**PROJECT NAME:**

**PALINDROME CHECKER**

****

**PRESENTED BY:**

**BHAVANI.A**

**INTERND ID:VN-JD-4W213**

**INTRODUCTION:**

The Palindrome Checker is a Java-based application designed to identify whether a given word, phrase, or number is a palindrome. A palindrome is a sequence of characters that reads the same backward as forward, disregarding spaces, punctuation, and case sensitivity. This project employs a graphical user interface (GUI) developed using Java Swing to provide an intuitive and user-friendly experience.

Key Features:

Interactive User Interface: The application features a clean and responsive GUI that allows users to enter text and view results instantly.

Core Logic Implementation: The program processes the input by removing non-alphanumeric characters and checks for palindrome properties.

Input Validation: Ensures robust handling of invalid or empty inputs to improve usability and feedback.

Extensibility: Modular code structure allows easy enhancements, such as database integration or exporting results.

Technologies Used:

Java: Core language for implementing the application.

Swing: GUI development framework for creating the interface.

Regular Expressions: Used for text cleaning and validation.

**ABSTRACT:**

The Palindrome Checker project is a Java-based application that determines whether a given word, phrase, or number is a palindrome. A palindrome is a sequence that reads the same backward as forward, irrespective of spaces, punctuation, and case sensitivity. This application uses a graphical user interface (GUI) built with Java Swing, providing an intuitive and interactive platform for users to input text and receive instant feedback.

The application is designed with a modular approach, ensuring clean and reusable code. It incorporates robust input validation to handle diverse user inputs, delivering accurate results even for complex phrases or alphanumeric strings. The core logic leverages string manipulation and regular expressions to strip non-essential characters and normalize the input for analysis.

By combining algorithmic problem-solving with user-friendly design, this project highlights the practical implementation of core Javaprogramming skills. It also serves as a foundation for future enhancements, such as integrating database support for storing user inputs or exporting results in various formats. The Palindrome Checker is an ideal tool for educational and practical demonstrations of programming concepts.

**SYSTEM ARCHITECTURE:**

The system architecture of the Palindrome Checker application is designed with modularity and scalability in mind, consisting of several key layers. The User Interface (UI) Layer provides an intuitive and interactive platform using Java Swing, allowing users to input text or numbers, trigger palindrome checks, and view results or error messages. This layer is responsible for managing userinteractions and ensuring a seamless experience. The Application Logic Layer contains the core functionality, including the palindrome-checking algorithm, which processes the input by removing non-alphanumeric characters, normalizing it, and comparing it to its reverse. This layer also handles input validation to ensure meaningful feedback and manages the flow of data between the UI and the core logic.

Optionally, a Data Management Layer can be integrated to handle storage and retrieval of user inputs and results. This layer may include in-memory storage for session data or persistent storage using databases like SQLite or JavaDB for historical tracking and report generation. Additionally, utilities such as regular expressions and string manipulation helpers are encapsulated in the Utilities Layer to support the application with reusable functions. This modular design ensures the application is robust, user-friendly, and adaptable for future enhancements, such as integrating database storage or exporting results in various formats.

**IMPLEMENTATION:**

The Palindrome Checker application has been implemented using Java with a focus on modularity, reusability, and a user-friendly graphical interface. The implementation is divided into several key components:

1. User Interface (UI)

The UI is developed using Java Swing, which provides a clean and interactive graphical interface for the user. The main frame includes the following:

Input Field: A text field where users can input words, phrases, or numbers.

Button: A "Check Palindrome" button to trigger the validation process.

Result Label: A dynamic label that displays the output, indicating whether the input is a palindrome.

The UI ensures real-time feedback and handles invalid inputs by displaying appropriate error messages.

2. Core Logic

The core functionality is implemented in a method called isPalindrome. This method:

Removes all non-alphanumeric characters using regular expressions.

Converts the sanitized input to lowercase for uniform comparison.

Compares the normalized string with its reversed version to determine if it is a palindrome.

This logic ensures accurate results even for complex inputs such as phrases with spaces and punctuation.

3. Input Validation

The application validates user input to ensure it is non-empty before processing. If the input is invalid (e.g., empty or only spaces), a clear error message is displayed to the user.

4. Error Handling

Comprehensive error handling ensures the application remains robust and user-friendly. Invalid inputs are caught, and meaningful feedback is provided to guide the user in entering valid data.

5. Modularity

The code is structured into distinct methods and components, ensuring readability and reusability. The core logic, input validation, and UI elements are separated to allow easy debugging and enhancements.

6. Extensibility

The application is designed to accommodate future enhancements. For instance:

Persistent data storage can be added using JDBC and SQLite for storing user inputs and history.

**SOURCE CODE:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class PalindromeChecker {

    public static void main(String[] args) {

        // Create the main frame

        JFrame frame = new JFrame("Palindrome Checker");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.setSize(400, 300);

        // Create components

        JLabel instructionLabel = new JLabel("Enter a word, phrase, or number:");

        JTextField inputField = new JTextField(20);

JButton checkButton = new JButton("Check Palindrome");

        JLabel resultLabel = new JLabel("Result will appear here", SwingConstants.CENTER);

// Configure components

        resultLabel.setFont(new Font("Arial", Font.BOLD, 14));

        resultLabel.setForeground(Color.BLUE);

        // Layout setup

        JPanel panel = new JPanel();

        panel.setLayout(new GridLayout(4, 1, 10, 10));

        panel.add(instructionLabel);

        panel.add(inputField);

        panel.add(checkButton);

        panel.add(resultLabel);

        frame.add(panel);

        // Add action listener for the button

        checkButton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                String input = inputField.getText().trim();

                if (input.isEmpty()) {

                    resultLabel.setText("Please enter a valid input.");

                } else {

                    if (isPalindrome(input)) {

                        resultLabel.setText("\"" + input + "\" is a Palindrome!");

                    } else {

                        resultLabel.setText("\"" + input + "\" is NOT a Palindrome.");

                    }

                }

            }

        });

        // Make the frame visible

        frame.setVisible(true);

    }

    // Core logic to check for palindrome

private static boolean isPalindrome(String input) {

        // Remove non-alphanumeric characters and convert to lowercase

        String cleanedInput = input.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();

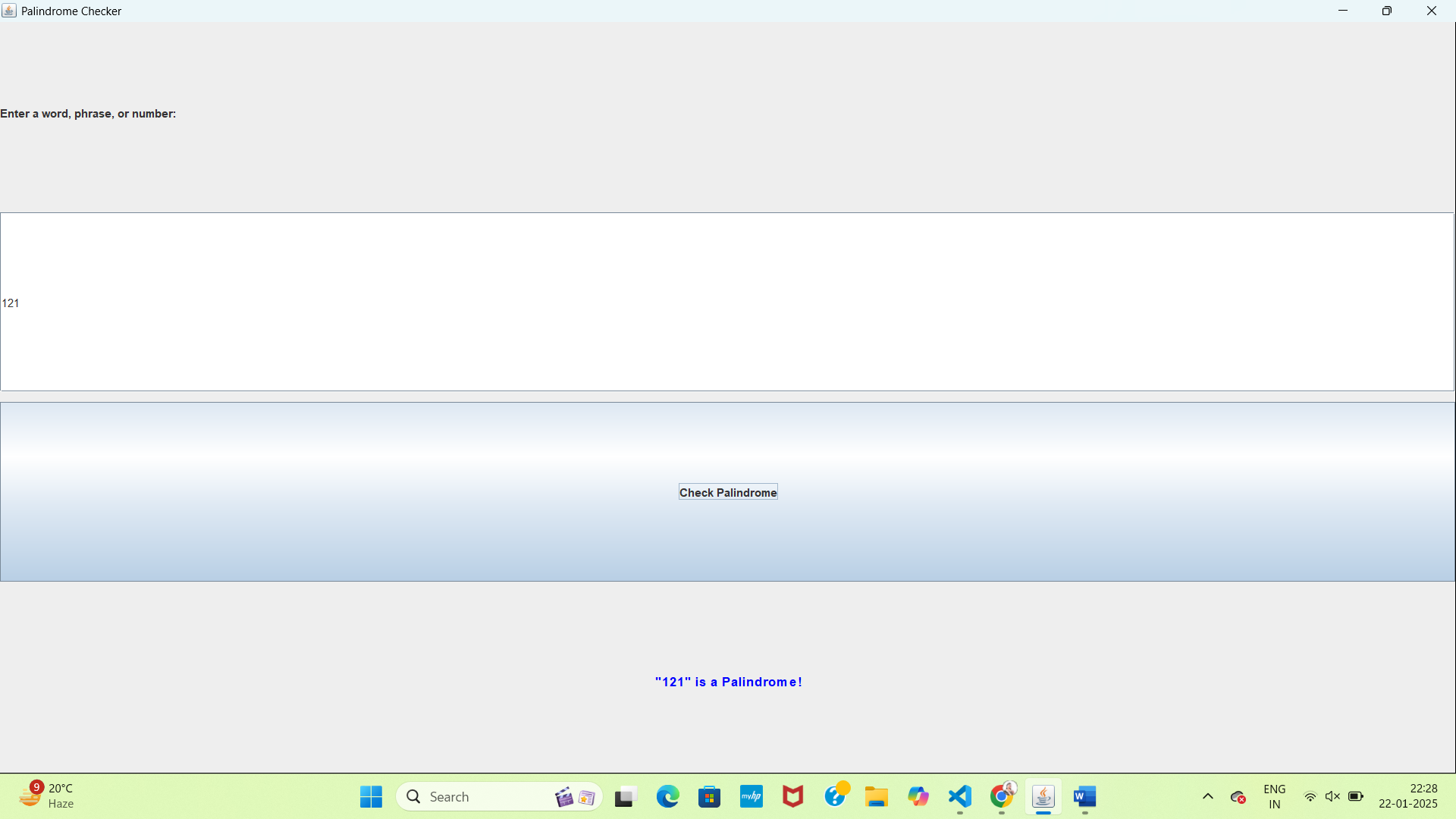
        String reversedInput = new StringBuilder(cleanedInput).reverse().toString();

        return cleanedInput.equals(reversedInput);

    }

}

OUTPUT:



**CONCLUSION:**

n this project, I developed a Palindrome Checker application using Java Swing to provide a graphical user interface (GUI). The program allows users to input a string, and upon clicking the "Check" button, it verifies whether the string is a palindrome or not.

The project demonstrates the use of Swing components such as JTextField, JButton, and JLabel for user interaction. I also implemented basic string manipulation and logic to check if the string reads the same forward and backward. The GUI offers an intuitive and responsive experience, providing feedback to users in real-time.

Key learnings from this project include:

Implementing event-driven programming with Swing, particularly focusing on button clicks to trigger actions.

Handling user input and managing the flow of information between GUI components.

Gaining experience in string manipulation and working with basic Java constructs.

This project serves as a great foundation for building more complex applications involving user input handling and GUI development.

**FUTURE ENHANCEMENTS:**

In this project, I developed a Palindrome Checker application using Java Swing to provide a graphical user interface (GUI). The program allows users to input a string, and upon clicking the "Check" button, it verifies whether the string is a palindrome or not.

The project demonstrates the use of Swing components such as JTextField, JButton,and JLabel for user interaction. I also implemented basic string manipulation and logic to check if the string reads the same forward and backward. The GUI offers an intuitive and responsive experience, providing feedback to users in real-time.

Key learnings from this project include:

Implementing event-driven programming with Swing, particularly focusing on button clicks to trigger actions.

Handling user input and managing the flow of information between GUI components.

Gaining experience in string manipulation and working with basic Java constructs.

This project serves as a great foundation for building more complex applications involving user input handling and GUI development.