DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI

DISTRICT- NAINITAL-263132

2023-2024

Term work

of

Java Programming Lab (PCS - 408)

Submitted in partial fulfillment of the requirement for the IV semester

Bachelor of Technology

By

Himanshu Joshi

University Roll No

2261267

Under the Guidance of

Mr. Ravindra Koranga

Assistant Professor

Deptt. of CSE

/*Q16.Write a java program to accept and print the employee details during runtime.The detailswill include employee id, name, dept_ Id. The program should raise an exception if user inputsincomplete or incorrect data. The entered value should meet the following conditions:

a. First Letter of employee name should be in capital letter.

b. Employee id should be between 2001 and 5001

c. Department id should be an integer between 1 and 5.

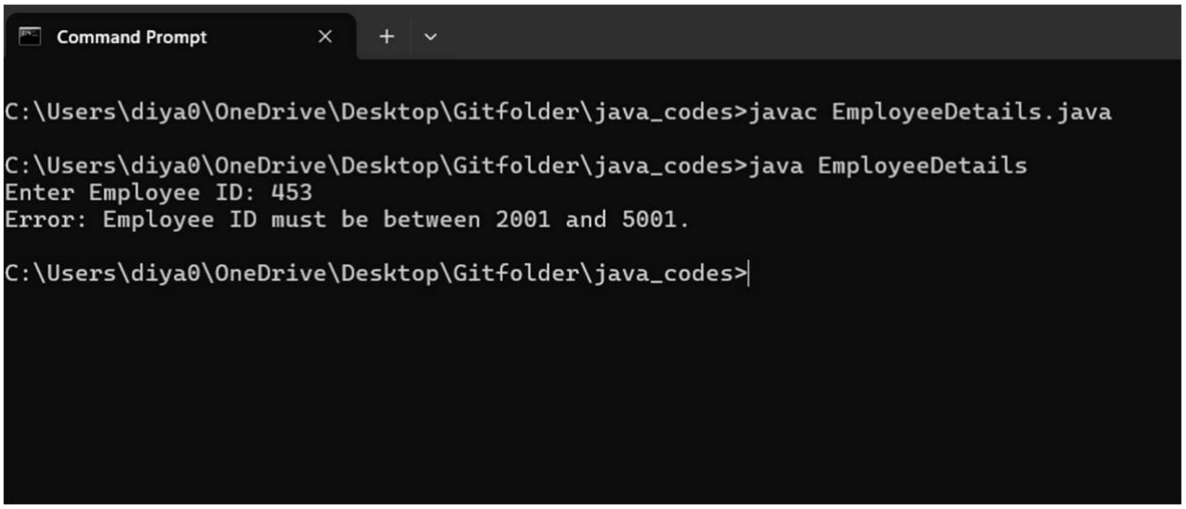
If the above conditions are not met, then the application should raise specific exception else should complete normal execution.*/

```
import java.util.Scanner;


class InvalidEmployeeNameException extends Exception { public

    InvalidEmployeeNameException(String message) {

        super(message);

    }

}


class InvalidEmployeeIdException extends Exception {

    public InvalidEmployeeIdException(String message) {

        super(message);

    }

}


class InvalidDepartmentIdException extends Exception {

    public InvalidDepartmentIdException(String message) {

        super(message);

    }

}


public class EmployeeDetails {
```

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

        try {
            // Accept Employee ID
            System.out.print("Enter Employee ID: ");
            int employeeId = scanner.nextInt();
            if (employeeId < 2001 || employeeId > 5001) {
                throw new InvalidEmployeeIdException("Employee ID must be between 2001 and
5001.");
            }

            // Accept Employee Name
            System.out.print("Enter Employee Name: ");
            String employeeName = scanner.next();
            if (Character.isLowerCase(employeeName.charAt(0))) {
                throw new InvalidEmployeeNameException("First letter of Employee Name must
be a capital letter.");
            }

            // Accept Department ID
            System.out.print("Enter Department ID:
            ");int deptId = scanner.nextInt();
            if (deptId < 1 || deptId > 5) {
                throw new InvalidDepartmentIdException("Department ID must be between 1 and
5.");

            }

            // Print Employee Details
            System.out.println("Employee Details:");
            System.out.println("Employee ID: " + employeeId);
```

```
            System.out.println("Employee Name: " + employeeName);
            System.out.println("Department ID: " + deptId);

        } catch (InvalidEmployeeIdException | InvalidEmployeeNameException |
InvalidDepartmentIdException e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

Output: Employee ID must be between 2001 and 5001



/*Q17.Create a class MyCalculator which consists of a single method power (int, int). This method takes two integers, n and p, as parameters and finds np. If either n or p is negative, then the method must throw an exception which says, "n and p should be non- negative".

Input Format

Each line of the input contains two integers, n and p. Output Format

Each line of the output contains the result, if neither of n and p is

negative.Otherwise, the output contains "n and p should be non-

negative".

Sample Input

-1 0

4 5

0 0

-2 -5

Sample Output

n and p should be non-

negative1024

n and p should not be zero

n and p should be non-negative

java.lang.Exception: n and p should not be zero. java.lang.Exception: n or p

should not be negative. java. lang. Exception: n or p should not be negative.

Explanation

In the first two cases, both n and p are positive. So, the power function

returnsthe answer correctly.

In the third case, both n and p are zero. So, the exception, "n and p should

notbe zero.” is printed.

In the last two cases, at least one out of n and p is negative. So, the

exception,"n or p should not be negative.” is printed for these two cases.

*/

import java.util.Scanner;

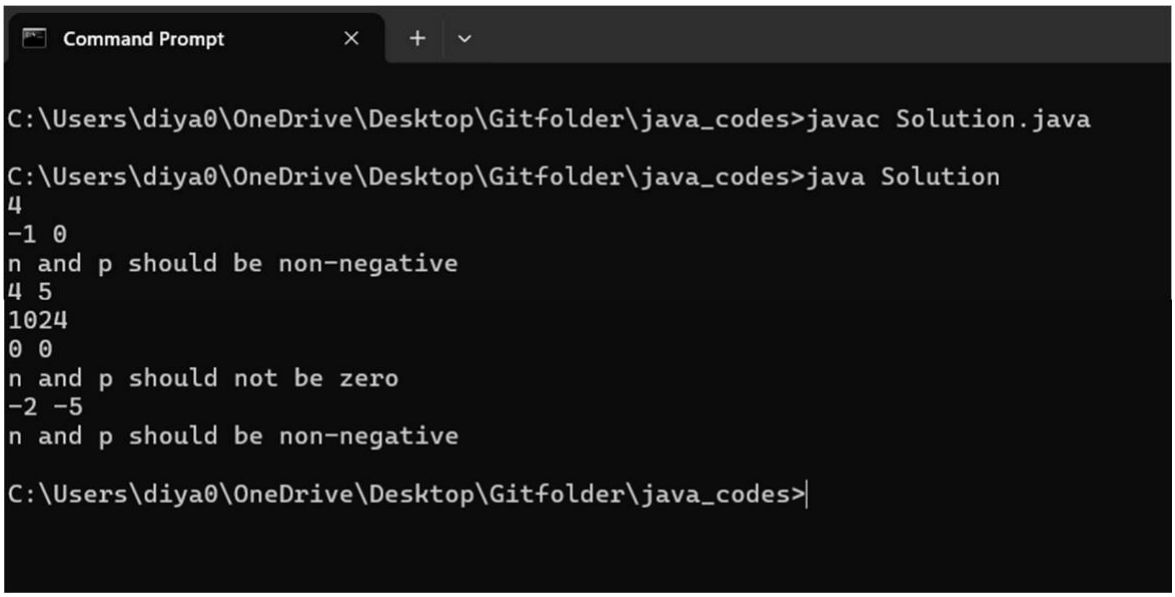
```
class MyCalculator {
    // Method to calculate n raised to the power of p
    public long power(int n, int p) throws Exception {
        if (n < 0 || p < 0) {
            throw new Exception("n and p should be non-negative");
        } else if (n == 0 && p == 0) {
            throw new Exception("n and p should not be zero");
        } else {
            return (long) Math.pow(n, p);
        }
    }
}

public class Solution {
    public  static  void  main(String[]  args)  {
        Scanner sc = new Scanner(System.in);
        MyCalculator myCalculator = new MyCalculator();
        int numberOfPairs = sc.nextInt(); // Assuming the first input is the number of pairs
        for (int i = 0; i < numberOfPairs; i++) {
            int n = sc.nextInt(); // Read the next integer
            nint p = sc.nextInt(); // Read the next integer
            ptry {
                System.out.println(myCalculator.power(n, p));
            }       catch       (Exception       e)       {
                System.out.println(e.getMessage())
                ;
            }
        }
        sc.close();
    }
}
```

Output: n and p should be non-

negative1024

n and p should not be zero

n and p should be non-negative



/*Q18.File Handling in Java:

Write a java file handling program to count and display the number of palindromes present ina text file "myfile.txt".

Example: If the file "myfile.txt" contains the following lines,

My name is NITIN

Hello aaa and bbb wordHow are You

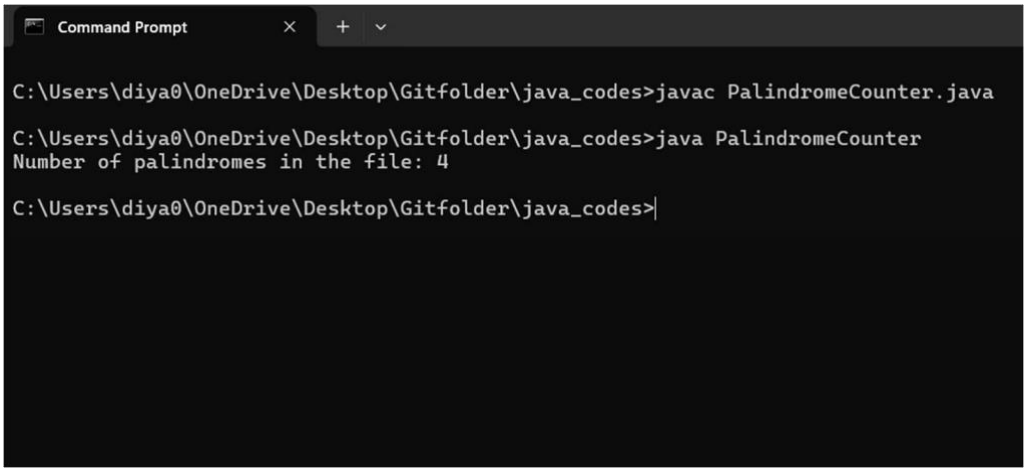
ARORA is my friend

Output will be => 4*/

```
import
java.io.BufferedReader;import
java.io.FileReader; import
java.io.IOException;

public class PalindromeCounter {
    public static void main(String[] args) {
        String filename = "myfile.txt";

        try (BufferedReader br = new BufferedReader(new FileReader(myfile.txt))) {
            String line;
            int palindromeCount = 0;

            while ((line = br.readLine()) != null) {
                String[] words = line.split("\\s+");
                for (String word : words) {
                    if (isPalindrome(word)) {
                        palindromeCount++;
                    }
                }
            }

            System.out.println("Number of palindromes in the file: " + palindromeCount);
```

```
        } catch (IOException e) {
            System.err.println("Error reading file: " + e.getMessage());
        }
    }

    // Method to check if a string is a palindrome
    private static boolean isPalindrome(String str) {
        str = str.toLowerCase();
        int left = 0;
        int right = str.length() - 1;

        while (left < right) {
            if (str.charAt(left) != str.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }
}
```

Output: 4



/*Q19.Multithreaded programming:

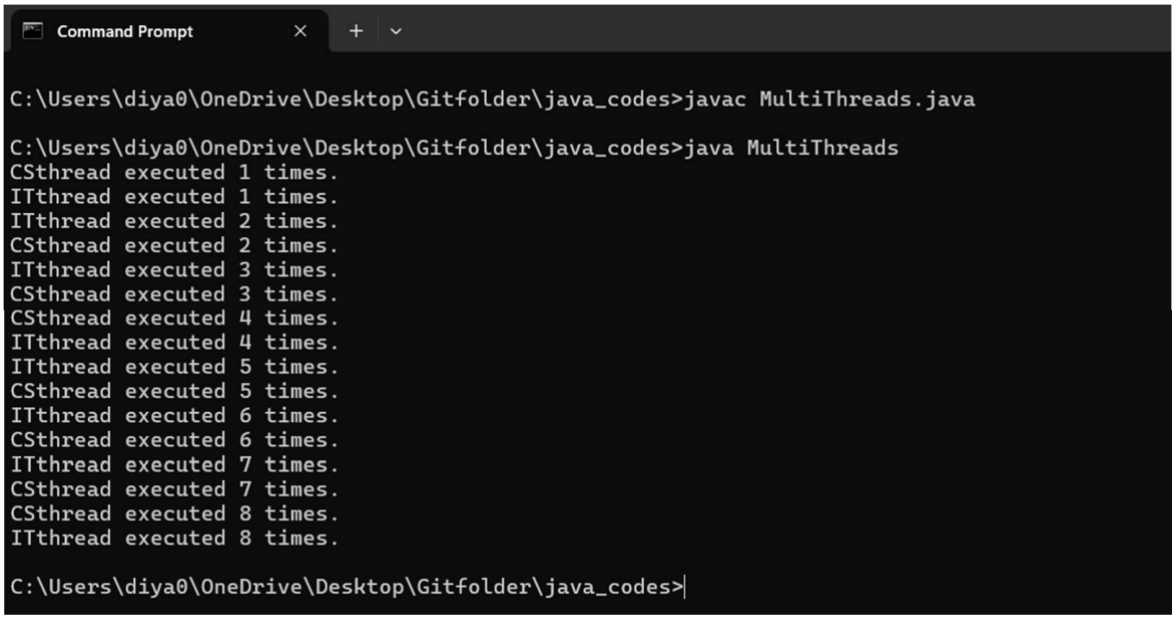
Write a program MultiThreads that creates two threads-one thread with the name CSthread andthe other thread named ITthread.

Each thread should display its respective name and execute after a gap of 500 milliseconds. Each thread should also display a number indicating the number of times it got a chance to execute.*/

```
public class MultiThreads {
    public static void main(String[] args) {
        // Create two threads with specified names
        Thread csThread = new Thread(new CustomRunnable("CSthread"), "CSthread");
        Thread itThread = new Thread(new CustomRunnable("ITthread"), "ITthread");

        // Start both
        threads
        csThread.start();
        itThread.start();
    }
}

class CustomRunnable implements Runnable {
    private final String threadName;
    private int executionCount = 0;

    public CustomRunnable(String threadName) {
        this.threadName = threadName;
    }

    public void run() {
        try {
            // Loop to repeatedly execute the
            threadwhile (executionCount < 8) {
```

```
            // Increment the execution
            countexecutionCount++;
            // Display the thread's name and execution count
            System.out.println(threadName + " executed " + executionCount + "
            times.");
            // Sleep for 500
            milliseconds
            Thread.sleep(500);
        }
    } catch (InterruptedException e) {
        System.err.println(threadName + " interrupted.");
    }
  }
}
```

Output:



```
/*Q20.Write a java program for to solve producer consumer problem in which
aproducer produces a value and consumer consume the value before
producer generate the next value.*/

class SharedBuffer {
    private int value;
    private boolean hasValue = false;

    public synchronized void produce(int newValue) throws InterruptedException {
        while (hasValue) {
            wait();
        }
        value = newValue;
        hasValue = true;
        System.out.println("Produced: " + value);
        notify();
    }

    public synchronized void consume() throws InterruptedException {
        while (!hasValue) {
            wait();
        }
        System.out.println("Consumed: " + value);
        hasValue = false;
        notify();
    }
}

class Producer implements Runnable {
    private SharedBuffer buffer;
```

```
    public Producer(SharedBuffer buffer) {
        this.buffer = buffer;
    }

    public void run() {
        int i = 0;
        while (true) {
            try {
                buffer.produce(i++);
                Thread.sleep(500); // Simulate time taken to produce
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class Consumer implements Runnable {
    private SharedBuffer buffer;

    public Consumer(SharedBuffer buffer) {
        this.buffer = buffer;
    }

    public void run() {
        while (true) {
            try {
                buffer.consume();
                Thread.sleep(500); // Simulate time taken to consume
```

```
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class ProducerConsumerProblem {
    public static void main(String[] args) {
        SharedBuffer buffer = new SharedBuffer();
        Thread producerThread = new Thread(new Producer(buffer));
        Thread consumerThread = new Thread(new Consumer(buffer));

        producerThread.start();
        consumerThread.start()
        ;
    }
}            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class ProducerConsumerProblem {
    public static void main(String[] args) {
        SharedBuffer buffer = new SharedBuffer();
        Thread producerThread = new Thread(new Producer(buffer));
        Thread consumerThread = new Thread(new Consumer(buffer));

        producerThread.start();
```

```
        consumerThread.start();
    }
}
```

Output:

```
Command Prompt

C:\Users\diya0\OneDrive\Desktop\Gitfolder\java_codes>javac ProducerConsumerProblem.java

C:\Users\diya0\OneDrive\Desktop\Gitfolder\java_codes>java ProducerConsumerProblem
Produced: 0
Consumed: 0
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
^C
C:\Users\diya0\OneDrive\Desktop\Gitfolder\java_codes>
```

/*Q21.Collection and Generic Framework:

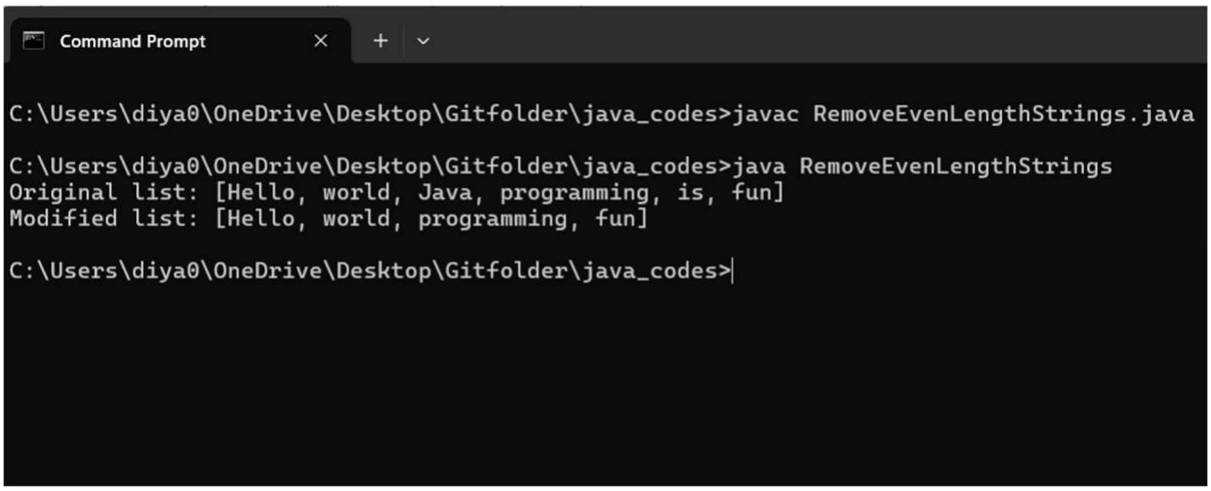
Write a method removeEvenLength that takes an ArrayList of Strings as a parameter and thatremoves all the strings of even length from the list.

(Use ArrayList)*/

```
import java.util.ArrayList;

import java.util.Iterator;


public class RemoveEvenLengthStrings {

    public static void removeEvenLength(ArrayList<String> list) {

        // Use an iterator to safely remove elements while

        iteratingIterator<String> iterator = list.iterator();


        while (iterator.hasNext()) {

            String str = iterator.next();

            // Check if the length of the string is

            evenif (str.length() % 2 == 0) {

                // Remove the string if its length is

                eveniterator.remove();

            }

        }

    }


    public static void main(String[] args) {

        // Example usage

        ArrayList<String> strings = new ArrayList<>();

        strings.add("Hello");

        strings.add("world");

        strings.add("Java");

        strings.add("programming")

        ;strings.add("is");
```

```
        strings.add("fun");

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

        // Remove strings of even
        length
        removeEvenLength(strings);

        System.out.println("Modified list: " + strings);
    }
}
```

Output: [Hello,world,programming,fun]



/*Q22.Write a method swapPairs that switches the order of values in an ArrayList of Strings in a pairwise fashion. Your method should switch the order of the first two values, then switch the order of the next two, switch the order of the next two, and so on.

For example, if the list initially stores these values: {"four", "score", "and","seven", "years",

"ago"} your method should switch the first pair, "four", "score", the second pair, "and", "seven", and the third pair, "years", "ago", to yield this list:{"score", "four", "seven", "and", "ago", "years"}.If there are an odd number of values in the list, the final element is not moved.

For example, if the original list had been: {"to", "be", "or", "not", "to", "be",

"hamlet"} It would again switch pairs of values, but the final value, "hamlet"

would not be moved, yielding this list: {"be", "to", "not", "or", "be", "to",

"hamlet"}

*/

```
import java.util.ArrayList;


public class SwapPairs {
    public static void swapPairs(ArrayList<String> list) {
        for (int i = 0; i < list.size() - 1; i += 2) {
            // Swap the elements at index i and
            i+1String temp = list.get(i);
            list.set(i, list.get(i + 1));
            list.set(i + 1, temp);
        }
    }
    public static void main(String[] args) {
        // Example 1
        ArrayList<String> list1 = new ArrayList<>();
        list1.add("four");
        list1.add("score");
        list1.add("and");
        list1.add("seven");
        list1.add("years");
```

```
        list1.add("ago");
        System.out.println("Original list 1: " + list1);
        swapPairs(list1);
        System.out.println("Modified list 1: " + list1);
        ArrayList<String> list2 = new ArrayList<>();
        list2.add("to");
        list2.add("be");
        list2.add("or");
        list2.add("not");
        list2.add("to");
        list2.add("be");
        list2.add("hamlet")
        ;
        System.out.println("Original list 2: " + list2);
        swapPairs(list2);
        System.out.println("Modified list 2: " + list2);
    }
}
```

Output:

[score,four,seven,and,ago,years

][be,to,not,or,be,to,hamlet]

/*Q23.Write a method called alternate that accepts two Lists of integers as its

parameters and returns a new List containing alternating elements from the twolists, in

thefollowing order:

• First element from first list

• First element from second list

• Second element from first list

• Second element from second list

• Third element from first list

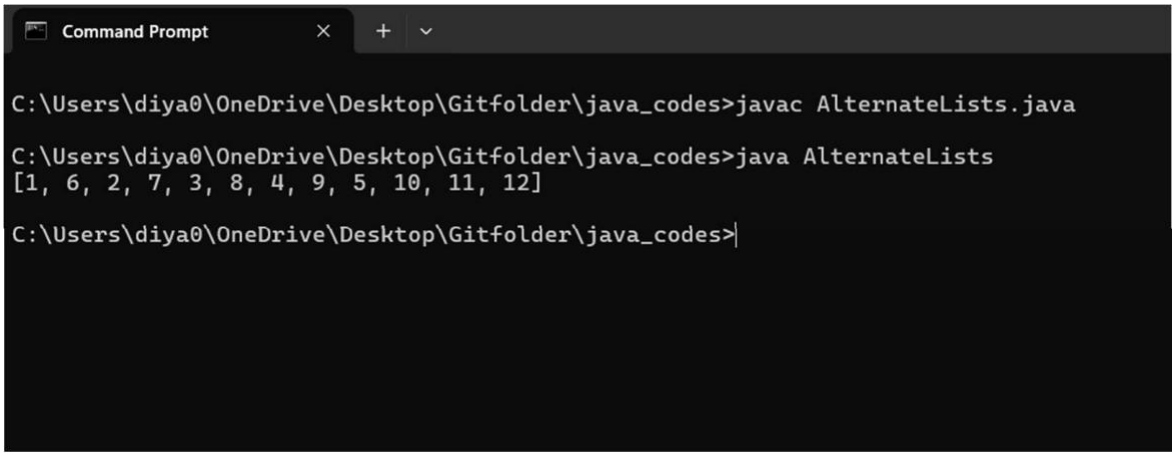
• Third element from second list

If the lists do not contain the same number of elements, the remaining elements from the longerlist should be placed consecutively at the end. For example, for a first list of (1, 2, 3, 4, 5) anda second list of (6, 7, 8, 9, 10, 11, 12), a call of alternate (list1,list2) should return a list
containing (1, 6, 2, 7, 3, 8, 4,9, 5, 10, 11, 12). Do not modify the parameter lists passed in.*/

```
import java.util.ArrayList;

import java.util.List;


public class AlternateLists {

    public static List<Integer> alternate(List<Integer> list1, List<Integer> list2) {

        List<Integer> result = new ArrayList<>();


        int size1 = list1.size();

        int size2 = list2.size();

        int maxSize = Math.max(size1, size2);


        // Iterate through both lists

        simultaneouslyfor (int i = 0; i < maxSize;

        i++) {

            if (i < size1) {

                result.add(list1.get(i)); // Add element from list1

            }

            if (i < size2) {
```

```
            result.add(list2.get(i)); // Add element from list2
        }
    }

    return result;
  }

  public static void main(String[] args) {
    List<Integer> list1 = List.of(1, 2, 3, 4, 5);
    List<Integer> list2 = List.of(6, 7, 8, 9, 10, 11, 12);
    List<Integer> result = alternate(list1, list2);
    System.out.println(result); // Output: [1, 6, 2, 7, 3, 8, 4, 9, 5, 10, 11, 12]
  }
}
```

Output: [1,6,2,7,3,8,4,9,5,10,11,12]



/*Q24.AWT & Swing, Event Handling:

Write a GUI program to develop an application that receives a string in one text field, and count number of vowels in a string and returns it in another text field, when the button named "CountVowel" is clicked.

When the button named "Reset" is clicked it will reset the value

oftextfield one and Textfield two.

When the button named "Exit" is clicked it will closed the application*/

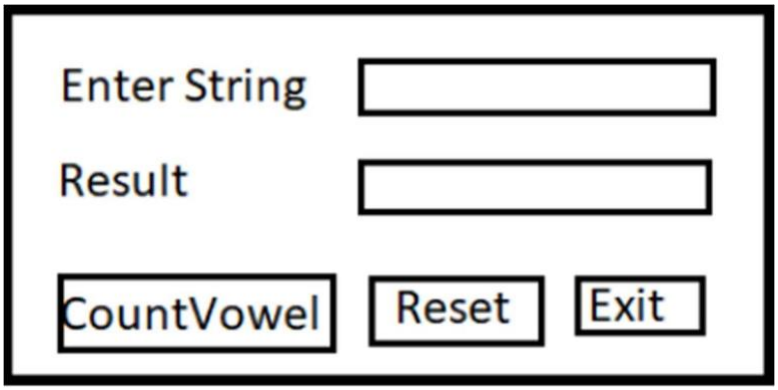
```
import javax.swing.*;

import java.awt.*;

import java.awt.event.ActionEvent;

import

java.awt.event.ActionListener;


public class VowelCounterApp {

    public static void main(String[] args) {

        // Create the frame

        JFrame frame = new JFrame("Vowel Counter");

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.setSize(400, 200);

        frame.setLayout(new GridLayout(3, 1));


        // Create the panel for input

        JPanel inputPanel = new JPanel(new BorderLayout());

        JLabel inputLabel = new JLabel("Enter a string:");

        JTextField inputField = new JTextField();

        inputPanel.add(inputLabel, BorderLayout.WEST);

        inputPanel.add(inputField, BorderLayout.CENTER);


        // Create the panel for result

        JPanel resultPanel = new JPanel(new BorderLayout());

        JLabel resultLabel = new JLabel("Result:");
```

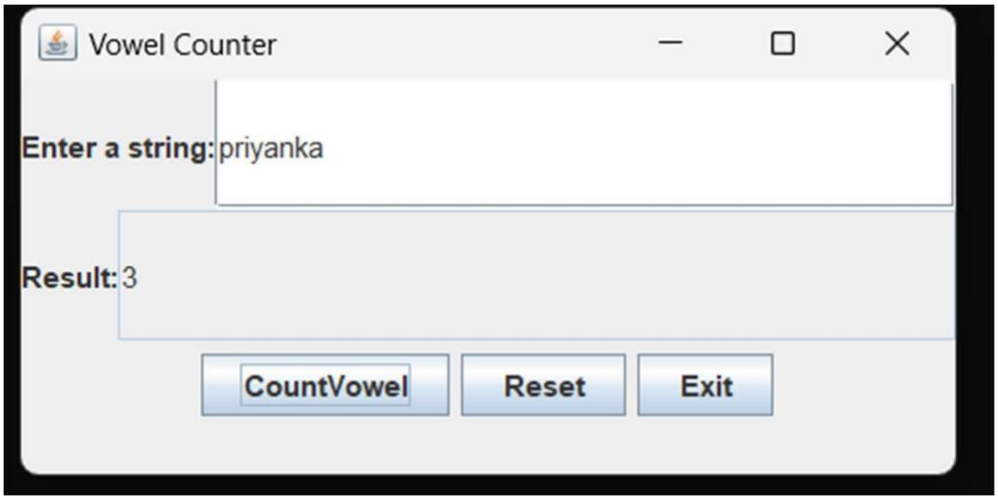
```
JTextField resultField = new JTextField();
resultField.setEditable(false);
resultPanel.add(resultLabel, BorderLayout.WEST);
resultPanel.add(resultField, BorderLayout.CENTER);

// Create the panel for buttons
JPanel buttonPanel = new JPanel();
buttonPanel.setLayout(new FlowLayout());

// Create the buttons
JButton countVowelButton = new JButton("CountVowel");
JButton resetButton = new JButton("Reset");
JButton exitButton = new JButton("Exit");

// Add action listeners to the buttons
countVowelButton.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        String inputText = inputField.getText();
        int vowelCount = countVowels(inputText);
        resultField.setText(String.valueOf(vowelCount));
    }
});

resetButton.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        inputField.setText("");
        resultField.setText("");
    }
```

```
    });

    exitButton.addActionListener(new ActionListener() {

        public void actionPerformed(ActionEvent e) {
            System.exit(0);
        }
    });

    // Add buttons to the panel
    buttonPanel.add(countVowelButton)
    ;buttonPanel.add(resetButton);
    buttonPanel.add(exitButton);

    // Add components to the
    frameframe.add(inputPanel);
    frame.add(resultPanel);
    frame.add(buttonPanel);

    // Make the frame
    visible
    frame.setVisible(true);
}

private static int countVowels(String input) {
    int count = 0;
    String vowels = "aeiouAEIOU";
    for (int i = 0; i < input.length(); i++) {
        if (vowels.indexOf(input.charAt(i)) != -1) {
            count++;
        }
```

```
        }
        return count;
    }
}
```

Output:





/*Q25.Java Database Connectivity (JDBC):

Create a database of employee with the following fields.

• Name

• Code

• Designation

• Salary

a) Write a java program to create GUI java application to take employee data from theTextFields and store it in database using JDBC connectivity.

*/

```
import javax.swing.*;

import java.awt.*;

import

java.awt.event.*;import

java.sql.*;


public class EmployeeDatabaseApp extends JFrame implements ActionListener {

    private JLabel nameLabel, codeLabel, designationLabel, salaryLabel;

    private JTextField nameField, codeField, designationField, salaryField;

    private JButton saveButton, resetButton, exitButton;


    // JDBC URL, username, and password of MySQL server

    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/employeedb";

    private static final String USERNAME = "root";

    private static final String PASSWORD = "your_sql_password”;


    public EmployeeDatabaseApp() {

        setTitle("Employee Data Entry");

        setSize(400, 250);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


        // Initialize components
```

```
nameLabel = new JLabel("Name:");
codeLabel = new JLabel("Code:");
designationLabel = new JLabel("Designation:");
salaryLabel = new JLabel("Salary:");

nameField = new JTextField(20);
codeField = new JTextField(10);
designationField = new JTextField(20);
salaryField = new JTextField(10);

saveButton = new JButton("Save");
resetButton = new
JButton("Reset");exitButton = new
JButton("Exit");

// Add action listeners to buttons
saveButton.addActionListener(this);
resetButton.addActionListener(this)
;exitButton.addActionListener(this);

// Create panel for buttons
JPanel buttonPanel = new JPanel(new FlowLayout());
buttonPanel.add(saveButton);
buttonPanel.add(resetButton);
buttonPanel.add(exitButton);

// Set layout
setLayout(new GridLayout(6, 2));

// Add components to the
frameadd(nameLabel);
```

```
    add(nameField);
    add(codeLabel);
    add(codeField);
    add(designationLabel);
    add(designationField);
    add(salaryLabel);
    add(salaryField);
    add(new JLabel()); // Placeholder for empty cell
    add(buttonPanel); // Add button panel

    setVisible(true);
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == saveButton) {
        saveEmployeeData();
    } else if (e.getSource() == resetButton) {
        resetFields();
    } else if (e.getSource() == exitButton) {
        System.exit(0);
    }
}

private void saveEmployeeData() {
    String name = nameField.getText();
    int code = Integer.parseInt(codeField.getText());
    String designation = designationField.getText();
    double salary = Double.parseDouble(salaryField.getText());

    try {
```

```
// Register MySQL JDBC driver
Class.forName("com.mysql.cj.jdbc.Driver");

// Open a connection
Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/employeedb
","root","your_sql_password");

// Create a prepared statement
String query = "INSERT INTO employee (NAME, CODE, DESIGNATION,
SALARY) VALUES (?, ?, ?, ?)";
PreparedStatement statement =
connection.prepareStatement(query);statement.setString(1,
name);
statement.setInt(2, code);
statement.setString(3,
designation);
statement.setDouble(4, salary);

// Execute the statement
int rowsInserted =
statement.executeUpdate();if (rowsInserted
> 0) {
   JOptionPane.showMessageDialog(this, "Employee data saved successfully!");
} else {
   JOptionPane.showMessageDialog(this, "Failed to save employee data.");
}

// Close resources
statement.close();
connection.close()
;
```

```
} catch (Exception ex) {
    ex.printStackTrace();
    JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());

    }
    }


    private void resetFields() { nameField.setText(""); codeField.setText(""); designationField.setText(""); salaryField.setText("");
    }


    public static void main(String[] args) { SwingUtilities.invokeLater(() -> new EmployeeDatabaseApp());
    }
    }
```

Output: