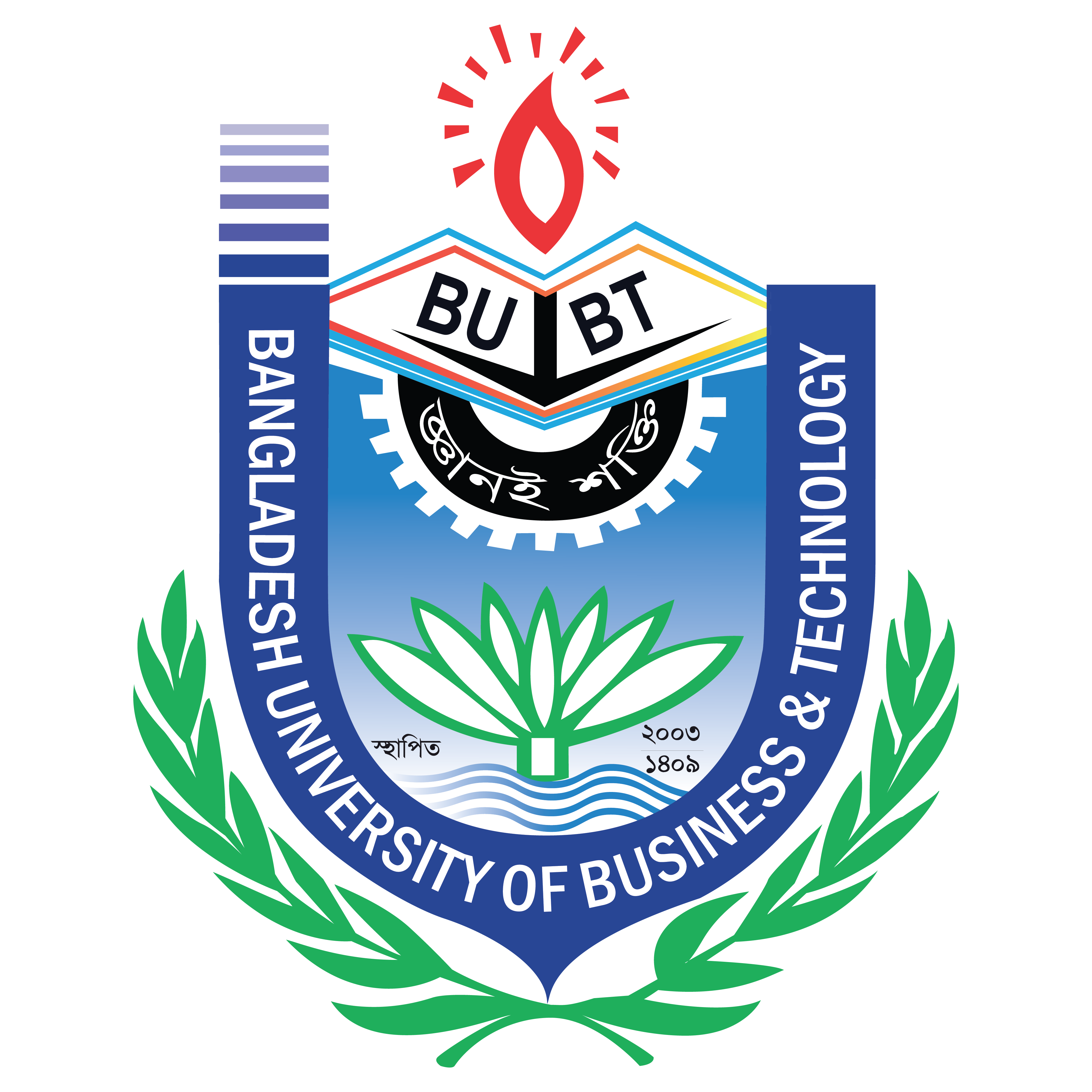
****

**Bangladesh University Of Business and Technology (BUBT)**

**Advanced Programming Lab**

**Course code: CSE 332**

**Final Lab Assignment**

**Submitted by:**

**Name: Anayat Hossain**

**ID: 21225103161**

**Section: 04**

**Intake: 49**

**Github Link:**

**https://github.com/AnayatHossain/Java-Final-Assignment**

**Submitted to:**

**Md. Mahbubur Rahman**

**Assistant Professor, Dept. of CSE, BUBT**

1. Write a Java program that reads data from a file named "data.txt". Implement

error handling using try-catch blocks to handle FileNotFoundException. If the

file is not found, print an error message indicating the issue.

**Code:**

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

public class ReadDataFromFile {

public static void main(String[] args) {

try { File file = new File("text.txt");

Scanner scanner = new Scanner(file);

while (scanner.hasNextLine()) {

String line = scanner.nextLine();

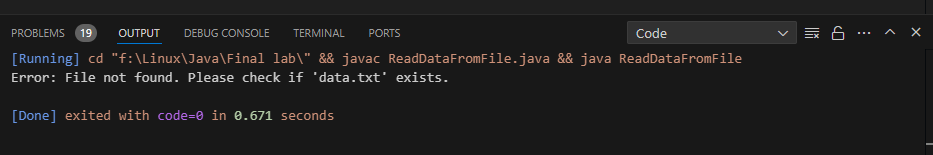
System.out.println(line); }

scanner.close(); }

catch (FileNotFoundException e) {

System.err.println("Error: File not found. Please check if 'text.txt' exists."); } }}

Output:



2. Write a Java program that initializes an array of integers and attempts to  access an element at an index beyond the array's length. Implement try-catch  blocks to handle the ArrayIndexOutOfBoundsException that may occur. If the  exception occurs, print a message indicating the invalid index.

Code:

public class ArrayAccessExample {

    public static void main(String[] args) {

        int[] numbers = { 10, 20, 30, 40, 50 };

        int index = 10;

        try {

            int value = numbers[index];

            System.out.println("Value at index " + index + ": " + value);

        } catch (ArrayIndexOutOfBoundsException e) {

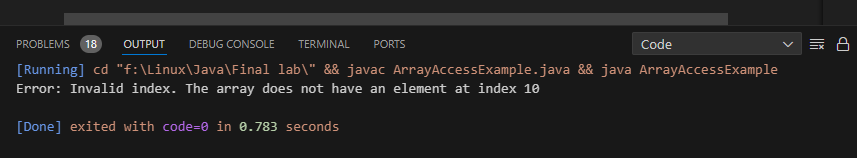
            System.err.println("Error: Invalid index. The array does not have an element at index " + index);

        }

    }

}

Output:



3. Write a Java program to simulate bank account transactions. Implement try catch blocks to handle exceptions that may occur during withdrawal or deposit  operations, such as InsufficientFundsException for insufficient balance and  NegativeAmountException for negative amounts. Use a finally block to  ensure that resources are properly released after each transaction.

Code:

class InsufficientFundsException extends Exception {

    public InsufficientFundsException(String message) {

        super(message);   }}

class NegativeAmountException extends Exception {

    public NegativeAmountException(String message) {

        super(message); }}

class BankAccount {

    private double balance;

    public BankAccount(double initialBalance) {

        this.balance = initialBalance;  }

    public void deposit(double amount) throws NegativeAmountException {

        if (amount < 0) {   throw new NegativeAmountException("Cannot deposit a negative amount."); }

        balance += amount; }

    public void withdraw(double amount) throws InsufficientFundsException, NegativeAmountException {

        if (amount < 0) { throw new NegativeAmountException("Cannot withdraw a negative amount.");  }

      if (balance < amount) { throw new InsufficientFundsException("Insufficient balance for withdrawal.");   }

        balance -= amount;  }

    public double getBalance() {

        return balance; }}

public class BankTransaction {

    public static void main(String[] args) {

        try {

            BankAccount account = new BankAccount(1000.0);

            account.deposit(1500.0);

            account.withdraw(300.0);

            System.out.println("Current balance: BDT " + account.getBalance());

        } catch (NegativeAmountException e) {

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

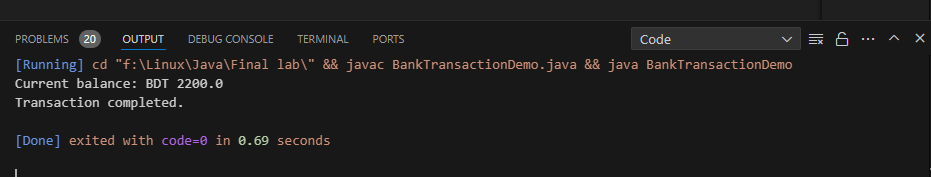
        } catch (InsufficientFundsException e) {

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

        } finally {

            System.out.println("Transaction completed."); }  }}

Output:



4. Imagine you have a bank account. You can deposit and withdraw money from  your account. You should keep in mind that the total amount of money  withdrawn from your account must not exceed the total balance present in  your account. If such a scenario happens, you need to safely execute from the  banking system. Implement the above case in Java with the proper utilization  of user-defined exception mechanism.

Code:

class InsufficientBalanceException extends Exception {

    public InsufficientBalanceException(String message) {

        super(message);  }}

class BankAccount {

    private double balance;

    public BankAccount(double initialBalance) {

        this.balance = initialBalance;  }

    public void deposit(double amount) {

        if (amount > 0) {

            balance += amount;

            System.out.println("Deposited $" + amount + ". New balance: $" + balance);

        } else {

            System.err.println("Error: Cannot deposit a negative or zero amount."); }  }

    public void withdraw(double amount) throws InsufficientBalanceException {

        if (amount <= 0) {

            System.err.println("Error: Withdrawal amount must be positive.");

            return;  }

        if (balance >= amount) {

            balance -= amount;

            System.out.println("Withdrawn $" + amount + ". New balance: $" + balance);

        } else {

            throw new InsufficientBalanceException("Insufficient balance for withdrawal.");   } }

    public double getBalance() {

        return balance;   } }

public class BankTransactionImagine {

    public static void main(String[] args) {

        try {

            BankAccount account = new BankAccount(1000.0);

            account.deposit(500.0);

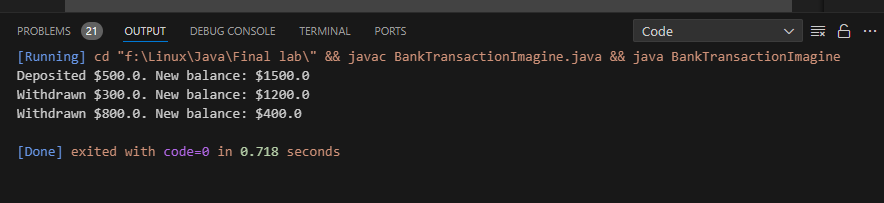
            account.withdraw(300.0);

            account.withdraw(800.0);

        } catch (InsufficientBalanceException e) {

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

Output:



5. Write a Java program to validate an email address entered by the user.  Implement multiple catch blocks to handle different types of exceptions that  may occur during validation, such as IllegalArgumentException for invalid  format and NullPointerException for null input. Use a finally block to close  any resources opened during validation.

Code:

import java.util.Scanner;

public class EmailValidator {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        try {

            System.out.print("Enter an email address: ");

            String email = scanner.nextLine();

            validateEmail(email);

            System.out.println("Valid email address: " + email);

        } catch (IllegalArgumentException e) {

            System.err.println("Error: Invalid email format. Please enter a valid email address.");

        } catch (NullPointerException e) {

            System.err.println("Error: Email address cannot be null.");

        } finally {

            scanner.close();  }  }

    private static void validateEmail(String email) {

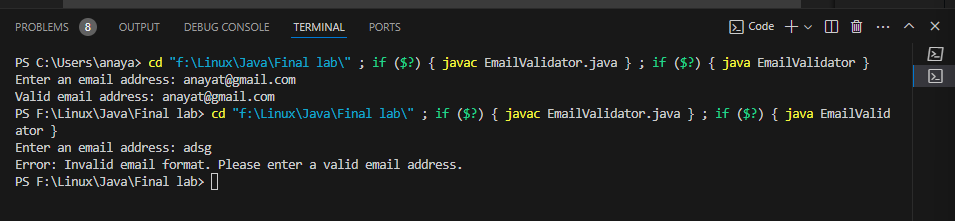
        if (email == null) {

            throw new NullPointerException();  }

        if (!email.matches("[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}")) {

            throw new IllegalArgumentException(); } }}

Output:



6. Write a program to create four threads. Inside the first thread print your Dept.  10 times but wait for 2 second before printing each time. Inside the second  thread print your Name 20 times. Inside the third thread print your ID 30  times. Make sure second thread gets more OS access than the first thread and  the third thread starts after finishing the second thread.

Code:

class PrintThread extends Thread {

    private final String message;

    private final int repetitions;

    private final long delayMillis;

    public PrintThread(String message, int repetitions, long delayMillis) {

        this.message = message;

        this.repetitions = repetitions;

        this.delayMillis = delayMillis;  }

    @Override

    public void run() {

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

            System.out.println(message);

            try {

                Thread.sleep(delayMillis);

            } catch (InterruptedException e) {

                Thread.currentThread().interrupt(); } } }}

public class ThreadDemo {

    public static void main(String[] args) {

        PrintThread deptThread = new PrintThread("Dept: CSE", 10, 2000);

        PrintThread nameThread = new PrintThread("Name: Anayat Hossain", 20, 1000);

        PrintThread idThread = new PrintThread("ID: 161", 30, 800);

        nameThread.setPriority(Thread.MAX\_PRIORITY);

        deptThread.setPriority(Thread.MIN\_PRIORITY);

        deptThread.start();

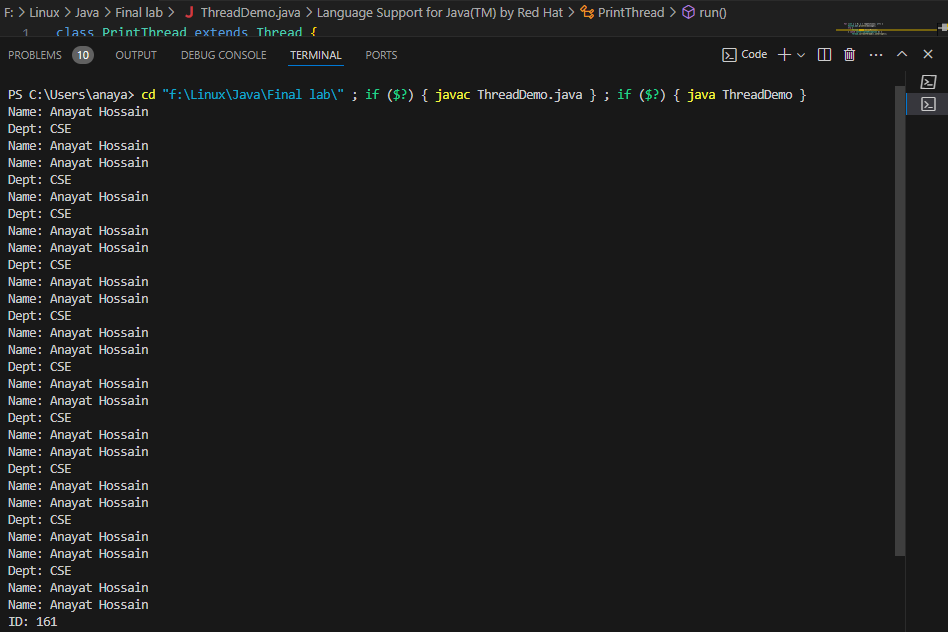
        nameThread.start();

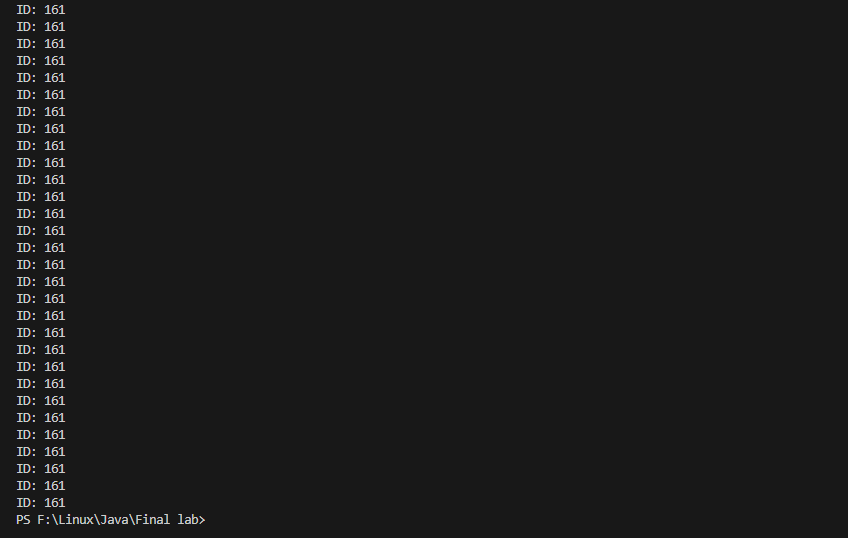
        try { nameThread.join();

        } catch (InterruptedException e) {

            Thread.currentThread().interrupt();  }

        idThread.start(); }}

Output:



7. Write a Java program to perform matrix multiplication using multithreading  for parallel computation. Implement a method that takes two matrices as input  and computes their product using multiple threads, each responsible for  computing a portion of the result matrix. Ensure efficient utilization of  resources and minimize thread synchronization overhead.

Code:

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.TimeUnit;

class MatrixMultiplier {

    private final int[][] resultMatrix;

    private final int[][] matrixA;

    private final int[][] matrixB;

    private final int numRowsA;

    private final int numColsA;

    private final int numColsB;

    public MatrixMultiplier(int[][] matrixA, int[][] matrixB) {

        this.matrixA = matrixA;

        this.matrixB = matrixB;

        this.numRowsA = matrixA.length;

        this.numColsA = matrixA[0].length;

        this.numColsB = matrixB[0].length;

        this.resultMatrix = new int[numRowsA][numColsB];  }

    public int[][] multiply() {

        ExecutorService executor = Executors.newFixedThreadPool(numRowsA);

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

            final int row = i;

            executor.submit(() -> computeRow(row));

        } executor.shutdown();

        try {

            executor.awaitTermination(Long.MAX\_VALUE, TimeUnit.NANOSECONDS);

        } catch (InterruptedException e) {

            Thread.currentThread().interrupt();}

        return resultMatrix; }

    private void computeRow(int row) {

        for (int col = 0; col < numColsB; col++) {

            int sum = 0;

            for (int k = 0; k < numColsA; k++) {

                sum += matrixA[row][k] \* matrixB[k][col];    }

            resultMatrix[row][col] = sum }}}

public class MatrixMultiplication {

    public static void main(String[] args) {

        int[][] matrixA = {

            {1, 2, 3}, {4, 5, 6}  };

        int[][] matrixB = {

            {7, 8}, {9, 10},{11, 12}};

        MatrixMultiplier multiplier = new MatrixMultiplier(matrixA, matrixB);

        int[][] result = multiplier.multiply();

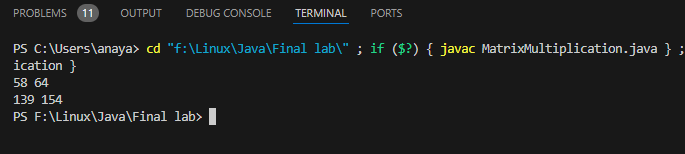
        for (int[] row : result) {

            for (int value : row) {

                System.out.print(value + " ");  }

            System.out.println(); } }}

Output:



8. Write a Java program to compute the factorial of a given number using  multithreading. Create two threads, one for computing the factorial of even  numbers and the other for computing the factorial of odd numbers. Combine  the results to get the final factorial.

Code:

class FactorialCalculator extends Thread {

    private final int start;

    private final int end;

    private long result = 1;

    public FactorialCalculator(int start, int end) {

        this.start = start;

        this.end = end;  }

    @Override

    public void run() {

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

            result \*= i;  }  }

    public long getResult() {

        return result;  }}

public class FactorialMultithreading {

    public static void main(String[] args) {

        int n = 5; // Change this to the desired number

        FactorialCalculator evenThread = new FactorialCalculator(2, n);

        FactorialCalculator oddThread = new FactorialCalculator(1, n);

        evenThread.start();

        oddThread.start();

        try {

            evenThread.join();

            oddThread.join();

            long finalFactorial = evenThread.getResult() \* oddThread.getResult();

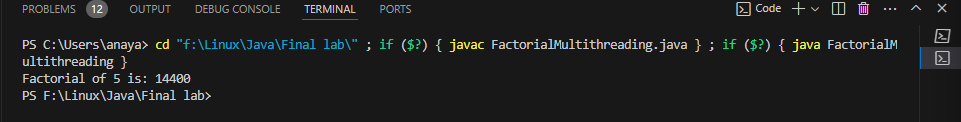
            System.out.println("Factorial of " + n + " is: " + finalFactorial);

        } catch (InterruptedException e) {

            Thread.currentThread().interrupt();

        } }}

Output:



9. Write a Java program that creates two threads, one for printing uppercase  letters from A to Z and the other for printing lowercase letters from a to z.  Ensure that the letters are printed in sequence, with uppercase letters followed  by lowercase letters.

Code:

class UppercaseThread extends Thread {

    @Override

    public void run() {

        System.out.print("Uppercase Thread: ");

        for (char c = 'A'; c <= 'Z'; c++) {

            System.out.print(c + " ");   }

        System.out.println(); }}

class LowercaseThread extends Thread {

    @Override

    public void run() {

        System.out.print("Lowercase Thread: ");

        for (char c = 'a'; c <= 'z'; c++) {

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

        }   }}

public class LetterPrinting {

    public static void main(String[] args) {

        UppercaseThread uppercaseThread = new UppercaseThread();

        LowercaseThread lowercaseThread = new LowercaseThread();

        uppercaseThread.start();

        try {

            uppercaseThread.join();

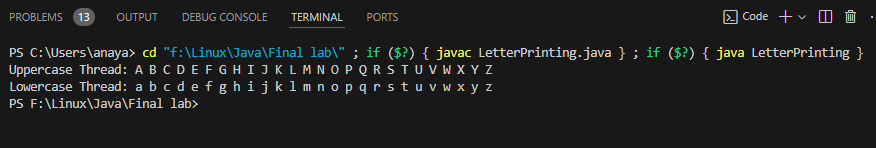
        } catch (InterruptedException e) {

            Thread.currentThread().interrupt();

        }

        lowercaseThread.start(); }}

Output:



10.Write a Java program that calculates the sum of all numbers from 1 to 100  using multiple threads. Divide the range of numbers into equal segments and  assign each thread to compute the sum of a segment. Then, combine the results  from all threads to get the final sum.

Code:

class SumCalculatorThread extends Thread {

    private final int start;

    private final int end;

    private long partialSum = 0;

    public SumCalculatorThread(int start, int end) {

        this.start = start;

        this.end = end;

    }

    @Override

    public void run() {

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

            partialSum += i;

        }

    }

    public long getPartialSum() {

        return partialSum;

    }

}

public class ParallelSumCalculation {

    public static void main(String[] args) {

        int numThreads = 5;

        int segmentSize = 100 / numThreads;

        SumCalculatorThread[] threads = new SumCalculatorThread[numThreads];

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

            int start = i \* segmentSize + 1;

            int end = (i + 1) \* segmentSize;

            threads[i] = new SumCalculatorThread(start, end);

            threads[i].start();

        }

        long totalSum = 0;

        for (SumCalculatorThread thread : threads) {

            try {

                thread.join();

                totalSum += thread.getPartialSum();

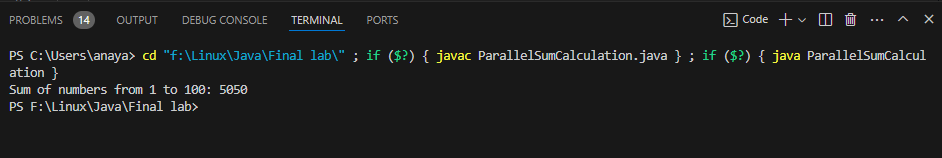
            } catch (InterruptedException e) {

                Thread.currentThread().interrupt();

            } }

        System.out.println("Sum of numbers from 1 to 100: " + totalSum);}}

Output:



11.Write a program that takes a paragraph of text as input and counts the  occurrences of each word. Additionally, identify the five most common words  and display them along with their frequencies.

Code:

import java.util.\*;

public class WordFrequencyCounter {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter a paragraph of text:");

        String paragraph = scanner.nextLine();

        scanner.close();

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

        Map<String, Integer> wordFrequencyMap = new HashMap<>();

        for (String word : words) {

            word = word.toLowerCase();

            wordFrequencyMap.put(word, wordFrequencyMap.getOrDefault(word, 0) + 1); }

List<Map.Entry<String, Integer>> sortedEntries = new ArrayList<>(wordFrequencyMap.entrySet());

        sortedEntries.sort((a, b) -> b.getValue().compareTo(a.getValue()));

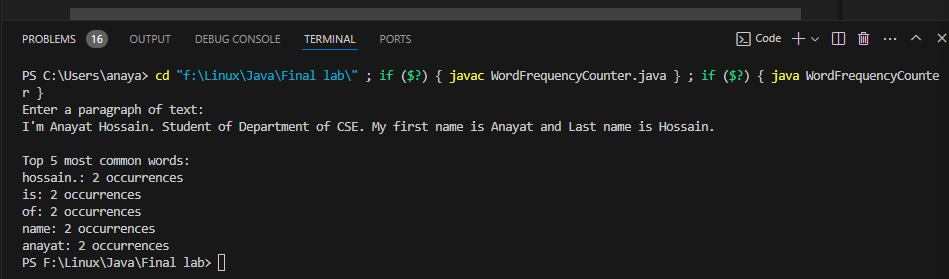
        System.out.println("\nTop 5 most common words:");

        for (int i = 0; i < Math.min(5, sortedEntries.size()); i++) {

            Map.Entry<String, Integer> entry = sortedEntries.get(i);

            System.out.println(entry.getKey() + ": " + entry.getValue() + " occurrences");   }  }}

Outputs:



12.Write a program that takes a sentence and a word as input and finds whether  the word is present as a substring in the sentence.

Code:

import java.util.Scanner;

public class SubstringChecker {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

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

        String sentence = scanner.nextLine();

        System.out.print("Enter a word to check: ");

        String word = scanner.next();

        boolean isSubstring = sentence.toLowerCase().contains(word.toLowerCase());

        if (isSubstring) {

            System.out.println("The word '" + word + "' is a substring in the sentence.");

        } else {

            System.out.println("The word '" + word + "' is not a substring in the sentence.");

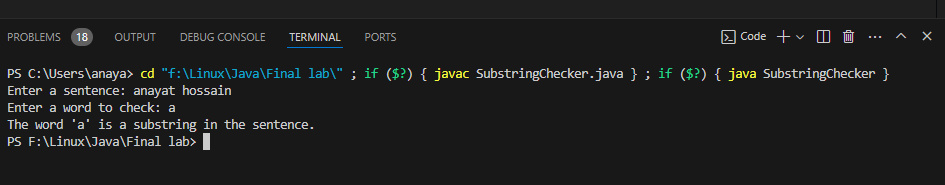
        }

        scanner.close();

    }

}

Output:



13.Write a program that takes a sentence as input and capitalizes the first letter  of each word. For example, "hello world" should become "Hello World".

Code:

import java.util.Scanner;

public class CapitalizeWords {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

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

        String sentence = scanner.nextLine();

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

        StringBuilder result = new StringBuilder();

        for (String word : words) {

            if (!word.isEmpty()) {

                char firstChar = Character.toUpperCase(word.charAt(0));

                String restOfWord = word.substring(1).toLowerCase();

                result.append(firstChar).append(restOfWord).append(" ");

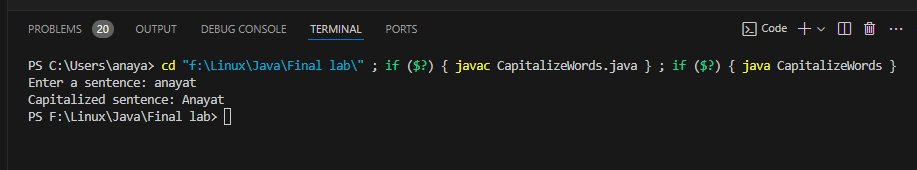
            }  }

        System.out.println("Capitalized sentence: " + result.toString().trim());

        scanner.close();

    }}

Output:



14.Create a function that takes a sentence as input and reverses the order of words  in it. For example, "Hello world" should become "world Hello".

Code:

public class WordReverser {

    public static void main(String[] args) {

        String inputSentence = "Hossain Anayat";

        String reversedSentence = reverseWords(inputSentence);

        System.out.println("Reversed sentence: " + reversedSentence);  }

    public static String reverseWords(String sentence) {

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

        StringBuilder reversed = new StringBuilder();

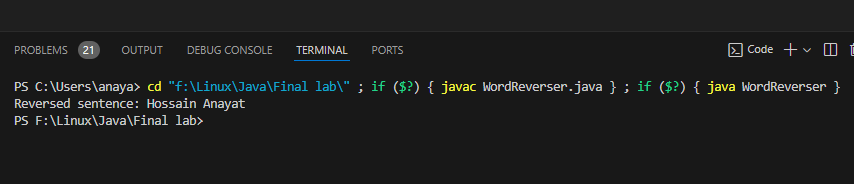
        for (int i = words.length - 1; i >= 0; i--) {

            reversed.append(words[i]).append(" ");  }

        return reversed.toString().trim();

    }}

Output:



15.Write a program that counts the occurrences of each character in a given string  and displays the count for each character.

Code:

public class CharacterCounter {

    public static void main(String[] args) {

        String inputString = "programming";

        countCharacterOccurrences(inputString);}

    public static void countCharacterOccurrences(String str) {

        int[] frequency = new int[26];

        for (char c : str.toCharArray()) {

            if (Character.isLetter(c)) {

                int index = c - 'a';

                frequency[index]++;    } }

        for (char c = 'a'; c <= 'z'; c++) {

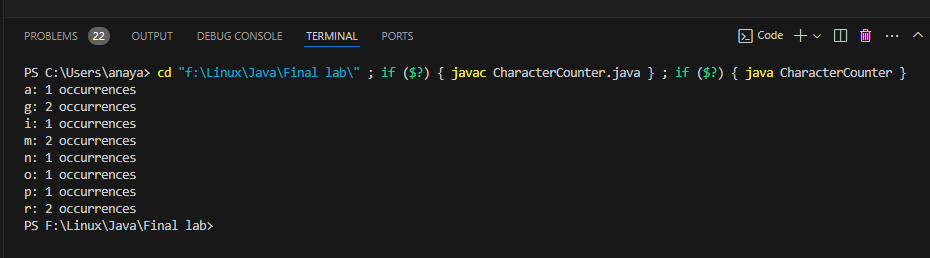
            int index = c - 'a';

            if (frequency[index] > 0) {

                System.out.println(c + ": " + frequency[index] + " occurrences");

            } }}}

Output:



16.Write a program that takes the first name and last name of a person as input  and concatenates them to form a full name.

Code:

import java.util.Scanner;

public class FullNameConcatenator {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

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

        String firstName = scanner.nextLine();

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

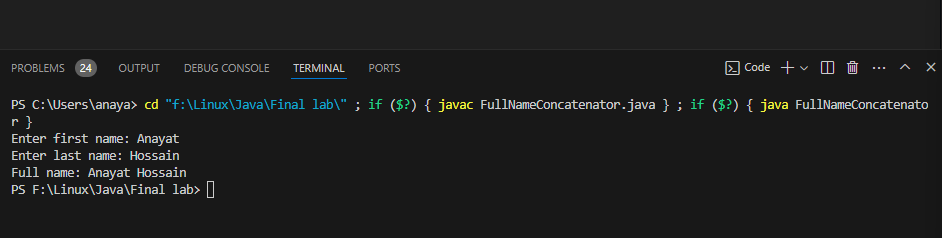
        String lastName = scanner.nextLine();

        String fullName = firstName + " " + lastName;

        System.out.println("Full name: " + fullName);

        scanner.close();    }}

Output:



17.Given the following strings: A = "The early bird catches the worm" B =  "Patience is a virtue" Your task is to extract the word "early" from A and  "virtue" from B. Then, concatenate these two words to form a sentence. After  that, capitalize the sentence and find the last occurrence of the letter 'V' from  the capitalized sentence. Perform all of these tasks using proper String class  functions.

Code:

public class StringManipulation {

    public static void main(String[] args) {

        String A = "The early bird catches the worm";

        String B = "Patience is a virtue";

       String wordFromA = A.substring(4, 9);

        String wordFromB = B.substring(10, 16);

        String concatenatedSentence = (wordFromA + " " + wordFromB).toLowerCase();

        String capitalizedSentence = concatenatedSentence.substring(0, 1).toUpperCase() + concatenatedSentence.substring(1);

        int lastIndexV = capitalizedSentence.lastIndexOf('V');

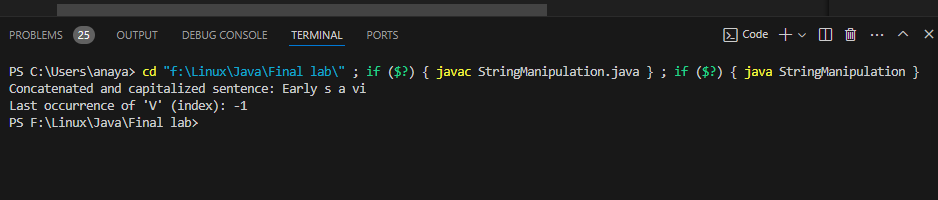
        System.out.println("Concatenated and capitalized sentence: " + capitalizedSentence);

        System.out.println("Last occurrence of 'V' (index): " + lastIndexV);

    }

}

Output:



18.You are developing a ticket booking system for a movie theater. Design a Java  program that uses a Queue to manage ticket requests, where each request  represents a customer wanting to book a ticket. Implement methods to add  new booking requests, process bookings in the order they were received, and  display the status of ticket bookings.

Code:

import java.util.LinkedList;

import java.util.Queue;

class TicketBookingSystem {

    private Queue<String> ticketRequests = new LinkedList<>();

    public void addBookingRequest(String customerName) {

        ticketRequests.offer(customerName);  }

    public void processBookings() {

        while (!ticketRequests.isEmpty()) {

            String customer = ticketRequests.poll();

            System.out.println("Booking processed for: " + customer);  }  }}

public class TicketBooking {

    public static void main(String[] args) {

        TicketBookingSystem bookingSystem = new TicketBookingSystem();

        bookingSystem.addBookingRequest("Anayat");

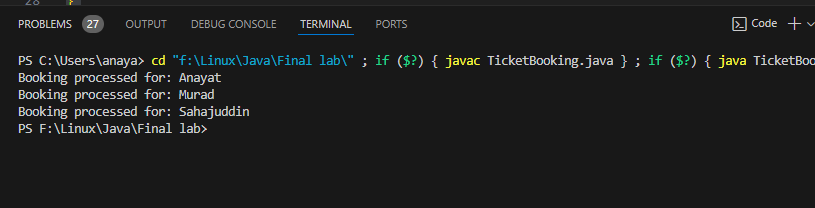
        bookingSystem.addBookingRequest("Murad");

        bookingSystem.addBookingRequest("Sahajuddin");

        bookingSystem.processBookings();

    }}

Output:



19.Create a class named Car with properties such as price (double), brand  (String), and speed (double). These properties will be initialized when an  object of the class is created. Create five objects of the Car class and add them  to an ArrayList. Display the cars whose price is over 2000000 takas. Complete  the program.

Code:

import java.util.ArrayList;

import java.util.List;

class Car {

    private double price;

    private String brand;

    private double speed;

    public Car(double price, String brand, double speed) {

        this.price = price;

        this.brand = brand;

        this.speed = speed;   }

    public double getPrice() {

        return price;  }

    public String getBrand() {

        return brand;    }}

public class CarList {

    public static void main(String[] args) {

        List<Car> cars = new ArrayList<>();

        cars.add(new Car(2500000, "Toyota", 180));

        cars.add(new Car(1800000, "Honda", 170));

        cars.add(new Car(3000000, "BMW", 220));

        cars.add(new Car(1500000, "Ford", 160));

        cars.add(new Car(2800000, "Mercedes", 200));

        System.out.println("Cars with price over 2000000 takas:");

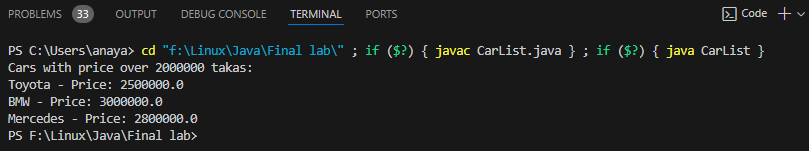
        for (Car car : cars) {

            if (car.getPrice() > 2000000) {

                System.out.println(car.getBrand() + " - Price: " + car.getPrice());

            } } }}

Output:



20.Write a basic Java program for managing student IDs and their grades in a  gradebook system. Implement methods to add new student IDs, remove  existing student IDs, display the list of student IDs, and store/display grades  for each student. Utilize simple data structures like arrays for storing student  IDs and grades.

Code:

import java.util.Scanner;

public class GradebookSystem {

    private static final int MAX\_STUDENTS = 100;

    private String[] studentIDs = new String[MAX\_STUDENTS];

    private double[] grades = new double[MAX\_STUDENTS];

    private int numStudents = 0;

    public void addStudent(String studentID, double grade) {

        if (numStudents < MAX\_STUDENTS) {

            studentIDs[numStudents] = studentID;

            grades[numStudents] = grade;

            numStudents++;

            System.out.println("Student added: " + studentID);

        } else {

            System.out.println("Gradebook is full. Cannot add more students.");  }    }

    public void removeStudent(String studentID) {

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

            if (studentIDs[i].equals(studentID)) {

                for (int j = i; j < numStudents - 1; j++) {

                    studentIDs[j] = studentIDs[j + 1];

                    grades[j] = grades[j + 1];     }

                numStudents--;

                System.out.println("Student removed: " + studentID);

                return;  }    }

        System.out.println("Student not found: " + studentID);    }

    public void displayStudents() {

        System.out.println("Student IDs and Grades:");

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

            System.out.println(studentIDs[i] + ": " + grades[i]);    }  }

    public static void main(String[] args) {

        GradebookSystem gradebook = new GradebookSystem();

        Scanner scanner = new Scanner(System.in);

        while (true) {

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

            System.out.println("1. Add Student");

            System.out.println("2. Remove Student");

            System.out.println("3. Display Students");

            System.out.println("4. Exit");

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

            int choice = scanner.nextInt();

            switch (choice) {

                case 1:

                    System.out.print("Enter student ID: ");

                    String studentID = scanner.next();

                    System.out.print("Enter grade: ");

                    double grade = scanner.nextDouble();

                    gradebook.addStudent(studentID, grade);

                    break;

                case 2:

                    System.out.print("Enter student ID to remove: ");

                    String idToRemove = scanner.next();

                    gradebook.removeStudent(idToRemove);

                    break;

                case 3:

                    gradebook.displayStudents();

                    break;

                case 4:

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

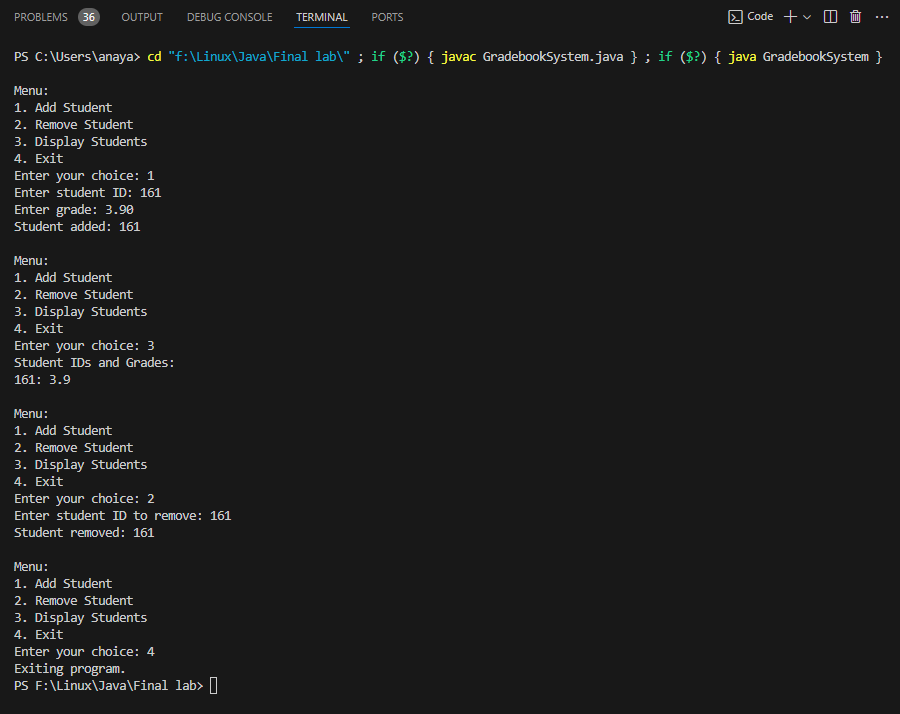
                    scanner.close();

                    System.exit(0);

                default:

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

            }   }  }}

Ooutput:

21.Create a class named Student. Write a program to insert 10 Student objects in  a Stack list. Now take user input for a variable named “menu”. If menu is 1  then insert another Student object. If menu is 2, delete the top Student object  from the stack list. If menu is 3, just output the top Student object. Use proper  stack list methods.

Code:

import java.util.Scanner;

import java.util.Stack;

class Student {

    private String name;

    public Student(String name) {

        this.name = name;    }

    public String getName() {

        return name;   }}

public class StudentStack {

    public static void main(String[] args) {

        Stack<Student> studentStack = new Stack<>();

        Scanner scanner = new Scanner(System.in);

        for (int i = 1; i <= 10; i++) {

            studentStack.push(new Student("Student " + i));        }

        while (true) {

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

            System.out.println("1. Insert Student");

            System.out.println("2. Delete Top Student");

            System.out.println("3. Display Top Student");

            System.out.println("4. Exit");

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

            int choice = scanner.nextInt();

            switch (choice) {

                case 1:

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

                    String studentName = scanner.next();

                    studentStack.push(new Student(studentName));

                    break;

                case 2:

                    if (!studentStack.isEmpty()) {

                        Student removedStudent = studentStack.pop();

                        System.out.println("Removed student: " + removedStudent.getName());

                    } else {

                        System.out.println("Stack is empty.");

                    }

                    break;

                case 3:

                    if (!studentStack.isEmpty()) {

                        System.out.println("Top student: " + studentStack.peek().getName());

                    } else {

                        System.out.println("Stack is empty.");

                    }

                    break;

                case 4:

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

                    scanner.close();

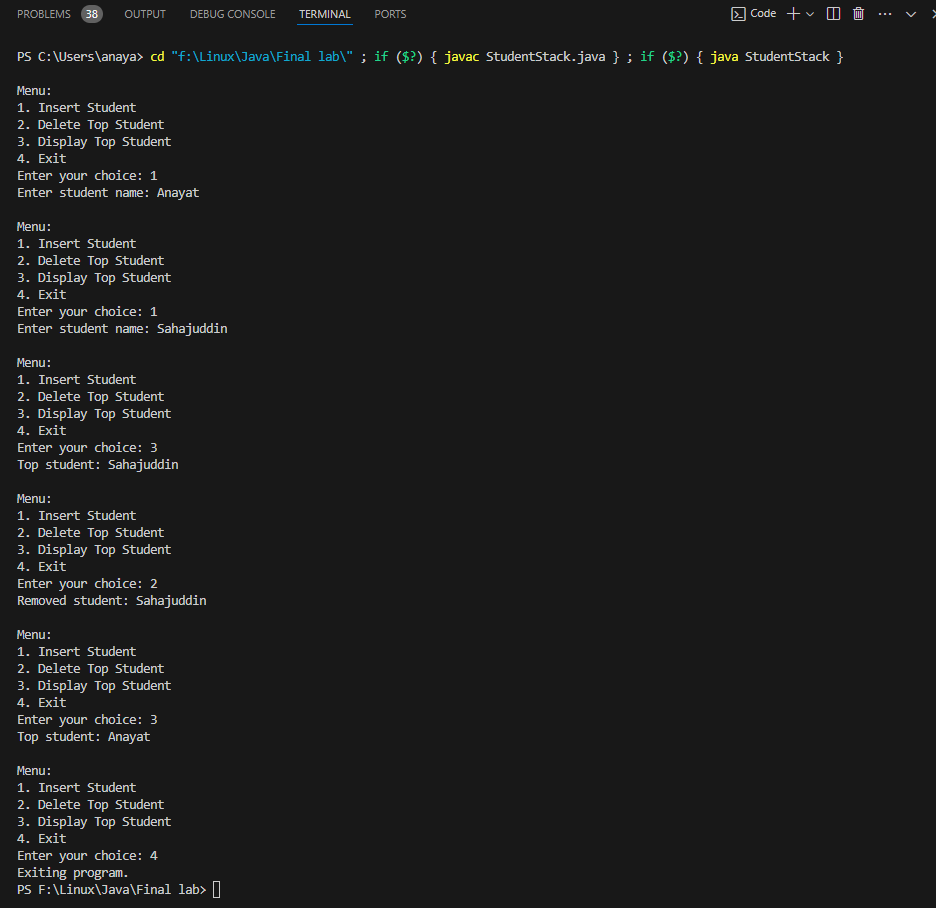
                    System.exit(0);

                default:

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

            }   }  }}

Output:



22.Write a Java program to remove duplicates from a list of strings. Implement  a method remove duplicates that takes a List of strings as input and removes  any duplicate elements, keeping only the first occurrence of each element.

Code:

import java.util.ArrayList;

import java.util.HashSet;

import java.util.List;

import java.util.Set;

public class RemoveDuplicates {

    public static void main(String[] args) {

        List<String> stringList = new ArrayList<>();

        stringList.add("apple");

        stringList.add("banana");

        stringList.add("apple");

        stringList.add("orange");

        stringList.add("banana");

        stringList.add("grape");

        System.out.println("Original list: " + stringList);

        removeDuplicates(stringList);

        System.out.println("List after removing duplicates: " + stringList);   }

    public static void removeDuplicates(List<String> list) {

        Set<String> seen = new HashSet<>();

        int writeIndex = 0;

        for (String str : list) {

            if (!seen.contains(str)) {

                seen.add(str);

                list.set(writeIndex++, str);    }    }

        list.subList(writeIndex, list.size()).clear();

    }}

Output:

