient balance.");
        }
    }

    public void checkBalance() {
        System.out.println("Account Number: " + this.accountNo);
        System.out.println("Customer Name: " + this.customerName);
        System.out.println("Account Type: " + this.accountType);
        System.out.println("Current Balance: " + this.currentBalance);
    }

    public static void main(String[] args) {
        CustomerAccount account = new CustomerAccount();
        account.createAccount();
        account.depositAmount();
        account.withdrawAmount();
        account.checkBalance();
    }
}
```

```
}  
}
```

```
PS E:\JAVA Lab> javac CustomerAccount.java  
PS E:\JAVA Lab> java CustomerAccount  
Enter Customer Name: Peter  
Enter Account Type (Current/Savings): Current  
Account Created Successfully! Account Number: 1  
Enter amount to deposit: 2000  
Amount Deposited Successfully! New Balance: 2000.0  
Enter amount to withdraw: 100  
Amount Withdrawn Successfully! Remaining Balance: 1900.0  
Account Number: 1  
Customer Name: Peter  
Account Type: Current  
Current Balance: 1900.0
```

2. Create a class time that has hour, minute and second as data members. Create a parameterized constructor to initialize Time Objects. Create a member Function Time Sum (Time, Time) to sum two time objects.

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

```
class Duration {  
    private int hours;  
    private int minutes;  
    private int seconds;  
  
    public Duration(int hours, int minutes, int seconds) {  
        this.hours = hours;  
        this.minutes = minutes;  
        this.seconds = seconds;  
    }  
  
    public static Duration add(Duration d1, Duration d2) {  
        int totalSeconds = d1.seconds + d2.seconds;  
        int carryMinutes = totalSeconds / 60;
```

```
int seconds = totalSeconds % 60;

int totalMinutes = d1.minutes + d2.minutes + carryMinutes;

int carryHours = totalMinutes / 60;
int minutes = totalMinutes % 60;

int totalHours = d1.hours + d2.hours + carryHours;

return new Duration(totalHours, minutes, seconds);
}

public void show() {
    System.out.println(hours + " hours " + minutes + " minutes " + seconds + "
seconds");
}

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

    System.out.println("Enter the first duration (hours minutes seconds): ");
    Duration d1 = new Duration(scanner.nextInt(), scanner.nextInt(),
scanner.nextInt());

    System.out.println("Enter the second duration (hours minutes seconds): ");
    Duration d2 = new Duration(scanner.nextInt(), scanner.nextInt(),
scanner.nextInt());

    Duration d3 = Duration.add(d1, d2);
    System.out.print("Total Duration: ");
    d3.show();
}
}
```



```

PS E:\JAVA Lab> javac Duration.java
PS E:\JAVA Lab> java Duration
Enter the first duration (hours minutes seconds):
5 23 46
Enter the second duration (hours minutes seconds):
1 11 22
Total Duration: 6 hours 35 minutes 8 seconds

```

3. Define a class with the Name, Basic salary and dearness allowance as data members. Calculate and print the Name, Basic salary(yearly), dearness allowance and tax deducted at source(TDS) and net salary, where TDS is charged on gross salary which is basic salary + dearness allowance and TDS rate is as per following table.

Gross Salary	TDS
Rs. 100000 and below	NIL
Above Rs. 100000	10% on excess over 100000

DA is 74% of Basic Salary for all. Use appropriate member function.

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

```

class Worker {
    private String employeeName;
    private double baseSalary;
    private double allowance;
    private double totalSalary;
    private double taxDeduction;
    private double finalSalary;

    public Worker(String employeeName, double baseSalary) {
        this.employeeName = employeeName;
        this.baseSalary = baseSalary;
        this.allowance = 0.74 * baseSalary;
        this.totalSalary = this.baseSalary + this.allowance;
    }
}

```

```
this.taxDeduction = (this.totalSalary > 100000) ? (this.totalSalary - 100000) * 0.10 : 0;
    this.finalSalary = this.totalSalary - this.taxDeduction;
}

public void displaySalaryInfo() {
    System.out.println("Employee Name: " + employeeName);
    System.out.println("Base Salary (Yearly): Rs. " + baseSalary);
    System.out.println("Allowance: Rs. " + allowance);
    System.out.println("Total Salary: Rs. " + totalSalary);
    System.out.println("Tax Deducted at Source: Rs. " + taxDeduction);
    System.out.println("Final Salary: Rs. " + finalSalary);
}

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

    System.out.print("Enter Employee Name: ");
    String employeeName = scanner.nextLine();

    System.out.print("Enter Base Salary: ");
    double baseSalary = scanner.nextDouble();

    Worker worker = new Worker(employeeName, baseSalary);
    worker.displaySalaryInfo();
}
}
```

```
PS E:\JAVA Lab> javac Worker.java
PS E:\JAVA Lab> java Worker
Enter Employee Name: Peter
Enter Base Salary: 150000
Employee Name: Peter
Base Salary (Yearly): Rs. 150000.0
Allowance: Rs. 111000.0
Total Salary: Rs. 261000.0
Tax Deducted at Source: Rs. 16100.0
Final Salary: Rs. 244900.0
```

## Practical 4 : Inheritance and interface

1. class Cricket having data members name, age and member methods display() and setdata(). class Match inherits Cricket and has data members no\_of\_odi, no\_of\_test. Create an array of 5 objects of class Match. Provide all the required data through command line and display the information.

```
class Player {
    String playerName;
    int playerAge;

    void setInfo(String pName, int pAge) {
        playerName = pName;
        playerAge = pAge;
    }

    void showInfo() {
        System.out.println("Player Name: " + playerName);
        System.out.println("Age: " + playerAge);
    }
}

class Game extends Player {
    int odiMatches;
    int testMatches;

    void setInfo(String pName, int pAge, int odi, int test) {
        super.setInfo(pName, pAge);
        odiMatches = odi;
        testMatches = test;
    }

    void showInfo() {
        super.showInfo();
    }
}
```

```

        System.out.println("ODI Matches: " + odiMatches);
        System.out.println("Test Matches: " + testMatches);
    }
}

```

```

class PlayerDemo {
    public static void main(String[] args) {
        Game[] players = new Game[5];
        int argIndex = 0;

        for(int i = 0; i < 5; i++) {
            players[i] = new Game();
            String name = args[argIndex++];
            int age = Integer.parseInt(args[argIndex++]);
            int odi = Integer.parseInt(args[argIndex++]);
            int test = Integer.parseInt(args[argIndex++]);

            players[i].setInfo(name, age, odi, test);
            System.out.println("\nPlayer " + i + " Details:");
            players[i].showInfo();
        }
    }
}

```

```

PS E:\JAVA Lab> javac PlayerDemo.java
PS E:\JAVA Lab> java PlayerDemo "Gill" 25 101 2 "Buttler" 42 35 21 "Rashid" 46 280 110 "Jason" 36 12 8 "Roshan" 30 10 20

Player 0 Details:
Player Name: Gill
Age: 25
ODI Matches: 101
Test Matches: 2

Player 1 Details:
Player Name: Buttler
Age: 42
ODI Matches: 35
Test Matches: 21

Player 2 Details:
Player Name: Rashid
Age: 46
ODI Matches: 280
Test Matches: 110

Player 3 Details:
Player Name: Jason
Age: 36
ODI Matches: 12
Test Matches: 8

Player 4 Details:
Player Name: Roshan
Age: 30
ODI Matches: 10
Test Matches: 20

```

2. Define a class Cipher with following data Field: String plainText; int key  
 Functions: Cipher(String plaintext,int key) abstract String Encryption( )  
 abstract String Decryption( ) Derived two classes Substitution\_Cipher and  
 Caesar\_Cipher override Encryption() and Decryption() Method. in substitute  
 cipher every character of string is replace with another character. For  
 example. In this method you will replace the letters using the following  
 scheme. Plain Text: a b c d e f g h i j k l m n o p q r s t u v w x y z Cipher Text:  
 q a z w s x e d c r f v t g b y h n u j m i k o l p So if string consist of letter  
 "gcet" then encrypted string will be "ezsj" and decrypt it to get original  
 string In ceaser cipher encrypt the string same as program 5 of LAB 5.

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

```
abstract class TextCoder {
    String inputText;
    int shiftValue;
```

```
    TextCoder(String text, int shift) {
        inputText = text;
        shiftValue = shift;
    }
```

```
    abstract String encode();
    abstract String decode();
}
```

```
class LetterSwapCoder extends TextCoder {
    String normal = "abcdefghijklmnopqrstuvwxyz";
    String coded = "qazwsxedcrfvtgbyhnujmikolp";
```

```
    LetterSwapCoder(String text, int shift) {
        super(text, shift);
    }
```

```
    String encode() {
        StringBuffer result = new StringBuffer();
```

```
        for(int i = 0; i < inputText.length(); i++) {
            int pos = normal.indexOf(inputText.charAt(i));
            result.append(coded.charAt(pos));
        }
        return result.toString();
    }

    String decode() {
        StringBuffer result = new StringBuffer();
        String encrypted = encode();
        for(int i = 0; i < encrypted.length(); i++) {
            int pos = coded.indexOf(encrypted.charAt(i));
            result.append(normal.charAt(pos));
        }
        return result.toString();
    }
}

class ShiftCoder extends TextCoder {
    ShiftCoder(String text, int shift) {
        super(text, shift);
    }

    String encode() {
        StringBuffer result = new StringBuffer();
        for(int i = 0; i < inputText.length(); i++) {
            char ch = (char)((inputText.charAt(i) - 'a' + shiftValue) % 26 + 'a');
            result.append(ch);
        }
        return result.toString();
    }

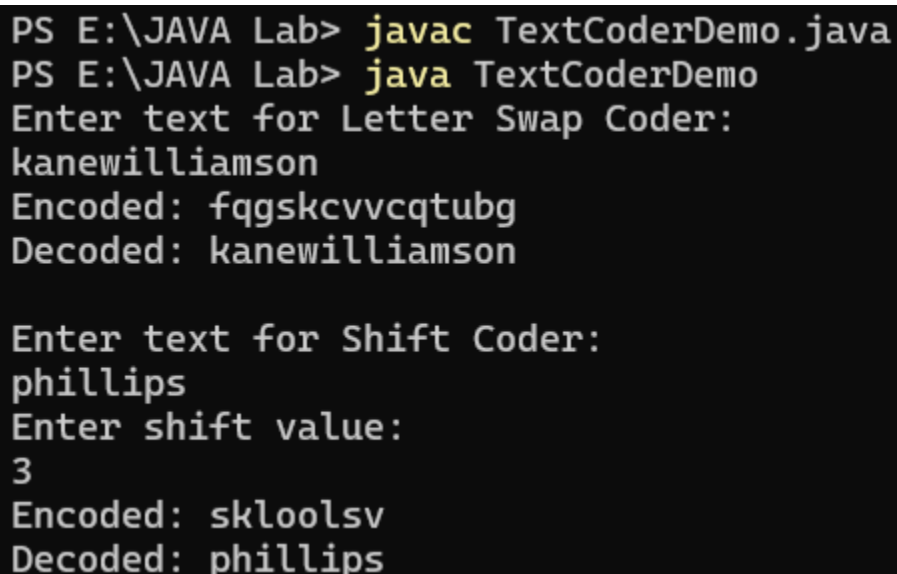
    String decode() {
        StringBuffer result = new StringBuffer();
        String encrypted = encode();
        for(int i = 0; i < encrypted.length(); i++) {
            char ch = (char)((encrypted.charAt(i) - 'a' - shiftValue + 26) % 26 + 'a');
```

```
        result.append(ch);
    }
    return result.toString();
}
}
```

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

        System.out.println("Enter text for Letter Swap Coder:");
        String swapText = input.nextLine();
        LetterSwapCoder swapCoder = new LetterSwapCoder(swapText, 0);
        System.out.println("Encoded: " + swapCoder.encode());
        System.out.println("Decoded: " + swapCoder.decode());

        System.out.println("\nEnter text for Shift Coder:");
        String shiftText = input.nextLine();
        System.out.println("Enter shift value:");
        int shift = input.nextInt();
        ShiftCoder shiftCoder = new ShiftCoder(shiftText, shift);
        System.out.println("Encoded: " + shiftCoder.encode());
        System.out.println("Decoded: " + shiftCoder.decode());
    }
}
```



```
PS E:\JAVA Lab> javac TextCoderDemo.java
PS E:\JAVA Lab> java TextCoderDemo
Enter text for Letter Swap Coder:
kanewilliamson
Encoded: fqgskcvvcqtubg
Decoded: kanewilliamson

Enter text for Shift Coder:
phillips
Enter shift value:
3
Encoded: skloolsv
Decoded: phillips
```

3. Declare an interface called Property containing a method computePrice to compute and return the price. The interface is to be implemented by following two classes i) Bungalow and ii) Flat. Both the classes have following data members

- name
- constructionArea

The class Bungalow has an additional data member called landArea. Define computePrice for both classes for computing total price. Use following rules for computing total price by summing up sub-costs:

Construction cost(for both classes):Rs.500/- per sq.feet

Additional cost ( for Flat) : Rs. 200000/-

( for Bungalow ): Rs. 200/- per sq.

feet for landArea Land cost ( only for Bungalow ): Rs. 400/- per sq. feet

Define method main to show usage of method computePrice.

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

```
interface Building {  
    double calculateCost();  
}
```

```
class House implements Building {
```

```
    String houseName;  
    double builtArea;  
    double gardenArea;
```

```
    House(String hName, double bArea, double gArea) {  
        houseName = hName;  
        builtArea = bArea;  
        gardenArea = gArea;  
    }
```

```
    public double calculateCost() {  
        System.out.println("House Name: " + houseName);  
        System.out.println("Built Area: " + builtArea);
```



```
        System.out.println("Garden Area: " + gardenArea);
        double buildCost = builtArea * 500;
        double extraCost = gardenArea * 200;
        double landCost = gardenArea * 400;
        return buildCost + extraCost + landCost;
    }
}
```

```
class Apartment implements Building {
    String aptName;
    double floorArea;

    Apartment(String aName, double fArea) {
        aptName = aName;
        floorArea = fArea;
    }

    public double calculateCost() {
        System.out.println("Apartment Name: " + aptName);
        System.out.println("Floor Area: " + floorArea);
        double buildCost = floorArea * 500;
        double extraCost = 200000;
        return buildCost + extraCost;
    }
}
```

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

        System.out.println("Enter House Details:");
        System.out.println("Name: ");
        String hName = input.nextLine();
        System.out.println("Built Area (sq.ft): ");
        double bArea = input.nextDouble();
        System.out.println("Garden Area (sq.ft): ");
        double gArea = input.nextDouble();
    }
}
```

```
House h = new House(hName, bArea, gArea);
System.out.println(hName + " Cost: " + h.calculateCost());

input.nextLine(); // clear buffer
System.out.println("\nEnter Apartment Details:");
System.out.println("Name: ");
String aName = input.nextLine();
System.out.println("Floor Area (sq.ft): ");
double fArea = input.nextDouble();

Apartment a = new Apartment(aName, fArea);
System.out.println(aName + " Cost: " + a.calculateCost());
}
}
```

```
PS E:\JAVA Lab> javac BuildingDemo.java
PS E:\JAVA Lab> java BuildingDemo
Enter House Details:
Name:
Jason
Built Area (sq.ft):
120
Garden Area (sq.ft):
100
House Name: Jason
Built Area: 120.0
Garden Area: 100.0
Jason Cost: 120000.0

Enter Apartment Details:
Name:
Roy
Floor Area (sq.ft):
12
Apartment Name: Roy
Floor Area: 12.0
Roy Cost: 206000.0
```

4. Define following classes and interfaces.

```
public interface GeometricShape {
    public void describe();
}

public interface TwoDShape extends GeometricShape {
    public double area();
}

public interface ThreeDShape extends GeometricShape {
    public double volume();
}

public class Cone implements ThreeDShape {
    private double radius;
    private double height;
    public Cone (double radius, double height)
    public double volume()
    public void describe()
}

public class Rectangle implements TwoDShape {
    private double width, height;
    public Rectangle (double width, double height)
    public double area()
    public double perimeter()
    public void describe()
}

public class Sphere implements ThreeDShape {
    private double radius;
    public Sphere (double radius)
    public double volume()
    public void describe()
}

Define test class to call various methods of Geometric Shape
```

```
import java.util.Scanner;

interface Shape {
    void info();
}

interface FlatShape extends Shape {
    double getArea();
}

interface SolidShape extends Shape {
    double getVolume();
}

class ConeShape implements SolidShape {
    private double baseRadius;
    private double coneHeight;

    ConeShape(double baseRadius, double coneHeight) {
        this.baseRadius = baseRadius;
        this.coneHeight = coneHeight;
    }

    public double getVolume() {
        return (1.0/3) * Math.PI * baseRadius * baseRadius * coneHeight;
    }

    public void info() {
        System.out.println("Cone - Radius: " + baseRadius + ", Height: " + coneHeight);
        System.out.println("Volume: " + getVolume());
    }
}

class RectShape implements FlatShape {
    private double rectWidth, rectHeight;

    RectShape(double rectWidth, double rectHeight) {
```

```
        this.rectWidth = rectWidth;
        this.rectHeight = rectHeight;
    }

    public double getArea() {
        return rectWidth * rectHeight;
    }

    public void info() {
        System.out.println("Rectangle - Width: " + rectWidth + ", Height: " + rectHeight);
        System.out.println("Area: " + getArea());
    }
}

class SphereShape implements SolidShape {
    private double sphereRadius;

    SphereShape(double sphereRadius) {
        this.sphereRadius = sphereRadius;
    }

    public double getVolume() {
        return (4.0/3) * Math.PI * sphereRadius * sphereRadius * sphereRadius;
    }

    public void info() {
        System.out.println("Sphere - Radius: " + sphereRadius);
        System.out.println("Volume: " + getVolume());
    }
}

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

        System.out.println("Enter Cone radius and height:");
        double cRad = input.nextDouble();
        double cHeight = input.nextDouble();
    }
}
```

```
ConeShape cone = new ConeShape(cRad, cHeight);
cone.info();

System.out.println("\nEnter Rectangle width and height:");
double rWidth = input.nextDouble();
double rHeight = input.nextDouble();
RectShape rect = new RectShape(rWidth, rHeight);
rect.info();

System.out.println("\nEnter Sphere radius:");
double sRad = input.nextDouble();
SphereShape sphere = new SphereShape(sRad);
sphere.info();
}
}
```

```
PS E:\JAVA Lab> javac ShapeTest.java
PS E:\JAVA Lab> java ShapeTest
Enter Cone radius and height:
3
5
Cone - Radius: 3.0, Height: 5.0
Volume: 47.12388980384689

Enter Rectangle width and height:
10
5
Rectangle - Width: 10.0, Height: 5.0
Area: 50.0

Enter Sphere radius:
4
Sphere - Radius: 4.0
Volume: 268.082573106329
```

## Practical 5 : Inner Class

Define two nested classes: Processor and RAM inside the outer class: CPU with following data members

```
class CPU {  
    double price;  
    class Processor{ // nested class  
        double cores;  
        double catch()  
        String manufacturer;  
        double getCache()  
        void displayProcesorDetail()  
    }  
    protected class RAM{  
        // nested protected class  
        // members of protected nested class  
        double memory;  
        String manufacturer;  
        Double clockSpeed;  
        double getClockSpeed()  
        void displayRAMDetail()  
    }  
}
```

1. Write appropriate Constructor and create instance of Outer and inner class and call the methods in main function

```
class CPU {
    double price;

    CPU(double p) {
        price = p;
    }

    class Processor {
        double cores;
        double cache;
        String manufacturer;

        Processor(double c, double ca, String m) {
            cores = c;
            cache = ca;
            manufacturer = m;
        }

        double getCache() {
            return cache;
        }

        void displayProcessorDetail() {
            System.out.println("Processor: Cores=" + cores + ", Cache=" + cache + "MB,
Manufacturer=" + manufacturer);
        }
    }

    protected class RAM {
        double memory;
        String manufacturer;
        Double clockSpeed;
    }
}
```



```
RAM(double m, String manu, Double cs) {
    memory = m;
    manufacturer = manu;
    clockSpeed = cs;
}

double getClockSpeed() {
    return clockSpeed;
}

void displayRAMDetail() {
    System.out.println("RAM: Memory=" + memory + "GB, ClockSpeed=" +
clockSpeed + "MHz, Manufacturer=" + manufacturer);
}

public static void main(String[] args) {
    CPU cpu = new CPU(299.99);
    CPU.Processor proc = cpu.new Processor(8, 16, "Intel");
    proc.displayProcessorDetail();
    System.out.println("Processor Cache: " + proc.getCache() + "MB");
    CPU.RAM ram = cpu.new RAM(32, "Corsair", 3200.0);
    ram.displayRAMDetail();
    System.out.println("RAM Clock Speed: " + ram.getClockSpeed() + "MHz");
    System.out.println("CPU Price: $" + cpu.price);
}
}
```

```
PS E:\JAVA Lab> javac CPU.java
PS E:\JAVA Lab> java CPU
Processor: Cores=8.0, Cache=16.0MB, Manufacturer=Intel
Processor Cache: 16.0MB
RAM: Memory=32.0GB, ClockSpeed=3200.0MHz, Manufacturer=Corsair
RAM Clock Speed: 3200.0MHz
CPU Price: $299.99
```

## 2. Write a program to demonstrate usage of static inner class, local inner class and anonymous inner class

```
class CPU {
    double price;

    CPU(double p) {
        price = p;
    }

    static class StaticFan {
        String type;

        StaticFan(String t) {
            type = t;
        }

        void showFan() {
            System.out.println("Fan Type: " + type);
        }
    }

    void cpuMethod() {
        class LocalCooler {
            int speed;

            LocalCooler(int s) {
                speed = s;
            }

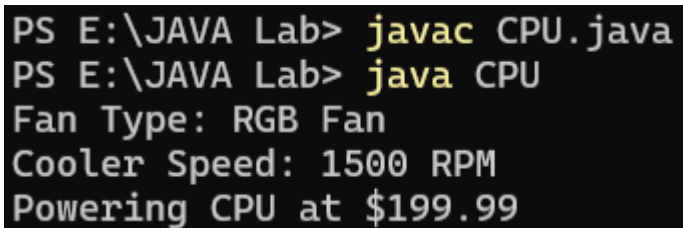
            void showCooler() {
                System.out.println("Cooler Speed: " + speed + " RPM");
            }
        }

        LocalCooler cooler = new LocalCooler(1500);
        cooler.showCooler();
    }
}
```

```
}

interface Power {
    void powerOn();
}

public static void main(String[] args) {
    CPU.StaticFan fan = new CPU.StaticFan("RGB Fan");
    fan.showFan();
    CPU cpu = new CPU(199.99);
    cpu.cpuMethod();
    Power powerSupply = new Power() {
        @Override
        public void powerOn() {
            System.out.println("Powering CPU at $" + cpu.price);
        }
    };
    powerSupply.powerOn();
}
```



```
PS E:\JAVA Lab> javac CPU.java
PS E:\JAVA Lab> java CPU
Fan Type: RGB Fan
Cooler Speed: 1500 RPM
Powering CPU at $199.99
```

## Practical 6 : Generics

1. Declare a class InvoiceDetail which accepts a type parameter which is of type Number with following data members class InvoiceDetail { private String invoiceName; private N amount; private N Discount // write getters, setters and constructors } Call the methods in Main class

```
class InvoiceDetail<N extends Number> {  
    private String invoiceName;  
    private N amount;  
    private N Discount;  
  
    public InvoiceDetail(String invoiceName, N amount, N Discount) {  
        this.invoiceName = invoiceName;  
        this.amount = amount;  
        this.Discount = Discount;  
    }  
  
    public String getInvoiceName() {  
        return invoiceName;  
    }  
  
    public void setInvoiceName(String invoiceName) {  
        this.invoiceName = invoiceName;  
    }  
  
    public N getAmount() {  
        return amount;  
    }  
  
    public void setAmount(N amount) {  
        this.amount = amount;  
    }  
}
```

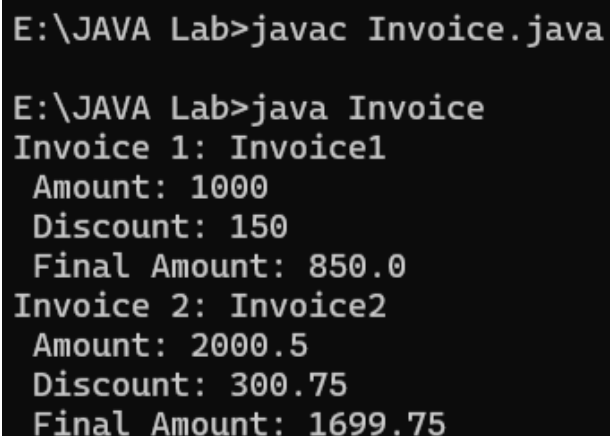
```
public N getDiscount() {
    return Discount;
}

public void setDiscount(N Discount) {
    this.Discount = Discount;
}
}

public class Invoice {
    public static void main(String[] args) {
        InvoiceDetail<Integer> invoice1 = new InvoiceDetail<>("Invoice1", 1000, 150);
        InvoiceDetail<Double> invoice2 = new InvoiceDetail<>("Invoice2", 2000.5,
300.75);

        System.out.println("Invoice 1: " + invoice1.getInvoiceName()
            + "\n Amount: " + invoice1.getAmount() + "\n Discount: " +
            invoice1.getDiscount() + "\n Final Amount: " +
            (invoice1.getAmount().doubleValue() - invoice1.getDiscount().doubleValue()));

        System.out.println("Invoice 2: " + invoice2.getInvoiceName()
            + "\n Amount: " + invoice2.getAmount() + "\n Discount: " +
            invoice2.getDiscount() + "\n Final Amount: " +
            (invoice2.getAmount().doubleValue() - invoice2.getDiscount().doubleValue()));
    }
}
```



```
E:\JAVA Lab>javac Invoice.java

E:\JAVA Lab>java Invoice
Invoice 1: Invoice1
Amount: 1000
Discount: 150
Final Amount: 850.0
Invoice 2: Invoice2
Amount: 2000.5
Discount: 300.75
Final Amount: 1699.75
```

## 2. Implement Generic Stack

```
import java.util.ArrayList;

class GenericStack<T> {
    private ArrayList<T> stack;

    public GenericStack() {
        stack = new ArrayList<>();
    }

    public void push(T item) {
        stack.add(item);
    }

    public T pop() {
        if (isEmpty()) {
            throw new RuntimeException("Stack is empty!");
        }
        return stack.remove(stack.size() - 1);
    }

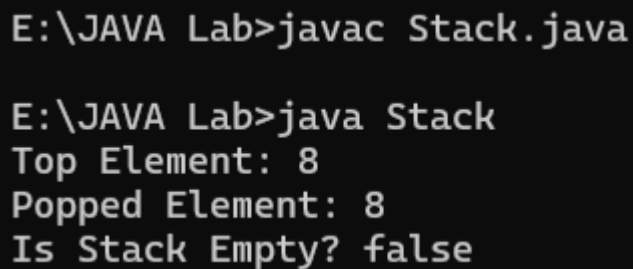
    public T peek() {
        if (isEmpty()) {
            throw new RuntimeException("Stack is empty!");
        }
        return stack.get(stack.size() - 1);
    }

    public boolean isEmpty() {
        return stack.isEmpty();
    }
}

public class Stack {
    public static void main(String[] args) {
        GenericStack<Integer> intStack = new GenericStack<>();
    }
}
```

```
intStack.push(2);
intStack.push(4);
intStack.push(8);

System.out.println("Top Element: " + intStack.peek());
System.out.println("Popped Element: " + intStack.pop());
System.out.println("Is Stack Empty? " + intStack.isEmpty());
}
}
```



```
E:\JAVA Lab>javac Stack.java

E:\JAVA Lab>java Stack
Top Element: 8
Popped Element: 8
Is Stack Empty? false
```

### 3. Write a program to sort the object of Book class using comparable

```
import java.util.*;

class Book implements Comparable<Book> {
    private int bookId;
    private String title;
    private String author;
    private String publisher;

    public Book(int bookId, String title, String author, String publisher) {
        this.bookId = bookId;
        this.title = title;
        this.author = author;
        this.publisher = publisher;
    }

    public int getBookId() {
        return bookId;
    }
}
```

```
public String getTitle() {
    return title;
}

public String getAuthor() {
    return author;
}

public String getPublisher() {
    return publisher;
}

public int compareTo(Book other) {
    return this.title.compareTo(other.title);
}

public String toString() {
    return "Book ID: " + bookId + ", Title: " + title + ", Author: " + author + ",
Publisher: " + publisher;
}
}

class AuthorComparator implements Comparator<Book> {
    public int compare(Book b1, Book b2) {
        return b1.getAuthor().compareTo(b2.getAuthor());
    }
}

public class Sort{
    public static void main(String[] args) {
        List<Book> books = new ArrayList<>();

        // One-word titles and authors
        books.add(new Book(401, "Java", "John", "Tech"));
        books.add(new Book(402, "Code", "Mike", "Soft"));
        books.add(new Book(403, "Data", "Anna", "Info"));
        books.add(new Book(404, "Web", "Lisa", "Net"));
```



```
Collections.sort(books);
System.out.println("Books sorted by Title:");
for (Book book : books) {
    System.out.println(book);
}

Collections.sort(books, new AuthorComparator());
System.out.println("\nBooks sorted by Author:");
for (Book book : books) {
    System.out.println(book);
}
}
```

```
E:\JAVA Lab>javac Sort.java
```

```
E:\JAVA Lab>java Sort
```

```
Books sorted by Title:
```

```
Book ID: 402, Title: Code, Author: Mike, Publisher: Soft
Book ID: 403, Title: Data, Author: Anna, Publisher: Info
Book ID: 401, Title: Java, Author: John, Publisher: Tech
Book ID: 404, Title: Web, Author: Lisa, Publisher: Net
```

```
Books sorted by Author:
```

```
Book ID: 403, Title: Data, Author: Anna, Publisher: Info
Book ID: 401, Title: Java, Author: John, Publisher: Tech
Book ID: 404, Title: Web, Author: Lisa, Publisher: Net
Book ID: 402, Title: Code, Author: Mike, Publisher: Soft
```

## Practical 7 : Exception Handling

1. Write a program for creating a Bank class, which is used to manage the bank account of customers. Class has two methods, Deposit () and withdraw (). Deposit method display old balance and new balance after depositing the specified amount. Withdrew method display old balance and new balance after withdrawing. If balance is not enough to withdraw the money, it throws ArithmeticException and if balance is less than 500rs after withdrawing then it throw custom exception, NotEnoughMoneyException.

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

```
class NotEnoughMoneyException extends Exception {  
    public NotEnoughMoneyException(String message) {  
        super(message);  
    }  
}
```

```
class Bank {  
    private double balance;  
  
    public Bank(double init_balance) {  
        this.balance = init_balance;  
    }  
  
    public void Deposit(double amount) {  
        System.out.println("Old Balance: " + balance);  
        balance += amount;  
        System.out.println("Deposited: " + amount);  
        System.out.println("New Balance: " + balance);  
    }  
}
```

```
public void withdraw(double amount) throws ArithmeticException,
NotEnoughMoneyException {
    System.out.println("Old Balance: " + balance);
    if (balance < amount) {
        throw new ArithmeticException("Insufficient balance to withdraw " + amount);
    }
    balance -= amount;
    if (balance < 500) {
        throw new NotEnoughMoneyException("Balance is below 500 after
withdrawal. Current balance: " + balance);
    }
    System.out.println("Withdrawn: " + amount);
    System.out.println("New Balance: " + balance);
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    Bank account = new Bank(1000);
    System.out.println("Welcome to the Bank! Your initial balance is: 1000");

    while (true) {
        System.out.println("\nChoose an option:");
        System.out.println("1. Deposit money");
        System.out.println("2. Withdraw money");
        System.out.println("3. Exit");
        System.out.print("Enter your choice (1/2/3): ");
        int choice = scanner.nextInt();

        if (choice == 1) {
            System.out.print("Enter the amount to deposit: ");
            double depositAmount = scanner.nextDouble();
            account.Deposit(depositAmount);
        } else if (choice == 2) {
            System.out.print("Enter the amount to withdraw: ");
            double withdrawAmount = scanner.nextDouble();
            try {
                account.withdraw(withdrawAmount);
            } catch (ArithmeticException | NotEnoughMoneyException e) {
                System.out.println(e.getMessage());
            }
        } else if (choice == 3) {
            System.out.println("Exiting the program...");
            break;
        }
    }
}
```

```
    } catch (ArithmeticException e) {
        System.out.println("Error: " + e.getMessage());
    } catch (NotEnoughMoneyException e) {
        System.out.println("Error: " + e.getMessage());
    }
} else if (choice == 3) {
    System.out.println("Exiting the bank system. Thank you!");
    break;
} else {
    System.out.println("Invalid choice. Please try again.");
}
}
scanner.close();
}
```

```
PS E:\JAVA Lab> javac Bank.java
PS E:\JAVA Lab> java Bank
Welcome to the Bank! Your initial balance is: 1000

Choose an option:
1. Deposit money
2. Withdraw money
3. Exit
Enter your choice (1/2/3): 1
Enter the amount to deposit: 1000
Old Balance: 1000.0
Deposited: 1000.0
New Balance: 2000.0

Choose an option:
1. Deposit money
2. Withdraw money
3. Exit
Enter your choice (1/2/3): 2
Enter the amount to withdraw: 1501
Old Balance: 2000.0
Error: Balance is below 500 after withdrawal. Current balance: 499.0

Choose an option:
1. Deposit money
2. Withdraw money
3. Exit
Enter your choice (1/2/3): 2
Enter the amount to withdraw: 501
Old Balance: 499.0
Error: Insufficient balance to withdraw 501.0

Choose an option:
1. Deposit money
2. Withdraw money
3. Exit
Enter your choice (1/2/3): 3
Exiting the bank system. Thank you!
```

2. Write a complete program for calculation average of n +ve integer numbers of Array A.
- a. Read the array form keyboard
  - b. Raise and handle Exception if
    - i. Element value is -ve or non-integer.
- If n is zero.

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

```
public class AverageCalculator {  
    static class InvalidNumberException extends Exception {  
        public InvalidNumberException(String message) {  
            super(message);  
        }  
    }  
}  
  
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
    try {  
        System.out.print("Enter the number of elements (n): ");  
        int n = scanner.nextInt();  
        if (n == 0) {  
            throw new IllegalArgumentException("Cannot calculate average for zero  
elements.");  
        }  
        int[] A = new int[n];  
        System.out.println("Enter " + n + " positive integers: ");  
        for (int i = 0; i < n; i++) {  
            System.out.print("Element " + (i + 1) + ": ");  
            String input = scanner.next();  
            try {  
                int value = Integer.parseInt(input);  
                if (value < 0) {  
                    throw new InvalidNumberException("Negative number entered: " +  
value);  
                }  
            }  
        }  
    }  
}
```

```

    }

    A[i] = value;
} catch (NumberFormatException e) {
    System.out.println("Error: Please enter a valid integer.");
    i--;
} catch (InvalidNumberException e) {
    System.out.println(e.getMessage());
    i--;
}
}
int sum = 0;
for (int i = 0; i < n; i++) {
    sum += A[i];
}
double average = (double) sum / n;
System.out.println("The average is: " + average);
} catch (IllegalArgumentException e) {
    System.out.println(e.getMessage());
} finally {
    scanner.close();
}
}
}

```

```

PS E:\JAVA Lab> javac AverageCalculator.java
PS E:\JAVA Lab> java AverageCalculator
Enter the number of elements (n): 4
Enter 4 positive integers:
Element 1: 20
Element 2: -4
Negative number entered: -4
Element 2: 12
Element 3: 0
Element 4: 3.5
Error: Please enter a valid integer.
Element 4: 2
The average is: 8.5
PS E:\JAVA Lab> javac AverageCalculator.java
PS E:\JAVA Lab> java AverageCalculator
Enter the number of elements (n): 0
Cannot calculate average for zero elements.

```

## Practical 8 : Threading

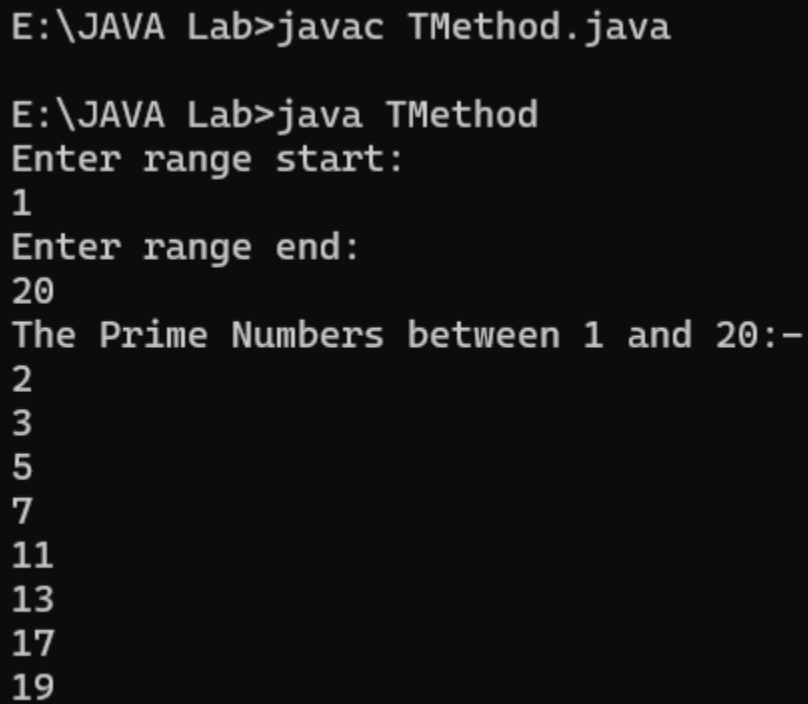
1. Write a program to find prime number in given range using both method of multithreading. Also run the same program using executor framework

- Extending Thread Class

```
import java.util.*;

class ThreadT extends Thread {
    public void run() {
        int flag;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter range start: ");
        int s = sc.nextInt();
        System.out.println("Enter range end: ");
        int e = sc.nextInt();
        System.out.println("The Prime Numbers between " + s + " and " + e + ":-");
        if (e > 2) {
            for (int i = s; i <= e; i++) {
                if (i < 2) continue;
                flag = 0;
                for (int j = 2; j < i; j++) {
                    if (i % j == 0) {
                        flag = 1;
                        break;
                    }
                }
                if (flag == 0) System.out.println(i);
            }
        } else {
            System.out.println("No Prime Number");
        }
    }
}
```

```
class TMethod {  
    public static void main(String[] args) {  
        ThreadT t = new ThreadT();  
        t.start();  
    }  
}
```



```
E:\JAVA Lab>javac TMethod.java  
  
E:\JAVA Lab>java TMethod  
Enter range start:  
1  
Enter range end:  
20  
The Prime Numbers between 1 and 20:-  
2  
3  
5  
7  
11  
13  
17  
19
```

- Implementing Runnable Interface

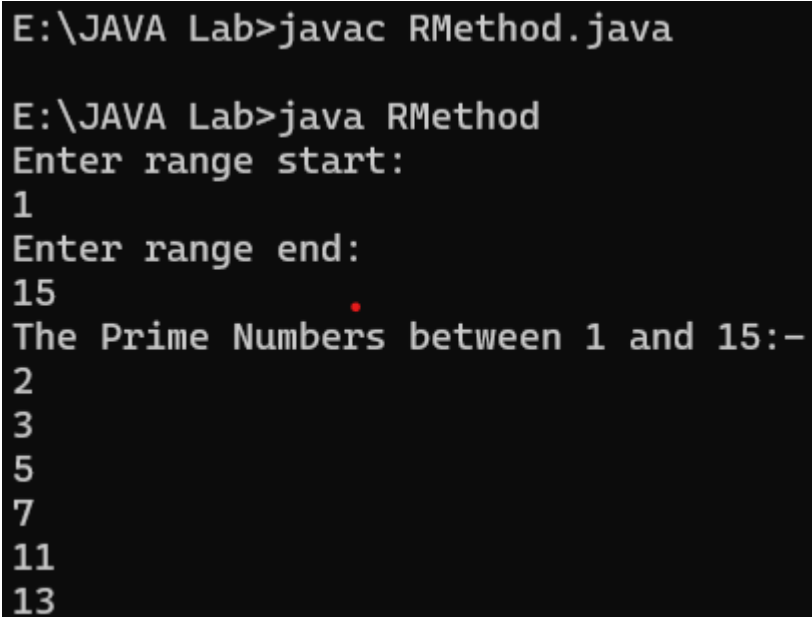
```
import java.util.*;
```

```
class RunnableT implements Runnable {  
    public void run() {  
        int flag;  
        Scanner sc = new Scanner(System.in);  
        System.out.println("Enter range start: ");  
        int s = sc.nextInt();  
        System.out.println("Enter range end: ");  
        int e = sc.nextInt();  
        System.out.println("The Prime Numbers between " + s + " and " + e + ":-");  
        if (e > 2) {  
            for (int i = s; i <= e; i++) {  
                if (i < 2) continue;  
                flag = 0;  
                for (int j = 2; j < i; j++) {
```



```
        if (i % j == 0) {
            flag = 1;
            break;
        }
    }
    if (flag == 0) System.out.println(i);
}
} else {
    System.out.println("No Prime Number");
}
}
}
```

```
class RMethod {
    public static void main(String[] args) {
        RunnableT r = new RunnableT();
        Thread t = new Thread(r);
        t.start();
    }
}
```



```
E:\JAVA Lab>javac RMethod.java

E:\JAVA Lab>java RMethod
Enter range start:
1
Enter range end:
15
The Prime Numbers between 1 and 15:-
2
3
5
7
11
13
```

- Using Executor Framework

```
import java.util.*;
import java.util.concurrent.*;

class RunnableT implements Runnable {
    public void run() {
        int flag;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter range start: ");
        int s = sc.nextInt();
        System.out.println("Enter range end: ");
        int e = sc.nextInt();
        System.out.println("The Prime Numbers between " + s + " and " + e + ":-");
        if (e > 2) {
            for (int i = s; i <= e; i++) {
                if (i < 2) continue;
                flag = 0;
                for (int j = 2; j < i; j++) {
                    if (i % j == 0) {
                        flag = 1;
                        break;
                    }
                }
                if (flag == 0) System.out.println(i);
            }
        } else {
            System.out.println("No Prime Number");
        }
    }
}

class EMethod {
    public static void main(String[] args) {
        ExecutorService executor = Executors.newSingleThreadExecutor();
        executor.execute(new RunnableT());
        executor.shutdown();
        try {
            executor.awaitTermination(10, TimeUnit.SECONDS);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

```
E:\JAVA Lab>javac EMethod.java

E:\JAVA Lab>java EMethod
Enter range start:
1
Enter range end:
10
The Prime Numbers between 1 and 10:-
2
3
5
7
```

2. Assume one class Queue that defines queue of fix size says 15.
- Assume one class producer which implements Runnable, having priority NORM\_PRIORITY +1
  - One more class consumer implements Runnable, having priority NORM\_PRIORITY-1
  - Class TestThread is having main method with maximum priority, which creates 1 thread for producer and 2 threads for consumer.
  - Producer produces number of elements and put on the queue. when queue becomes full it notifies other threads.
- Consumer consumes number of elements and notifies other thread when queue become empty.

```
class Queue {
    private final int[] queue;
    private int front, rear, size;
    private final int CAPACITY = 15;

    public Queue() {
        queue = new int[CAPACITY];
        front = rear = size = 0;
    }

    synchronized void produce(int value) throws InterruptedException {
        while (size == CAPACITY) {
```

```
        wait();
    }
    queue[rear] = value;
    rear = (rear + 1) % CAPACITY;
    size++;
    System.out.println("Produced: " + value);
    notifyAll();
}

synchronized int consume() throws InterruptedException {
    while (size == 0) {
        wait();
    }
    int value = queue[front];
    front = (front + 1) % CAPACITY;
    size--;
    System.out.println("Consumed: " + value);
    notifyAll();
    return value;
}
}

class Producer implements Runnable {
    private final Queue queue;

    public Producer(Queue queue) {
        this.queue = queue;
    }

    public void run() {
        try {
            for (int i = 1; i <= 30; i++) {
                queue.produce(i);
                Thread.sleep(100);
            }
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}

class Consumer implements Runnable {
    private final Queue queue;

    public Consumer(Queue queue) {
        this.queue = queue;
    }
}
```

```
        public void run() {
            try {
                for (int i = 0; i < 15; i++) {
                    queue.consume();
                    Thread.sleep(150);
                }
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }

    public class TestThread {
        public static void main(String[] args) {
            Queue queue = new Queue();

            Thread producerThread = new Thread(new Producer(queue));
            Thread consumerThread1 = new Thread(new Consumer(queue));
            Thread consumerThread2 = new Thread(new Consumer(queue));

            producerThread.setPriority(Thread.NORM_PRIORITY + 1);
            consumerThread1.setPriority(Thread.NORM_PRIORITY - 1);
            consumerThread2.setPriority(Thread.NORM_PRIORITY - 1);

            Thread.currentThread().setPriority(Thread.MAX_PRIORITY);

            producerThread.start();
            consumerThread1.start();
            consumerThread2.start();
        }
    }
```

```
E:\JAVA Lab>javac TestThread.java

E:\JAVA Lab>java TestThread
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Consumed: 3
Produced: 4
Consumed: 4
Produced: 5
Consumed: 5
Produced: 6
Consumed: 6
Produced: 7
Consumed: 7
Produced: 8
Consumed: 8
Produced: 9
Consumed: 9
Produced: 10
Consumed: 10
Produced: 11
Consumed: 11
Produced: 12
Consumed: 12
Produced: 13
Consumed: 13
Produced: 14
Consumed: 14
Produced: 15
Consumed: 15
Produced: 16
Consumed: 16
Produced: 17
Consumed: 17
Produced: 18
Consumed: 18
Produced: 19
Consumed: 19
Produced: 20
Consumed: 20
Produced: 21
Consumed: 21
Produced: 22
Consumed: 22
Produced: 23
Consumed: 23
Produced: 24
Consumed: 24
Produced: 25
Consumed: 25
Produced: 26
Consumed: 26
Produced: 27
Consumed: 27
Produced: 28
Consumed: 28
Produced: 29
Consumed: 29
Produced: 30
```

## Practical 9 : Collection API

1. Write a program to demonstrate user of ArrayList, LinkedList, LinkedHashMap, TreeMap and HashSet Class. And also implement CRUD operation without database connection using Collection API. Write a program to Sort Array, ArrayList, String, List, Map and Set

```
import java.util.*;

public class CollectionOperations {

    private ArrayList<Double> arrayList = new ArrayList<>();
    private LinkedList<Character> linkedList = new LinkedList<>();
    private LinkedHashMap<String, Double> linkedHashMap = new LinkedHashMap<>();
    private TreeMap<Integer, String> treeMap = new TreeMap<>();
    private HashSet<Integer> hashSet = new HashSet<>();

    public void performCRUD() {
        Scanner scanner = new Scanner(System.in);
        int choice;
        do {
            System.out.println("\nCollection Operations Menu:");
            System.out.println("1. Add to ArrayList (Doubles)");
            System.out.println("2. Display ArrayList");
            System.out.println("3. Update ArrayList");
            System.out.println("4. Remove from ArrayList");
            System.out.println("5. Add to LinkedList (Characters)");
            System.out.println("6. Display LinkedList");
```

```
System.out.println("7. Add to LinkedHashMap (String-Double)");
System.out.println("8. Display LinkedHashMap");
System.out.println("9. Add to TreeMap (Int-String)");
System.out.println("10. Display TreeMap");
System.out.println("11. Add to HashSet (Integers)");
System.out.println("12. Display HashSet");
System.out.println("13. Sort Demonstrations");
System.out.println("0. Exit");
System.out.print("Enter your choice: ");

choice = scanner.nextInt();
scanner.nextLine();

switch (choice) {
    case 1:
        System.out.print("Enter a double value: ");
        double dValue = scanner.nextDouble();
        arrayList.add(dValue);
        System.out.println("Added: " + dValue);
        break;

    case 2:
        System.out.println("ArrayList: " + arrayList);
        break;

    case 3:
        System.out.print("Enter index to update (0-" + (arrayList.size()-1) + "): ");
        int aIndex = scanner.nextInt();
        if (aIndex >= 0 && aIndex < arrayList.size()) {
            System.out.print("Enter new double value: ");
```



```
double newDValue = scanner.nextDouble();  
arrayList.set(aIndex, newDValue);  
System.out.println("Updated at index " + aIndex);  
} else {  
    System.out.println("Invalid index!");  
}  
break;
```

case 4:

```
System.out.print("Enter index to remove (0-" + (arrayList.size()-1) + "): ");  
int delIndex = scanner.nextInt();  
if (delIndex >= 0 && delIndex < arrayList.size()) {  
    arrayList.remove(delIndex);  
    System.out.println("Removed element at index " + delIndex);  
} else {  
    System.out.println("Invalid index!");  
}  
break;
```

case 5:

```
System.out.print("Enter a character: ");  
char cValue = scanner.nextLine().charAt(0);  
linkedList.add(cValue);  
System.out.println("Added: " + cValue);  
break;
```

case 6:

```
System.out.println("LinkedList: " + linkedList);  
break;
```

case 7:

```
System.out.print("Enter key (String): ");  
String lKey = scanner.nextLine();  
System.out.print("Enter value (Double): ");  
double lValue = scanner.nextDouble();  
linkedHashMap.put(lKey, lValue);  
System.out.println("Added to LinkedHashMap");  
break;
```

case 8:

```
System.out.println("LinkedHashMap: " + linkedHashMap);  
break;
```

case 9:

```
System.out.print("Enter key (Integer): ");  
int tKey = scanner.nextInt();  
scanner.nextLine();  
System.out.print("Enter value (String): ");  
String tValue = scanner.nextLine();  
treeMap.put(tKey, tValue);  
System.out.println("Added to TreeMap");  
break;
```

case 10:

```
System.out.println("TreeMap: " + treeMap);  
break;
```

case 11:

```
System.out.print("Enter an integer: ");  
int hValue = scanner.nextInt();
```

```
        hashSet.add(hValue);

        System.out.println("Added to HashSet (if unique)");

        break;

    case 12:

        System.out.println("HashSet: " + hashSet);

        break;

    case 13:

        demonstrateSorting();

        break;

    case 0:

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

        break;

    default:

        System.out.println("Invalid choice! Try again.");

    }

} while (choice != 0);

scanner.close();

}

private void demonstrateSorting() {

    int[] intArray = {45, 12, 89, 34, 67};

    Arrays.sort(intArray);

    System.out.println("Sorted Integer Array: " + Arrays.toString(intArray));

    ArrayList<String> sortList = new ArrayList<>(Arrays.asList("Zebra", "Apple", "Cat", "Dog"));

    Collections.sort(sortList);
```

```
        System.out.println("Sorted ArrayList: " + sortList);

        String original = "hello";
        char[] charArray = original.toCharArray();
        Arrays.sort(charArray);
        String sortedString = new String(charArray);
        System.out.println("Sorted String: " + sortedString);

        List<Double> numberList = new ArrayList<>(Arrays.asList(3.5, 1.2, 4.8, 2.1));
        Collections.sort(numberList);
        System.out.println("Sorted List: " + numberList);

        Map<Integer, String> map = new TreeMap<>();
        map.put(3, "Three");
        map.put(1, "One");
        map.put(2, "Two");
        System.out.println("Sorted Map by Keys: " + map);

        Set<String> set = new TreeSet<>(Arrays.asList("Banana", "Apple", "Cherry", "Date"));
        System.out.println("Sorted Set: " + set);
    }

    public static void main(String[] args) {
        CollectionOperations demo = new CollectionOperations();
        demo.performCRUD();
    }
}
```

```

E:\JAVA Lab>javac CollectionOperations.java

E:\JAVA Lab>java CollectionOperations

Collection Operations Menu:
1. Add to ArrayList (Doubles)
2. Display ArrayList
3. Update ArrayList
4. Remove from ArrayList
5. Add to LinkedList (Characters)
6. Display LinkedList
7. Add to LinkedHashMap (String-Double)
8. Display LinkedHashMap
9. Add to TreeMap (Int-String)
10. Display TreeMap
11. Add to HashSet (Integers)
12. Display HashSet
13. Sort Demonstrations
0. Exit
Enter your choice: 1
Enter a double value: 10.5
Added: 10.5

Collection Operations Menu:
1. Add to ArrayList (Doubles)
2. Display ArrayList
3. Update ArrayList
4. Remove from ArrayList
5. Add to LinkedList (Characters)
6. Display LinkedList
7. Add to LinkedHashMap (String-Double)
8. Display LinkedHashMap
9. Add to TreeMap (Int-String)
10. Display TreeMap
11. Add to HashSet (Integers)
12. Display HashSet
13. Sort Demonstrations
0. Exit
Enter your choice: 2
ArrayList: [10.5]

```

```

Collection Operations Menu:
1. Add to ArrayList (Doubles)
2. Display ArrayList
3. Update ArrayList
4. Remove from ArrayList
5. Add to LinkedList (Characters)
6. Display LinkedList
7. Add to LinkedHashMap (String-Double)
8. Display LinkedHashMap
9. Add to TreeMap (Int-String)
10. Display TreeMap
11. Add to HashSet (Integers)
12. Display HashSet
13. Sort Demonstrations
0. Exit
Enter your choice: 5
Enter a character: A
Added: A

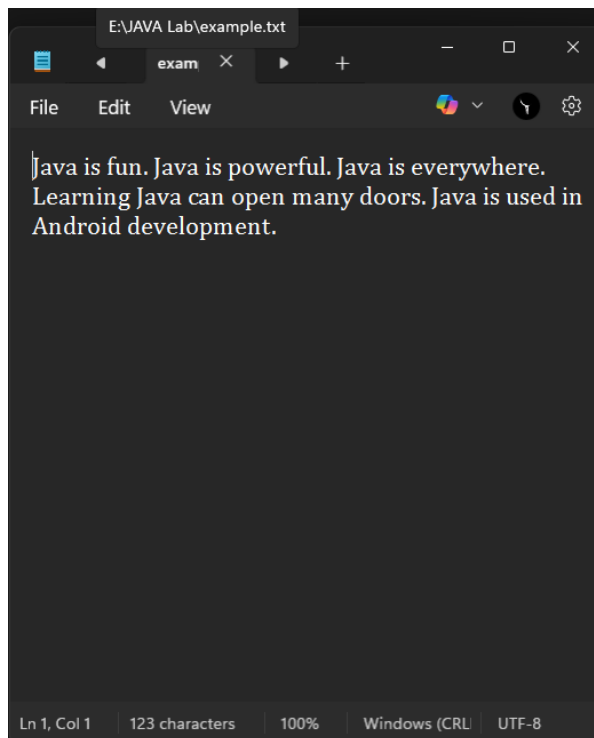
Collection Operations Menu:
1. Add to ArrayList (Doubles)
2. Display ArrayList
3. Update ArrayList
4. Remove from ArrayList
5. Add to LinkedList (Characters)
6. Display LinkedList
7. Add to LinkedHashMap (String-Double)
8. Display LinkedHashMap
9. Add to TreeMap (Int-String)
10. Display TreeMap
11. Add to HashSet (Integers)
12. Display HashSet
13. Sort Demonstrations
0. Exit
Enter your choice: 13
Sorted Integer Array: [12, 34, 45, 67, 89]
Sorted ArrayList: [Apple, Cat, Dog, Zebra]
Sorted String: ehlllo
Sorted List: [1.2, 2.1, 3.5, 4.8]
Sorted Map by Keys: {1=One, 2=Two, 3=Three}

```

## Practical 10 : File Handling Using Java

1. Write a program to count occurrence of a given words in a file.

Create one text file:



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

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

        System.out.print("Enter file path: ");
        String filePath = scanner.nextLine();
```

```
System.out.print("Enter word to count: ");

String word = scanner.nextLine();


int count = 0;

try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {

    String line;

    while ((line = br.readLine()) != null) {

        String[] words = line.split("\\s+");

        for (String w : words) {

            if (w.equalsIgnoreCase(word)) {

                count++;

            }

        }

    }

    System.out.println("The word '" + word + "' occurred " + count + " times.");

} catch (IOException e) {

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

}


scanner.close();

}
```

```
E:\JAVA Lab>javac WordCountInFile.java

E:\JAVA Lab>java WordCountInFile
Enter file path: E:\JAVA Lab\example.txt
Enter word to count: Java
The word 'Java' occurred 5 times.
```

## 2. Write a program to print itself.

```
import java.io.*;

public class SelfPrinting {
    public static void main(String[] args) {
        try (BufferedReader br = new BufferedReader(new FileReader("SelfPrinting.java"))) {
            String line;
            while ((line = br.readLine()) != null) {
                System.out.println(line);
            }
        } catch (IOException e) {
            System.out.println("File not found or error reading file.");
        }
    }
}
```

```
E:\JAVA Lab>javac SelfPrinting.java

E:\JAVA Lab>java SelfPrinting
import java.io.*;

public class SelfPrinting {
    public static void main(String[] args) {
        try (BufferedReader br = new BufferedReader(new FileReader("SelfPrinting.java"))) {
            String line;
            while ((line = br.readLine()) != null) {
                System.out.println(line);
            }
        } catch (IOException e) {
            System.out.println("File not found or error reading file.");
        }
    }
}
```

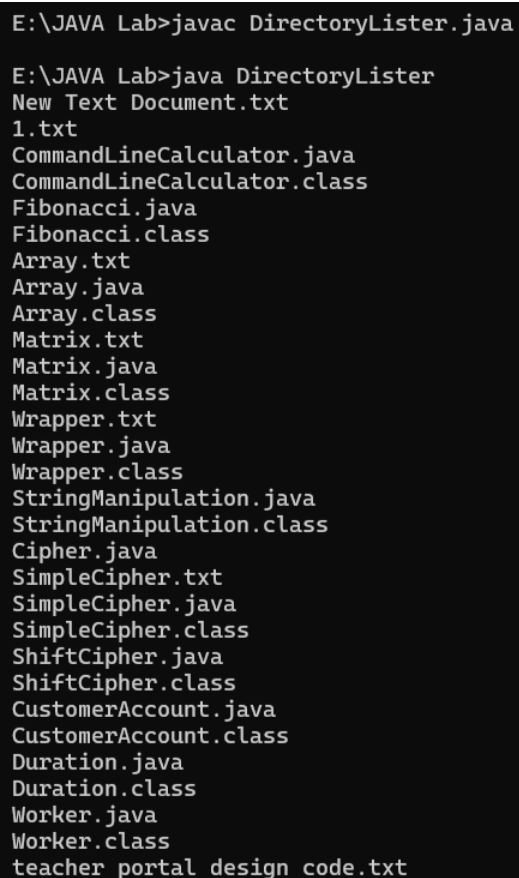


### 3. Write a program to display list of all the files of given directory

```
import java.io.File;

public class DirectoryLister {
    public static void main(String[] args) {
        String directoryPath = "E:/JAVA Lab";
        File directory = new File(directoryPath);

        if (directory.isDirectory()) {
            File[] files = directory.listFiles();
            if (files != null) {
                for (File file : files) {
                    System.out.println(file.getName());
                }
            }
        } else {
            System.out.println("Not a valid directory");
        }
    }
}
```



```
E:\JAVA Lab>javac DirectoryLister.java

E:\JAVA Lab>java DirectoryLister
New Text Document.txt
1.txt
CommandLineCalculator.java
CommandLineCalculator.class
Fibonacci.java
Fibonacci.class
Array.txt
Array.java
Array.class
Matrix.txt
Matrix.java
Matrix.class
Wrapper.txt
Wrapper.java
Wrapper.class
StringManipulation.java
StringManipulation.class
Cipher.java
SimpleCipher.txt
SimpleCipher.java
SimpleCipher.class
ShiftCipher.java
ShiftCipher.class
CustomerAccount.java
CustomerAccount.class
Duration.java
Duration.class
Worker.java
Worker.class
teacher portal design code.txt
```

# Practical 11 : Networking

## 1. Implement Echo client/server program using TCP

- Save as EchoServer.java

```
import java.io.*;
import java.net.*;

public class EchoServer {
    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(12345);
            System.out.println("Server is running on port 12345...");

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("New client connected");

                new Thread(() -> {
                    try {
                        BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                        PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

                        String message;
                        while ((message = in.readLine()) != null) {
                            System.out.println("Received: " + message);
```

```
        out.println("Echo: " + message);
    }

    clientSocket.close();
} catch (IOException e) {
    System.out.println("Error: " + e.getMessage());
}
}).start();
}
} catch (IOException e) {
    System.out.println("Server error: " + e.getMessage());
}
}
}
```

- EchoClient.java

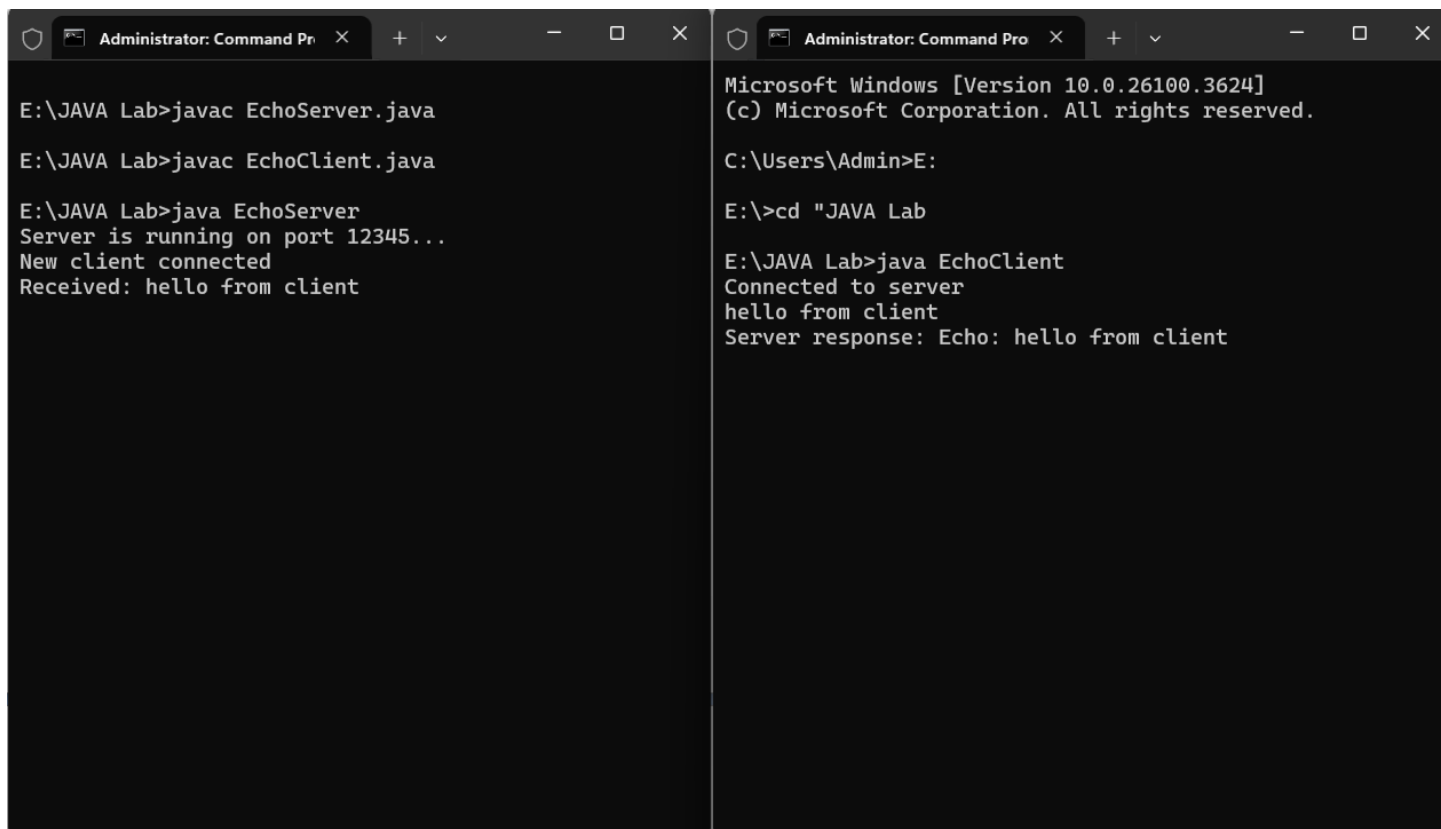
```
import java.io.*;
import java.net.*;

public class EchoClient {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("localhost", 12345);
            System.out.println("Connected to server");

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));
```

```
String message;
while (true) {
    message = userInput.readLine();
    if (message.equals("exit")) break;

    out.println(message);
    String response = in.readLine();
    System.out.println("Server response: " + response);
}
socket.close();
} catch (IOException e) {
    System.out.println("Client error: " + e.getMessage());
}
}
```



```
Administrator: Command Pr x + - □ x
E:\JAVA Lab>javac EchoServer.java
E:\JAVA Lab>javac EchoClient.java
E:\JAVA Lab>java EchoServer
Server is running on port 12345...
New client connected
Received: hello from client

Administrator: Command Pro x + - □ x
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Admin>E:
E:>cd "JAVA Lab
E:\JAVA Lab>java EchoClient
Connected to server
hello from client
Server response: Echo: hello from client
```

## 2. Write a program using UDP which give name of the audio file to server and server reply with content of audio file

- Save as UDPAudioServer.java:

```
import java.io.*;
import java.net.*;

public class UDPAudioServer {
    public static void main(String[] args) {
        try {
            DatagramSocket serverSocket = new DatagramSocket(12346);
            System.out.println("UDP Server is running on port 12346...");

            while (true) {
                byte[] receiveData = new byte[1024];
                DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
                serverSocket.receive(receivePacket);

                String filename = new String(receivePacket.getData(), 0, receivePacket.getLength());
                System.out.println("Received filename request: " + filename);

                // Read audio file
                File audioFile = new File(filename);
                byte[] fileData = new byte[(int)audioFile.length()];
                FileInputStream fis = new FileInputStream(audioFile);
                fis.read(fileData);
                fis.close();

                // Send file content
```

```
        InetAddress clientAddress = receivePacket.getAddress();

        int clientPort = receivePacket.getPort();

        DatagramPacket sendPacket = new DatagramPacket(fileData, fileData.length, clientAddress,
clientPort);

        serverSocket.send(sendPacket);

        System.out.println("Sent audio file content");
    }
} catch (IOException e) {

    System.out.println("Server error: " + e.getMessage());
}
}
}
```

- Save as UDPAudioClient.java:

```
import java.io.*;

import java.net.*;

public class UDPAudioClient {

    public static void main(String[] args) {

        try {

            DatagramSocket clientSocket = new DatagramSocket();

            InetAddress serverAddress = InetAddress.getByName("localhost");

            int serverPort = 12346;

            // Send filename

            String filename = "audio.mp3"; // Change this to your audio file name

            byte[] sendData = filename.getBytes();

            DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
serverAddress, serverPort);
```

```
clientSocket.send(sendPacket);

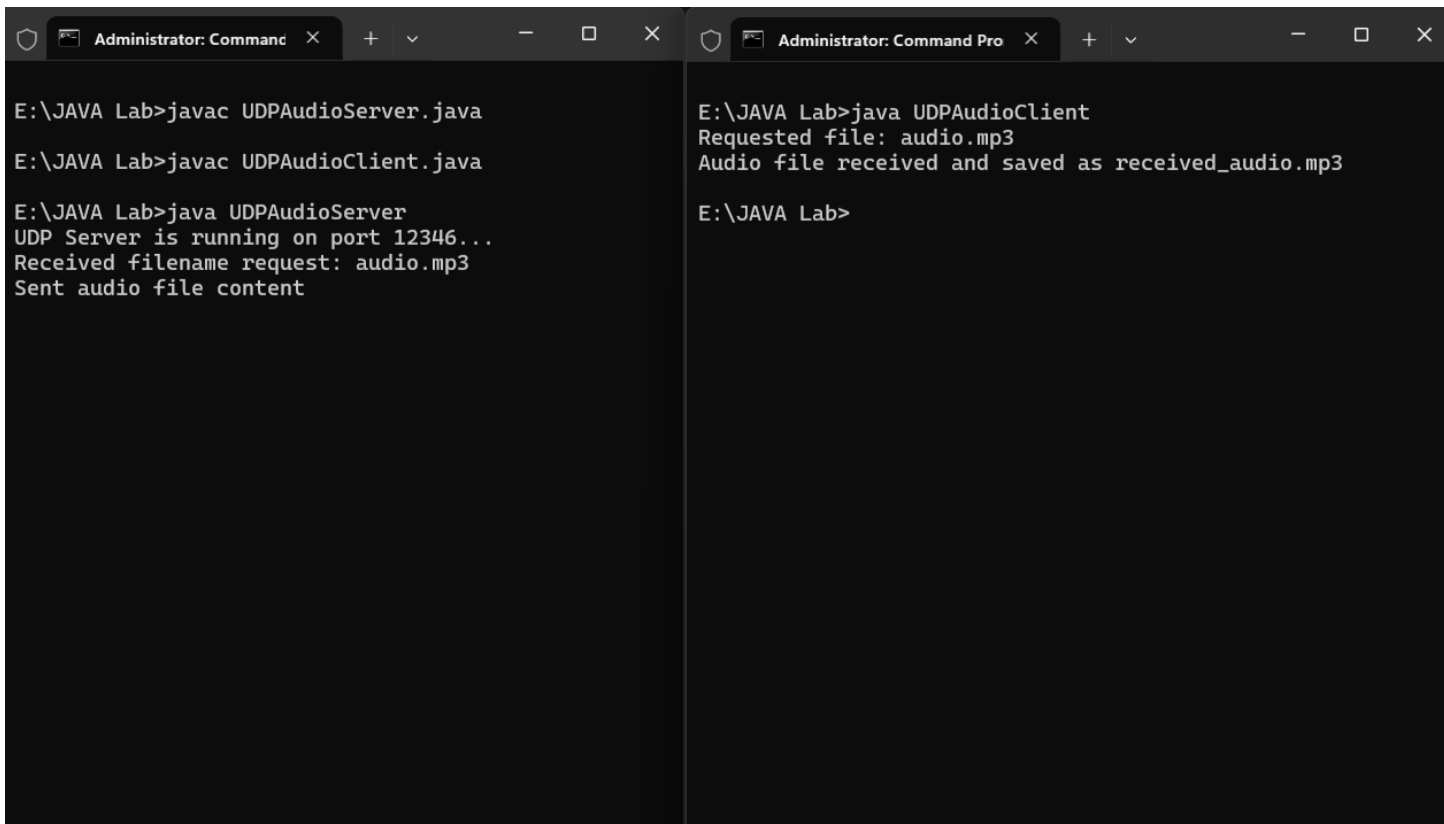
System.out.println("Requested file: " + filename);


// Receive file content
byte[] receiveData = new byte[65000]; // Larger buffer for audio data
DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
clientSocket.receive(receivePacket);


// Save received data to file
String outputFile = "received_audio.mp3";
FileOutputStream fos = new FileOutputStream(outputFile);
fos.write(receivePacket.getData(), 0, receivePacket.getLength());
fos.close();


System.out.println("Audio file received and saved as " + outputFile);
clientSocket.close();


} catch (IOException e) {
    System.out.println("Client error: " + e.getMessage());
}
}
}
```



The image shows two side-by-side Windows Command Prompt windows. The left window, titled 'Administrator: Command Prompt', shows the compilation and execution of a UDP audio server. The right window, titled 'Administrator: Command Prompt', shows the execution of a UDP audio client that requests a file and receives it.

```
E:\JAVA Lab>javac UDPAudioServer.java
E:\JAVA Lab>javac UDPAudioClient.java
E:\JAVA Lab>java UDPAudioServer
UDP Server is running on port 12346...
Received filename request: audio.mp3
Sent audio file content

E:\JAVA Lab>java UDPAudioClient
Requested file: audio.mp3
Audio file received and saved as received_audio.mp3
E:\JAVA Lab>
```



## Practical 12 : GUI

1. Write a programme to implement an investement value calculator using the data inputed by user. textFields to be included are amount, year, interest rate and future value. The field “future value” (shown in gray) must not be altered by user.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class InvestmentCalculator extends JFrame {
    private JTextField amountField, yearField, interestField, futureValueField;

    public InvestmentCalculator() {
        setTitle("Investment Calculator");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new GridLayout(5, 2, 10, 10));
        setSize(300, 200);

        add(new JLabel("Amount:"));
        amountField = new JTextField(10);
        add(amountField);

        add(new JLabel("Year:"));
        yearField = new JTextField(10);
        add(yearField);

        add(new JLabel("Interest Rate:"));
```

```
        interestField = new JTextField(10);

        add(interestField);


        add(new JLabel("Future Value:"));
        futureValueField = new JTextField(10);
        futureValueField.setEditable(false);
        futureValueField.setBackground(Color.LIGHT_GRAY);
        add(futureValueField);


        JButton calculateButton = new JButton("Calculate");
        calculateButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                calculateFutureValue();
            }
        });
        add(calculateButton);

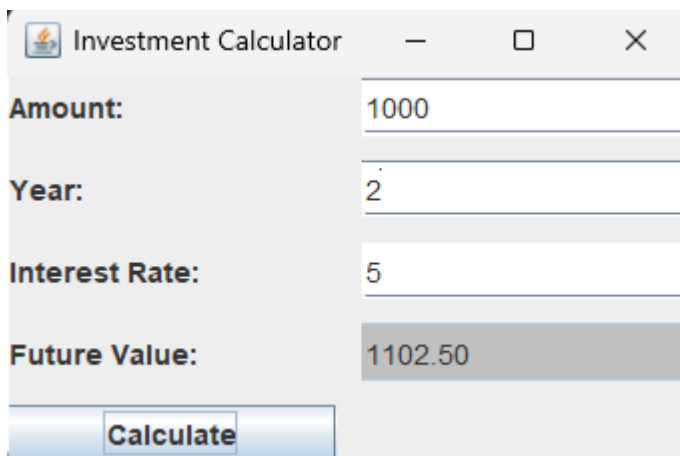

        setLocationRelativeTo(null);
    }


    private void calculateFutureValue() {
        try {
            double amount = Double.parseDouble(amountField.getText());
            int year = Integer.parseInt(yearField.getText());
            double interestRate = Double.parseDouble(interestField.getText()) / 100;

            double futureValue = amount * Math.pow(1 + interestRate, year);
            futureValueField.setText(String.format("%.2f", futureValue));
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "Please enter valid numbers!");
        }
    }
}
```

```
    } catch (Exception ex) {  
        JOptionPane.showMessageDialog(this, "An error occurred!");  
    }  
}  
  
public static void main(String[] args) {  
    SwingUtilities.invokeLater(() -> {  
        InvestmentCalculator calc = new InvestmentCalculator();  
        calc.setVisible(true);  
    });  
}
```

```
E:\JAVA Lab>javac InvestmentCalculator.java  
E:\JAVA Lab>java InvestmentCalculator
```



The screenshot shows a Java Swing window titled "Investment Calculator". It contains four input fields and one output field, all with a light gray background. The "Amount:" field contains the value "1000". The "Year:" field contains the value "2". The "Interest Rate:" field contains the value "5". The "Future Value:" field contains the value "1102.50". Below these fields is a blue button with the text "Calculate".

Field	Value
Amount:	1000
Year:	2
Interest Rate:	5
Future Value:	1102.50

## 2. Write a program which fill the rectangle with the selected color when button pressed.

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

public class ColorRectangle extends JFrame {

    private JPanel colorPanel;

    public ColorRectangle() {

        setTitle("I am a JFrame");

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setLayout(new BorderLayout(10, 10));

        setSize(300, 200);

        JPanel buttonPanel = new JPanel();

        buttonPanel.setLayout(new GridLayout(3, 1, 5, 5));

        JButton redButton = new JButton("Red");

        JButton blueButton = new JButton("Blue");

        JButton greenButton = new JButton("Green");

        redButton.addActionListener(e -> changeColor(Color.RED));

        blueButton.addActionListener(e -> changeColor(Color.BLUE));

        greenButton.addActionListener(e -> changeColor(Color.GREEN));

        buttonPanel.add(redButton);

        buttonPanel.add(blueButton);

        buttonPanel.add(greenButton);
```

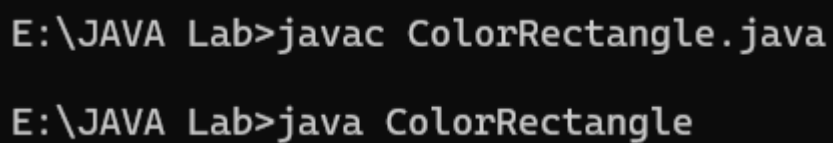
```
        colorPanel = new JPanel();
        colorPanel.setBackground(Color.BLUE);
        colorPanel.setPreferredSize(new Dimension(150, 150));

        add(buttonPanel, BorderLayout.WEST);
        add(colorPanel, BorderLayout.CENTER);

        setLocationRelativeTo(null);
    }

    private void changeColor(Color color) {
        colorPanel.setBackground(color);
        colorPanel.repaint();
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            ColorRectangle frame = new ColorRectangle();
            frame.setVisible(true);
        });
    }
}
```



```
E:\JAVA Lab>javac ColorRectangle.java
E:\JAVA Lab>java ColorRectangle
```

