

**VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES -**

**TECHNICAL CAMPUS**

**Grade A++ Accredited Institution by NAAC**

NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;

Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE An ISO 9001:2015 Certified Institution

**SCHOOL OF ENGINEERING & TECHNOLOGY****Department of Computer Science:**

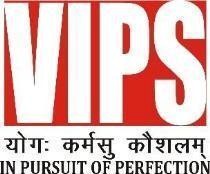
**Course Title: Advance Java Programming**

# Course Code: CIE-306P

**Submitted To: Submitted By:**

**Dr. Nihar Ranjan Roy Name: Ishaan Jain**

**HOD, Associate Professor Enrollment No: 06117702722**



**VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS**

**Grade A++ Accredited Institution by NAAC**

NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;

Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE An ISO 9001:2015 Certified Institution

### SCHOOL OF ENGINEERING & TECHNOLOGY

VISION OF INSTITUTE

To be an educational institution that empowers the field of engineering to build a sustainable future by providing quality education with innovative practices that support people, planet and profit.

MISSION OF THE INSTITUTE

To groom the future engineers by providing value based education and awakening students’ curiosity nurturing creativity and building capabilities to enable them to make significant contribution to the world.

## Experiment-1

**Aim : Write a Java program to print numbers from 1 to n. For multiple of 3, print “Fizz” instead of the number, and for multiple of 5, print “Buzz”. For numbers that are the multiples of both 3 and 5 print “FizzBuzz”**

**Code :**

import java.util.Scanner;

public class exp1 {

    public static void main(String[] args){

        Scanner S = new Scanner(System.in);

        System.out.print("Enter a number: ");

        int num = S.nextInt();

        for(int i = 1; i<=num;i++){

            if (i%3==0 && i%5==0){

            System.out.println("fizzbuzz");

            }

            else if (i%3 == 0){

                System.out.println("fizz");

            }

            else if (i%5==0){

                System.out.println("buzz");

            }

            else{

                System.out.println(i);

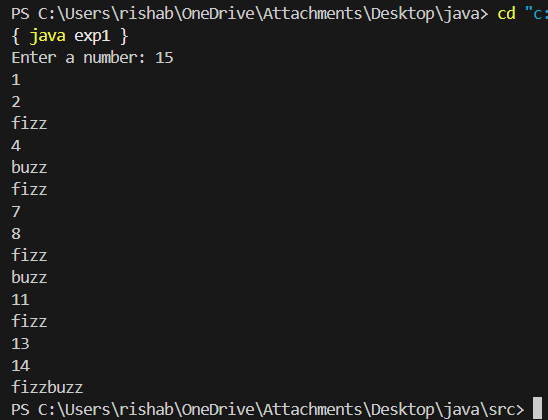
            }

        }

    }

}

**Output:**



**Learning Outcome:**

## Experiment-2

**Aim :** Create an abstract class BankAccount with the following:

1. An accountNumber (String) and balance (double) as instance variables.
2. A constructor to initialize the account number and balance.
3. Abstract methods:
   * deposit(double amount)
   * withdraw(double amount)
4. Create two subclasses:

SavingsAccount:

* + Has an additional variable interestRate (double).
  + Overrides the deposit method to add interest to the balance. • Withdrawals are allowed only if the balance remains above a certain minimum (e.g., 500).

CurrentAccount:

* + Has an additional variable overdraftLimit (double).
  + Overrides the withdraw method to allow overdraft up to the specified limit.

Write a program that:

1. Creates objects of both subclasses.
2. Performs deposit and withdrawal operations.
3. Displays the final account details for each object.

**Code:**

class exp2 {

    abstract class BankAccount {

        protected String AccountNumber;

        protected double balance;

        public BankAccount(String AccountNumber, double balance) {

            this.AccountNumber = AccountNumber;

            this.balance = balance;

        }

        public abstract void deposit(double amount);

        public abstract void withdraw(double amount);

        public void display() {

            System.out.println("Account Number: " + AccountNumber);

            System.out.println("Balance: Rs " + balance);

        }

    }

    class SavingsAccount extends BankAccount {

        private double interestRate;

        public SavingsAccount(String AccountNumber, double balance, double interestRate) {

            super(AccountNumber, balance);

            this.interestRate = interestRate;

        }

        @Override

        public void deposit(double amount) {

            balance += amount + (amount \* interestRate / 100);

        }

        @Override

        public void withdraw(double amount) {

            if (balance - amount >= 500) {

                balance -= amount;

            } else {

                System.out.println("Withdraw denied. Minimum balance must be 500");

            }

        }

    }

    class CurrentAccount extends BankAccount {

        private double overdraftLimit;

        public CurrentAccount(String AccountNumber, double balance, double overdraftLimit) {

            super(AccountNumber, balance);

            this.overdraftLimit = overdraftLimit;

        }

        @Override

        public void deposit(double amount) {

            balance += amount;

        }

        @Override

        public void withdraw(double amount) {

            if (balance - amount >= -overdraftLimit) {

                balance -= amount;

            } else {

                System.out.println("Withdraw denied. Overdraft limit exceeded");

            }

        }

    }

    public static void main(String[] args) {

        exp2 expInstance = new exp2();

        SavingsAccount savings = expInstance.new SavingsAccount("SA123", 1000, 5);

        CurrentAccount current = expInstance.new CurrentAccount("CA123", 2000, 1000);

        savings.deposit(500);

        savings.withdraw(200);

        savings.display();

        current.deposit(1000);

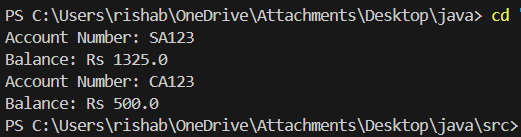
        current.withdraw(2500);

        current.display();

    }

}

**Output:**

****

**Learning Outcome:**

## Experiment-3

**Aim :**

**Write an efficient code in java to check all the prime numbers in a given range of numbers.**

**Code :**

import java.util.Scanner;

public class exp3 {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        System.out.println("Enter the starting range: ");

        int start = scan.nextInt();

        System.out.println("Enter the ending range: ");

        int end = scan.nextInt();

        scan.close();

        for (int i = start; i <= end; i++) {

            if (isPrime(i)) {

                System.out.print(i + " ");

            }

        }

    }

    public static boolean isPrime(int num) {

        if (num <= 1) {

            return false;

        }

        for (int i = 2; i <= Math.sqrt(num); i++) {

            if (num % i == 0) {

                return false;

            }

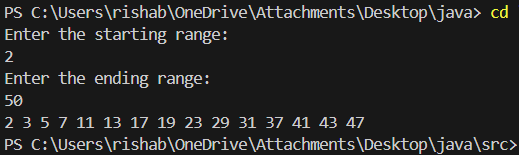
        }

        return true;

    }

}

**Output:**

****

**Learning Outcome:**

## Experiment-4

**Aim:**

**N soldiers (or people) stand in a circle. The king gives a sword to the first soldier, who kills the person to their left and passes the sword to the next surviving person. This process continues until only one person remains. Write a java program to find the safe position to stand. Assuming first position is=1.**

**Code :**

import java.util.Scanner;

public class exp4 {

    public static int findSafePosition(int n) {

        if (n == 1) {

            return 1;

        } else {

            return (findSafePosition(n - 1) + 1) % n + 1;

        }

    }

    public static void main(String[] args) {

        System.out.print("Enter the number of people: ");

        Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

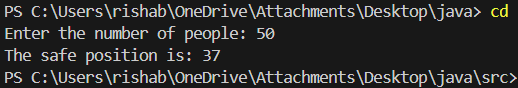
        System.out.println("The safe position is: " + findSafePosition(n));

        scan.close();

    }

}

**Output:**

****

**Learning Outcome:**

**EXPERIMENT 5**

**AIM:**

**Write a Java program using multithreading where:**

**One thread calculates the sum of all even numbers.**

**Another thread calculates the sum of all odd numbers.**

**A file is created that stores randomly generated numbers within a specified range.**

**CODE:**

import java.io.\*;

import java.util.Random;

class NumberGenerator {

public static final String FILE\_NAME = "numbers.txt";

public static final int RANGE\_START = 1;

public static final int RANGE\_END = 100;

public static final int COUNT = 50;

public static void generateNumbersToFile() {

Random random = new Random();

try (BufferedWriter writer = new BufferedWriter(new FileWriter(numbers.txt))) {

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

int num = random.nextInt(RANGE\_END - RANGE\_START + 1) + RANGE\_START;

writer.write(num + "\n");

}

System.out.println("Random numbers generated in file: " + numbers.txt);

} catch (IOException e) {

e.printStackTrace();

}

}

public static int[] readNumbersFromFile() {

int[] numbers = new int[COUNT];

try (BufferedReader reader = new BufferedReader(new FileReader(numbers.txt))) {

String line;

int index = 0;

while ((line = reader.readLine()) != null && index < COUNT) {

numbers[index++] = Integer.parseInt(line.trim());

}

} catch (IOException e) {

e.printStackTrace();

}

return numbers;

}

}

class EvenSumThread extends Thread {

public int[] numbers;

public int evenSum = 0;

public EvenSumThread(int[] numbers) {

this.numbers = numbers;

}

@Override

public void run() {

for (int num : numbers) {

if (num % 2 == 0) {

evenSum += num;

}

}

System.out.println("Sum of even numbers: " + evenSum);

}

}

class OddSumThread extends Thread {

public int[] numbers;

public int oddSum = 0;

public OddSumThread(int[] numbers) {

this.numbers = numbers;

}

@Override

public void run() {

for (int num : numbers) {

if (num % 2 != 0) {

oddSum += num;

}

}

System.out.println("Sum of odd numbers: " + oddSum);

}

}

public class MultiThreadingExample {

public static void main(String[] args) {

NumberGenerator.generateNumbersToFile();

int[] numbers = NumberGenerator.readNumbersFromFile();

EvenSumThread evenThread = new EvenSumThread(numbers);

OddSumThread oddThread = new OddSumThread(numbers);

evenThread.start();

oddThread.start();

try {

evenThread.join();

oddThread.join();

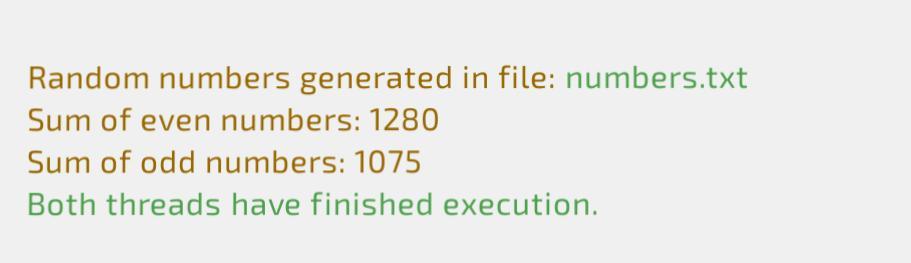
} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Both threads have finished execution.");

}}

**OUTPUT:**

**LEARNING OUTCOMES:**

**EXPERIMENT 6**

**AIM:**

**Write a java program to: -**

1. **Fetch all the records from the employee table.**
2. **Search with employee id.**
3. **Update specific employee.**
4. **Delete a particular record.**

**CODE:**

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

import java.sql.ResultSet;

import java.sql.PreparedStatement;

import java.util.Scanner;

class exp6 {

public static void main(String args[]) throws SQLException, ClassNotFoundException {

String url = "jdbc:mysql://localhost:3306/ishaan";

String username = "root";

String password = "";

Scanner sc = new Scanner(System.in);

Class.forName("com.mysql.cj.jdbc.Driver");

Connection conn = DriverManager.getConnection(url, username, password);

System.out.println("Database Connected Successfully");

Statement stm = conn.createStatement();

while (true) {

System.out.println("\nMenu:");

System.out.println("1. Fetch all records");

System.out.println("2. Search by id");

System.out.println("3. Update specific employee");

System.out.println("4. Delete a particular record");

System.out.println("5. Exit");

System.out.print("Enter your choice: ");

int choice = sc.nextInt();

sc.nextLine(); // Consume newline

switch (choice) {

case 1:

fetchAllRecords(stm);

break;

case 2:

searchById(conn, sc);

break;

case 3:

updateEmployee(conn, sc);

break;

case 4:

deleteEmployee(conn, sc);

break;

case 5:

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

conn.close();

sc.close();

return;

default:

System.out.println("Invalid choice. Please try again.");

}

}

}

}

public static void fetchAllRecords(Statement stm) throws SQLException {

String query = "SELECT \* FROM employee";

ResultSet rs = stm.executeQuery(query);

while (rs.next()) {

System.out.println(rs.getInt("id") + " | " + rs.getString("name") + " | " + rs.getString("dep"));

}

}

public static void searchById(Connection conn, Scanner sc) throws SQLException {

System.out.print("Enter id to search: ");

int id = sc.nextInt();

sc.nextLine();

String query = "SELECT \* FROM employee WHERE id = ?";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setInt(1, id);

ResultSet rs = pstmt.executeQuery();

if (rs.next()) {

System.out.println(rs.getInt("id") + " | " + rs.getString("name") + " | " + rs.getString("dep"));

} else {

System.out.println("Record not found.");

}

}

public static void updateEmployee(Connection conn, Scanner sc) throws SQLException {

System.out.print("Enter id to update: ");

int id = sc.nextInt();

sc.nextLine();

System.out.print("Enter new name: ");

String name = sc.nextLine();

System.out.print("Enter new department: ");

String department = sc.nextLine();

String query = "UPDATE employee SET name = ?, dep = ? WHERE id = ?";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setString(1, name);

pstmt.setString(2, department);

pstmt.setInt(3, id);

int rowsUpdated = pstmt.executeUpdate();

if (rowsUpdated > 0) {

System.out.println("Record updated successfully.");

} else {

System.out.println("Record not found.");

}

}

public static void deleteEmployee(Connection conn, Scanner sc) throws SQLException {

System.out.print("Enter id to delete: ");

int id = sc.nextInt();

sc.nextLine();

String query = "DELETE FROM employee WHERE id = ?";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setInt(1, id);

int rowsDeleted = pstmt.executeUpdate();

if (rowsDeleted > 0) {

System.out.println("Record deleted successfully.");

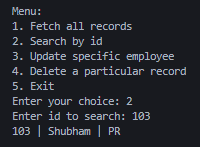
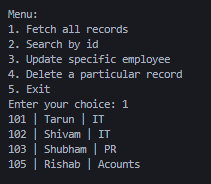
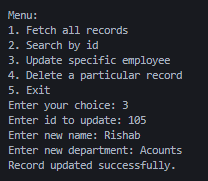
} else {

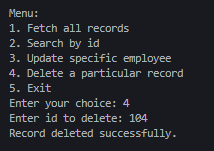
System.out.println("Record not found.");

}

}

**Output:**





**Learning Outcome:**

**EXPERIMENT 7**

**PROGRAM 1**

**AIM:**

**Write a client server based program in java where a client sends 2 numbers, server adds and returns sum to the client.**

**CODE:**

**CLIENT-SIDE CODE**

import java.io.\*;

import java.net.\*;

public class AdditionClient {

    public static void main(String[] args) {

        try {

            // Connect to server on localhost, port 5000

            Socket socket = new Socket("localhost", 5000);

            System.out.println("Connected to server!");

            // Input and Output streams

            BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));

            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            System.out.print("Enter first number: "); // Get two numbers from user

            int num1 = Integer.parseInt(userInput.readLine());

            System.out.print("Enter second number: ");

            int num2 = Integer.parseInt(userInput.readLine());

            // Send numbers to server

            out.println(num1);

            out.println(num2);

            int sum = Integer.parseInt(in.readLine()); // Receive result from server

            System.out.println("Sum received from server: " + sum);

            // Close resources

            userInput.close();

            out.close();

            in.close();

            socket.close();

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

**SERVER-SIDE CODE**

import java.io.\*;

import java.net.\*;

public class AdditionServer {

    public static void main(String[] args) {

        try {

            // Create server socket on port 5000

            ServerSocket serverSocket = new ServerSocket(5000);

            System.out.println("Server is waiting for client...");

            Socket socket = serverSocket.accept(); // Accept client connection

            System.out.println("Client connected!");

            // Input and Output streams

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

            int num1 = Integer.parseInt(in.readLine()); // Read two numbers from client

            int num2 = Integer.parseInt(in.readLine());

            int sum = num1 + num2; // Calculate sum

            out.println(sum); // Send result back to client

            // Close resources

            in.close();

            out.close();

            socket.close();

            serverSocket.close();

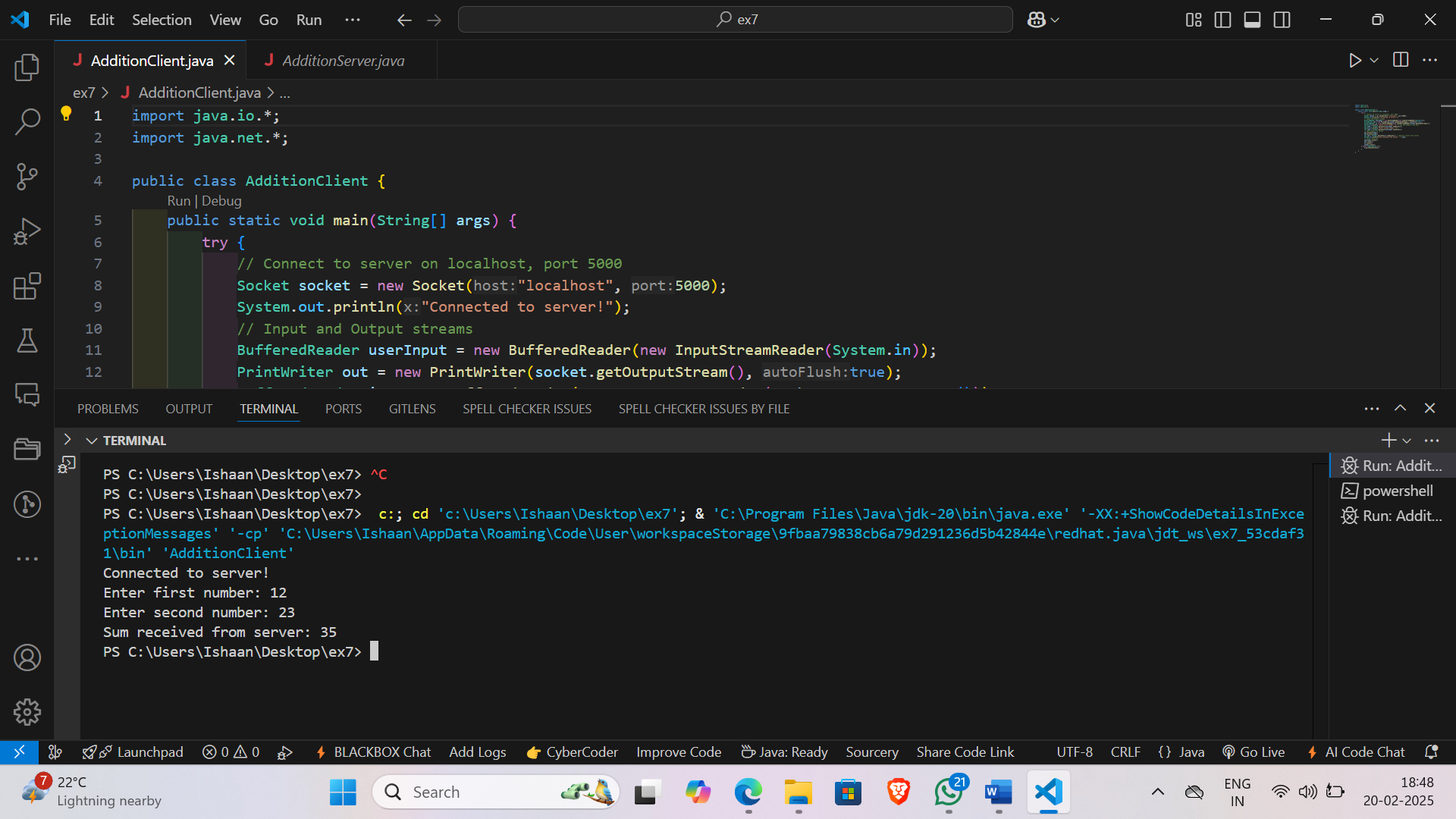
        } catch (IOException e) {

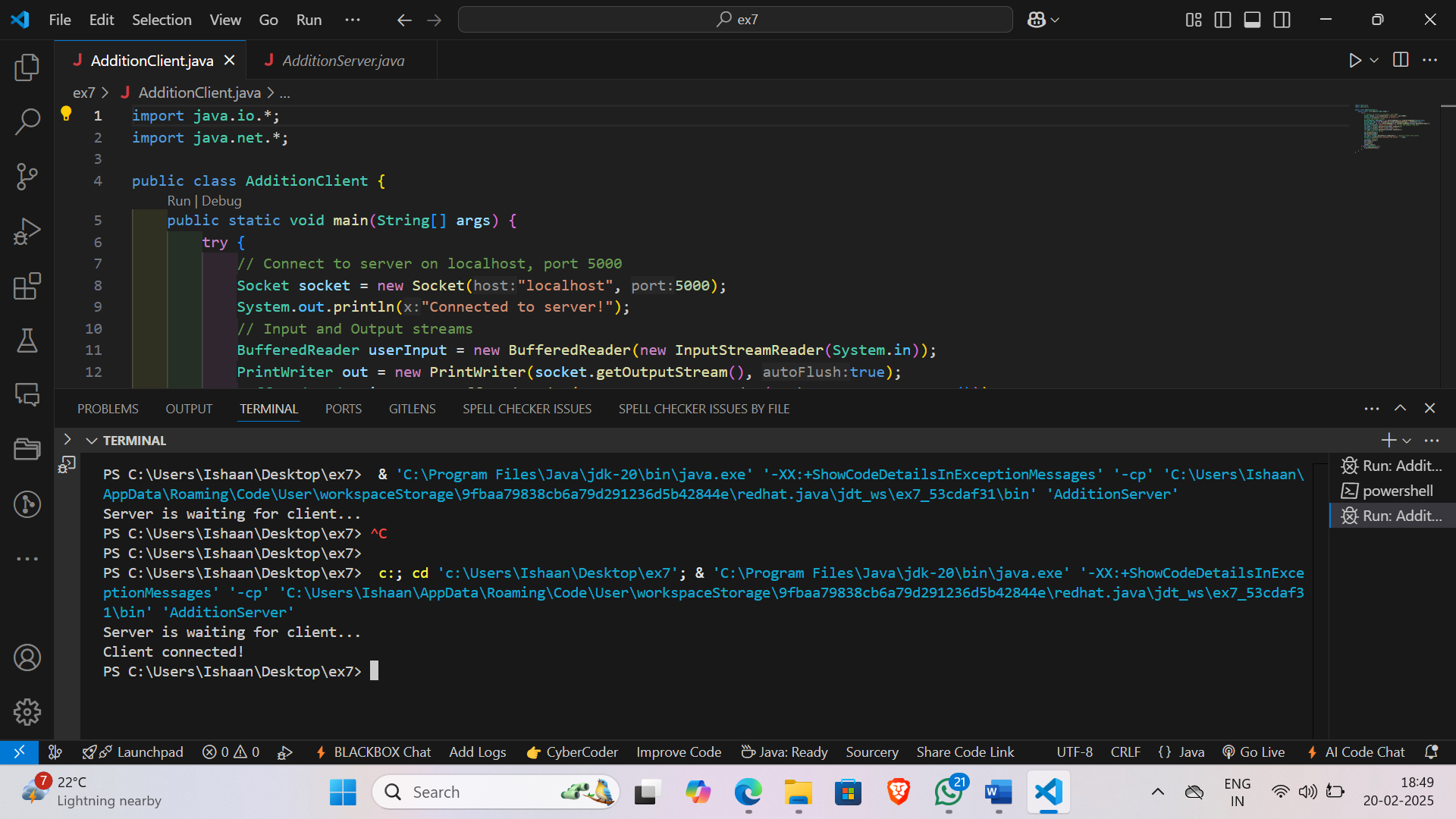
            e.printStackTrace();

        }

    }

}

**Output:**



**Learning Outcome:**

**PROGRAM 2**

**AIM:**

**Write a client server based program in java where client requests server a particular file, if this file exists in current folder of the server then server sends this to the client**

**CODE:**

**CLIENT-SIDE CODE**

import java.io.\*;

import java.net.\*;

public class FileClient {

    public static void main(String[] args) {

        try (Socket socket = new Socket("localhost", 5000);

                BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

                PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

                BufferedInputStream fileIn = new BufferedInputStream(socket.getInputStream())) {

            System.out.println("Connected to server. Requesting file...");

            out.println("example.txt"); // Request a file from the server

            String response = in.readLine();

            if (response.equals("File found. Sending...")) {

                System.out.println("Receiving file...");

                receiveFile(fileIn, "received\_example.txt");

                System.out.println("File received successfully.");

            } else {

                System.out.println("Error: " + response);

            }

        } catch (UnknownHostException e) {

            System.err.println("Unknown host: " + e.getMessage());

        } catch (IOException e) {

            System.err.println("Error connecting to server: " + e.getMessage());

        }

    }

    private static void receiveFile(BufferedInputStream fileIn, String fileName) throws IOException {

        try (FileOutputStream fileOut = new FileOutputStream(fileName)) {

            byte[] buffer = new byte[1024];

            int bytesRead;

            while ((bytesRead = fileIn.read(buffer)) != -1) {

                fileOut.write(buffer, 0, bytesRead);

            }

            fileOut.flush();

        } catch (IOException e) {

            System.err.println("Error receiving file: " + e.getMessage());

            throw e;

        }

    }

}

**SERVER-SIDE CODE**

import java.io.\*;

import java.net.\*;

public class FileServer {

    public static void main(String[] args) {

        try (ServerSocket serverSocket = new ServerSocket(5000)) {

            System.out.println("Server is waiting for client...");

            Socket socket = serverSocket.accept();

            System.out.println("Client connected!");

            handleClientRequest(socket);

        } catch (IOException e) {

            System.err.println("Error starting server: " + e.getMessage());

        }

    }

    private static void handleClientRequest(Socket socket) throws IOException {

        try (BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

                PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

                BufferedOutputStream fileOut = new BufferedOutputStream(socket.getOutputStream())) {

            String fileName = in.readLine();

            System.out.println("Client requested file: " + fileName);

            File file = new File(fileName);

            if (file.exists() && file.isFile()) {

                System.out.println("File exists. Sending to client...");

                sendFile(file, fileOut, out);

            } else {

                System.out.println("File does not exist.");

                out.println("File does not exist.");

            }

        } catch (IOException e) {

            System.err.println("Error handling client request: " + e.getMessage());

        } finally {

            socket.close();

        }

    }

    private static void sendFile(File file, BufferedOutputStream fileOut, PrintWriter out) throws IOException {

        try (FileInputStream fileIn = new FileInputStream(file)) {

            out.println("File found. Sending...");

            byte[] buffer = new byte[1024];

            int bytesRead;

            while ((bytesRead = fileIn.read(buffer)) != -1) {

                fileOut.write(buffer, 0, bytesRead);

            }

            fileOut.flush();

            out.println("File sent successfully.");

        } catch (IOException e) {

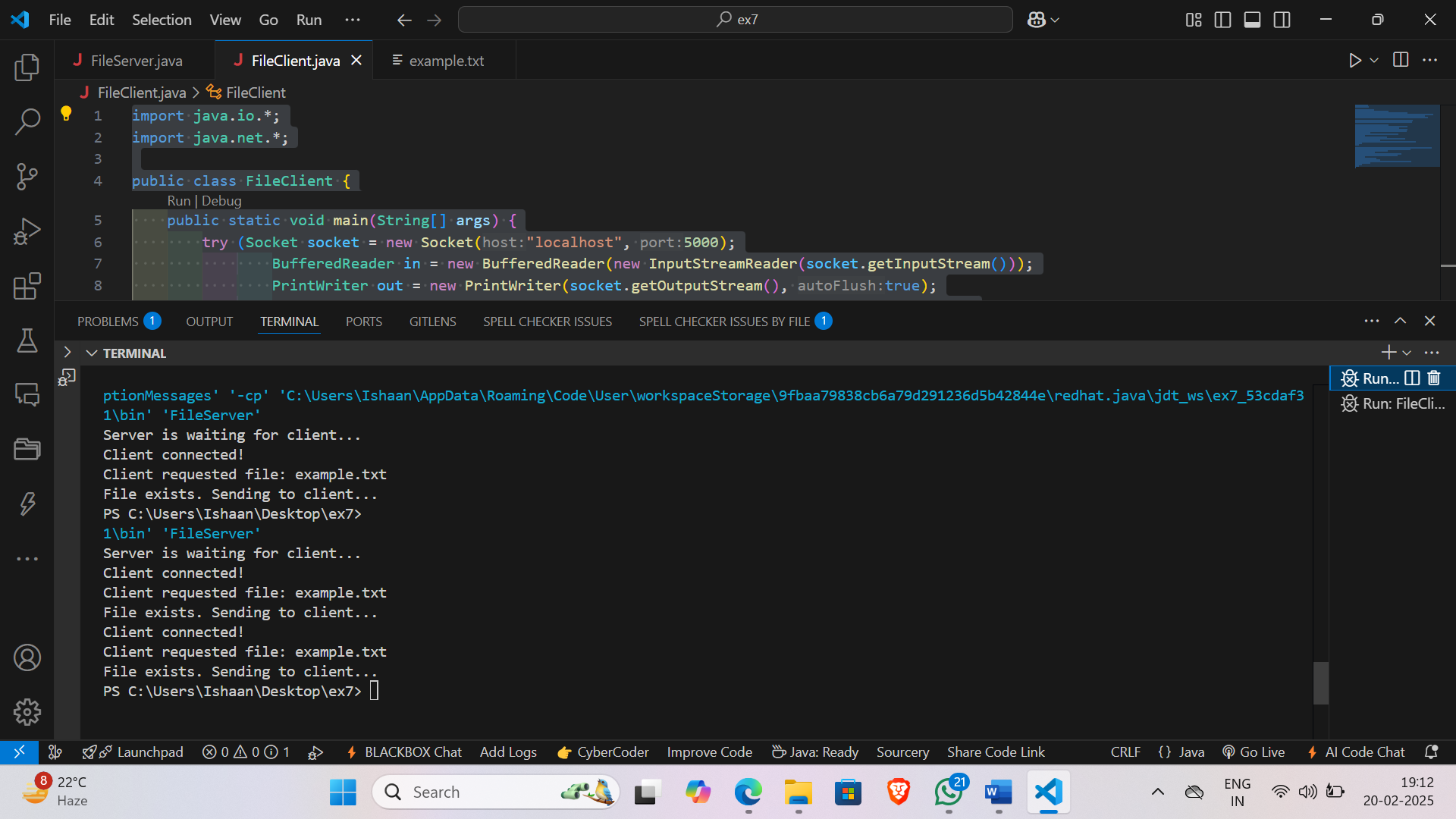
            out.println("Error sending file: " + e.getMessage());

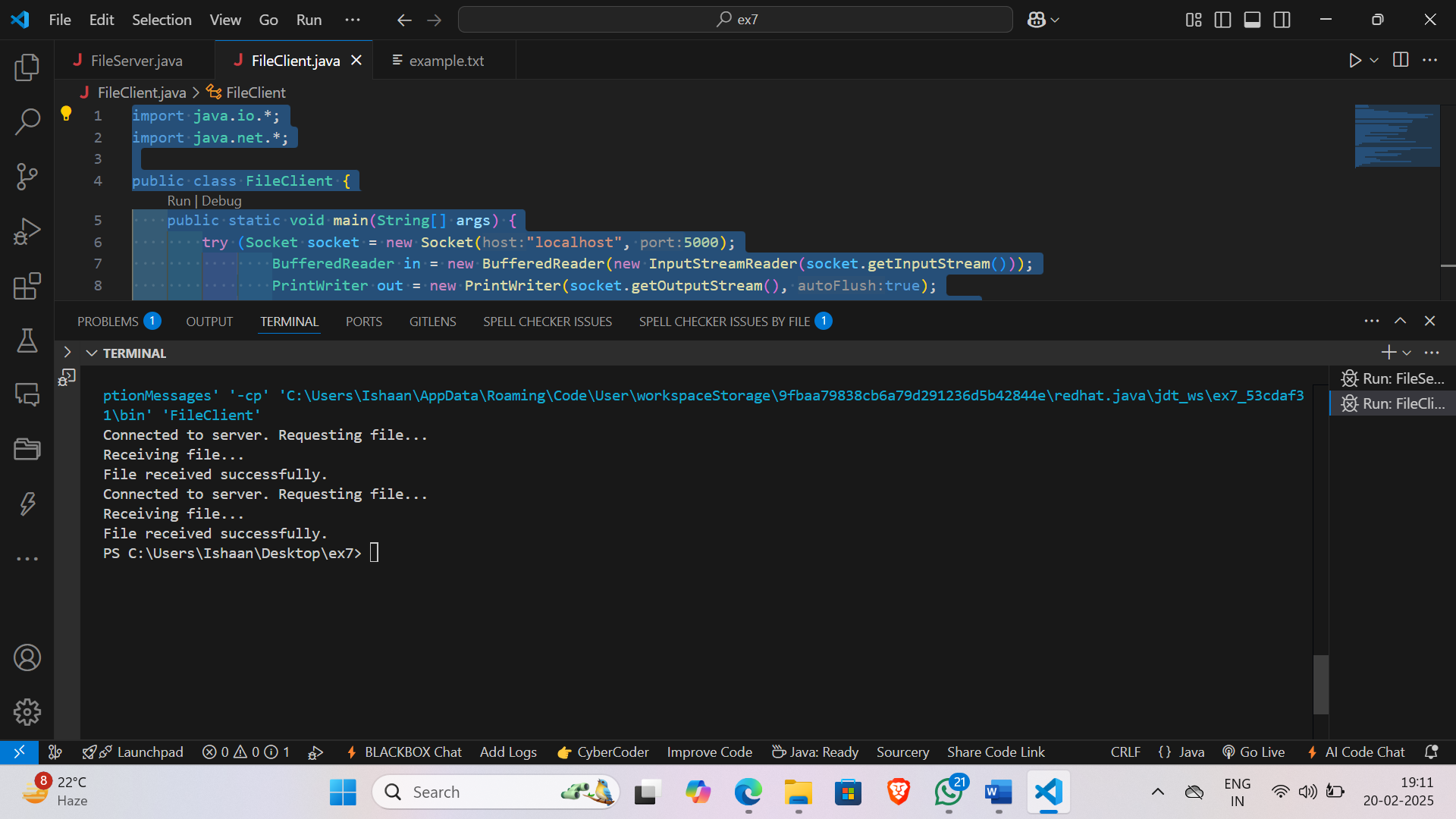
            throw e;

        }

    }

}

**Output:**



**Learning Outcome:**