My grade management system

Java Programming II - Project

Anna Ha
12/13/2022

Contents

1. Overview of the Design

1.1 Project description

*Note: This project is a modified version of the program from the material *MySQLWithNetBeansAndJDBCTutorial.pdf* in this course.

The program will check whether a student is eligible for a degree or a certificate based on the course information they input.
It will check things as follows

1. A student must take 2 core courses (course number: CO100, CO200) with at least B grade.
2. A student must take at least 5 courses with at least B grade.

This program is implemented by Java with GUI interfaces using swing. And the data is stored in MySQL DB using JDBC.

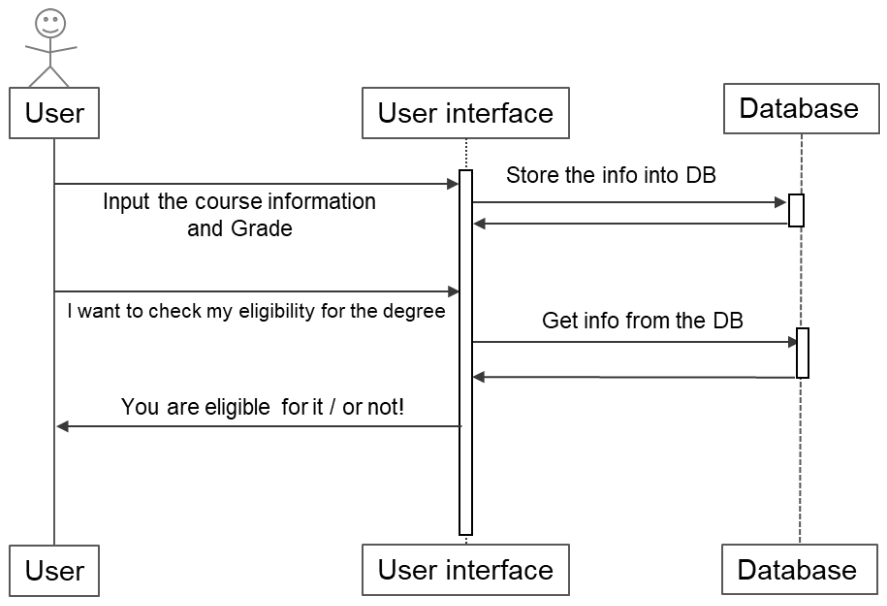
1.2 Brief Use case and sequence diagram

Main actor: Student

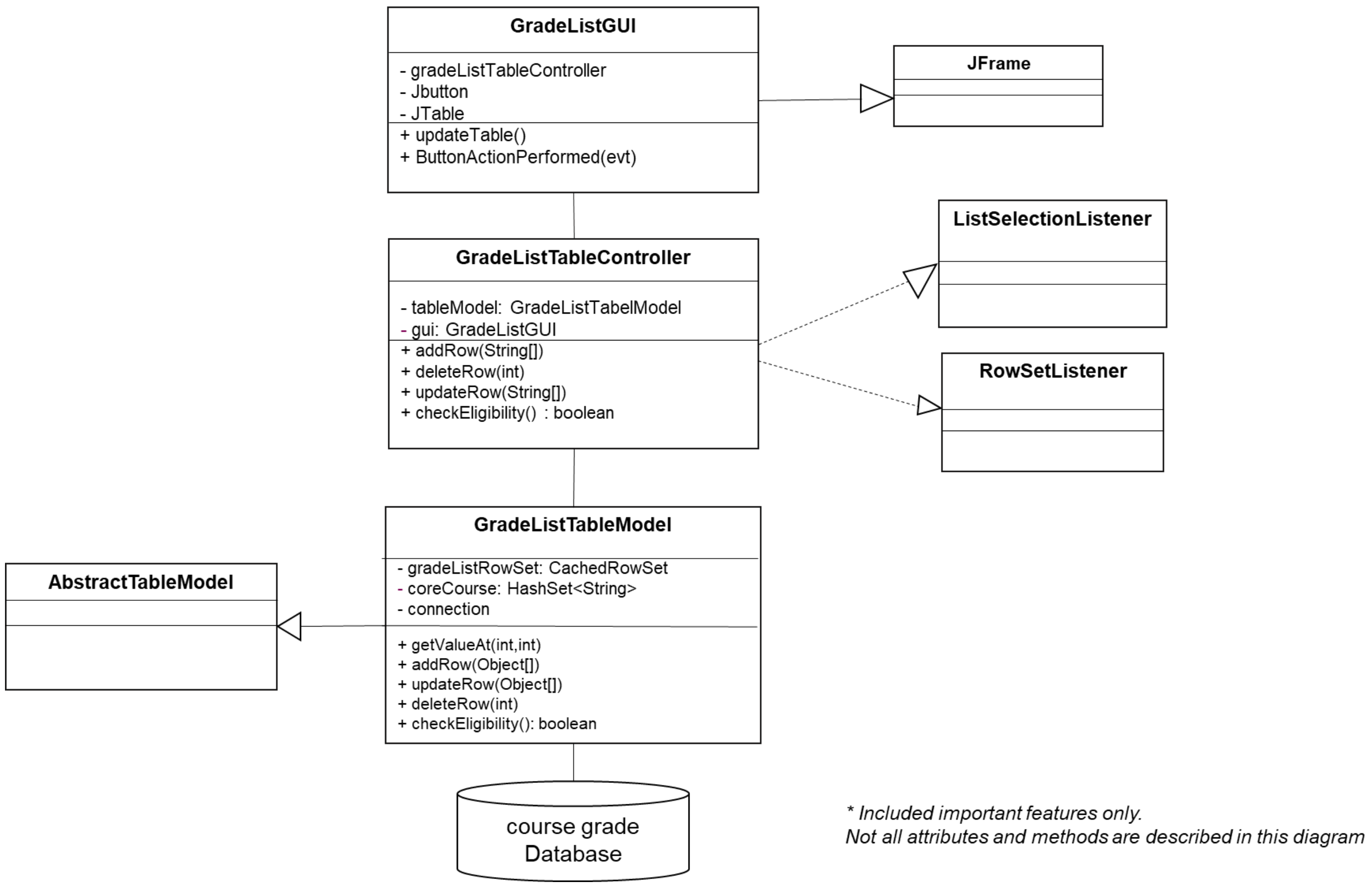
A student turns on the GUI program. A window pops up displaying the course list and the grade the user took over the last quarters.

There are buttons to add, delete or update a course. When the user clicks these buttons, it updates course list in database and display updated list in the GUI window.

There is also a button to calculate whether the student is eligible for the certificate based on the database.
When the button is clicked, additional window pops up saying ‘You are eligible!” or “You are not qualified yet”


User
User interface
Database
Input the course information and Grade
Store the info into DB
I want to check my eligibility for the degree
Get info from the DB
You are eligible for it / or not!
User
User interface
Database

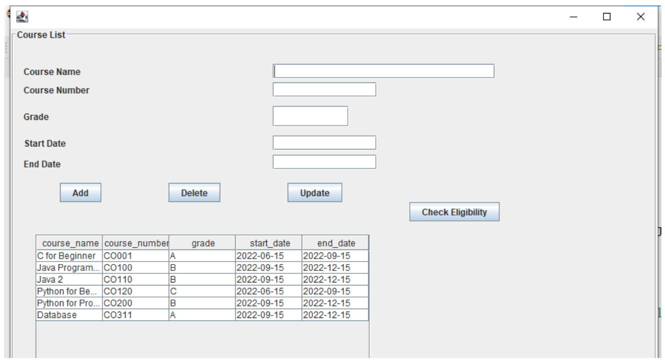
1.3 UML Class Diagram



▲ Class diagram

- This program use MVC pattern. There are 3 main classes which are GradeListGUI, GradeListTableController, GradeListTableModel.

1) GradeListGUI class as a view
   - It is the graphic user interface using swing.JFrame
   - get actions from user clicking buttons and deliver the event to controller



course_name	course_number	grade	start_date	end_date
C for Beginner	CO001	A	2022-06-15	2022-09-15
Java Program...	CO100	B	2022-09-15	2022-12-15
Java 2	CO110	B	2022-09-15	2022-12-15
Python for Be...	CO120	C	2022-06-15	2022-09-15
Python for Pro...	CO200	B	2022-09-15	2022-12-15
Database	CO311	A	2022-09-15	2022-12-15

▲ *GUI window*

2) GradeListTableController class as a controller

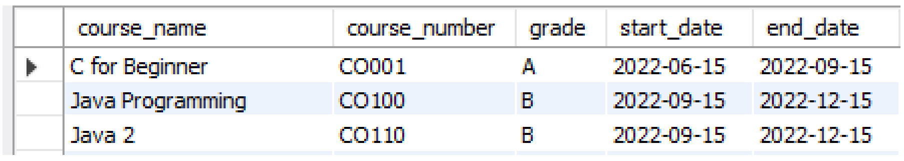
- It implements ListSelectionListener and RowSetListener interface so that it is notified when a lists selection value or RowSet value changed
- perform as a glue between view(GradeListGUI) and the model (GradeListTableModel)
- Get the events from GUI and deliver them to model and vice versa.

3) GradeListTableModel class as a model

- It is a subclass of AbstractTableModel class that provides default implementations for most of the methods in the TableModel interface. It takes care of the management of listeners
- It interacts with database directly using JDBC
- It has data from database 'coursegrade' table.

* Creation 'coursegrade' table in your MySQL server is needed in advance as follows.

```
create table if not exists coursegrade (
    course_name varchar(100),
    course_number varchar(50),
    grade varchar(10),
    start_date varchar(10),
    end_date varchar(10),
    primary key (course_number));
```



	course_name	course_number	grade	start_date	end_date
▶	C for Beginner	CO001	A	2022-06-15	2022-09-15
	Java Programming	CO100	B	2022-09-15	2022-12-15
	Java 2	CO110	B	2022-09-15	2022-12-15

▲ *coursegrade table example in MySQL database*

2. Implementation details

2.1 Class details

*Note: This project is a modified version of the program from the material “MySQLWithNetBeansAndJDBCTutorial.pdf” in this course. Therefore, overall structure of this program is based on the previous program (CourseListGUI, CourseListTableController, CourseListTableModel) and the code has been modified and added for this project as follows.

- Added Eligibility Check feature.
- Added implementation of delete, update feature
- minor changes in overall structure.

**Only meaningful features of this project are described in this document. Please refer to the source code files if full context is needed.

1) GradeListGUI class

* swing library used to implement GUI

a. Attributes of GradeListGUI class

1. JButtons

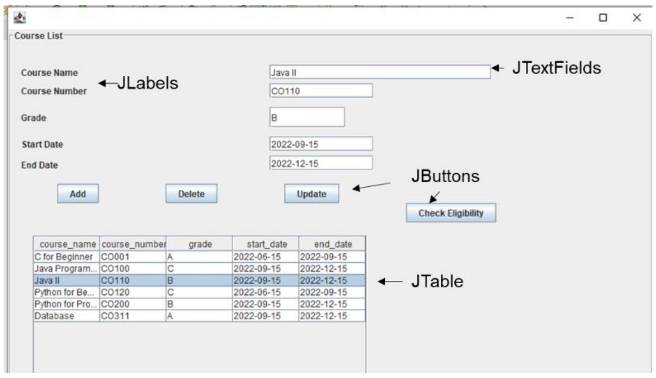
- There are 4 buttons in GUI and each button will their roles when clicked as follows

```
private JButton addButton;     // add a new row with information in textfield
private JButton deleteButton;  // delete the selected row
private JButton updateButton;  // update the selected row with new info in textfields
private JButton calculateButton;  // calculate eligibility for a certificate
```

2. JTable, JLabels, JTextFields

- There are a JTable that represents table from database as well as 5 JLabels and JTextFields to represent the text and text input field.

```
private JTable jtable1;          // the table displayed on the GUI
private JTextField courseNameTextField;        // get a course name as a input
private JTextField courseNumberTextField;      // get a course number
private JTextField gradeTextField;             // get a letter grade
private JTextField startDateTextField;         // get a course start date
private JTextField endDateTextField;           // and end date
```



Course Name: Java II
Course Number: CO110
Grade: B
Start Date: 2022-09-15
End Date: 2022-12-15

course_name	course_number	grade	start_date	end_date
C for Beginner	CO001	A	2022-06-15	2022-09-15
Java Program...	CO100	C	2022-09-15	2022-12-15
Java II	CO110	B	2022-09-15	2022-12-15
Python for Be...	CO120	C	2022-06-15	2022-09-15
Python for Pro...	CO200	B	2022-09-15	2022-12-15
Database	CO311	A	2022-09-15	2022-12-15

▲ *GUI window*

3. gradeListTableController
- Besides window components, GradeListGUI class has a controller as an attribute that deliver action to a model object (GradeListTableModel)

```
private GradeListTableController gradeListTableController;
```

A controller object is created using this GUI object (GradeListGUI) as an argument so it can communicate each other.

```
gradeListTableController = new GradeListTableController(this);
```

And this controller is added to the JTable as a ListSelectionLister as well

```
jtable1 = new JTable(gradeListTableController.getTableModel());
// add a ListSelectionListener to the table
jtable1.getSelectionModel().addListSelectionListener(gradeListTableController);
```

b. Methods of GradeListGUI class

1. initComponents()
- This method does initialize components in window by creating component objects and setting them in given layout. Most of the code in this method is from CourseListGUI class and modified for this program.

```
// initialize components in window
private void initComponents()
```

2. addButtonActionPerformed()
- When the buttons(Add, Delete, Update, Check Eligibility) is clicked, actions is performed by each method.

- Add button invoke the addRow method of the controller with string array that contains texts in text fields.

```
// code for the Add button ** This code is from @author rgrover
private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {
    //add a row to the table
    String[] array = new String[jtable1.getColumnCount()];
    array[0] = courseNameTextField.getText();
    array[1] = courseNumberTextField.getText();
    array[2] = gradeTextField.getText();
    array[3] = startDateTextField.getText();
    array[4] = endDateTextField.getText();
    gradeListTableController.addRow(array);
 }
```

3. deleteButtonActionPerformed()

- Delete button invoke the deleteRow method of the controller with the index of the selected row. The selected row index is from getSelectedRow() method of JTable object. And if there is no selected row, it returns -1. This index is for checking if it is -1 or not in order to prevent the unselected data from changing.

```
 private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt) {
    int selectedRow = jtable1.getSelectedRow();
    gradeListTableController.deleteRow(selectedRow);
}
```

4. updateButtonActionPerformed()

- Update button invoke the updateRow method of the controller with string array that contains updated texts in text fields and the selected row index for the same purpose as the delete action.

```
    private void updateButtonActionPerformed(java.awt.event.ActionEvent evt) {
        //update the current row in the table
        String[] array = new String[jtable1.getColumnCount()];
        array[0] = courseNameTextField.getText();
        array[1] = courseNumberTextField.getText();
        array[2] = gradeTextField.getText();
        array[3] = startDateTextField.getText();
        array[4] = endDateTextField.getText();

        int selectedRow = jtable1.getSelectedRow();
        gradeListTableController.updateRow(selectedRow, array);
}
```

5. calculateButtonActionPerformed()

- This is for Check Eligibility. When the button clicked, a new message dialog window pops up and say the result message. It says “Congrats! You are eligible….” if the student is eligible for the certificate based on the data from table. And “Sorry! You are not….. “ if it is not the case.
The message is dependent on the boolean type return value of checkEligibility method of the controller. (if true: eligible, false: not eligible)

```
    private void calculateButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // message dialog will display the result after check
        JFrame dWindow = new JFrame();
        String message = "";
       // check eligibility with the database
        boolean result = gradeListTableController.checkEligibility();
        if (result)
                message = "Congrats!\nYou are Eligible for the certificate.";
        else
                message = "Sorry!\nYou are Not eligible yet. Keep it up!";
        JOptionPane.showMessageDialog(dWindow, message, "Eligibility check",
        JOptionPane.PLAIN_MESSAGE);  }
```

2) GradeListTableController class

a. Attributes of GradeListTableController

```
private GradeListTableModel tableModel;
private GradeListGUI gui;
```

1. GradeListTableModel
- A controller has a GradeListTableModel object to access and control data in the table model.
And the controller itself is registered to the RowSet object of this TableModel as a RowSetListener so that it will be notified of events that occur on this RowSet object.

```
tableModel.getRowSet().addRowSetListener(this);
```

2. GradeListGUI
 - A controller has a GradeListGUI object to communicate with for window displaying.

b. Methods of GradeListTableController class

- The major role in methods of controller is delivery the action from GUI to table model object.

1. addRow(): invokes addRow method of the table model object

```
public void addRow(String[] array) {
       tableModel.addRow(array);
}
```

2. updateRow(): invokes updateRow method of the table model object.

```
public void updateRow(int index, String[] array) {
       tableModel.updateRow(index, array);
}
```

3. deleteRow(): invokes deleteRow method of the table model object. and replace tableModel with new table model object of the changed RowSet after deletion. This is for avoiding cursor confusion problem by decreased row number when check eligibility is clicked right after a delete operation in the same window.

```
public void deleteRow(int row) {
tableModel.deleteRow(row);
// create a new table model with the new data
tableModel = new GradeListTableModel(tableModel.getRowSet(),
tableModel.getConnection());
gui.updateTable(); // update the JTable with the data  }
```

4. checkEligibility(): invokes checkEligibility method of the table model object.

```
// check eligibility based on the current database
// return true if the user is eligible else return false.
public boolean checkEligibility() {
     return tableModel.checkEligibility();
}
```

3) GradeListTableModel class

*JDBC and HashSet library used here

a. Attributes

```
private CachedRowSet gradeListRowSet;    // contains data from database
private ResultSetMetaData metadata;      // additional information about the data
private Connection connection;          // DB connection
private int numcols, numrows;           // number of rows and columns
private HashSet<String> coreCourse;     // core course list for certificate
```

1. CachedRowSet

- This object stores all the data from 'coursegrade' table in database by executing SQL query through JDBC connection.

```
gradeListRowSet = RowSetProvider.newFactory().createCachedRowSet();
gradeListRowSet.setCommand("SELECT * FROM coursegrade");
gradeListRowSet.execute(connection);
```

2. ResultSetMetaData

- This object has additional information about the dataset such as number of columns and all.

```
  metadata = gradeListRowSet.getMetaData();
  numcols = metadata.getColumnCount();
  numrows = gradeListRowSet.size();
```

3. Connection

- It makes connection with SQL local server using JDBC. You need to change the database name, username, password as your environment.

```
connection = DriverManager.getConnection(
           "jdbc:mysql://localhost:3306/test1", "root", "1234");
```

4. coreCourse

- It contains core courses numbers ("CO100", "CO200") for the certificates. You can modify the setCoreCourse method in this class if you want to change or add course numbers.
  A HashSet structure is used here for a set operation when we check the eligibility later.

```
public void setCoreCourse() {
       // add core course numbers into a hashset
       coreCourse = new HashSet<String>();
       coreCourse.add("CO100");
       coreCourse.add("CO200");
}
```

b. Methods of GradeListTableModel class

1. addRow(Object[] array)

- It operates addition a new row into the *RowSet* object with information of the array argument passed. and propagates the change to database through *connection.*

It is done by the methods `updateString`, `insertRow` and `acceptChanges` of `CachedRowSet` Object.

```
public void addRow(Object[] array) {
  try {
          gradeListRowSet.last(); // set the current row to rowIndex
          gradeListRowSet.moveToInsertRow();
          // set the strings to add using the array passed
          gradeListRowSet.updateString("course_name", (String) array[0]);
          gradeListRowSet.updateString("course_number", (String) array[1]);
          gradeListRowSet.updateString("grade",  (String) array[2]);
          gradeListRowSet.updateString("start_date", (String) array[3]);
          gradeListRowSet.updateString("end_date", (String) array[4]);

          gradeListRowSet.insertRow(); // add a new row into the RowSet
          gradeListRowSet.moveToCurrentRow();
          // propagates all the changes to database
          gradeListRowSet.acceptChanges(connection);
       } catch(SQLException err) {
              err.getMessage();
              err.printStackTrace();
       }
}
```

2. updateRow(int index, Object[] array)

- It updates the selected row of the RowSet object with information in the array argument passed. and propagates the change to database through *connection.*

It is done by the methods `updateString`, `updateRow` and `acceptChanges` of `CachedRowSet` Object.

```
// update the selected row with new information in the text fields of the window
public void updateRow(int index, Object[] array) {
       if (index != -1) {
          try {
              gradeListRowSet.updateString("course_name", (String) array[0]);
              gradeListRowSet.updateString("course_number", (String) array[1]);
              gradeListRowSet.updateString("grade",  (String) array[2]);
              gradeListRowSet.updateString("start_date", (String) array[3]);
              gradeListRowSet.updateString("end_date", (String) array[4]);

              gradeListRowSet.updateRow();
              gradeListRowSet.acceptChanges(connection);
              } catch(SQLException err) {
                     err.getMessage();
                     err.printStackTrace();
              }
       }
}
```

3. deleteRow(int index)

 - This method remove the selected row in the RowSet and propagates the changes to database as well. It is done by `deleteRow` and `acceptChanges` method of `CachedRowSet` Object.

```
        public void deleteRow(int index) {
            if (index != -1) { // delete a selected row from gradeListRowSet
                try {
                        gradeListRowSet.deleteRow();
                        gradeListRowSet.acceptChanges(connection);
                        } catch (SQLException e) {
                                System.out.println("Fail to delete");
                                e.printStackTrace();
                          }               }
         }
```

4. checkEligibility()

- This method checks whether the user is eligible for a certificate based on the database and requirements. It checks requirements as follows and return true or false as a result.

    1. check core courses and the grade using set operation of HashSet object.
    2. check the total number of course (need to be >=5 with >= B grade)

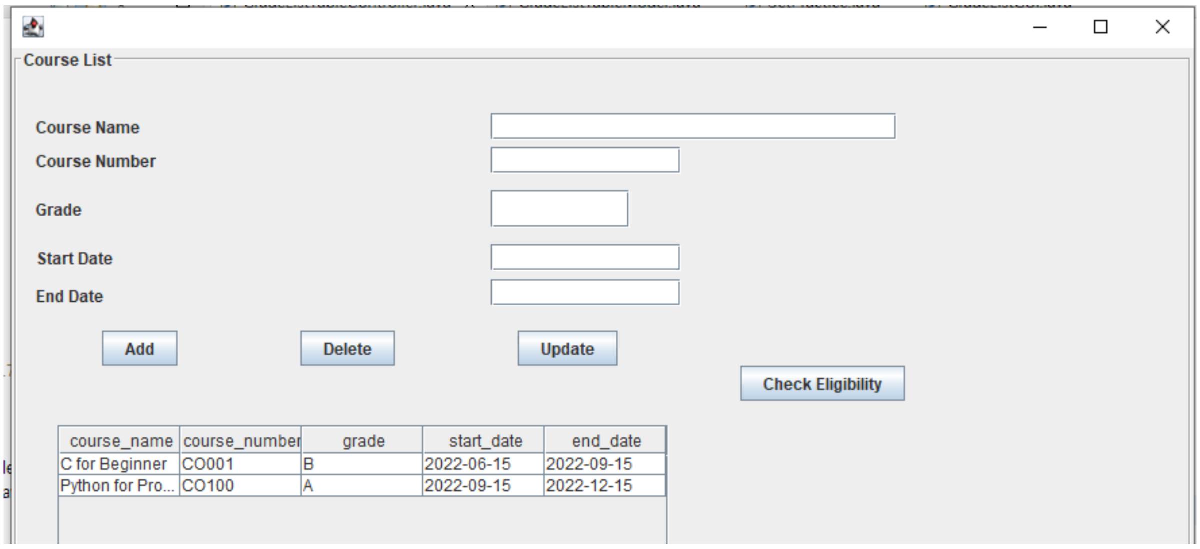
 * If the requirement changes, this method is the place where you want to modify

```
public boolean checkEligibility()  {
        try {
            // 1.check core courses and the grade
                gradeListRowSet.first(); // make sure to move the cursor to the first row
              // store taken course numbers from database into a HashSet(courseNumberSet)
                HashSet<String> courseNumberSet = new HashSet<String>();
                for (int i=0;i<numrows;i++) {
                        // only A,B grade counts
                        if ("AB".contains(gradeListRowSet.getString("grade")))
                          courseNumberSet.add(gradeListRowSet.getString("course_number"));
                          gradeListRowSet.next();
                        }
                this.setCoreCourse(); // set core course list
                int coreSize = coreCourse.size();  // store the size of core course
                coreCourse.retainAll(courseNumberSet); // get the intersection of 2 sets
                // examine whether all core courses are taken by checking the size
                if (coreCourse.size() != coreSize) { // now coreCourse is the intersection
                        return false;
                }

            // 2.check the total number of course if it is at least 5.
                if (courseNumberSet.size() < 5) {
                        return false;
                }

        } catch (SQLException e) {
                System.out.println("A SQLException occurred.");
                e.printStackTrace();
        }
        // return true if it passes all conditions
        return true;
}
```

2.2 Screenshots of the output

a. Start program

- The table display the course list information in current database

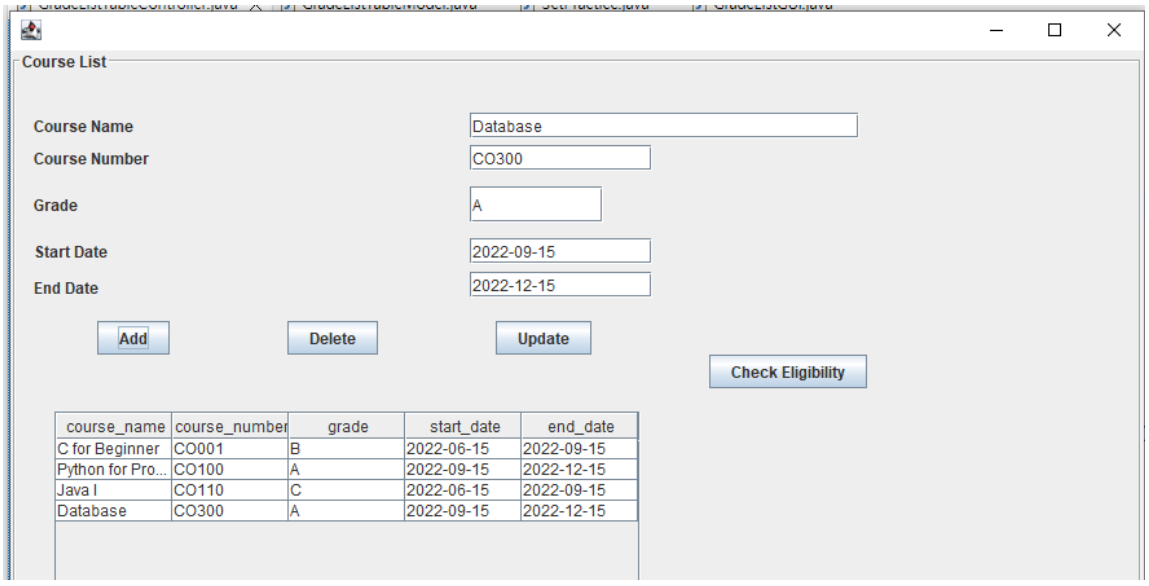


Course Name	
Course Number	
Grade	
Start Date	
End Date	

course_name	course_number	grade	start_date	end_date
C for Beginner	CO001	B	2022-06-15	2022-09-15
Python for Pro...	CO100	A	2022-09-15	2022-12-15

b. Add a new course information taken

- new rows are added the table as below

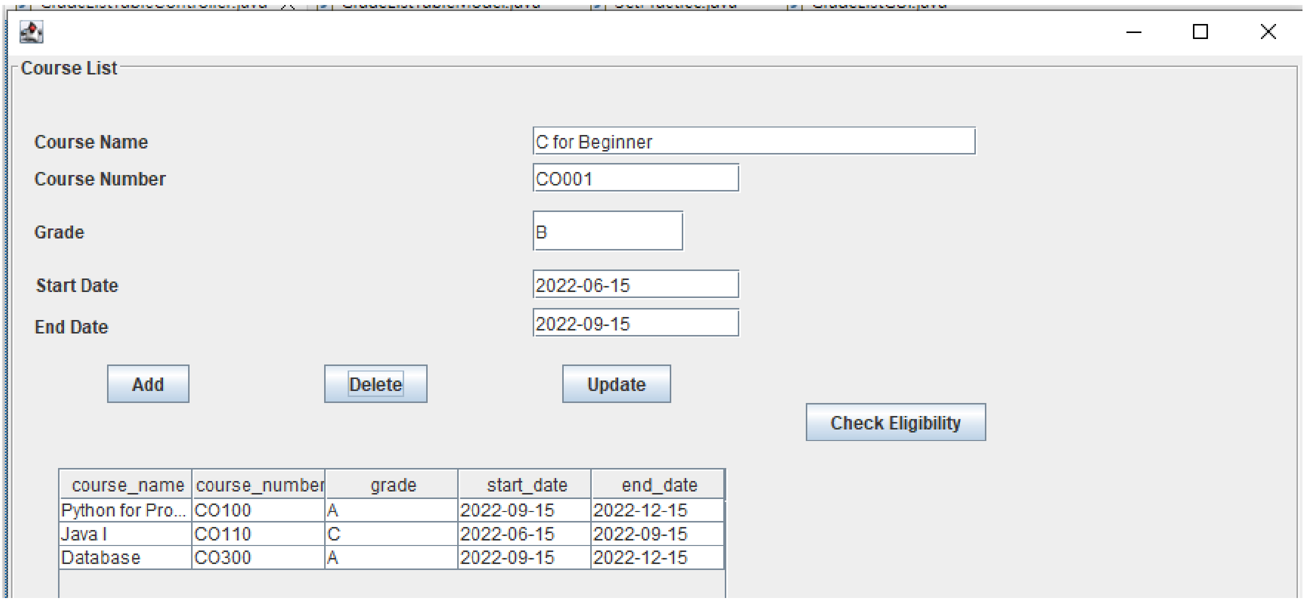


Course Name	Database
Course Number	CO300
Grade	A
Start Date	2022-09-15
End Date	2022-12-15

course_name	course_number	grade	start_date	end_date
C for Beginner	CO001	B	2022-06-15	2022-09-15
Python for Pro...	CO100	A	2022-09-15	2022-12-15
Java I	CO110	C	2022-06-15	2022-09-15
Database	CO300	A	2022-09-15	2022-12-15

c. Delete a selected row

- action: select the first row in the table and click Delete button.
- result: The row is deleted



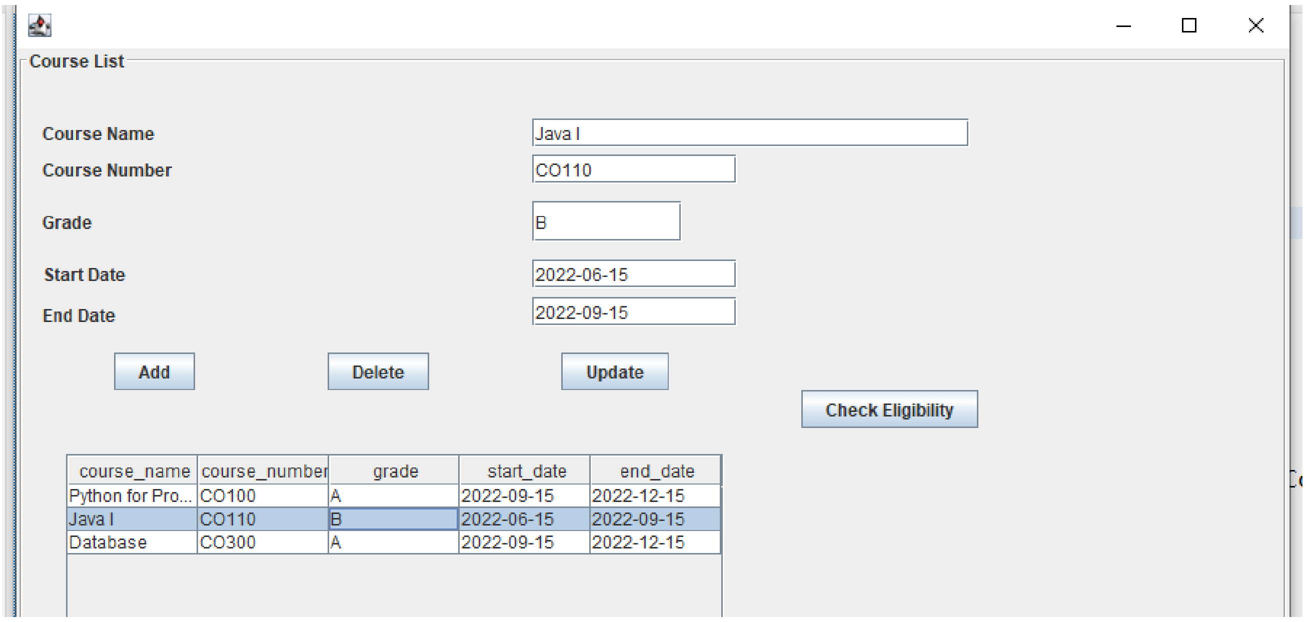
Course List

Course Name	C for Beginner
Course Number	CO001
Grade	B
Start Date	2022-06-15
End Date	2022-09-15

course_name	course_number	grade	start_date	end_date
Python for Pro...	CO100	A	2022-09-15	2022-12-15
Java I	CO110	C	2022-06-15	2022-09-15
Database	CO300	A	2022-09-15	2022-12-15

d. Update a selected row

- action: select the second row and input the grade 'B' in the Grade textfield. then click Update button.
- result: The grade of second row is updated to 'B' from 'C'



Course List

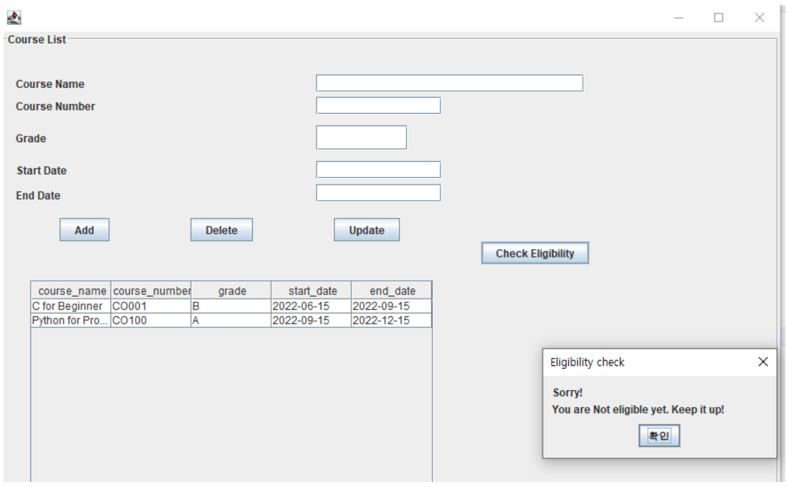
Course Name	Java I
Course Number	CO110
Grade	B
Start Date	2022-06-15
End Date	2022-09-15

course_name	course_number	grade	start_date	end_date
Python for Pro...	CO100	A	2022-09-15	2022-12-15
Java I	CO110	B	2022-06-15	2022-09-15
Database	CO300	A	2022-09-15	2022-12-15

e. Check Eligibility

- action: click Check Eligibility button

Case 1. Not enough courses and core course taken → It says "Sorry! Not eligible"



Course Name

Course Number

Grade

Start Date

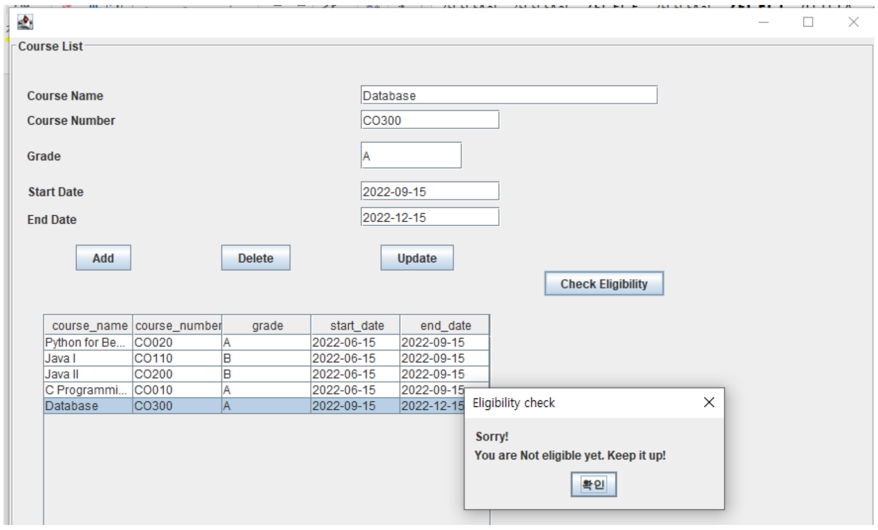
End Date

course_name	course_number	grade	start_date	end_date
C for Beginner	CO001	B	2022-06-15	2022-09-15
Python for Pro...	CO100	A	2022-09-15	2022-12-15

Eligibility check

Sorry!
You are Not eligible yet. Keep it up!

Case 2. 5 courses taken with at least B grade but lack of core course taken → It says "Sorry! Not eligible"



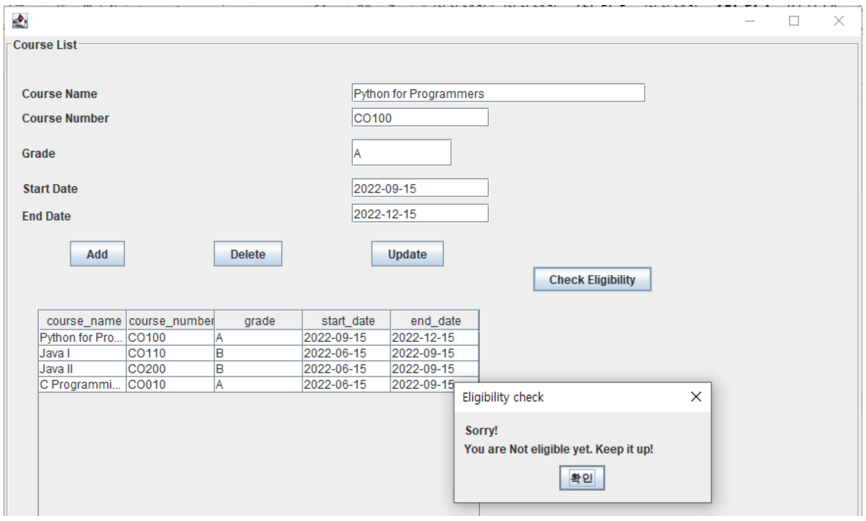
Course Name	Database
Course Number	CO300
Grade	A
Start Date	2022-09-15
End Date	2022-12-15

course_name	course_number	grade	start_date	end_date
Python for Be...	CO020	A	2022-06-15	2022-09-15
Java I	CO110	B	2022-06-15	2022-09-15
Java II	CO200	B	2022-06-15	2022-09-15
C Programmi...	CO010	A	2022-06-15	2022-09-15
Database	CO300	A	2022-09-15	2022-12-15

Eligibility check

Sorry!
You are Not eligible yet. Keep it up!

Case 3. All core course taken but not enough course taken → It says "Sorry! Not eligible"



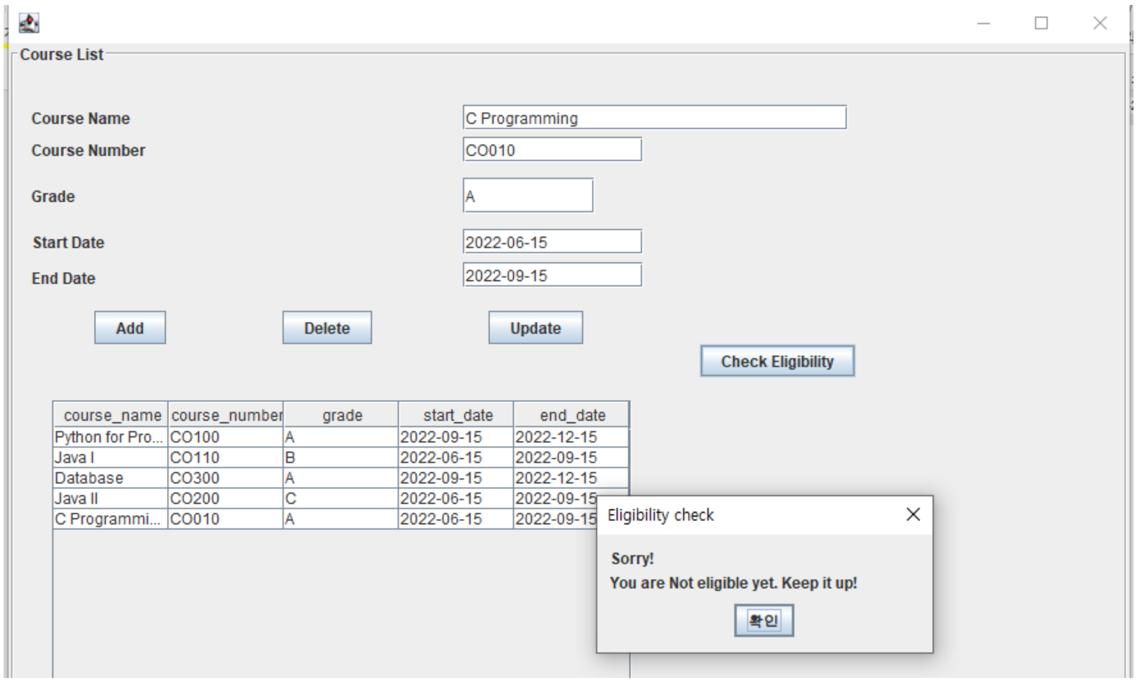
Course Name	Python for Programmers
Course Number	CO100
Grade	A
Start Date	2022-09-15
End Date	2022-12-15

course_name	course_number	grade	start_date	end_date
Python for Pro...	CO100	A	2022-09-15	2022-12-15
Java I	CO110	B	2022-06-15	2022-09-15
Java II	CO200	B	2022-06-15	2022-09-15
C Programmi...	CO010	A	2022-06-15	2022-09-15

Eligibility check

Sorry!
You are Not eligible yet. Keep it up!

Case 4. 5 Course taken and All core course taken but the grade below B → It says "Sorry! Not eligible"

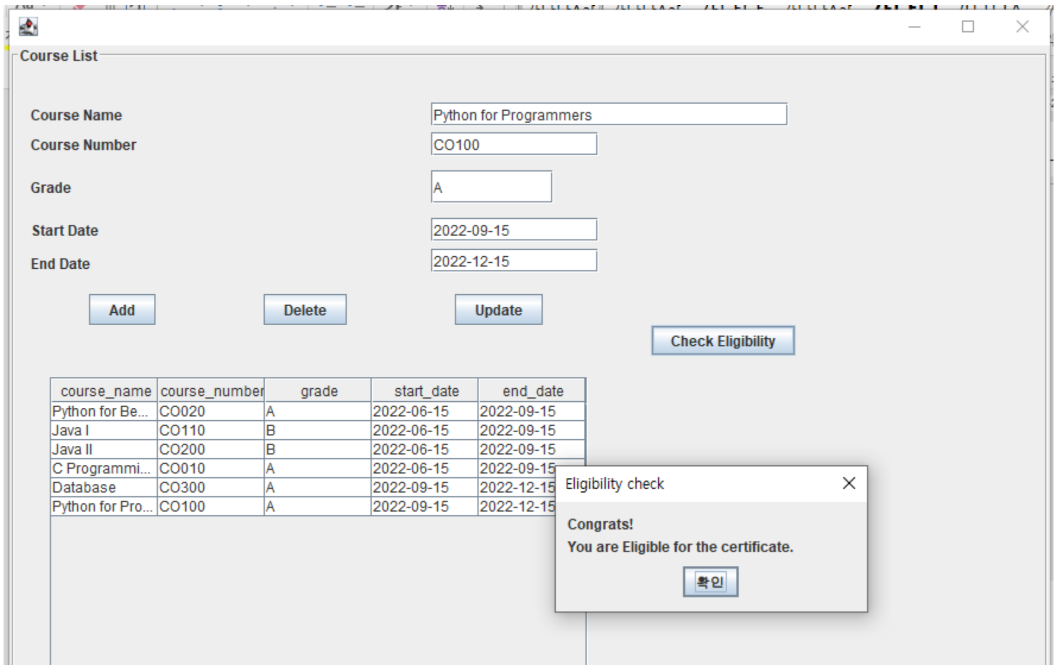


Course Name	C Programming
Course Number	CO010
Grade	A
Start Date	2022-06-15
End Date	2022-09-15

course_name	course_number	grade	start_date	end_date
Python for Pro...	CO100	A	2022-09-15	2022-12-15
Java I	CO110	B	2022-06-15	2022-09-15
Database	CO300	A	2022-09-15	2022-12-15
Java II	CO200	C	2022-06-15	2022-09-15
C Programmi...	CO010	A	2022-06-15	2022-09-15

Sorry!
You are Not eligible yet. Keep it up!

Case 5. All requirement clear → "Congrats! You are eligible"



Course Name	Python for Programmers
Course Number	CO100
Grade	A
Start Date	2022-09-15
End Date	2022-12-15

course_name	course_number	grade	start_date	end_date
Python for Be...	CO020	A	2022-06-15	2022-09-15
Java I	CO110	B	2022-06-15	2022-09-15
Java II	CO200	B	2022-06-15	2022-09-15
C Programmi...	CO010	A	2022-06-15	2022-09-15
Database	CO300	A	2022-09-15	2022-12-15
Python for Pro...	CO100	A	2022-09-15	2022-12-15

Congrats!
You are Eligible for the certificate.

3. Conclusion

- My grade management program is developed to check whether a student is eligible for a certificate.
- It has features such as add, delete, update courses by user and check eligibility based on the certificate course requirement.
- This program is implemented using java with MVC pattern and following technologies are used.
  - swing for implementation GUI
  - JDBC for database connection
  - Collection (HashSet structure) for checking core courses.

4. References

Java Programming II (CMPR.X412) Lecture materials by Radhika Grover

MySQLWithNetBeansAndJDBCTutorial.pdf
6634_Chapter06_Inheritance.pdf
6634_Chapter09_GUI_Programming.pdf
6634_Chapter12_Generics+Collections.pdf

Java SE Documentation

https://docs.oracle.com/javase/7/docs/api/javax/sql/rowset/CachedRowSet.html
https://docs.oracle.com/javase/7/docs/api/javax/swing/table/AbstractTableModel.html