

AIR TRAFFIC CONTROL

A MINI-PROJECT BY:

Prathyush R. 230701240

Naveed Ahmed Basha 230701204

in partial fulfillment of the award of the degree

OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous

Institute

CHENNAI

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project “**AIR TRAFFIC CONTROL**” is the bonafide work of “**PRATHYUSH R, NAVEED AHMED BASHA**” who carried out the project work under my supervision.

Submitted for the practical examination held on _____

SIGNATURE

Ms. ASWANA LAL
Asst. Professor
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam, Chennai-602105

SIGNATURE

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

An Air Traffic Control (ATC) application is a dynamic system designed to ensure the safe, efficient, and coordinated movement of aircraft within controlled airspace. This project aims to streamline air traffic management processes and provide support for ATC operators.

The application integrates real-time data from various sources, including aircraft, weather systems, and surrounding airspace, to manage both incoming and outgoing aircraft. Key features include tracking aircraft positions, coordinating flight plans, managing airspace sectors dynamically, and optimizing departure and arrival processes.

Aircraft tracking and monitoring are achieved using advanced technologies such as radar and satellite systems, ensuring real-time awareness of their position, speed, and altitude. The system uses predictive algorithms to handle flight paths and resolve potential conflicts, adjusting for weather conditions, congestion, or emergencies.

Operators are supported with automated alerts and a communication interface for seamless interaction with pilots and other control centers. The ATC application ensures smooth departure sequencing, efficient arrival procedures, and real-time decision-making to maintain safety and minimize delays.

The project will help modernize air traffic control systems, enabling safer and more efficient air travel.

TABLE OF CONTENTS

1. INTRODUCTION

- 1.1 INTRODUCTION
- 1.2 IMPLEMENTATION
- 1.3 SCOPE OF THE PROJECT
- 1.4 WEBSITE FEATURES

2. SYSTEM SPECIFICATION

- 2.1 HARDWARE SPECIFICATION
- 2.2 SOFTWARE SPECIFICATION

3. SAMPLE CODE

- 3.1 Creating the menu
- 3.2 Database connection and operations
- 3.3 Flightentry class
- 3.4 UI components layout
- 3.5 Background panel
- 3.6 Main Application

4. SNAPSHOTS

5. CONCLUSION

6. REFERENCES

INTRODUCTION

1.1 INTRODUCTION

The ATC Aircraft Logging System is a Java-based application designed to manage and store flight information for air traffic control. It provides a simple interface for adding, updating, and deleting flight records. The system connects to a MySQL database, ensuring efficient data management and accessibility for ATC personnel.

1.2 IMPLEMENTATION

The system is built using Java and Swing for the user interface, providing a platform-independent solution. It uses MySQL to store flight records. The app allows users to interact with a table displaying flight information and uses SQL queries to handle data operations, such as inserting, updating, and deleting records.

1.3 SCOPE OF THE PROJECT

This project focuses on creating a basic logging system for ATC operations. The main features include managing flight data and ensuring smooth CRUD (Create, Read, Update, Delete) operations. Future improvements could include flight tracking, automated conflict resolution, and additional reporting features.

1.5 WEBSITE FEATURES

The system offers several features for efficient flight management:

- **Flight Entry:** Users can add, update, or delete flight records.
- **Data Display:** Flight details are shown in a table format for easy access.
- **Conflict Check:** It checks for conflicts like duplicate flight numbers.
- **User Interface:** The design is modern and user-friendly.
- **Database Integration:** All flight data is stored in a MySQL database, making it accessible across sessions.

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS:

PROCESSOR	:	Intel i5
MEMORY SIZE	:	4GB(Minimum)
HARD DISK	:	500 GB of free space

2.2 SOFTWARE SPECIFICATIONS:

PROGRAMMING LANGUAGE	:	Java, MySQL
FRONT-END	:	Java
BACK-END	:	MySQL
OPERATING SYSTEM	:	Windows 11

SAMPLE CODE

3.1 Creating the Menu

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

public class ATCWindow extends JFrame {

    public ATCWindow() {
        setTitle("ATC Aircraft Logging System");
        setSize(900, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setupMenuBar();

        setLayout(new BorderLayout());

        BackgroundPanel backgroundPanel = new BackgroundPanel();
        setContentPane(backgroundPanel);

        setVisible(true);
    }

    private void setupMenuBar() {
        JMenuBar menuBar = new JMenuBar();

        JMenu fileMenu = new JMenu("File");
        JMenuItem exitMenuItem = new JMenuItem("Exit");
        exitMenuItem.addActionListener(new ExitMenuItemListener());
        fileMenu.add(exitMenuItem);
        menuBar.add(fileMenu);

        JMenu operationsMenu = new JMenu("Operations");
        JMenuItem addMenuItem = new JMenuItem("Add New Flight");
        addMenuItem.addActionListener(new AddNewFlightListener());
        operationsMenu.add(addMenuItem);

        JMenuItem updateMenuItem = new JMenuItem("Update Flight");
        updateMenuItem.addActionListener(new UpdateFlightListener());
        operationsMenu.add(updateMenuItem);

        JMenuItem deleteMenuItem = new JMenuItem("Delete Flight");
        deleteMenuItem.addActionListener(new DeleteFlightListener());
        operationsMenu.add(deleteMenuItem);

        menuBar.add(operationsMenu);

        JMenu viewMenu = new JMenu("View");
```

```

JMenuItem viewMenuItem = new JMenuItem("View All Flights");
viewMenuItem.addActionListener(new ViewFlightsListener());
viewMenu.add(viewMenuItem);
menuBar.add(viewMenu);

setJMenuBar(menuBar);
}

private class ExitMenuItemListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
}

private class AddNewFlightListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(ATCWindow.this, "Add New Flight clicked.");
    }
}

private class UpdateFlightListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(ATCWindow.this, "Update Flight clicked.");
    }
}

private class DeleteFlightListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(ATCWindow.this, "Delete Flight clicked.");
    }
}

private class ViewFlightsListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(ATCWindow.this, "View All Flights clicked.");
    }
}

private class BackgroundPanel extends JPanel {
    private Image backgroundImage;

    public BackgroundPanel() {
        backgroundImage = new ImageIcon("KAIA-ATCtower.jpg").getImage();
    }
}

```



```

@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);
}

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

```

3.2 Database Connection and Operations

```

import java.sql.*;

public class DatabaseManager {
    // Database credentials
    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/atc_db";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "1234";

    // Load data from the database into the table
    public static ResultSet loadData() throws SQLException {
        Connection connection = DriverManager.getConnection(JDBC_URL, DB_USER, DB_PASSWORD);
        Statement statement = connection.createStatement();
        return statement.executeQuery("SELECT * FROM aircraft_log");
    }

    // Save a new flight entry to the database
    public static void saveData(FlightEntry entry) throws SQLException {
        String query = "INSERT INTO aircraft_log (aircraft_id, flight_number, runway, arrival_time, departure_time, arrival_airport, departure_airport) VALUES (?, ?, ?, ?, ?, ?, ?)";
        try (Connection connection = DriverManager.getConnection(JDBC_URL, DB_USER, DB_PASSWORD);
            PreparedStatement statement = connection.prepareStatement(query)) {

            statement.setString(1, entry.aircraftId);
            statement.setString(2, entry.flightNumber);
            statement.setString(3, entry.runway);
            statement.setTimestamp(4, new Timestamp(entry.arrivalTime.getTime()));
            statement.setTimestamp(5, new Timestamp(entry.departureTime.getTime()));
            statement.setString(6, entry.arrivalAirport);
            statement.setString(7, entry.departureAirport);

            statement.executeUpdate();
        }
    }
}

```

```

// Check if a flight number already exists in the database
public static boolean isFlightNumberExists(String flightNumber) throws SQLException {
    String query = "SELECT COUNT(*) FROM aircraft_log WHERE flight_number = ?";
    try (Connection connection = DriverManager.getConnection(JDBC_URL, DB_USER,
DB_PASSWORD);
        PreparedStatement stmt = connection.prepareStatement(query)) {

        stmt.setString(1, flightNumber);
        ResultSet rs = stmt.executeQuery();
        return rs.next() && rs.getInt(1) > 0;
    }
}

// Delete a flight record from the database by flight number
public static void deleteData(String flightNumber) throws SQLException {
    String query = "DELETE FROM aircraft_log WHERE flight_number = ?";
    try (Connection connection = DriverManager.getConnection(JDBC_URL, DB_USER,
DB_PASSWORD);
        PreparedStatement stmt = connection.prepareStatement(query)) {

        stmt.setString(1, flightNumber);
        stmt.executeUpdate();
    }
}
}

```

3.3 FlightEntry Class

```

import java.util.Date;

public class FlightEntry {
    String aircraftId, flightNumber, runway, arrivalAirport, departureAirport;
    Date arrivalTime, departureTime;

    public FlightEntry(String aircraftId, String flightNumber, String runway, Date arrivalTime, Date
departureTime, String arrivalAirport, String departureAirport) {
        this.aircraftId = aircraftId;
        this.flightNumber = flightNumber;
        this.runway = runway;
        this.arrivalTime = arrivalTime;
        this.departureTime = departureTime;
        this.arrivalAirport = arrivalAirport;
        this.departureAirport = departureAirport;
    }
}

```

3.4 UI Components and Layout

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
import java.util.Date;

public class ATCWindow extends JFrame {
    private JTextField aircraftIdField, flightNumberField, runwayField, arrivalAirportField,
departureAirportField;
    private JSpinner arrivalTimeSpinner, departureTimeSpinner;
    private JTable recordsTable;
    private DefaultTableModel tableModel;

    public ATCWindow() {
        setTitle("ATC Aircraft Logging System");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(900, 600);

        // Setup layout
        setLayout(new BorderLayout());

        // Add background panel
        BackgroundPanel backgroundPanel = new BackgroundPanel();
        setContentPane(backgroundPanel);

        // Add input form and buttons
        JPanel inputPanel = createInputPanel();
        backgroundPanel.add(inputPanel, BorderLayout.EAST);

        // Add records table
        JScrollPane tableScrollPane = createTable();
        backgroundPanel.add(tableScrollPane, BorderLayout.CENTER);

        loadData();
        setVisible(true);
    }

    // Creates and returns the input form panel
    private JPanel createInputPanel() {
        JPanel inputPanel = new JPanel(new GridLayout(9, 2, 10, 10));
        inputPanel.setOpaque(false);
        inputPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

        // Add labels and input fields for Aircraft ID, Flight Number, Runway, etc.
        inputPanel.add(new JLabel("Aircraft ID:"));
        aircraftIdField = new JTextField();
        inputPanel.add(aircraftIdField);
```

```

inputPanel.add(new JLabel("Flight Number:"));
flightNumberField = new JTextField();
inputPanel.add(flightNumberField);

inputPanel.add(new JLabel("Runway:"));
runwayField = new JTextField();
inputPanel.add(runwayField);

inputPanel.add(new JLabel("Arrival Time:"));
arrivalTimeSpinner = new JSpinner(new SpinnerDateModel());
inputPanel.add(arrivalTimeSpinner);

inputPanel.add(new JLabel("Departure Time:"));
departureTimeSpinner = new JSpinner(new SpinnerDateModel());
inputPanel.add(departureTimeSpinner);

inputPanel.add(new JLabel("Arrival Airport:"));
arrivalAirportField = new JTextField();
inputPanel.add(arrivalAirportField);

inputPanel.add(new JLabel("Departure Airport:"));
departureAirportField = new JTextField();
inputPanel.add(departureAirportField);

// Add buttons
JPanel buttonPanel = createButtonPanel();
inputPanel.add(buttonPanel);

return inputPanel;
}

// Creates and returns a panel containing the action buttons
private JPanel createButtonPanel() {
    JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 10, 10));
    buttonPanel.setOpaque(false);

    JButton submitButton = new JButton("Submit");
    submitButton.addActionListener(new SubmitButtonListener());
    buttonPanel.add(submitButton);

    JButton updateButton = new JButton("Update");
    updateButton.addActionListener(new UpdateButtonListener());
    buttonPanel.add(updateButton);

    JButton deleteButton = new JButton("Delete");
    deleteButton.addActionListener(new DeleteButtonListener());
    buttonPanel.add(deleteButton);

    return buttonPanel;
}

```

```

// Creates and returns the table for displaying records
private JScrollPane createTable() {
    tableModel = new DefaultTableModel(new String[]{"Aircraft ID", "Flight Number", "Runway",
"Arrival Time", "Departure Time", "Arrival Airport", "Departure Airport"}, 0);
    recordsTable = new JTable(tableModel);
    recordsTable.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
    recordsTable.setFillsViewportHeight(true);
    recordsTable.setShowGrid(false);
    recordsTable.setRowHeight(30);
    JScrollPane scrollPane = new JScrollPane(recordsTable);
    return scrollPane;
}

// Load data from the database and update the table
private void loadData() {
    try {
        ResultSet resultSet = DatabaseManager.loadData();
        tableModel.setRowCount(0);
        while (resultSet.next()) {
            String aircraftId = resultSet.getString("aircraft_id");
            String flightNumber = resultSet.getString("flight_number");
            String runway = resultSet.getString("runway");
            Date arrivalTime = resultSet.getTimestamp("arrival_time");
            Date departureTime = resultSet.getTimestamp("departure_time");
            String arrivalAirport = resultSet.getString("arrival_airport");
            String departureAirport = resultSet.getString("departure_airport");

            tableModel.addRow(new Object[]{aircraftId, flightNumber, runway, arrivalTime, departureTime,
arrivalAirport, departureAirport});
        }
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(this, "Error loading data: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

// Submit button listener for adding data
private class SubmitButtonListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        String aircraftId = aircraftIdField.getText().trim();
        String flightNumber = flightNumberField.getText().trim();
        String runway = runwayField.getText().trim();
        Date arrivalTime = (Date) arrivalTimeSpinner.getValue();
        Date departureTime = (Date) departureTimeSpinner.getValue();
        String arrivalAirport = arrivalAirportField.getText().trim();
        String departureAirport = departureAirportField.getText().trim();
    }
}

```

```

        if (aircraftId.isEmpty() || flightNumber.isEmpty() || runway.isEmpty() || arrivalAirport.isEmpty() ||
departureAirport.isEmpty()) {
            JOptionPane.showMessageDialog(ATCWindow.this, "Please fill in all fields", "Validation Error",
JOptionPane.ERROR_MESSAGE);
            return;
        }

        try {
            if (DatabaseManager.isFlightNumberExists(flightNumber)) {
                JOptionPane.showMessageDialog(ATCWindow.this, "Flight number already exists!",
"Validation Error", JOptionPane.ERROR_MESSAGE);
                return;
            }

            FlightEntry entry = new FlightEntry(aircraftId, flightNumber, runway, arrivalTime, departureTime,
arrivalAirport, departureAirport);
            DatabaseManager.saveData(entry);
            loadData(); // Reload data to refresh the table
            JOptionPane.showMessageDialog(ATCWindow.this, "Flight record added successfully.");
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(ATCWindow.this, "Error saving data: " + ex.getMessage(),
"Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

```

// Update button listener for editing data
private class UpdateButtonListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        int selectedRow = recordsTable.getSelectedRow();
        if (selectedRow >= 0) {
            String flightNumber = (String) tableModel.getValueAt(selectedRow, 1);
            aircraftIdField.setText((String) tableModel.getValueAt(selectedRow, 0));
            flightNumberField.setText(flightNumber);
            runwayField.setText((String) tableModel.getValueAt(selectedRow, 2));
            arrivalTimeSpinner.setValue(tableModel.getValueAt(selectedRow, 3));
            departureTimeSpinner.setValue(tableModel.getValueAt(selectedRow, 4));
            arrivalAirportField.setText((String) tableModel.getValueAt(selectedRow, 5));
            departureAirportField.setText((String) tableModel.getValueAt(selectedRow, 6));
        } else {
            JOptionPane.showMessageDialog(ATCWindow.this, "Please select a flight record to update",
"Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

```

// Delete button listener for removing data
private class DeleteButtonListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        int selectedRow = recordsTable.getSelectedRow();

```

```

if (selectedRow >= 0) {
    String flightNumber = (String) tableModel.getValueAt(selectedRow, 1);
    try {
        DatabaseManager.deleteData(flightNumber);
        loadData();
        JOptionPane.showMessageDialog(ATCWindow.this, "Flight record deleted successfully.");
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(ATCWindow.this, "Error deleting record: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
    } else {
        JOptionPane.showMessageDialog(ATCWindow.this, "Please select a flight record to delete",
"Error", JOptionPane.ERROR_MESSAGE);
    }
}
}
}

```

3.5 Background Panel

```

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

public class BackgroundPanel extends JPanel {
    private Image backgroundImage;

    public BackgroundPanel() {
        backgroundImage = new ImageIcon("KAIA-ATCtower.jpg").getImage(); // Replace with the correct
path to your image
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);
    }
}

```

3.6 Main Application

```

public class Main {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new ATCWindow());
    }
}

```

SNAPSHOTS

THE MENU:

ATC Aircraft Logging System

Aircraft ID	Flight Number	Runway	Arrival Time	Departure Time	Arrival Airport	Departure Airport

Aircraft ID:

Flight Number:

Runway:

Arrival Time:

2024-11-22 18:00:35

Departure Time:

2024-11-22 18:00:35

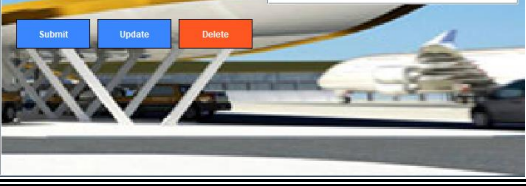
Arrival Airport:

Departure Airport:

Submit

Update

Delete



TO ENTER DATA

ENTER THE DATAS IN THE DEIGNATED SLOTS

Aircraft ID:

SV

Flight Number:

672

Runway:

34L

Arrival Time:

2024-11-22 18:00:35

Departure Time:

2024-11-22 18:00:35

Arrival Airport:

JED

Departure Airport:

MAA

Submit

Update

Delete

AFTER DATAS ARE ENTERED IT IS UPDATED LIKE SUCH

Aircraft ID	Flight Number	Runway	Arrival Time	Departure Time	Arrival Airport	Departure Airport
SV	672	34L	2024-11-22 18:00:00.0	2024-11-22 18:00:35.0	JED	MAA

ERROR HANDLING:

IF DATA SLOT(S) ARE EMPTY

Aircraft ID	Flight Number	Runway	Arrival Time	Departure Time	Arrival Airport	Departure Airport
sv	672	34L	2024-11-22 18:00:00.0	2024-11-22 18:00:35.0	JED	MAA

Validation Error

Please fill all fields.

OK

Aircraft ID: sv

Flight Number: 672

Runway: 34L

Arrival Time: 2024-11-22 18:00:35

Departure Time: 2024-11-22 18:00:35

Arrival Airport: JED

Departure Airport: MAA

Submit Update Delete

DUPLICATE DATA

Aircraft ID	Flight Number	Runway	Arrival Time	Departure Time	Arrival Airport	Departure Airport
sv	672	34L	2024-11-22 18:00:00.0	2024-11-22 18:00:35.0	JED	MAA

Duplicate Data

Error: Duplicate flight number!

OK

Aircraft ID: sv

Flight Number: 672

Runway: 34L

Arrival Time: 2024-11-22 18:00:35

Departure Time: 2024-11-22 18:00:35

Arrival Airport: JED

Departure Airport: MAA

Submit Update Delete

IF ARRIVAL TIME IS SAME IT ADDS 5 MINUTES TO IT

sv	672	34L	2024-11-22 18:00:00.0	2024-11-22 18:00:35.0	JED	MAA
ai	873	34L	2024-11-22 18:05:00.0	2024-11-22 18:00:35.0	JED	MAA

UPDATE DATA

Select the record and change the desired data

Aircraft ID	Flight Number	Runway	Arrival Time	Departure Time	Arrival Airport	Departure Airport
sv	672	34L	2024-11-22 18:00:00.0	2024-11-22 18:00:35.0	JED	MAA
ai	873	34L	2024-11-22 18:05:00.0	2024-11-22 18:00:35.0	JED	MAA

sv	672	34L	2024-11-22 18:00:00.0	2024-11-22 18:00:35.0	JED	MAA
am	873	34L	2024-11-22 18:05:00.0	2024-11-22 18:00:35.0	JED	MAA

DELETE DATA

Select the data you want to delete and press delete

Aircraft ID	Flight Number	Runway	Arrival Time	Departure Time	Arrival Airport	Departure Airport
sv	672	34L	2024-11-22 18:00:00.0	2024-11-22 18:00:35.0	JED	MAA
ai	873	34L	2024-11-22 18:05:00.0	2024-11-22 18:00:35.0	JED	MAA

Aircraft ID	Flight Number	Runway	Arrival Time	Departure Time	Arrival Airport	Departure Airport
sv	873	34L	2024-11-22 18:05:00.0	2024-11-22 18:00:35.0	JED	MAA

Aircraft ID:

sv

Flight Number:

873

Runway:

34L

Arrival Time:

2024-11-22 18:00:35

Departure Time:

2024-11-22 18:00:35

Arrival Airport:

JED

Departure Airport:

MAA

Submit

Update

Delete

CONCLUSION

In conclusion, the ATC Aircraft Logging System provides an easy and efficient way to manage aircraft data. It allows air traffic controllers to add, update, and delete flight records while handling issues like arrival time conflicts. The system also prevents duplicate flight numbers, ensuring accurate data. Overall, this project serves as a useful tool for tracking flight operations and can be further enhanced in the future with additional features like real-time tracking or integration with other systems.

REFERENCES

<https://www.geeksforgeeks.org/java-swing/>
<https://www.tutorialspoint.com/jdbc/index.htm>
<https://www.geeksforgeeks.org/java-sql-jdbc/>
<https://www.w3schools.com/sql/>
<https://www.geeksforgeeks.org/javax-swing-jtable-in-java/>
<https://www.geeksforgeeks.org/java/>