MSA UNIVERSITY
IN EGYPT SINCE 1996
Established by Dr.Nawal El Degwi

OCTOBER UNIVERISTY
FOR MODERN SCIENCES AND ARTS
جامعة أكتوبر للعلوم الحديثة والآداب

COMPUTER SYSTEMS ENGINEERING

School Management System

Student Name :Farah Mohamed Ahmed Habib ID :243897

Student Name :Sarah Mohamed Ali ID :248097

Student Name :Toni Wagdi Wadie ID:244743

Student Name :Ahmed Hisham Mohsen ID :242435

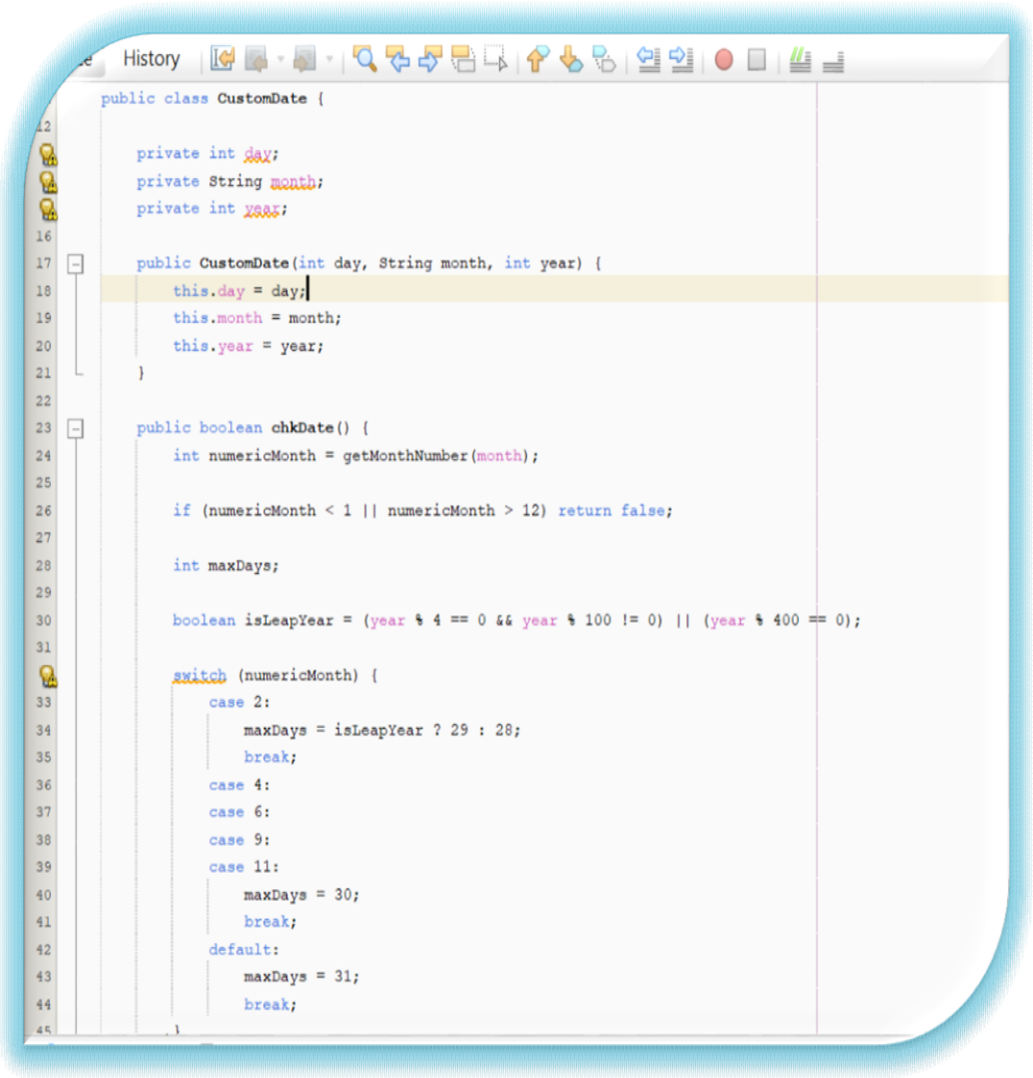
Student Name :Youssef Mohamed ID :234833

Presented to:Eng. Gehad Ehab

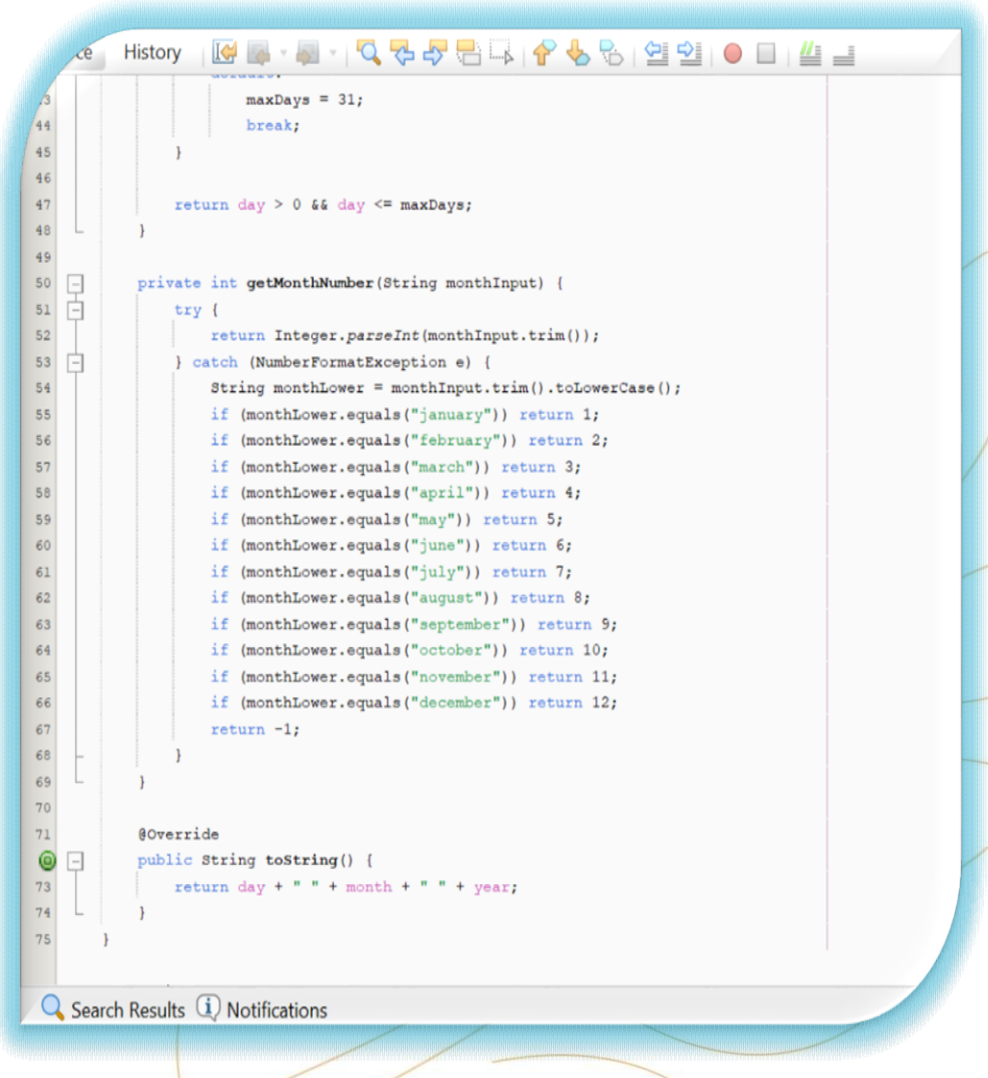
1.CustomDate

- **Role:** Validates date inputs with leap year logic and month formats.

- **Purpose:** Ensures that all date entries are accurately formatted and consistently stored in the database, maintaining data integrity and reliability across the entire application.



```java
public class CustomDate {

    private int day;
    private String month;
    private int year;

    public CustomDate(int day, String month, int year) {
        this.day = day;
        this.month = month;
        this.year = year;
    }

    public boolean chkDate() {
        int numericMonth = getMonthNumber(month);

        if (numericMonth < 1 || numericMonth > 12) return false;

        int maxDays;

        boolean isLeapYear = (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);

        switch (numericMonth) {
            case 2:
                maxDays = isLeapYear ? 29 : 28;
                break;
            case 4:
            case 6:
            case 9:
            case 11:
                maxDays = 30;
                break;
            default:
                maxDays = 31;
                break;
```

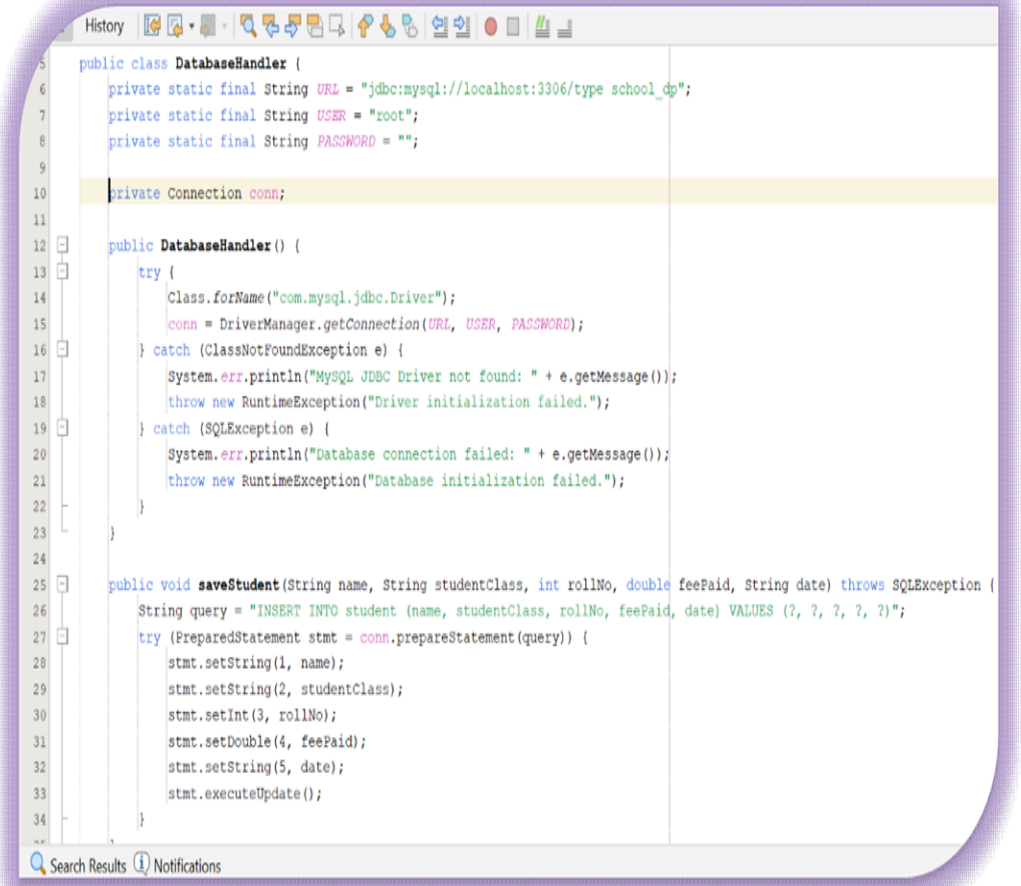


```java
                maxDays = 31;
                break;
        }

        return day > 0 && day <= maxDays;
    }

    private int getMonthNumber(String monthInput) {
        try {
            return Integer.parseInt(monthInput.trim());
        } catch (NumberFormatException e) {
            String monthLower = monthInput.trim().toLowerCase();
            if (monthLower.equals("january")) return 1;
            if (monthLower.equals("february")) return 2;
            if (monthLower.equals("march")) return 3;
            if (monthLower.equals("april")) return 4;
            if (monthLower.equals("may")) return 5;
            if (monthLower.equals("june")) return 6;
            if (monthLower.equals("july")) return 7;
            if (monthLower.equals("august")) return 8;
            if (monthLower.equals("september")) return 9;
            if (monthLower.equals("october")) return 10;
            if (monthLower.equals("november")) return 11;
            if (monthLower.equals("december")) return 12;
            return -1;
        }
    }

    @Override
    public String toString() {
        return day + " " + month + " " + year;
    }
}
```

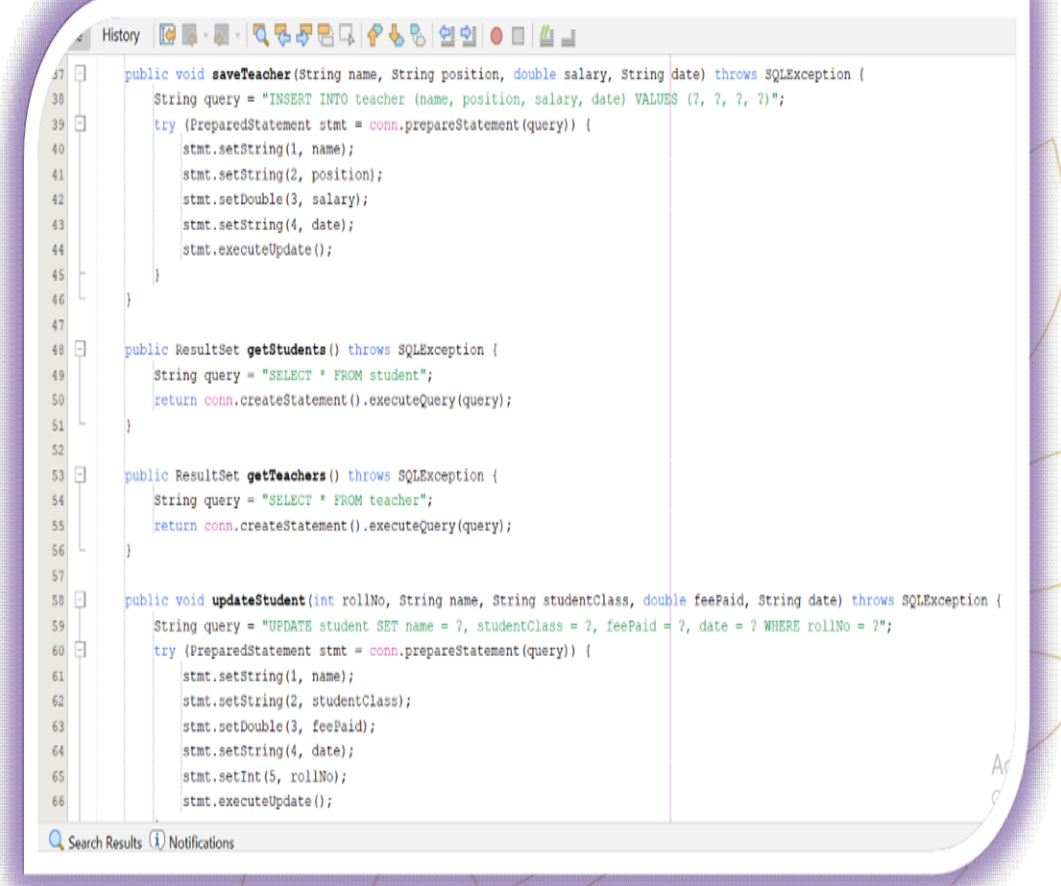
2. DatabaseHandler

- **Role:** Manages database connectivity and performs CRUD operations.

- **Purpose:** Facilitates secure and efficient interactions with the MySQL database through JDBC, enabling the system to handle data operations while protecting against vulnerabilities like SQL injection.

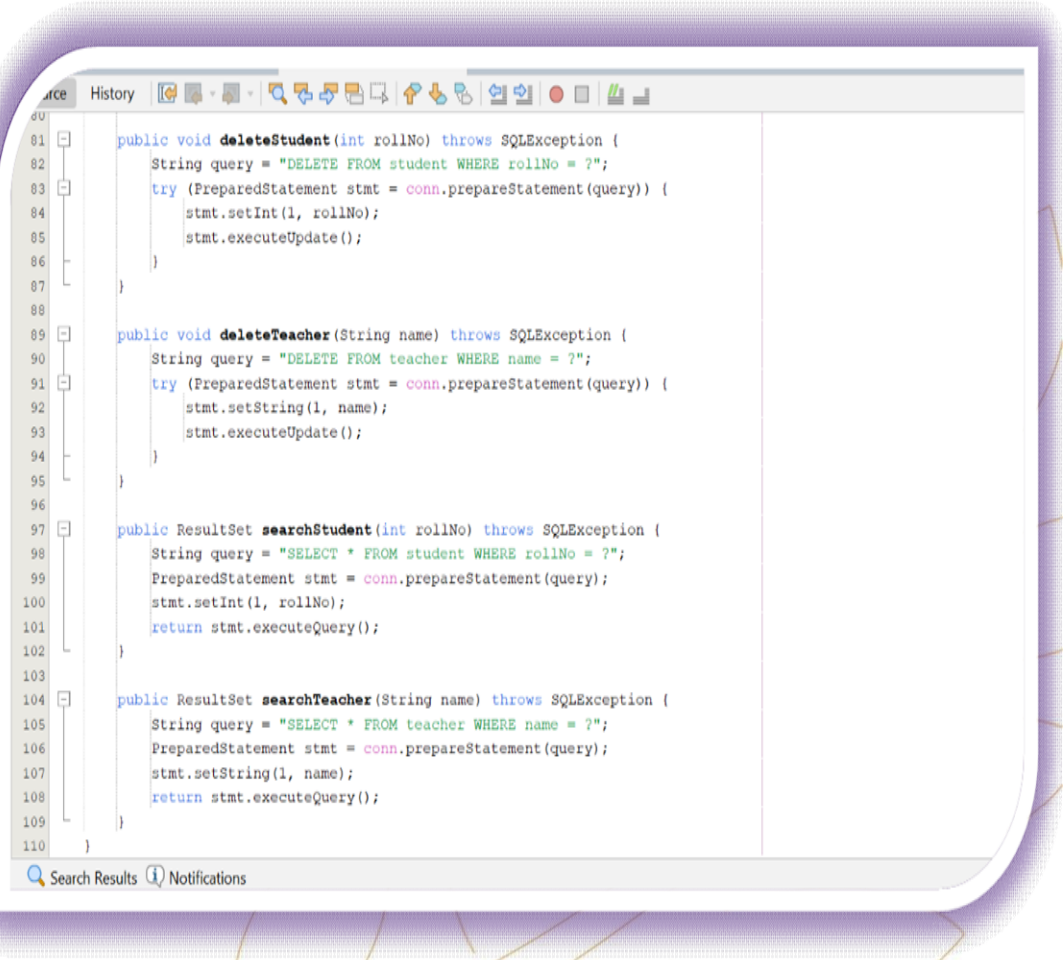


```
public class DatabaseHandler {
    private static final String URL = "jdbc:mysql://localhost:3306/type school_db";
    private static final String USER = "root";
    private static final String PASSWORD = "";

    private Connection conn;

    public DatabaseHandler() {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            conn = DriverManager.getConnection(URL, USER, PASSWORD);
        } catch (ClassNotFoundException e) {
            System.err.println("MySQL JDBC Driver not found: " + e.getMessage());
            throw new RuntimeException("Driver initialization failed.");
        } catch (SQLException e) {
            System.err.println("Database connection failed: " + e.getMessage());
            throw new RuntimeException("Database initialization failed.");
        }
    }

    public void saveStudent(String name, String studentClass, int rollNo, double feePaid, String date) throws SQLException {
        String query = "INSERT INTO student (name, studentClass, rollNo, feePaid, date) VALUES (?, ?, ?, ?, ?)";
        try (PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setString(1, name);
            stmt.setString(2, studentClass);
            stmt.setInt(3, rollNo);
            stmt.setDouble(4, feePaid);
            stmt.setString(5, date);
            stmt.executeUpdate();
        }
```



```
    public void saveTeacher(String name, String position, double salary, String date) throws SQLException {
        String query = "INSERT INTO teacher (name, position, salary, date) VALUES (?, ?, ?, ?)";
        try (PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setString(1, name);
            stmt.setString(2, position);
            stmt.setDouble(3, salary);
            stmt.setString(4, date);
            stmt.executeUpdate();
        }
    }

    public ResultSet getStudents() throws SQLException {
        String query = "SELECT * FROM student";
        return conn.createStatement().executeQuery(query);
    }

    public ResultSet getTeachers() throws SQLException {
        String query = "SELECT * FROM teacher";
        return conn.createStatement().executeQuery(query);
    }

    public void updateStudent(int rollNo, String name, String studentClass, double feePaid, String date) throws SQLException {
        String query = "UPDATE student SET name = ?, studentClass = ?, feePaid = ?, date = ? WHERE rollNo = ?";
        try (PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setString(1, name);
            stmt.setString(2, studentClass);
            stmt.setDouble(3, feePaid);
            stmt.setString(4, date);
            stmt.setInt(5, rollNo);
            stmt.executeUpdate();
```

```
History
70  public void updateTeacher(String name, String position, double salary, String date) throws SQLException {
71      String query = "UPDATE teacher SET position = ?, salary = ?, date = ? WHERE name = ?";
72      try (PreparedStatement stmt = conn.prepareStatement(query)) {
73          stmt.setString(1, position);
74          stmt.setDouble(2, salary);
75          stmt.setString(3, date);
76          stmt.setString(4, name);
77          stmt.executeUpdate();
78      }
79  }
80
81  public void deleteStudent(int rollNo) throws SQLException {
82      String query = "DELETE FROM student WHERE rollNo = ?";
83      try (PreparedStatement stmt = conn.prepareStatement(query)) {
84          stmt.setInt(1, rollNo);
85          stmt.executeUpdate();
86      }
87  }
88
89  public void deleteTeacher(String name) throws SQLException {
90      String query = "DELETE FROM teacher WHERE name = ?";
91      try (PreparedStatement stmt = conn.prepareStatement(query)) {
92          stmt.setString(1, name);
93          stmt.executeUpdate();
94      }
95  }
96
97  public ResultSet searchStudent(int rollNo) throws SQLException {
98      String query = "SELECT * FROM student WHERE rollNo = ?";
99      PreparedStatement stmt = conn.prepareStatement(query);
Search Results  Notifications
```

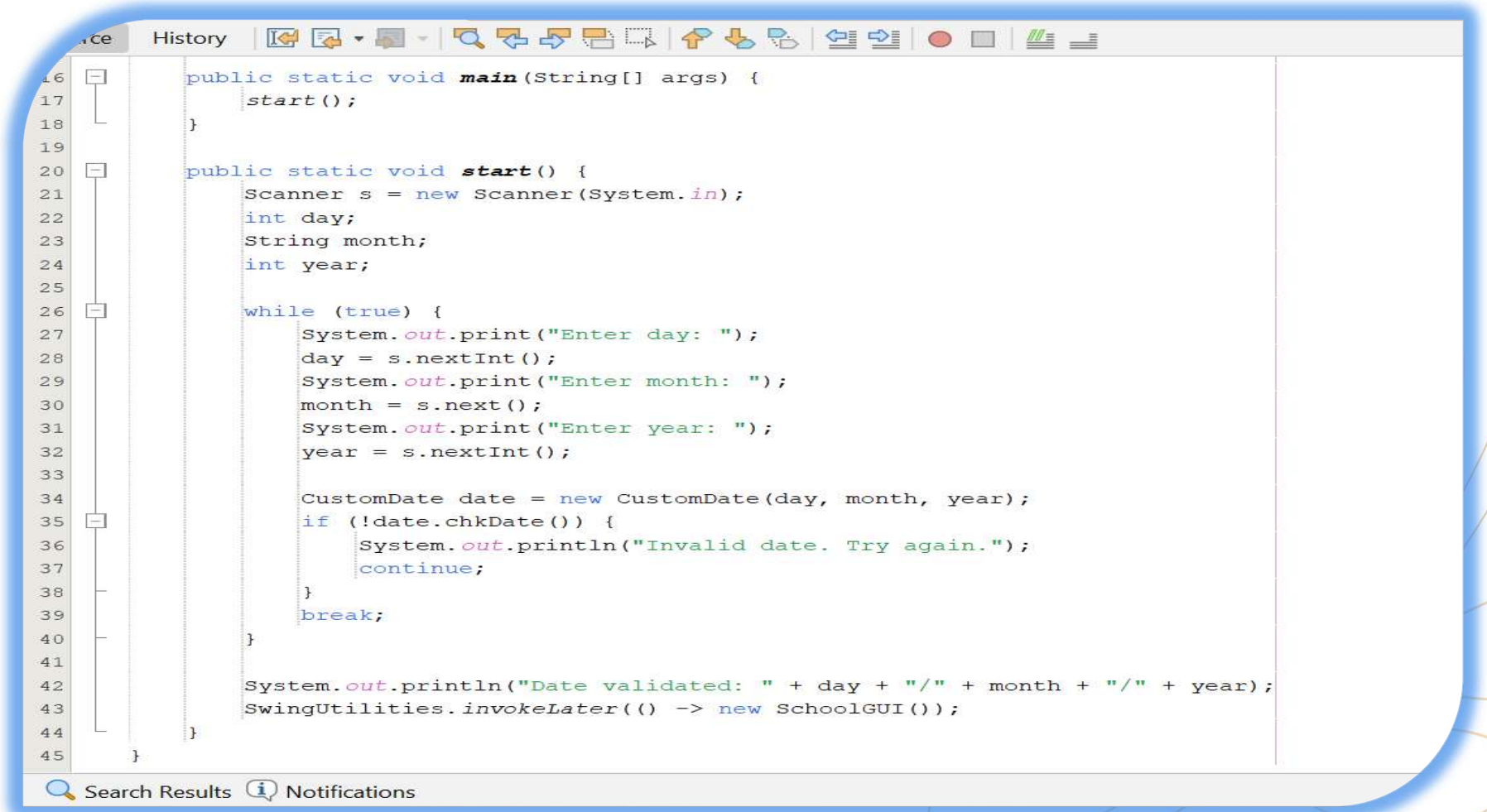


```
History
81  public void deleteStudent(int rollNo) throws SQLException {
82      String query = "DELETE FROM student WHERE rollNo = ?";
83      try (PreparedStatement stmt = conn.prepareStatement(query)) {
84          stmt.setInt(1, rollNo);
85          stmt.executeUpdate();
86      }
87  }
88
89  public void deleteTeacher(String name) throws SQLException {
90      String query = "DELETE FROM teacher WHERE name = ?";
91      try (PreparedStatement stmt = conn.prepareStatement(query)) {
92          stmt.setString(1, name);
93          stmt.executeUpdate();
94      }
95  }
96
97  public ResultSet searchStudent(int rollNo) throws SQLException {
98      String query = "SELECT * FROM student WHERE rollNo = ?";
99      PreparedStatement stmt = conn.prepareStatement(query);
100     stmt.setInt(1, rollNo);
101     return stmt.executeQuery();
102 }
103
104 public ResultSet searchTeacher(String name) throws SQLException {
105     String query = "SELECT * FROM teacher WHERE name = ?";
106     PreparedStatement stmt = conn.prepareStatement(query);
107     stmt.setString(1, name);
108     return stmt.executeQuery();
109 }
110 }
Search Results  Notifications
```

3. SchoolSystemGSE122i

- **Role:** Acts as the entry point with console-based date validation.

- **Purpose:** Validates the initial date input provided by the user and subsequently launches the graphical user interface, ensuring the application starts with a verified foundation.

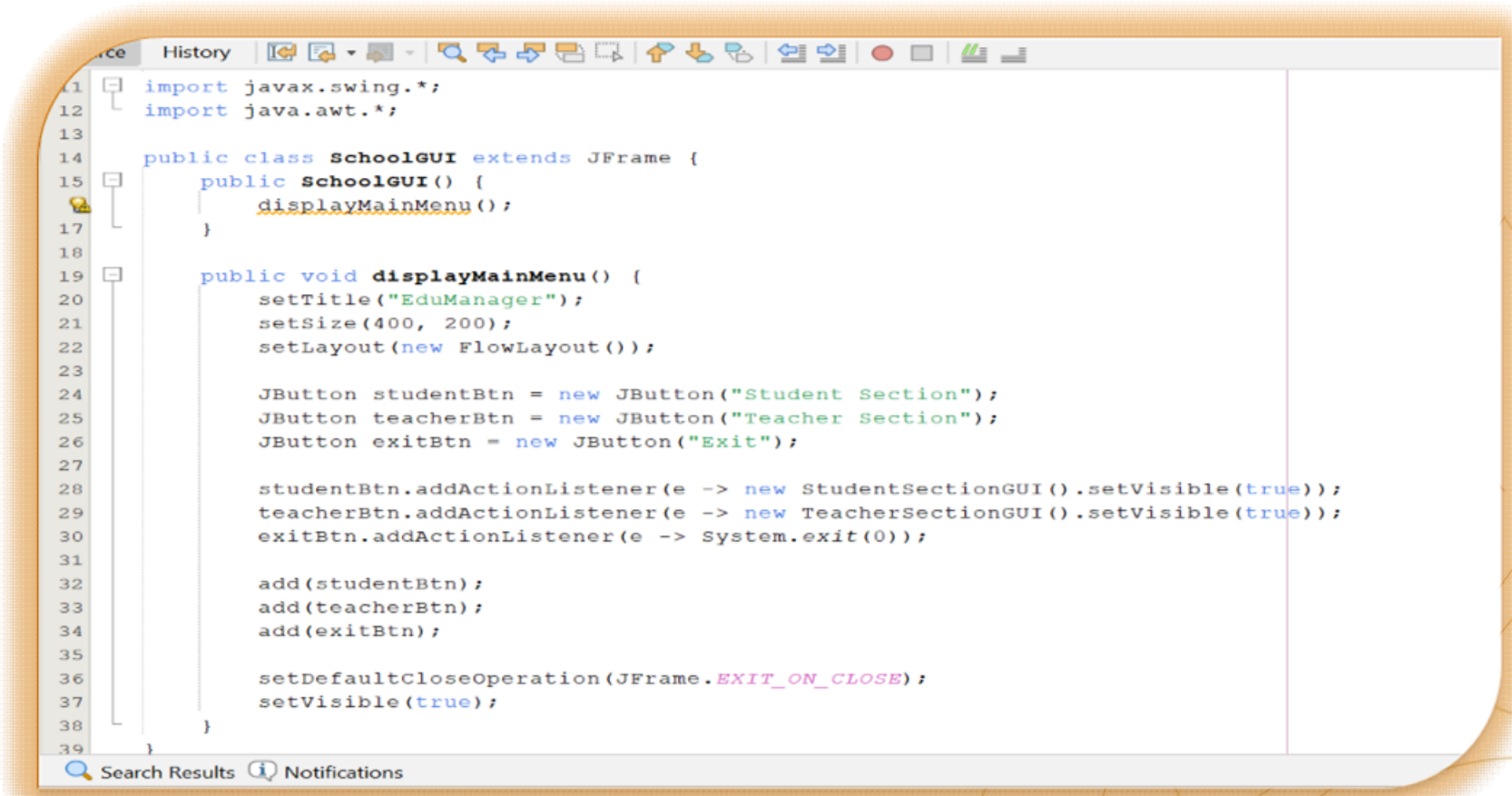



```
    public static void main(String[] args) {
        start();
    }

    public static void start() {
        Scanner s = new Scanner(System.in);
        int day;
        String month;
        int year;

        while (true) {
            System.out.print("Enter day: ");
            day = s.nextInt();
            System.out.print("Enter month: ");
            month = s.next();
            System.out.print("Enter year: ");
            year = s.nextInt();

            CustomDate date = new CustomDate(day, month, year);
            if (!date.chkDate()) {
                System.out.println("Invalid date. Try again.");
                continue;
            }
            break;
        }

        System.out.println("Date validated: " + day + "/" + month + "/" + year);
        SwingUtilities.invokeLater(() -> new SchoolGUI());
    }
}
```

4. SchoolGUI

- **Role:** Displays the main menu with navigation buttons.

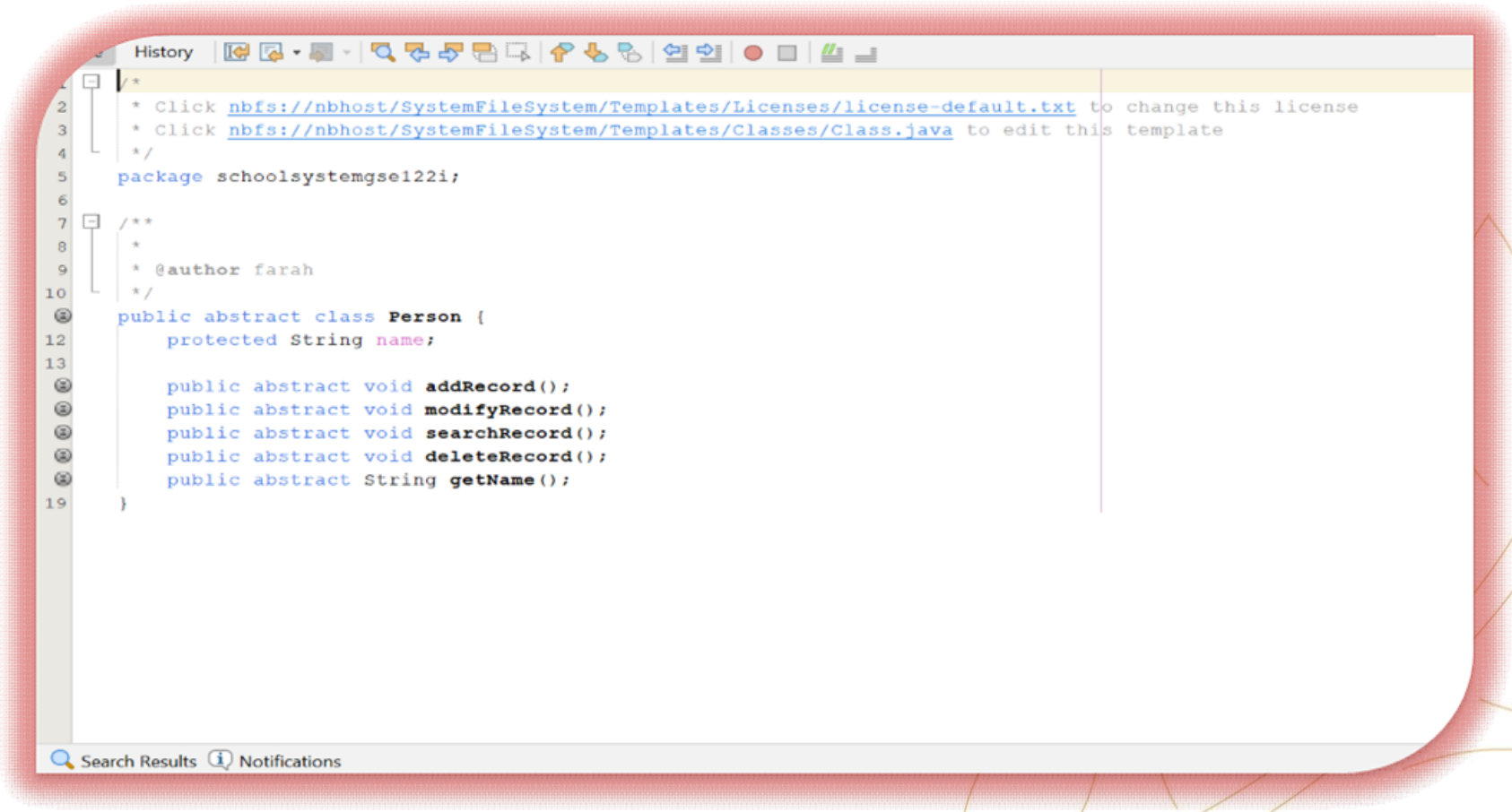
- **Purpose:** Guides users to access either the student or teacher management sections or exit the application, providing an intuitive navigation experience to enhance usability.


History
import javax.swing.*;
import java.awt.*;
public class SchoolGUI extends JFrame {
public SchoolGUI() {
displayMainMenu();
}
public void displayMainMenu() {
setTitle("EduManager");
setSize(400, 200);
setLayout(new FlowLayout());
JButton studentBtn = new JButton("Student Section");
JButton teacherBtn = new JButton("Teacher Section");
JButton exitBtn = new JButton("Exit");
studentBtn.addActionListener(e -> new StudentSectionGUI().setVisible(true));
teacherBtn.addActionListener(e -> new TeacherSectionGUI().setVisible(true));
exitBtn.addActionListener(e -> System.exit(0));
add(studentBtn);
add(teacherBtn);
add(exitBtn);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setVisible(true);
}
}
Search Results
Notifications

5. Person

- **Role:** Defines an abstract superclass for record management.

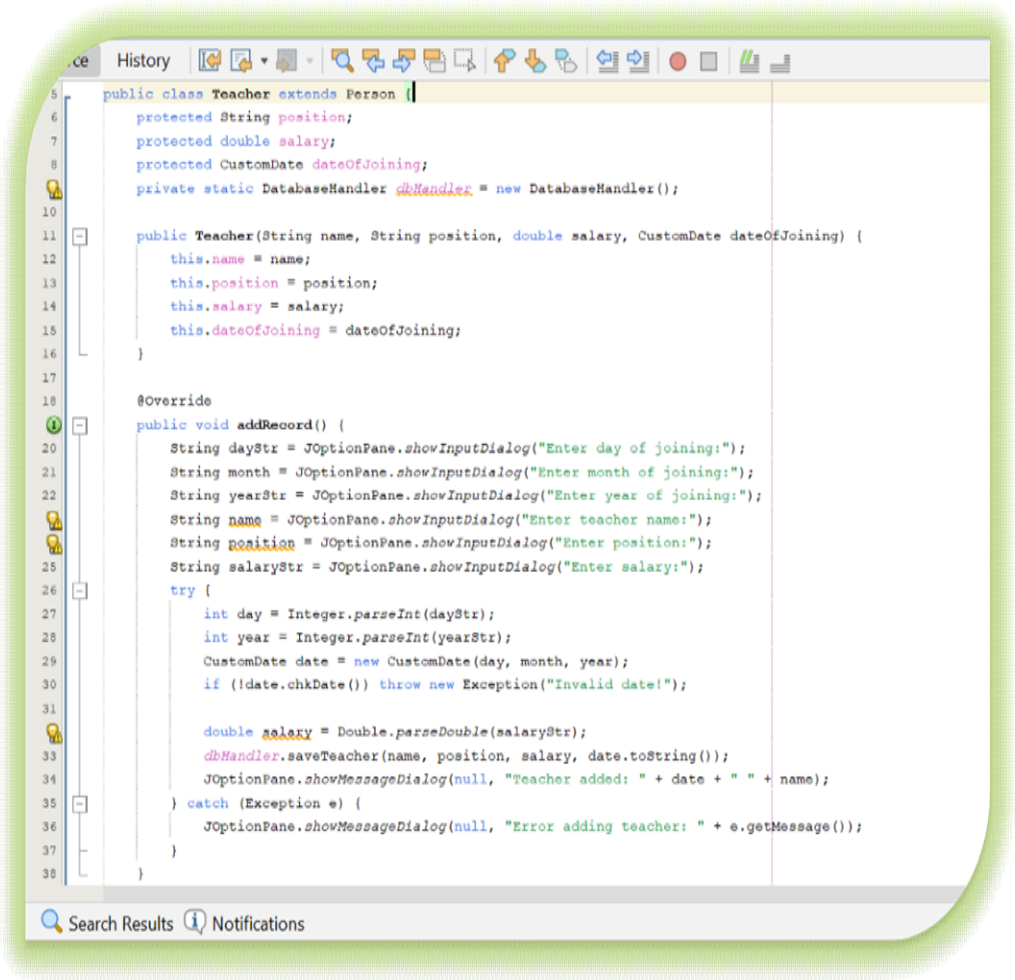
- **Purpose:** Ensures consistency in the implementation of record management operations across all derived classes, promoting a standardized approach in the object-oriented design.


History
/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
*/
package schoolsystemgse122i;
/**
*
* @author farah
*/
public abstract class Person {
protected String name;
public abstract void addRecord();
public abstract void modifyRecord();
public abstract void searchRecord();
public abstract void deleteRecord();
public abstract String getName();
}
Search Results
Notifications

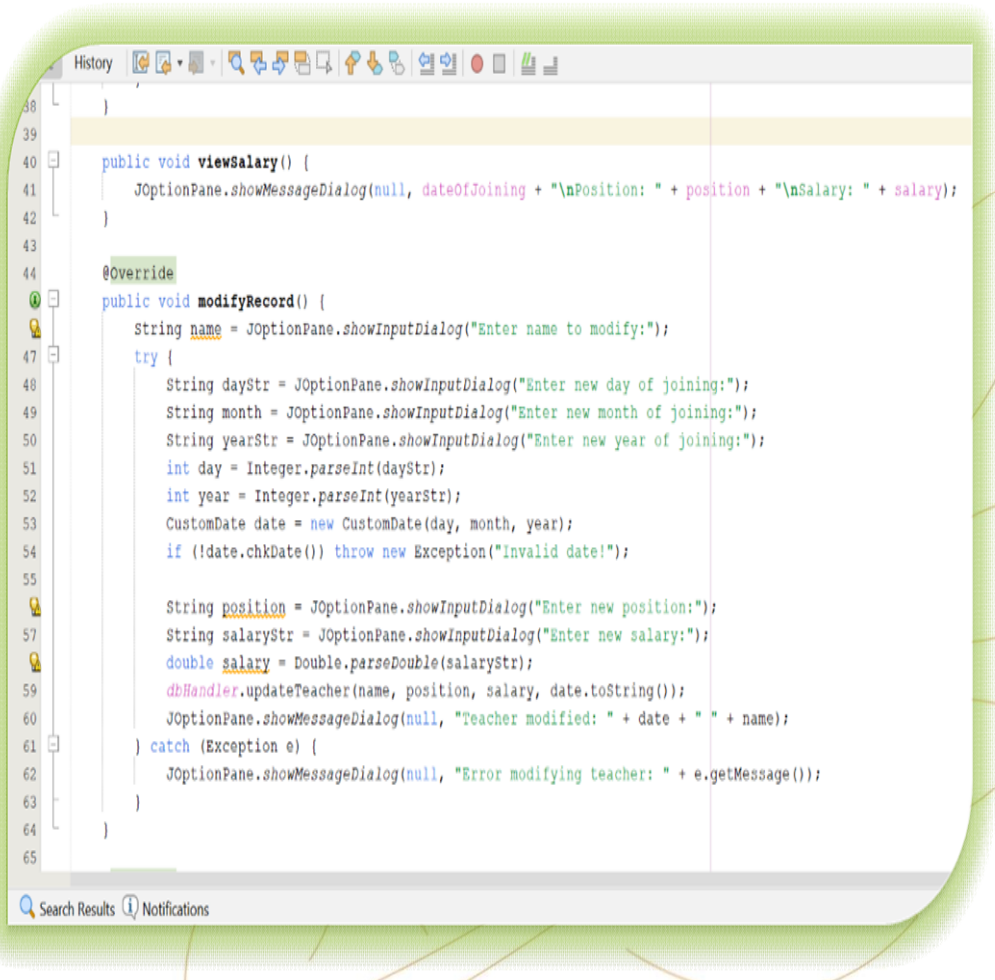
6. Teacher

- **Role:** Manages teacher records, including saving and modifying.

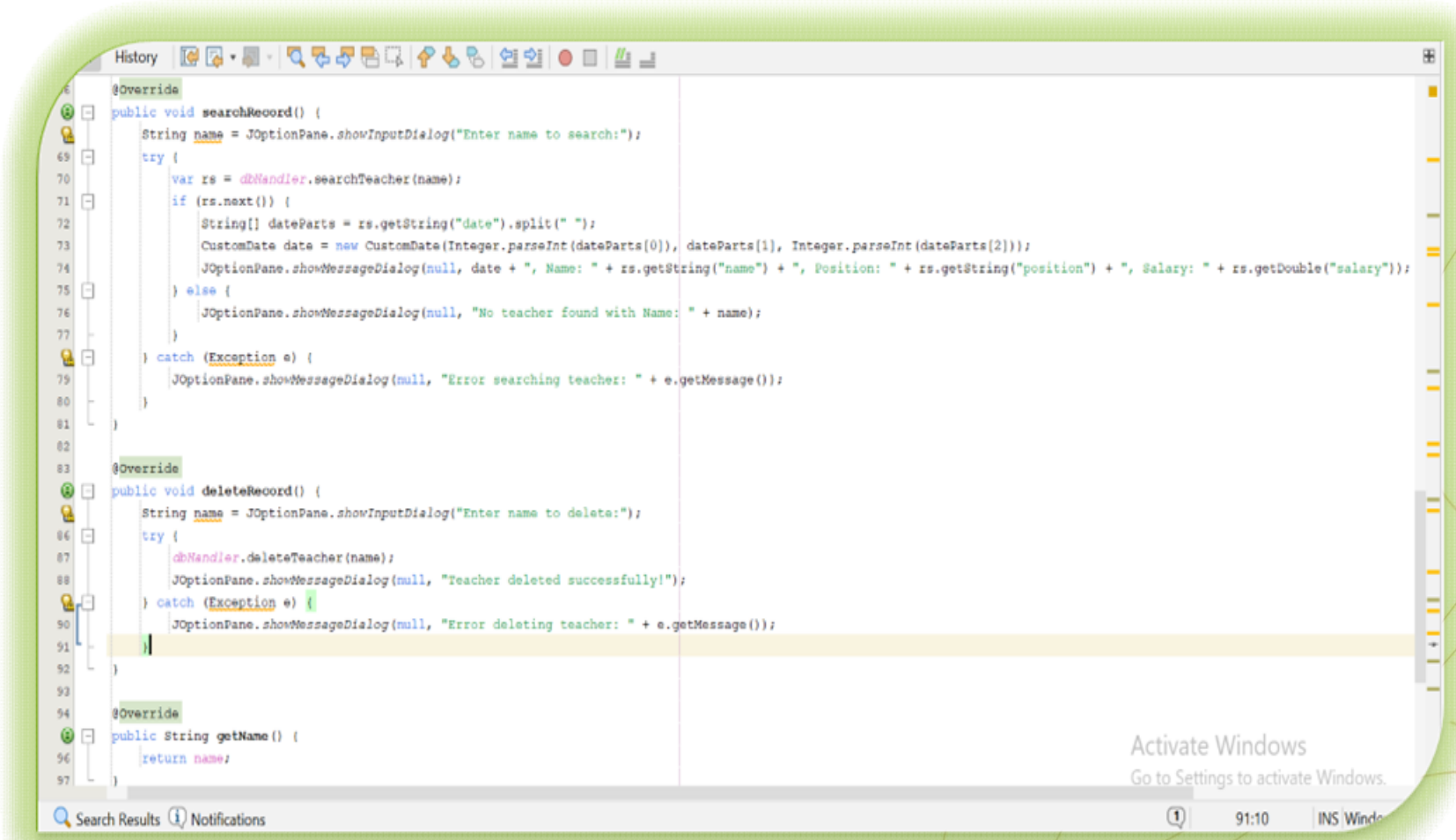
- **Purpose:** Handles the storage, retrieval, and modification of teacher data in the database, supporting administrative tasks like salary viewing and record updates with precision.

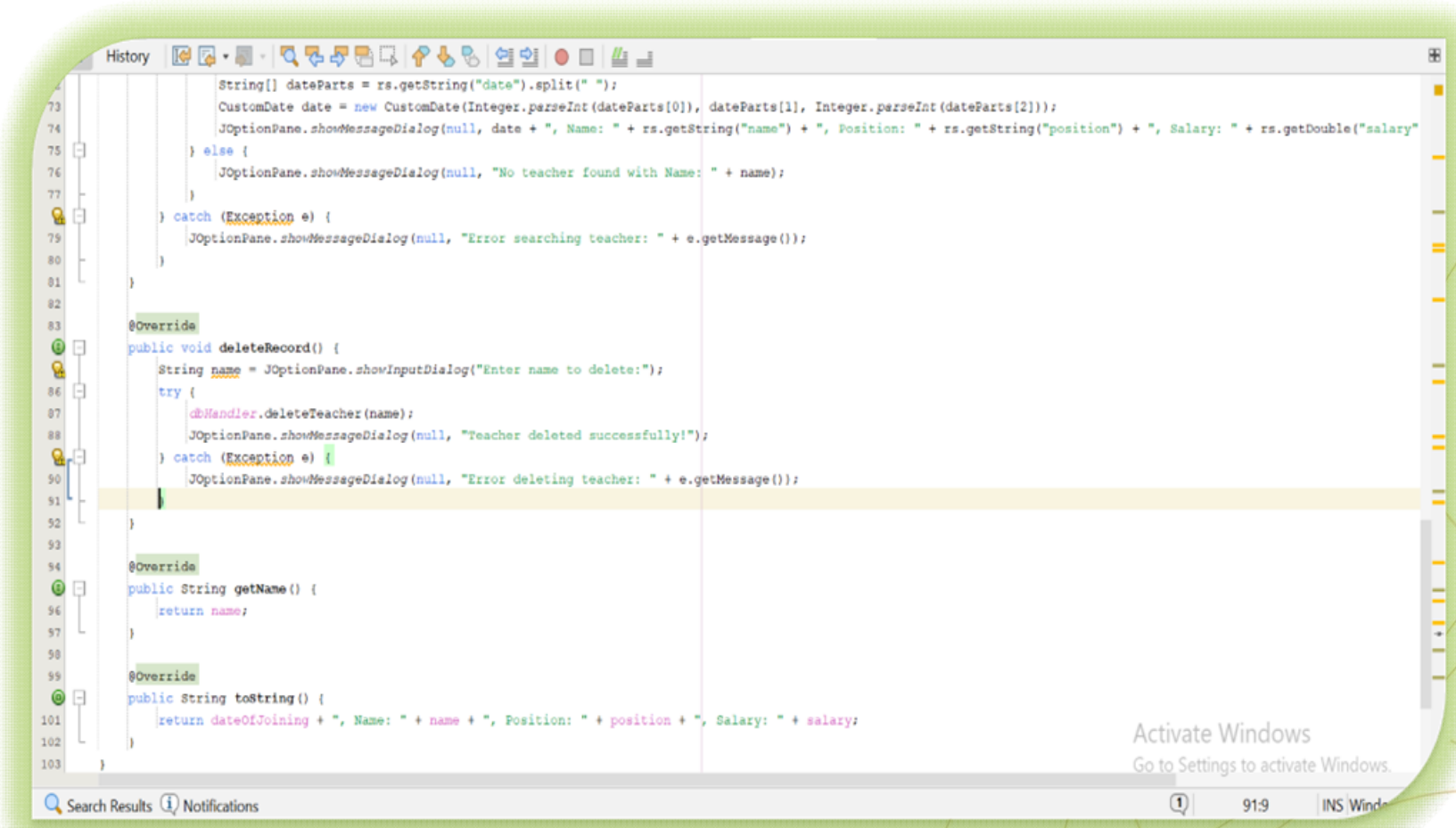


```java
public class Teacher extends Person {
    protected String position;
    protected double salary;
    protected CustomDate dateOfJoining;
    private static DatabaseHandler dbHandler = new DatabaseHandler();

    public Teacher(String name, String position, double salary, CustomDate dateOfJoining) {
        this.name = name;
        this.position = position;
        this.salary = salary;
        this.dateOfJoining = dateOfJoining;
    }

    @Override
    public void addRecord() {
        String dayStr = JOptionPane.showInputDialog("Enter day of joining:");
        String month = JOptionPane.showInputDialog("Enter month of joining:");
        String yearStr = JOptionPane.showInputDialog("Enter year of joining:");
        String name = JOptionPane.showInputDialog("Enter teacher name:");
        String position = JOptionPane.showInputDialog("Enter position:");
        String salaryStr = JOptionPane.showInputDialog("Enter salary:");
        try {
            int day = Integer.parseInt(dayStr);
            int year = Integer.parseInt(yearStr);
            CustomDate date = new CustomDate(day, month, year);
            if (!date.chkDate()) throw new Exception("Invalid date!");

            double salary = Double.parseDouble(salaryStr);
            dbHandler.saveTeacher(name, position, salary, date.toString());
            JOptionPane.showMessageDialog(null, "Teacher added: " + date + " " + name);
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Error adding teacher: " + e.getMessage());
        }
    }
```



```java
    }

    public void viewSalary() {
        JOptionPane.showMessageDialog(null, dateOfJoining + "\nPosition: " + position + "\nSalary: " + salary);
    }

    @Override
    public void modifyRecord() {
        String name = JOptionPane.showInputDialog("Enter name to modify:");
        try {
            String dayStr = JOptionPane.showInputDialog("Enter new day of joining:");
            String month = JOptionPane.showInputDialog("Enter new month of joining:");
            String yearStr = JOptionPane.showInputDialog("Enter new year of joining:");
            int day = Integer.parseInt(dayStr);
            int year = Integer.parseInt(yearStr);
            CustomDate date = new CustomDate(day, month, year);
            if (!date.chkDate()) throw new Exception("Invalid date!");

            String position = JOptionPane.showInputDialog("Enter new position:");
            String salaryStr = JOptionPane.showInputDialog("Enter new salary:");
            double salary = Double.parseDouble(salaryStr);
            dbHandler.updateTeacher(name, position, salary, date.toString());
            JOptionPane.showMessageDialog(null, "Teacher modified: " + date + " " + name);
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Error modifying teacher: " + e.getMessage());
        }
    }
```

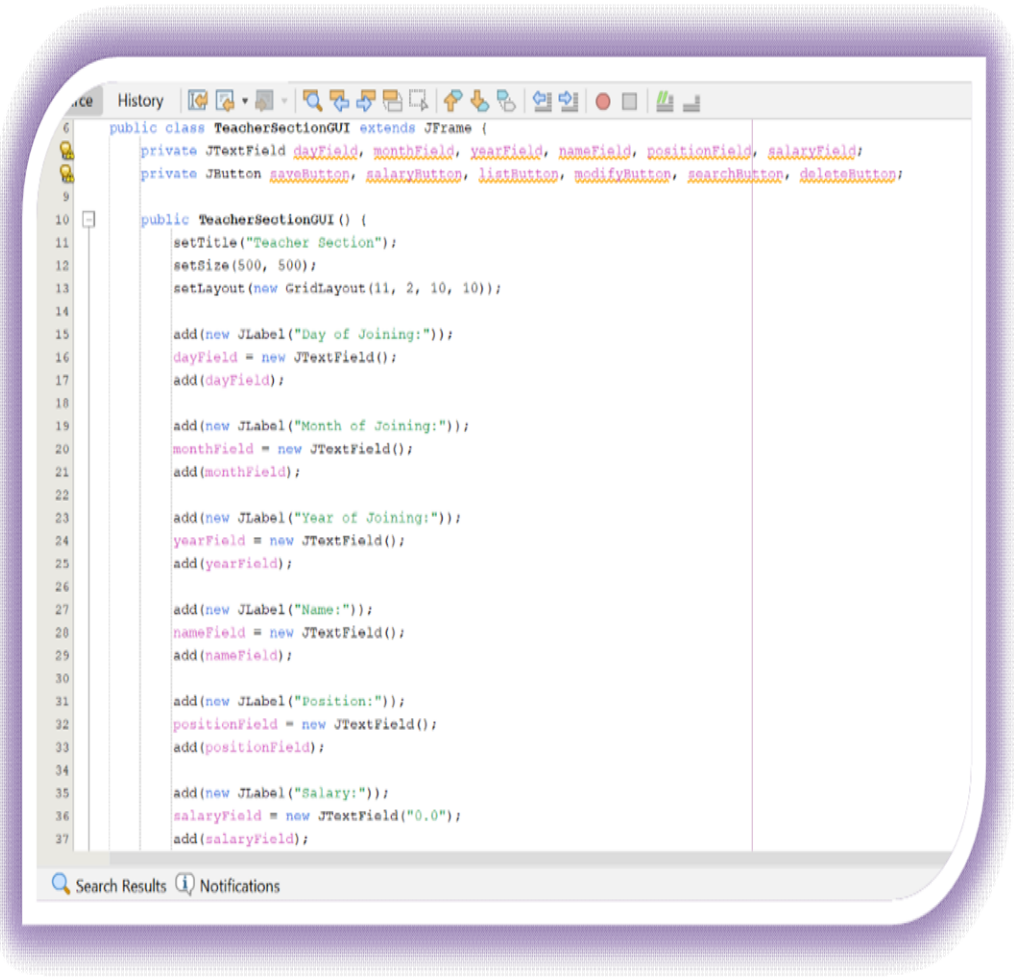

History
@Override
public void searchRecord() {
String name = JOptionPane.showInputDialog("Enter name to search:");
try {
var rs = dbHandler.searchTeacher(name);
if (rs.next()) {
String[] dateParts = rs.getString("date").split(" ");
CustomDate date = new CustomDate(Integer.parseInt(dateParts[0]), dateParts[1], Integer.parseInt(dateParts[2]));
JOptionPane.showMessageDialog(null, date + ", Name: " + rs.getString("name") + ", Position: " + rs.getString("position") + ", Salary: " + rs.getDouble("salary"));
} else {
JOptionPane.showMessageDialog(null, "No teacher found with Name: " + name);
}
} catch (Exception e) {
JOptionPane.showMessageDialog(null, "Error searching teacher: " + e.getMessage());
}
}
@Override
public void deleteRecord() {
String name = JOptionPane.showInputDialog("Enter name to delete:");
try {
dbHandler.deleteTeacher(name);
JOptionPane.showMessageDialog(null, "Teacher deleted successfully!");
} catch (Exception e) {
JOptionPane.showMessageDialog(null, "Error deleting teacher: " + e.getMessage());
}
}
@Override
public String getName() {
return name;
}
Activate Windows
Go to Settings to activate Windows.
Search Results
Notifications
91:10
INS


History
String[] dateParts = rs.getString("date").split(" ");
CustomDate date = new CustomDate(Integer.parseInt(dateParts[0]), dateParts[1], Integer.parseInt(dateParts[2]));
JOptionPane.showMessageDialog(null, date + ", Name: " + rs.getString("name") + ", Position: " + rs.getString("position") + ", Salary: " + rs.getDouble("salary"
} else {
JOptionPane.showMessageDialog(null, "No teacher found with Name: " + name);
} catch (Exception e) {
JOptionPane.showMessageDialog(null, "Error searching teacher: " + e.getMessage());
@Override
public void deleteRecord() {
String name = JOptionPane.showInputDialog("Enter name to delete:");
try {
dbHandler.deleteTeacher(name);
JOptionPane.showMessageDialog(null, "Teacher deleted successfully!");
} catch (Exception e) {
JOptionPane.showMessageDialog(null, "Error deleting teacher: " + e.getMessage());
@Override
public String getName() {
return name;
@Override
public String toString() {
return dateOfJoining + ", Name: " + name + ", Position: " + position + ", Salary: " + salary;
Activate Windows
Go to Settings to activate Windows.
Search Results
Notifications
91:9
INS

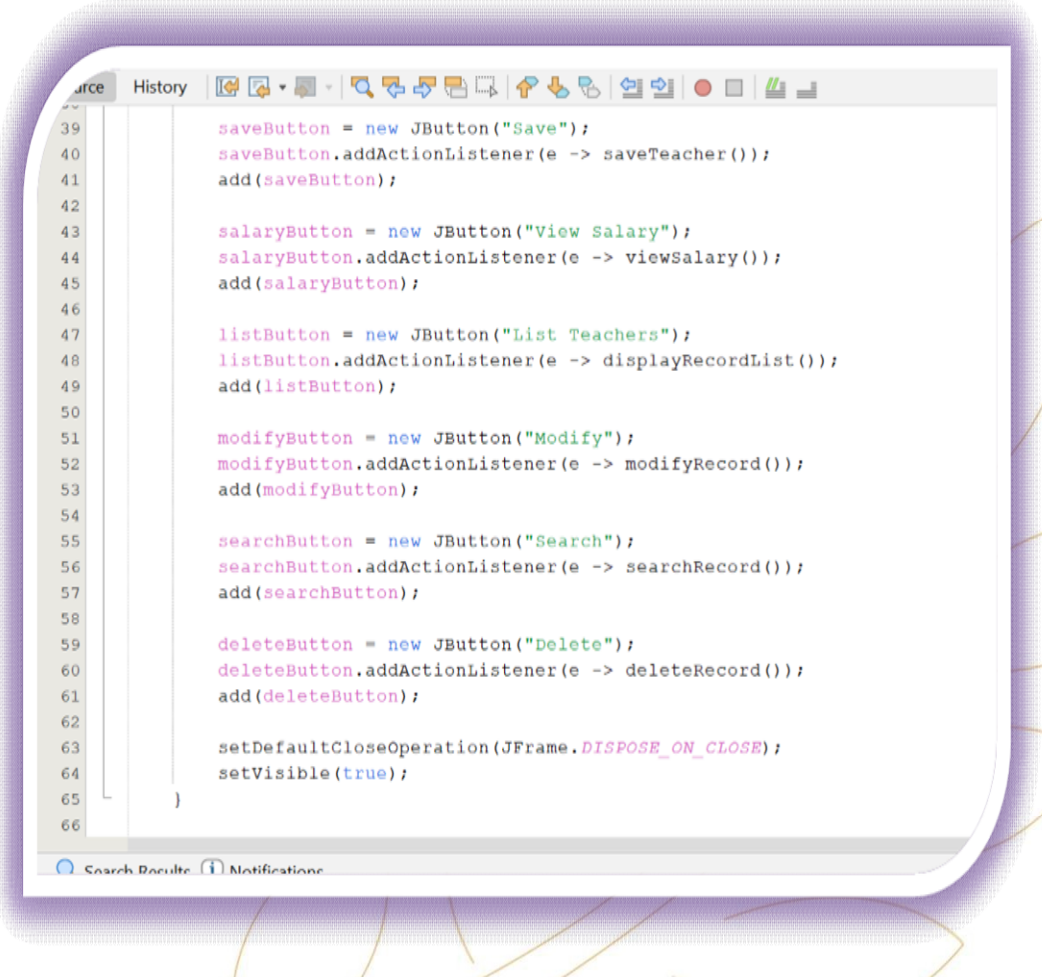
7. TeacherSectionGUI

- **Role:** Provides a GUI for teacher record management.

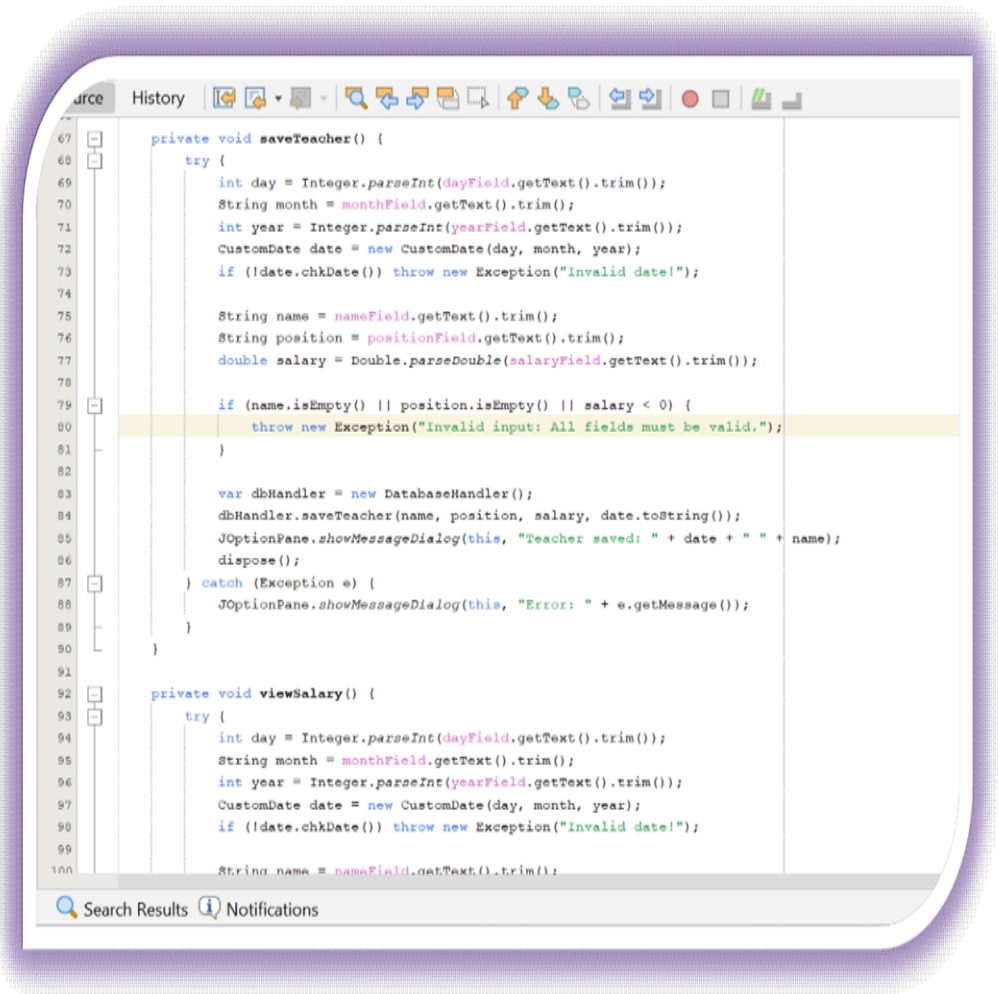
- **Purpose:** Allows users to perform create, read, update, and delete operations on teacher records, offering a user-friendly interface to streamline administrative tasks effectively.



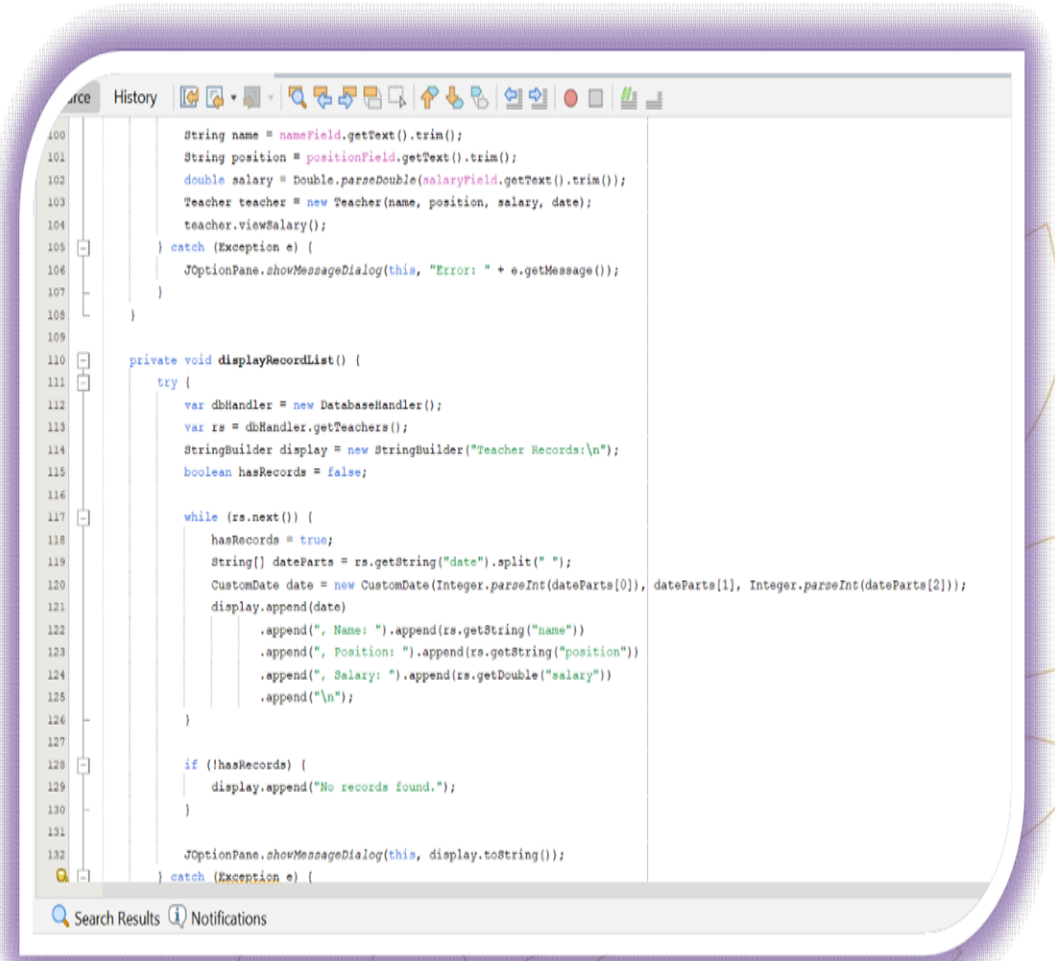
```
public class TeacherSectionGUI extends JFrame {
    private JTextField dayField, monthField, yearField, nameField, positionField, salaryField;
    private JButton saveButton, salaryButton, listButton, modifyButton, searchButton, deleteButton;

    public TeacherSectionGUI() {
        setTitle("Teacher Section");
        setSize(500, 500);
        setLayout(new GridLayout(11, 2, 10, 10));

        add(new JLabel("Day of Joining:"));
        dayField = new JTextField();
        add(dayField);

        add(new JLabel("Month of Joining:"));
        monthField = new JTextField();
        add(monthField);

        add(new JLabel("Year of Joining:"));
        yearField = new JTextField();
        add(yearField);

        add(new JLabel("Name:"));
        nameField = new JTextField();
        add(nameField);

        add(new JLabel("Position:"));
        positionField = new JTextField();
        add(positionField);

        add(new JLabel("Salary:"));
        salaryField = new JTextField("0.0");
        add(salaryField);
```



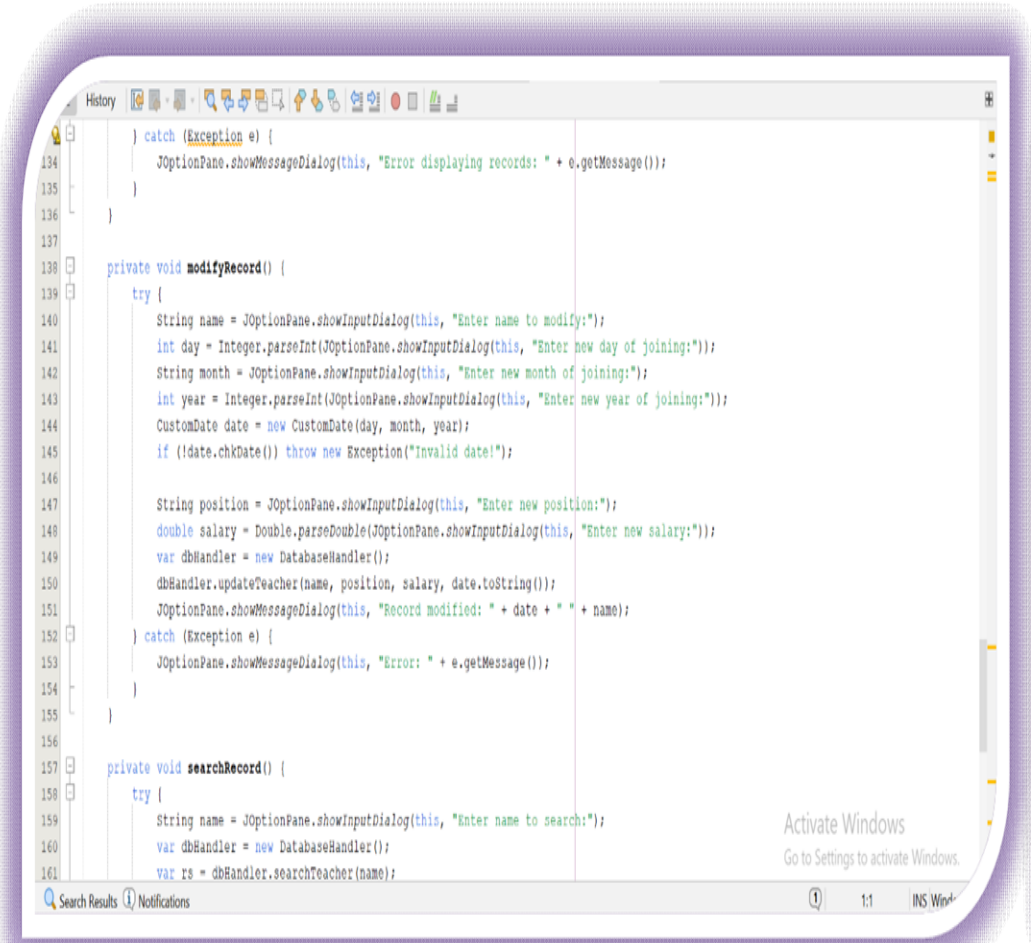
```
        saveButton = new JButton("Save");
        saveButton.addActionListener(e -> saveTeacher());
        add(saveButton);

        salaryButton = new JButton("View Salary");
        salaryButton.addActionListener(e -> viewSalary());
        add(salaryButton);

        listButton = new JButton("List Teachers");
        listButton.addActionListener(e -> displayRecordList());
        add(listButton);

        modifyButton = new JButton("Modify");
        modifyButton.addActionListener(e -> modifyRecord());
        add(modifyButton);

        searchButton = new JButton("Search");
        searchButton.addActionListener(e -> searchRecord());
        add(searchButton);

        deleteButton = new JButton("Delete");
        deleteButton.addActionListener(e -> deleteRecord());
        add(deleteButton);

        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setVisible(true);
    }
```



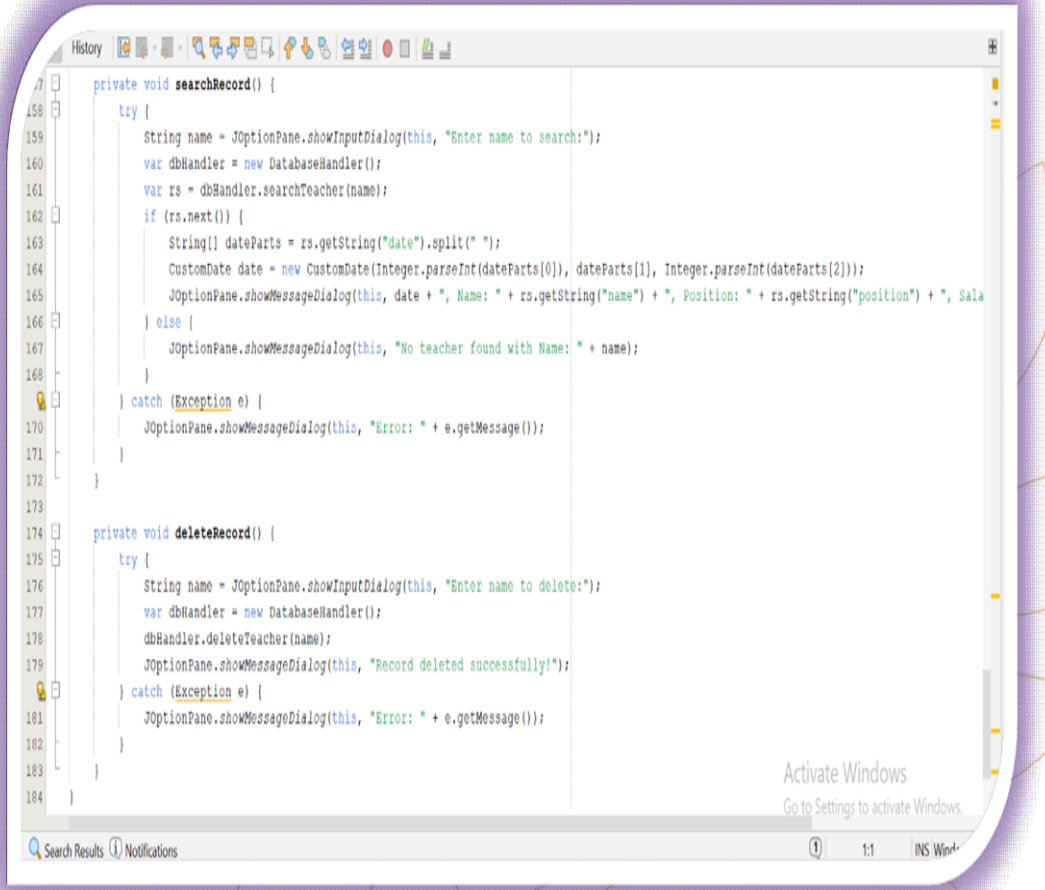
```
    private void saveTeacher() {
        try {
            int day = Integer.parseInt(dayField.getText().trim());
            String month = monthField.getText().trim();
            int year = Integer.parseInt(yearField.getText().trim());
            CustomDate date = new CustomDate(day, month, year);
            if (!date.chkDate()) throw new Exception("Invalid date!");

            String name = nameField.getText().trim();
            String position = positionField.getText().trim();
            double salary = Double.parseDouble(salaryField.getText().trim());

            if (name.isEmpty() || position.isEmpty() || salary < 0) {
                throw new Exception("Invalid input: All fields must be valid.");
            }

            var dbHandler = new DatabaseHandler();
            dbHandler.saveTeacher(name, position, salary, date.toString());
            JOptionPane.showMessageDialog(this, "Teacher saved: " + date + " " + name);
            dispose();
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
        }
    }

    private void viewSalary() {
        try {
            int day = Integer.parseInt(dayField.getText().trim());
            String month = monthField.getText().trim();
            int year = Integer.parseInt(yearField.getText().trim());
            CustomDate date = new CustomDate(day, month, year);
            if (!date.chkDate()) throw new Exception("Invalid date!");

            String name = nameField.getText().trim();
```



```
            String name = nameField.getText().trim();
            String position = positionField.getText().trim();
            double salary = Double.parseDouble(salaryField.getText().trim());
            Teacher teacher = new Teacher(name, position, salary, date);
            teacher.viewSalary();
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
        }
    }

    private void displayRecordList() {
        try {
            var dbHandler = new DatabaseHandler();
            var rs = dbHandler.getTeachers();
            StringBuilder display = new StringBuilder("Teacher Records:\n");
            boolean hasRecords = false;

            while (rs.next()) {
                hasRecords = true;
                String[] dateParts = rs.getString("date").split(" ");
                CustomDate date = new CustomDate(Integer.parseInt(dateParts[0]), dateParts[1], Integer.parseInt(dateParts[2]));
                display.append(date)
                        .append(", Name: ").append(rs.getString("name"))
                        .append(", Position: ").append(rs.getString("position"))
                        .append(", Salary: ").append(rs.getDouble("salary"))
                        .append("\n");
            }

            if (!hasRecords) {
                display.append("No records found.");
            }

            JOptionPane.showMessageDialog(this, display.toString());
        } catch (Exception e) {
```



```
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Error displaying records: " + e.getMessage());
        }
    }

    private void modifyRecord() {
        try {
            String name = JOptionPane.showInputDialog(this, "Enter name to modify:");
            int day = Integer.parseInt(JOptionPane.showInputDialog(this, "Enter new day of joining:"));
            String month = JOptionPane.showInputDialog(this, "Enter new month of joining:");
            int year = Integer.parseInt(JOptionPane.showInputDialog(this, "Enter new year of joining:"));
            CustomDate date = new CustomDate(day, month, year);
            if (!date.chkDate()) throw new Exception("Invalid date!");

            String position = JOptionPane.showInputDialog(this, "Enter new position:");
            double salary = Double.parseDouble(JOptionPane.showInputDialog(this, "Enter new salary:"));
            var dbHandler = new DatabaseHandler();
            dbHandler.updateTeacher(name, position, salary, date.toString());
            JOptionPane.showMessageDialog(this, "Record modified: " + date + " " + name);
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
        }
    }

    private void searchRecord() {
        try {
            String name = JOptionPane.showInputDialog(this, "Enter name to search:");
            var dbHandler = new DatabaseHandler();
            var rs = dbHandler.searchTeacher(name);
```

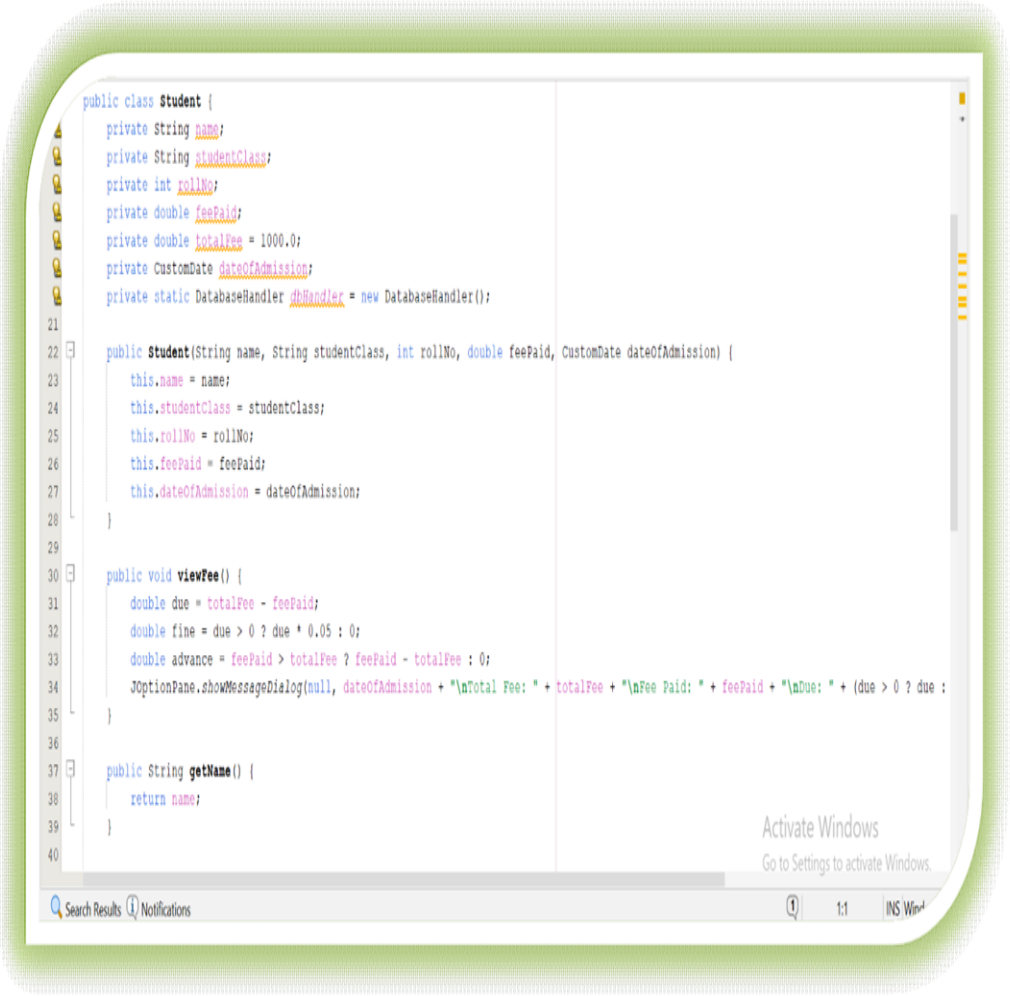



```
    private void searchRecord() {
        try {
            String name = JOptionPane.showInputDialog(this, "Enter name to search:");
            var dbHandler = new DatabaseHandler();
            var rs = dbHandler.searchTeacher(name);
            if (rs.next()) {
                String[] dateParts = rs.getString("date").split(" ");
                CustomDate date = new CustomDate(Integer.parseInt(dateParts[0]), dateParts[1], Integer.parseInt(dateParts[2]));
                JOptionPane.showMessageDialog(this, date + ", Name: " + rs.getString("name") + ", Position: " + rs.getString("position") + ", Sala
            } else {
                JOptionPane.showMessageDialog(this, "No teacher found with Name: " + name);
            }
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
        }
    }

    private void deleteRecord() {
        try {
            String name = JOptionPane.showInputDialog(this, "Enter name to delete:");
            var dbHandler = new DatabaseHandler();
            dbHandler.deleteTeacher(name);
            JOptionPane.showMessageDialog(this, "Record deleted successfully!");
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
        }
    }
}
```

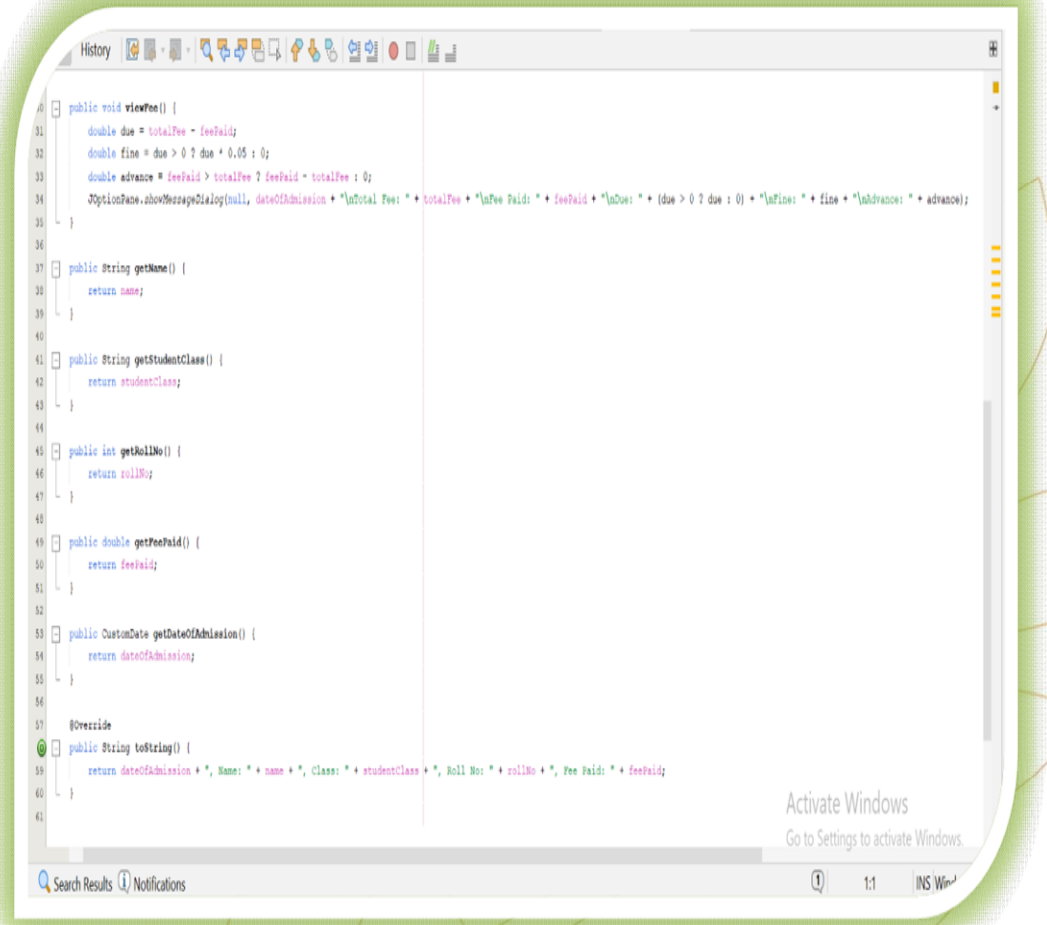
8. Student

- **Role:** Manages student records and calculates fees.

- **Purpose:** Tracks student information in the database and computes fee details such as due amounts, fines, and advances, supporting financial oversight within the educational system.



```java
public class Student {
    private String name;
    private String studentClass;
    private int rollNo;
    private double feePaid;
    private double totalFee = 1000.0;
    private CustomDate dateOfAdmission;
    private static DatabaseHandler dbHandler = new DatabaseHandler();

    public Student(String name, String studentClass, int rollNo, double feePaid, CustomDate dateOfAdmission) {
        this.name = name;
        this.studentClass = studentClass;
        this.rollNo = rollNo;
        this.feePaid = feePaid;
        this.dateOfAdmission = dateOfAdmission;
    }

    public void viewFee() {
        double due = totalFee - feePaid;
        double fine = due > 0 ? due * 0.05 : 0;
        double advance = feePaid > totalFee ? feePaid - totalFee : 0;
        JOptionPane.showMessageDialog(null, dateOfAdmission + "\nTotal Fee: " + totalFee + "\nFee Paid: " + feePaid + "\nDue: " + (due > 0 ? due :
    }

    public String getName() {
        return name;
    }
```

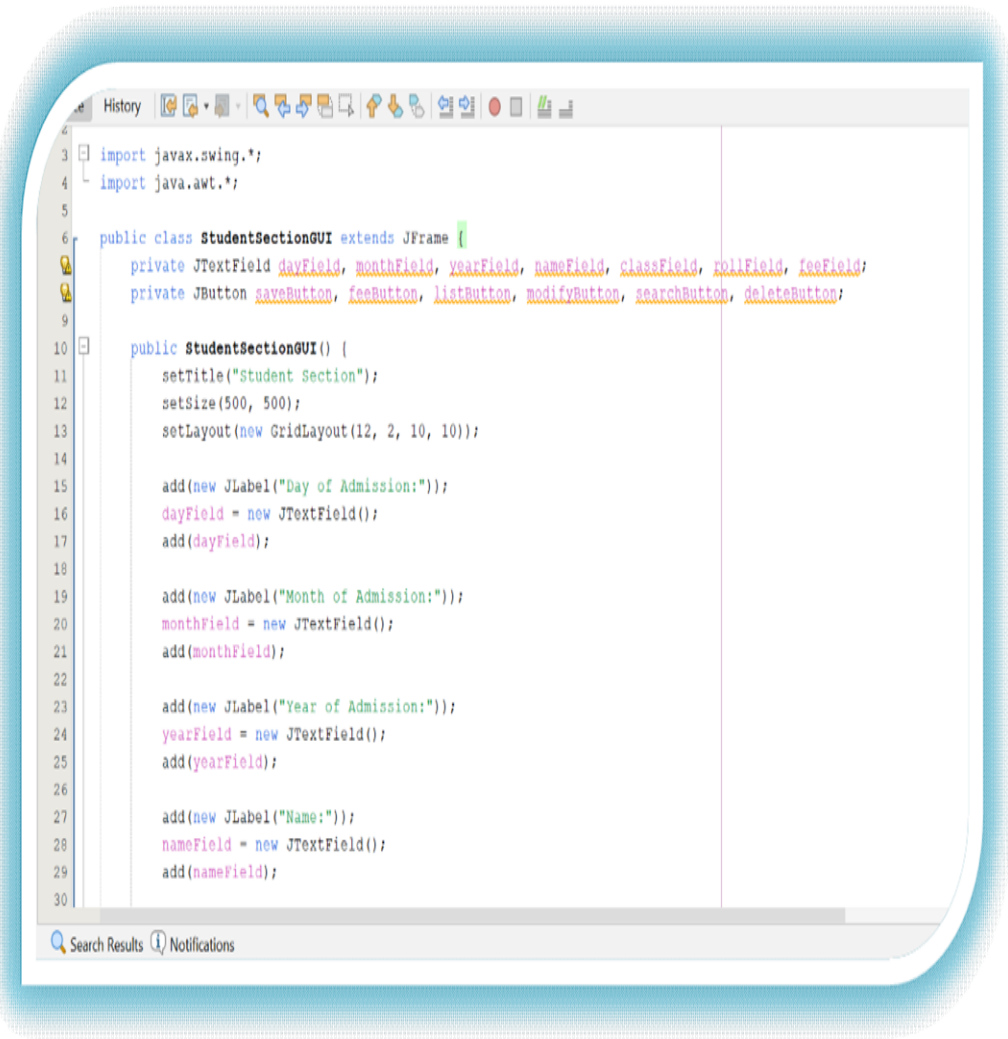



```java
    public void viewFee() {
        double due = totalFee - feePaid;
        double fine = due > 0 ? due * 0.05 : 0;
        double advance = feePaid > totalFee ? feePaid - totalFee : 0;
        JOptionPane.showMessageDialog(null, dateOfAdmission + "\nTotal Fee: " + totalFee + "\nFee Paid: " + feePaid + "\nDue: " + (due > 0 ? due : 0) + "\nFine: " + fine + "\nAdvance: " + advance);
    }

    public String getName() {
        return name;
    }

    public String getStudentClass() {
        return studentClass;
    }

    public int getRollNo() {
        return rollNo;
    }

    public double getFeePaid() {
        return feePaid;
    }

    public CustomDate getDateOfAdmission() {
        return dateOfAdmission;
    }

    @Override
    public String toString() {
        return dateOfAdmission + ", Name: " + name + ", Class: " + studentClass + ", Roll No: " + rollNo + ", Fee Paid: " + feePaid;
    }
```

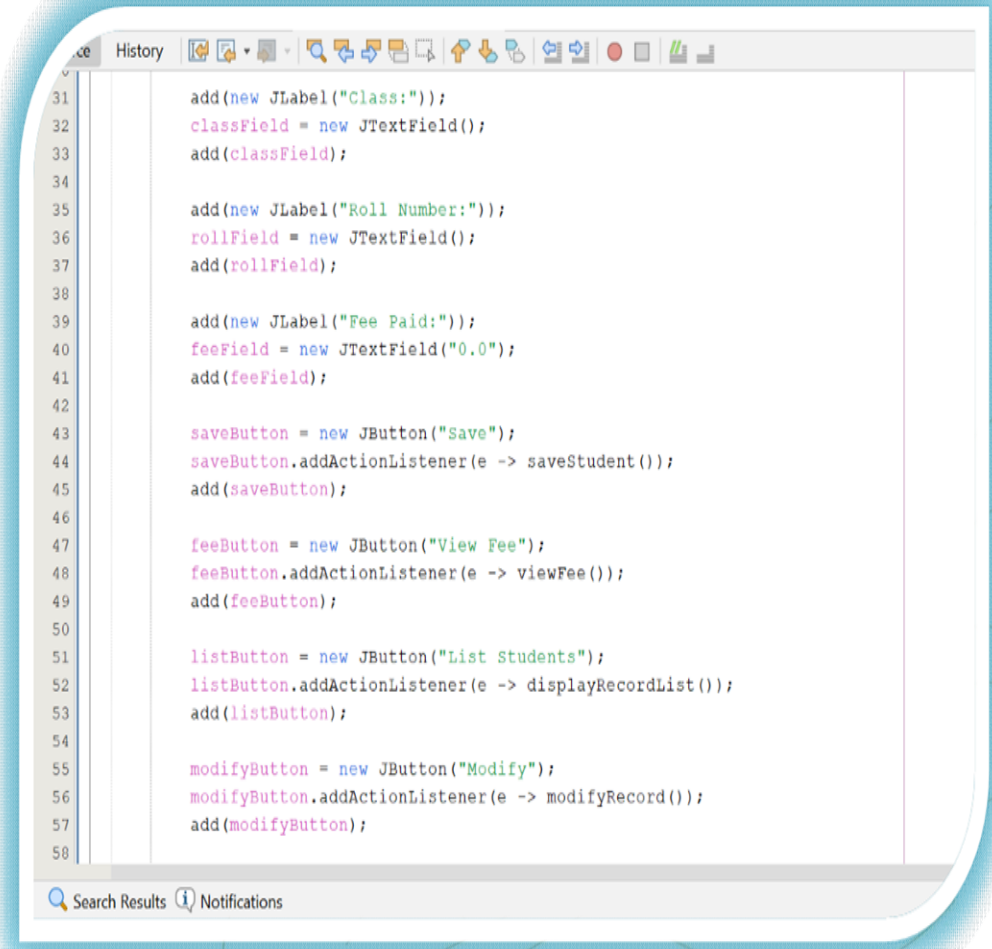
9. StudentSectionGUI

- **Role:** Offers a GUI for student record management.

- **Purpose:** Facilitates the handling of student data and displays fee-related information, enabling users to manage records efficiently and access financial details with ease.

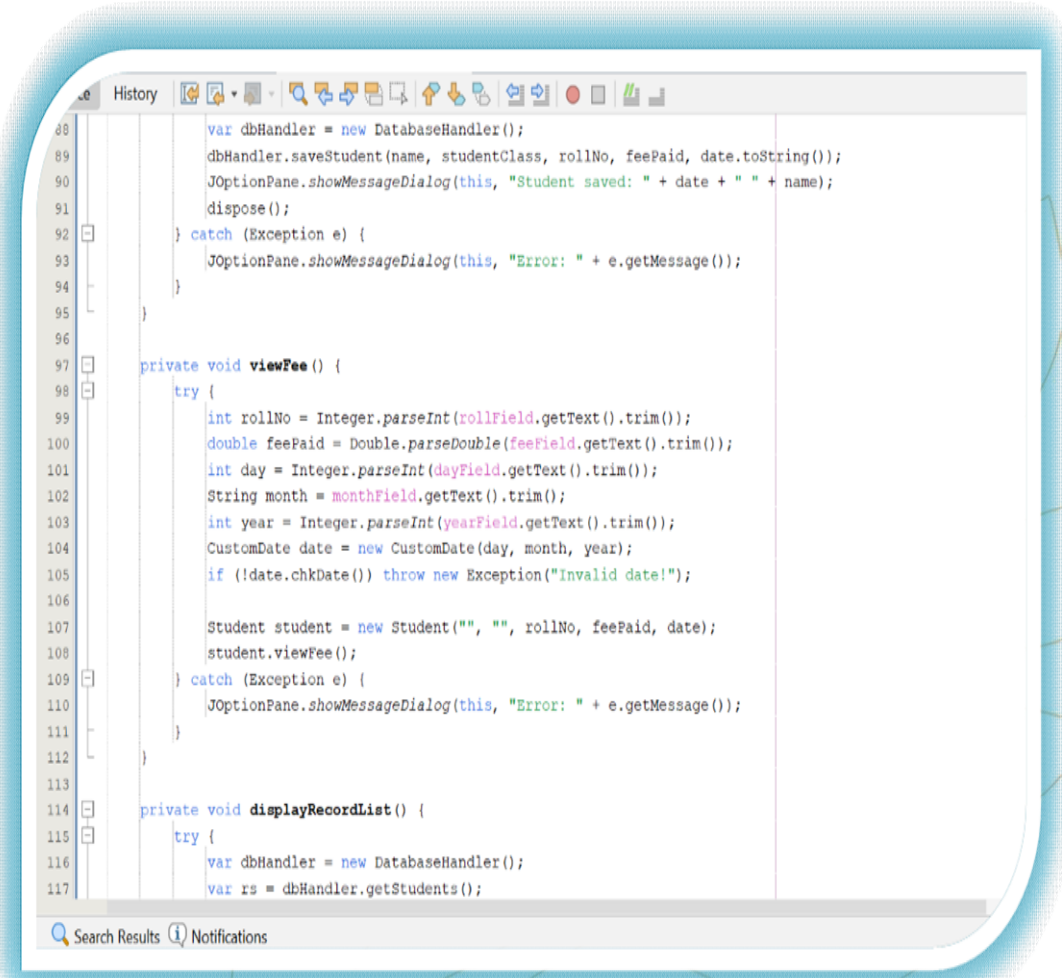


```
import javax.swing.*;
import java.awt.*;

public class StudentSectionGUI extends JFrame {
    private JTextField dayField, monthField, yearField, nameField, classField, rollField, feeField;
    private JButton saveButton, feeButton, listButton, modifyButton, searchButton, deleteButton;

    public StudentSectionGUI() {
        setTitle("Student Section");
        setSize(500, 500);
        setLayout(new GridLayout(12, 2, 10, 10));

        add(new JLabel("Day of Admission:"));
        dayField = new JTextField();
        add(dayField);

        add(new JLabel("Month of Admission:"));
        monthField = new JTextField();
        add(monthField);

        add(new JLabel("Year of Admission:"));
        yearField = new JTextField();
        add(yearField);

        add(new JLabel("Name:"));
        nameField = new JTextField();
        add(nameField);
```



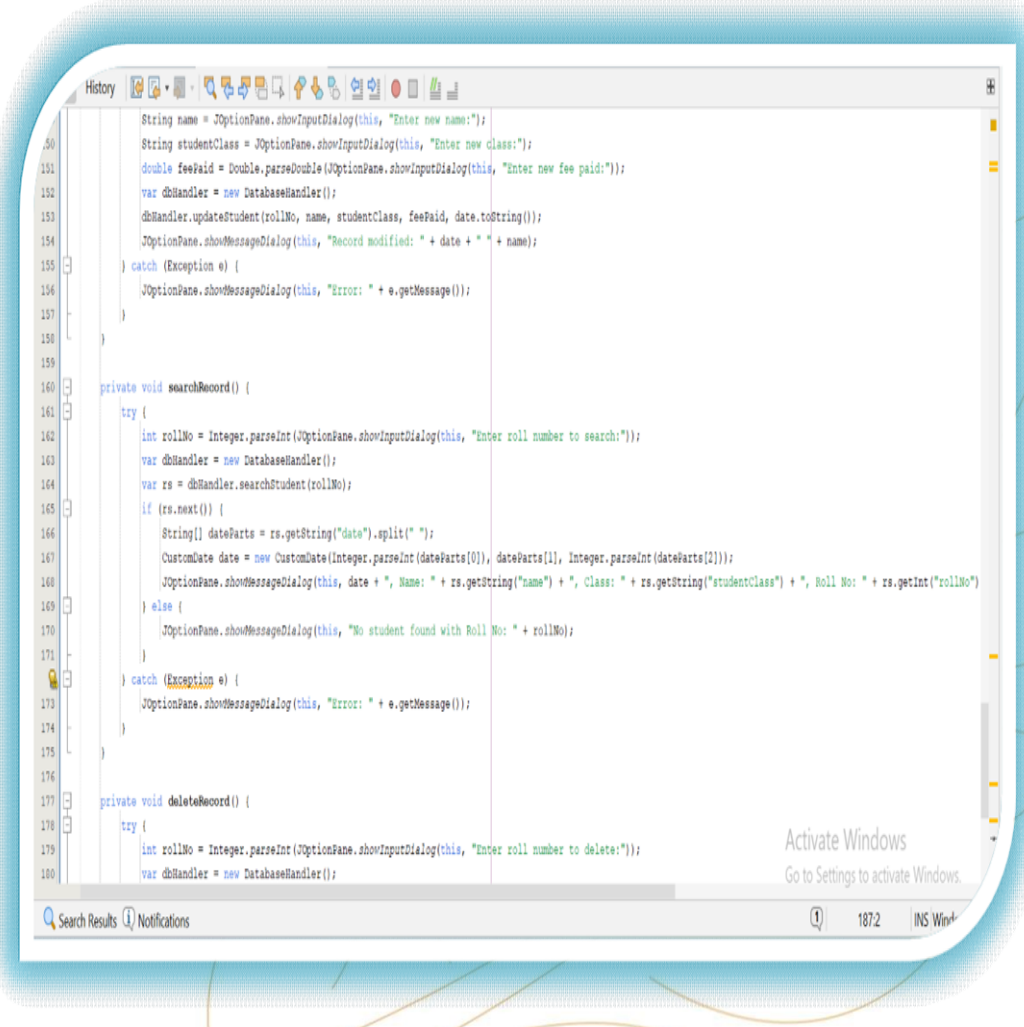
```
        add(new JLabel("Class:"));
        classField = new JTextField();
        add(classField);

        add(new JLabel("Roll Number:"));
        rollField = new JTextField();
        add(rollField);

        add(new JLabel("Fee Paid:"));
        feeField = new JTextField("0.0");
        add(feeField);

        saveButton = new JButton("Save");
        saveButton.addActionListener(e -> saveStudent());
        add(saveButton);

        feeButton = new JButton("View Fee");
        feeButton.addActionListener(e -> viewFee());
        add(feeButton);

        listButton = new JButton("List Students");
        listButton.addActionListener(e -> displayRecordList());
        add(listButton);

        modifyButton = new JButton("Modify");
        modifyButton.addActionListener(e -> modifyRecord());
        add(modifyButton);
```

```
59          searchButton = new JButton("Search");
60          searchButton.addActionListener(e -> searchRecord());
61          add(searchButton);
62
63          deleteButton = new JButton("Delete");
64          deleteButton.addActionListener(e -> deleteRecord());
65          add(deleteButton);
66
67          setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
68          setVisible(true);
69      }
70
71      private void saveStudent() {
72          try {
73              int day = Integer.parseInt(dayField.getText().trim());
74              String month = monthField.getText().trim();
75              int year = Integer.parseInt(yearField.getText().trim());
76              CustomDate date = new CustomDate(day, month, year);
77              if (!date.chkDate()) throw new Exception("Invalid date!");
78
79              String name = nameField.getText().trim();
80              String studentClass = classField.getText().trim();
81              int rollNo = Integer.parseInt(rollField.getText().trim());
82              double feePaid = Double.parseDouble(feeField.getText().trim());
83
84              if (name.isEmpty() || studentClass.isEmpty() || rollNo <= 0 || feePaid < 0) {
85                  throw new Exception("Invalid input: All fields must be valid.");
86              }
```

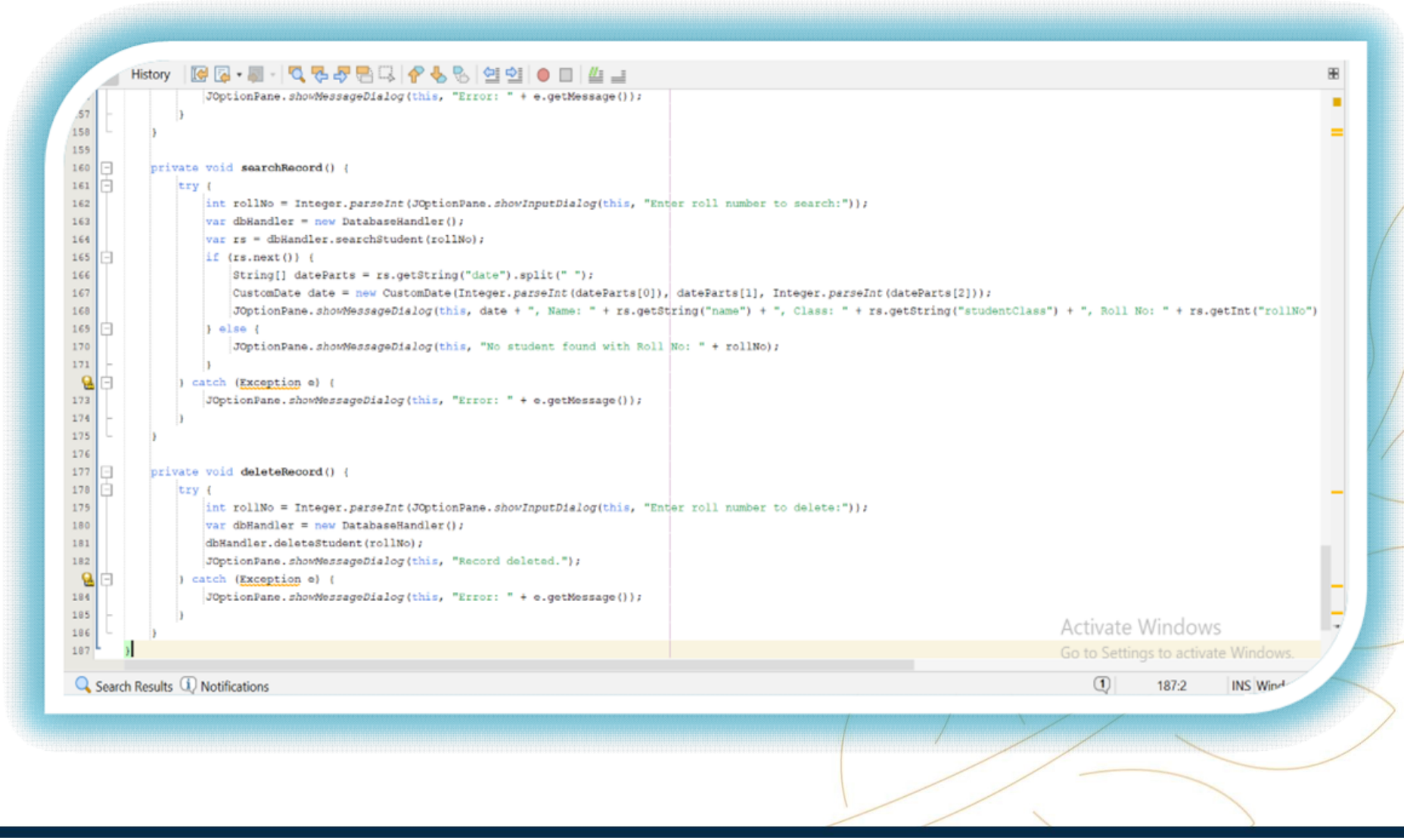


```
88              var dbHandler = new DatabaseHandler();
89              dbHandler.saveStudent(name, studentClass, rollNo, feePaid, date.toString());
90              JOptionPane.showMessageDialog(this, "Student saved: " + date + " " + name);
91              dispose();
92          } catch (Exception e) {
93              JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
94          }
95      }
96
97      private void viewFee() {
98          try {
99              int rollNo = Integer.parseInt(rollField.getText().trim());
100             double feePaid = Double.parseDouble(feeField.getText().trim());
101             int day = Integer.parseInt(dayField.getText().trim());
102             String month = monthField.getText().trim();
103             int year = Integer.parseInt(yearField.getText().trim());
104             CustomDate date = new CustomDate(day, month, year);
105             if (!date.chkDate()) throw new Exception("Invalid date!");
106
107             Student student = new Student("", "", rollNo, feePaid, date);
108             student.viewFee();
109         } catch (Exception e) {
110             JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
111         }
112     }
113
114     private void displayRecordList() {
115         try {
116             var dbHandler = new DatabaseHandler();
117             var rs = dbHandler.getStudents();
```

```
StringBuilder display = new StringBuilder("Student Records:\n");
boolean hasRecords = false;

while (rs.next()) {
    hasRecords = true;
    String[] dateParts = rs.getString("date").split(" ");
    CustomDate date = new CustomDate(Integer.parseInt(dateParts[0]), dateParts[1], Integer.parseInt(dateParts[2]));
    display.append(date)
           .append(", Name: ").append(rs.getString("name"))
           .append(", Class: ").append(rs.getString("studentClass"))
           .append(", Roll No: ").append(rs.getInt("rollNo"))
           .append(", Fee Paid: ").append(rs.getDouble("feePaid"))
           .append("\n");
}

if (!hasRecords) display.append("No records found.");
JOptionPane.showMessageDialog(this, display.toString());
} catch (Exception e) {
    JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
}
}

private void modifyRecord() {
    try {
        int rollNo = Integer.parseInt(JOptionPane.showInputDialog(this, "Enter roll number to modify:"));
        int day = Integer.parseInt(JOptionPane.showInputDialog(this, "Enter new day of admission:"));
        String month = JOptionPane.showInputDialog(this, "Enter new month of admission:");
        int year = Integer.parseInt(JOptionPane.showInputDialog(this, "Enter new year of admission:"));
        CustomDate date = new CustomDate(day, month, year);
        if (!date.chkDate()) throw new Exception("Invalid date!");
```



```
String name = JOptionPane.showInputDialog(this, "Enter new name:");
String studentClass = JOptionPane.showInputDialog(this, "Enter new class:");
double feePaid = Double.parseDouble(JOptionPane.showInputDialog(this, "Enter new fee paid:"));
var dbHandler = new DatabaseHandler();
dbHandler.updateStudent(rollNo, name, studentClass, feePaid, date.toString());
JOptionPane.showMessageDialog(this, "Record modified: " + date + " " + name);
} catch (Exception e) {
    JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
}
}

private void searchRecord() {
    try {
        int rollNo = Integer.parseInt(JOptionPane.showInputDialog(this, "Enter roll number to search:"));
        var dbHandler = new DatabaseHandler();
        var rs = dbHandler.searchStudent(rollNo);
        if (rs.next()) {
            String[] dateParts = rs.getString("date").split(" ");
            CustomDate date = new CustomDate(Integer.parseInt(dateParts[0]), dateParts[1], Integer.parseInt(dateParts[2]));
            JOptionPane.showMessageDialog(this, date + ", Name: " + rs.getString("name") + ", Class: " + rs.getString("studentClass") + ", Roll No: " + rs.getInt("rollNo")
        } else {
            JOptionPane.showMessageDialog(this, "No student found with Roll No: " + rollNo);
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
    }
}

private void deleteRecord() {
    try {
        int rollNo = Integer.parseInt(JOptionPane.showInputDialog(this, "Enter roll number to delete:"));
        var dbHandler = new DatabaseHandler();
```


History
JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
}
}
private void searchRecord() {
try {
int rollNo = Integer.parseInt(JOptionPane.showInputDialog(this, "Enter roll number to search:"));
var dbHandler = new DatabaseHandler();
var rs = dbHandler.searchStudent(rollNo);
if (rs.next()) {
String[] dateParts = rs.getString("date").split(" ");
CustomDate date = new CustomDate(Integer.parseInt(dateParts[0]), dateParts[1], Integer.parseInt(dateParts[2]));
JOptionPane.showMessageDialog(this, date + ", Name: " + rs.getString("name") + ", Class: " + rs.getString("studentClass") + ", Roll No: " + rs.getInt("rollNo")
} else {
JOptionPane.showMessageDialog(this, "No student found with Roll No: " + rollNo);
}
} catch (Exception e) {
JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
}
}
private void deleteRecord() {
try {
int rollNo = Integer.parseInt(JOptionPane.showInputDialog(this, "Enter roll number to delete:"));
var dbHandler = new DatabaseHandler();
dbHandler.deleteStudent(rollNo);
JOptionPane.showMessageDialog(this, "Record deleted.");
} catch (Exception e) {
JOptionPane.showMessageDialog(this, "Error: " + e.getMessage());
}
}
}
Activate Windows
Go to Settings to activate Windows.
Search Results
Notifications
187:2
INS

MSA
UNIVERSITY
IN EGYPT SINCE 1996
Established by Dr.Nawal El Degwi
October Univeristy for Modern Sciences and Arts
جامعة أكتوبر للعلوم الحديثة والآداب

Thank You!

THE FIRST BRITISH HIGHER EDUCATION IN EGYPT
26th July Mehwar Road Intersection with Wahat Road, 6th of October City, Egypt
Tel: 00238371113 Postal code: 12451 Email: info@msa.edu.eg
Hotline: 16672 Website: www.msa.edu.eg