UML Document

For

Course Registration System

Prepared By:

Mengyun Xie

Table of Contents

1. UML 3

2. Descriptions 4

2.1 UML of the User Interface. 4

2.2 UML of Components and Interface. 5

2.3 UML of Entity. 6

2.4 UML of Database related. 7

1. UML

The course registration system provides users with a service for course registration, including enrolling in courses, dropping courses, creating courses, and processing student requests to register for courses. To better understand and visualize the structure of the system, we will present and describe the UML of the course registration system in the following content.

Here is the whole picture of the UML.

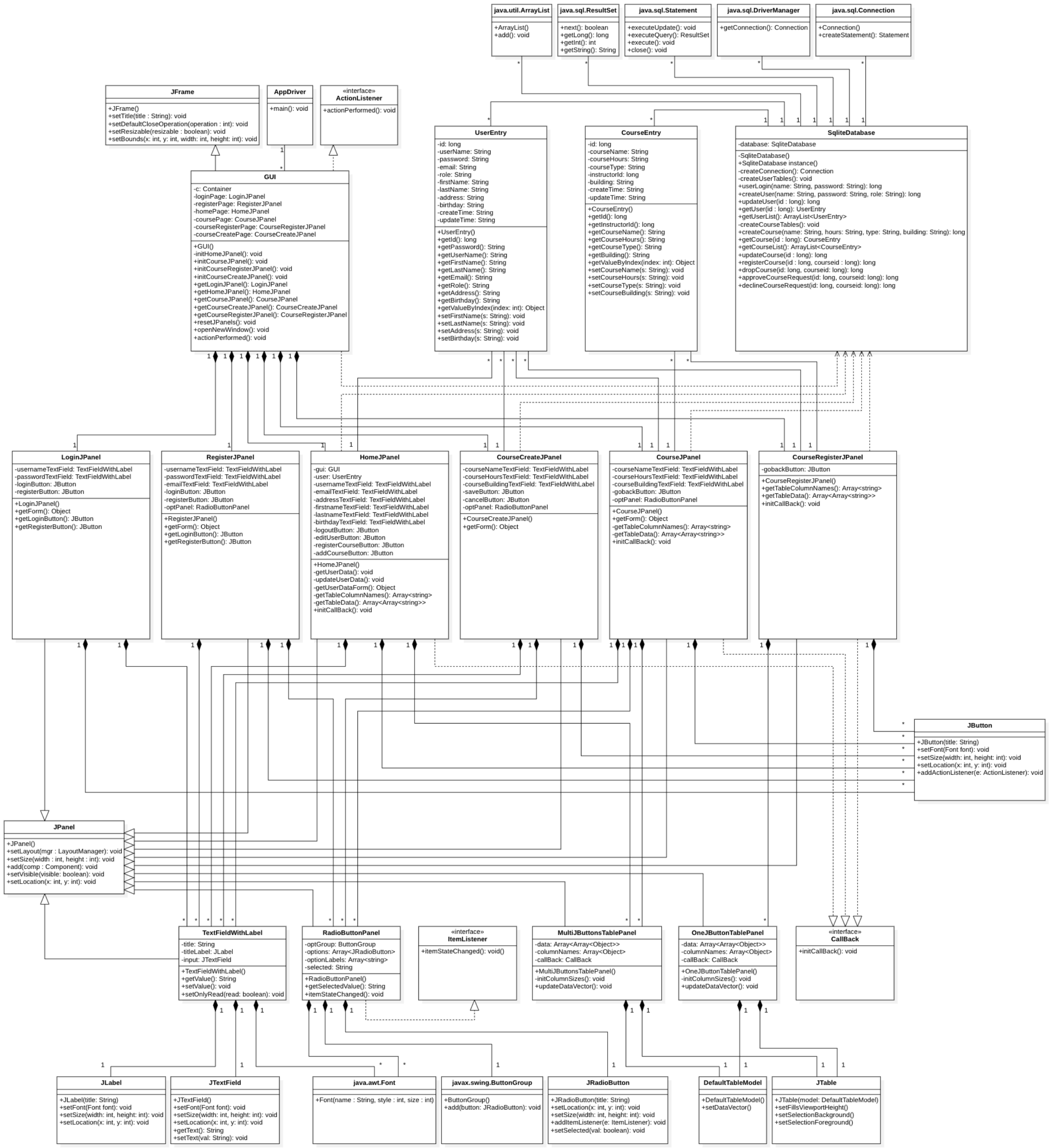
java.util.ArrayList
+ArrayList()
+add(): void
java.sql.ResultSet
+next(): boolean
+getLong(): long
+getInt(): int
+getString(): String
java.sql.Statement
+executeUpdate(): void
+executeQuery(): ResultSet
+execute(): void
+close(): void
java.sql.DriverManager
+getConnection(): Connection
java.sql.Connection
+Connection()
+createStatement(): Statement
JFrame
+JFrame()
+setTitle(title : String): void
+setDefaultCloseOperation(operation : int): void
+setResizable(resizable : boolean): void
+setBounds(x: int, y: int, width: int, height: int): void
AppDriver
+main(): void
«interface»
ActionListener
+actionPerformed(): void
GUI
-c: Container
-loginPage: LoginJPanel
-registerPage: RegisterJPanel
-homePage: HomeJPanel
-coursePage: CourseJPanel
-courseRegisterPage: CourseRegisterJPanel
-courseCreatePage: CourseCreateJPanel
+GUI()
-initHomeJPanel(): void
+initCourseJPanel(): void
+initCourseRegisterJPanel(): void
+initCourseCreateJPanel(): void
+getLoginJPanel(): LoginJPanel
+getHomeJPanel(): HomeJPanel
+getCourseJPanel(): CourseJPanel
+getCourseCreateJPanel(): CourseCreateJPanel
+getCourseRegisterJPanel(): CourseRegisterJPanel
+resetJPanels(): void
+openNewWindow(): void
+actionPerformed(): void
UserEntry
-id: long
-userName: String
-password: String
-email: String
-role: String
-firstName: String
-lastName: String
-address: String
-birthday: String
-createTime: String
-updateTime: String
+UserEntry()
+getId(): long
+getPassword(): String
+getUserName(): String
+getFirstName(): String
+getLastName(): String
+getEmail(): String
+getRole(): String
+getAddress(): String
+getBirthday(): String
+getValueByIndex(index: int): Object
+setFirstName(s: String): void
+setLastName(s: String): void
+setAddress(s: String): void
+setBirthday(s: String): void
CourseEntry
-id: long
-courseName: String
-courseHours: String
-courseType: String
-instructorId: long
-building: String
-createTime: String
-updateTime: String
+CourseEntry()
+getId(): long
+getInstructorId(): long
+getCourseName(): String
+getCourseHours(): String
+getCourseType(): String
+getBuilding(): String
+getValueByIndex(index: int): Object
+setCourseName(s: String): void
+setCourseHours(s: String): void
+setCourseType(s: String): void
+setCourseBuilding(s: String): void
SqliteDatabase
-database: SqliteDatabase
-SqliteDatabase()
+SqliteDatabase instance()
-createConnection(): Connection
-createUserTables(): void
+userLogin(name: String, password: String): long
+createUser(name: String, password: String, role: String): long
+updateUser(id : long): long
+getUser(id : long): UserEntry
+getUserList(): ArrayList<UserEntry>
-createCourseTables(): void
+createCourse(name: String, hours: String, type: String, building: String): long
+getCourse(id : long): CourseEntry
+getCourseList(): ArrayList<CourseEntry>
+updateCourse(id : long): long
+registerCourse(id : long, courseid : long): long
+dropCourse(id: long, courseid: long): long
+approveCourseRequest(id: long, courseid: long): long
+declineCourseRequest(id: long, courseid: long): long
LoginJPanel
-usernameTextField: TextFieldWithLabel
-passwordTextField: TextFieldWithLabel
-loginButton: JButton
-registerButton: JButton
+LoginJPanel()
+getForm(): Object
+getLoginButton(): JButton
+getRegisterButton(): JButton
RegisterJPanel
-usernameTextField: TextFieldWithLabel
-passwordTextField: TextFieldWithLabel
-emailTextField: TextFieldWithLabel
-loginButton: JButton
-registerButton: JButton
-optPanel: RadioButtonPanel
+RegisterJPanel()
+getForm(): Object
+getLoginButton(): JButton
+getRegisterButton(): JButton
HomeJPanel
-gui: GUI
-user: UserEntry
-usernameTextField: TextFieldWithLabel
-emailTextField: TextFieldWithLabel
-addressTextField: TextFieldWithLabel
-firstnameTextField: TextFieldWithLabel
-lastnameTextField: TextFieldWithLabel
-birthdayTextField: TextFieldWithLabel
-logoutButton: JButton
-editUserButton: JButton
-registerCourseButton: JButton
-addCourseButton: JButton
+HomeJPanel()
-getUserData(): void
-updateUserData(): void
-getUserDataForm(): Object
-getTableColumnNames(): Array<string>
-getTableData(): Array<Array<string>>
+initCallBack(): void
CourseCreateJPanel
-courseNameTextField: TextFieldWithLabel
-courseHoursTextField: TextFieldWithLabel
-courseBuildingTextField: TextFieldWithLabel
-saveButton: JButton
-cancelButton: JButton
-optPanel: RadioButtonPanel
+CourseCreateJPanel()
+getForm(): Object
CourseJPanel
-courseNameTextField: TextFieldWithLabel
-courseHoursTextField: TextFieldWithLabel
-courseBuildingTextField: TextFieldWithLabel
-gobackButton: JButton
-optPanel: RadioButtonPanel
+CourseJPanel()
+getForm(): Object
-getTableColumnNames(): Array<string>
-getTableData(): Array<Array<string>>
+initCallBack(): void
CourseRegisterJPanel
-gobackButton: JButton
+CourseRegisterJPanel()
+getTableColumnNames(): Array<string>
+getTableData(): Array<Array<string>>
+initCallBack(): void
JButton
+JButton(title: String)
+setFont(Font font): void
+setSize(width: int, height: int): void
+setLocation(x: int, y: int): void
+addActionListener(e: ActionListener): void
JPanel
+JPanel()
+setLayout(mgr : LayoutManager): void
+setSize(width : int, height : int): void
+add(comp : Component): void
+setVisible(visible: boolean): void
+setLocation(x: int, y: int): void
TextFieldWithLabel
-title: String
-titleLabel: JLabel
-input: JTextField
+TextFieldWithLabel()
+getValue(): String
+setValue(): void
+setOnlyRead(read: boolean): void
RadioButtonPanel
-optGroup: ButtonGroup
-options: Array<JRadioButton>
-optionLabels: Array<string>
-selected: String
+RadioButtonPanel()
+getSelectedValue(): String
+itemStateChanged(): void
«interface»
ItemListener
+itemStateChanged(): void()
MultiJButtonsTablePanel
-data: Array<Array<Object>>
-columnNames: Array<Object>
-callBack: CallBack
+MultiJButtonsTablePanel()
-initColumnSizes(): void
+updateDataVector(): void
OneJButtonTablePanel
-data: Array<Array<Object>>
-columnNames: Array<Object>
-callBack: CallBack
+OneJButtonTablePanel()
-initColumnSizes(): void
+updateDataVector(): void
«interface»
CallBack
+initCallBack(): void
JLabel
+JLabel(title: String)
+setFont(Font font): void
+setSize(width: int, height: int): void
+setLocation(x: int, y: int): void
JTextField
+JTextField()
+setFont(Font font): void
+setSize(width: int, height: int): void
+setLocation(x: int, y: int): void
+getText(): String
+setText(val: String): void
java.awt.Font
+Font(name : String, style : int, size : int)
javax.swing.ButtonGroup
+ButtonGroup()
+add(button: JRadioButton): void
JRadioButton
+JRadioButton(title: String)
+setLocation(x: int, y: int): void
+setSize(width: int, height: int): void
+addItemListener(e: ItemListener): void
+setSelected(val: boolean): void
DefaultTableModel
+DefaultTableModel()
+setDataVector()
JTable
+JTable(model: DefaultTableModel)
+setFillsViewportHeight()
+setSelectionBackground()
+setSelectionForeground()

Figure 1: The UML of the system.

2. Descriptions

2.1 UML of the User Interface.

This is the part of UML that describes the user interface for this system.

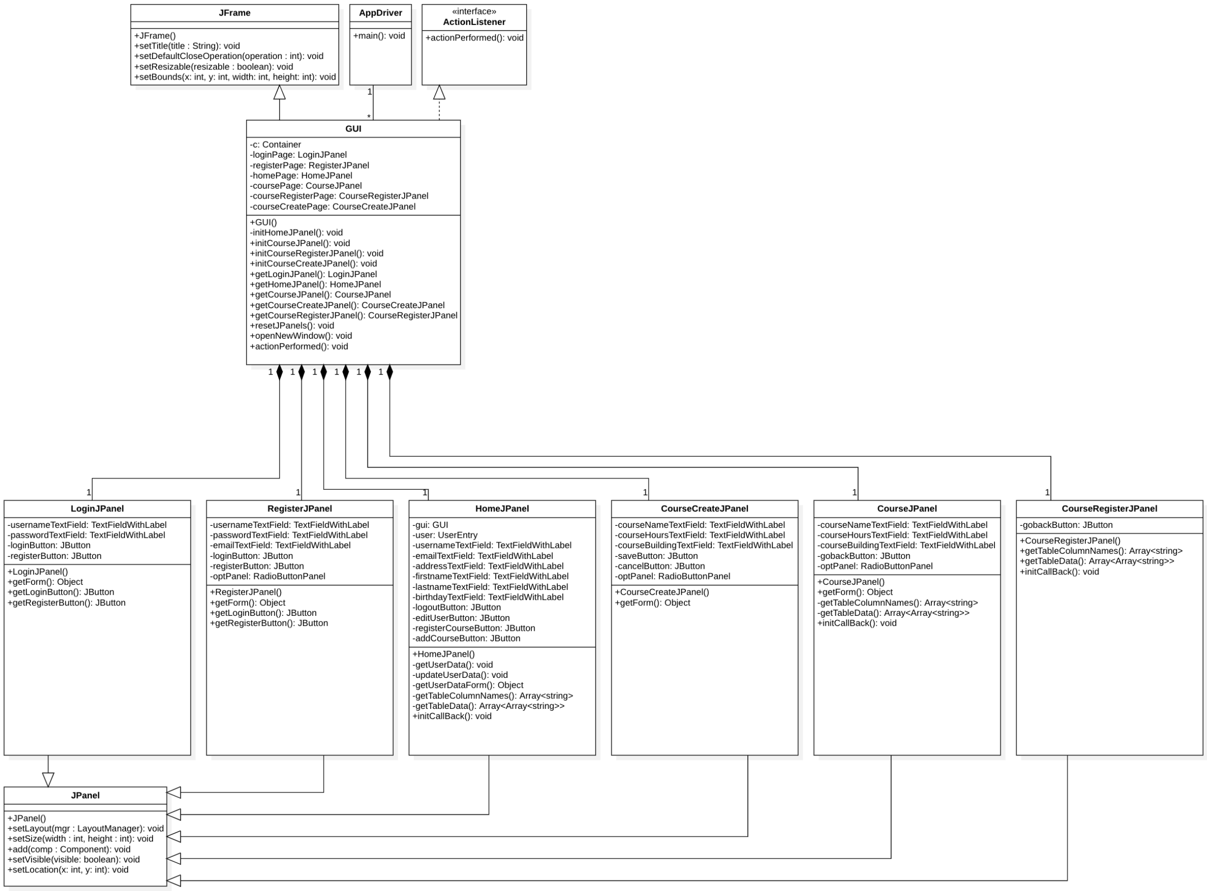
JFrame
+JFrame()
+setTitle(title : String): void
+setDefaultCloseOperation(operation : int): void
+setResizable(resizable : boolean): void
+setBounds(x: int, y: int, width: int, height: int): void
AppDriver
+main(): void
«interface»
ActionListener
+actionPerformed(): void
1
*
GUI
-c: Container
-loginPage: LoginJPanel
-registerPage: RegisterJPanel
-homePage: HomeJPanel
-coursePage: CourseJPanel
-courseRegisterPage: CourseRegisterJPanel
-courseCreatePage: CourseCreateJPanel
+GUI()
-initHomeJPanel(): void
+initCourseJPanel(): void
+initCourseRegisterJPanel(): void
+initCourseCreateJPanel(): void
+getLoginJPanel(): LoginJPanel
+getHomeJPanel(): HomeJPanel
+getCourseJPanel(): CourseJPanel
+getCourseCreateJPanel(): CourseCreateJPanel
+getCourseRegisterJPanel(): CourseRegisterJPanel
+resetJPanels(): void
+openNewWindow(): void
+actionPerformed(): void
1 1 1 1 1 1
1
LoginJPanel
-usernameTextField: TextFieldWithLabel
-passwordTextField: TextFieldWithLabel
-loginButton: JButton
-registerButton: JButton
+LoginJPanel()
+getForm(): Object
+getLoginButton(): JButton
+getRegisterButton(): JButton
1
RegisterJPanel
-usernameTextField: TextFieldWithLabel
-passwordTextField: TextFieldWithLabel
-emailTextField: TextFieldWithLabel
-loginButton: JButton
-registerButton: JButton
-optPanel: RadioButtonPanel
+RegisterJPanel()
+getForm(): Object
+getLoginButton(): JButton
+getRegisterButton(): JButton
1
HomeJPanel
-gui: GUI
-user: UserEntry
-usernameTextField: TextFieldWithLabel
-emailTextField: TextFieldWithLabel
-addressTextField: TextFieldWithLabel
-firstnameTextField: TextFieldWithLabel
-lastnameTextField: TextFieldWithLabel
-birthdayTextField: TextFieldWithLabel
-logoutButton: JButton
-editUserButton: JButton
-registerCourseButton: JButton
-addCourseButton: JButton
+HomeJPanel()
-getUserData(): void
-updateUserData(): void
-getUserDataForm(): Object
-getTableColumnNames(): Array<string>
-getTableData(): Array<Array<string>>
+initCallBack(): void
1
CourseCreateJPanel
-courseNameTextField: TextFieldWithLabel
-courseHoursTextField: TextFieldWithLabel
-courseBuildingTextField: TextFieldWithLabel
-saveButton: JButton
-cancelButton: JButton
-optPanel: RadioButtonPanel
+CourseCreateJPanel()
+getForm(): Object
1
CourseJPanel
-courseNameTextField: TextFieldWithLabel
-courseHoursTextField: TextFieldWithLabel
-courseBuildingTextField: TextFieldWithLabel
-gobackButton: JButton
-optPanel: RadioButtonPanel
+CourseJPanel()
+getForm(): Object
-getTableColumnNames(): Array<string>
-getTableData(): Array<Array<string>>
+initCallBack(): void
1
CourseRegisterJPanel
-gobackButton: JButton
+CourseRegisterJPanel()
+getTableColumnNames(): Array<string>
+getTableData(): Array<Array<string>>
+initCallBack(): void
JPanel
+JPanel()
+setLayout(mgr : LayoutManager): void
+setSize(width : int, height : int): void
+add(comp : Component): void
+setVisible(visible: boolean): void
+setLocation(x: int, y: int): void

Figure 2: The UML of the User Interface.

In this system, the entry is the AppDriver, where the program creates an instance of the GUI class that is the system's main container, a JFrame. GUI inherits the attributes and methods of JFrame and implements the ActionListener interface to respond to button actions.

Here shows the user interface structure of the system.

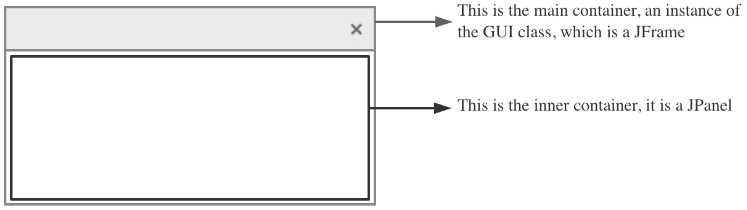
×
This is the main container, an instance of the GUI class, which is a JFrame
This is the inner container, it is a JPanel

Figure 3: The structure of the user interface.

And the inner container is an instance of corresponding JPanel class, such as LoginJPanel class, RegisterJPanel class, HomeJPanel class and so on. These JPanel instance serves as different pages, like login page, register page, home page and so on. Since these inner container as different pages, the relationship between GUI and one of these JPanel is one to one, as shown in Figure 1. Specifically, a GUI can only have one LoginJPanel, one RegisterJPanel, one HomeJPanel, one CourseJPanel, one CourseCreateJPanel, and one CourseRegisterJPanel.

There are the various classes used to build the user interface for this system.

Class Name	Description
AppDriver	The entry of the system.
GUI	The main container to contain different pages, it’s a JFrame.
LoginJPanel	A JPanel serves as a login page.
RegisterJPanel	A JPanel serves as a user register page.
HomeJPanel	A JPanel serves as a home page.
CourseJPanel	A JPanel serves as a course information page.
CourseCreateJPanel	A JPanel serves as a course create page.
CourseRegisterJPanel	A JPanel serves as a course register page.

2.2 UML of Components and Interface.

The following shows the UML of custom components and interface.

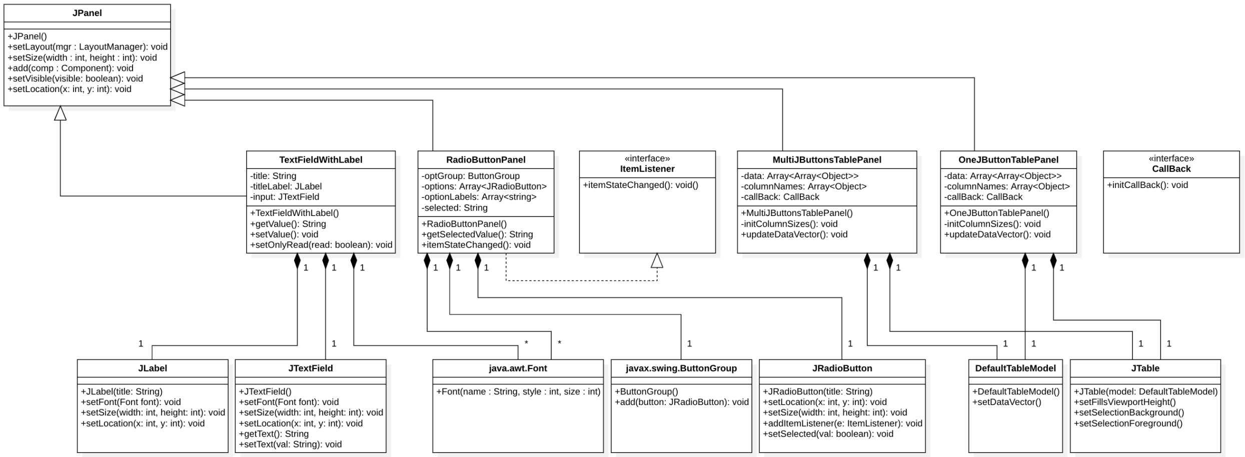
JPanel
+JPanel()
+setLayout(mgr : LayoutManager): void
+setSize(width : int, height : int): void
+add(comp : Component): void
+setVisible(visible: boolean): void
+setLocation(x: int, y: int): void
TextFieldWithLabel
-title: String
-titleLabel: JLabel
-input: JTextField
+TextFieldWithLabel()
+getValue(): String
+setValue(): void
+setOnlyRead(read: boolean): void
RadioButtonPanel
-optGroup: ButtonGroup
-options: Array<JRadioButton>
-optionLabels: Array<string>
-selected: String
+RadioButtonPanel()
+getSelectedValue(): String
+itemStateChanged(): void
«interface»
ItemListener
+itemStateChanged(): void()
MultiJButtonsTablePanel
-data: Array<Array<Object>>
-columnNames: Array<Object>
-callBack: CallBack
+MultiJButtonsTablePanel()
-initColumnSizes(): void
+updateDataVector(): void
OneJButtonTablePanel
-data: Array<Array<Object>>
-columnNames: Array<Object>
-callBack: CallBack
+OneJButtonTablePanel()
-initColumnSizes(): void
+updateDataVector(): void
«interface»
CallBack
+initCallBack(): void
JLabel
+JLabel(title: String)
+setFont(Font font): void
+setSize(width: int, height: int): void
+setLocation(x: int, y: int): void
JTextField
+JTextField()
+setFont(Font font): void
+setSize(width: int, height: int): void
+setLocation(x: int, y: int): void
+getText(): String
+setText(val: String): void
java.awt.Font
+Font(name : String, style : int, size : int)
javax.swing.ButtonGroup
+ButtonGroup()
+add(button: JRadioButton): void
JRadioButton
+JRadioButton(title: String)
+setLocation(x: int, y: int): void
+setSize(width: int, height: int): void
+addItemListener(e: ItemListener): void
+setSelected(val: boolean): void
DefaultTableModel
+DefaultTableModel()
+setDataVector()
JTable
+JTable(model: DefaultTableModel)
+setFillsViewportHeight()
+setSelectionBackground()
+setSelectionForeground()
1
*

Figure 4: The UML of custom components and interface.

Since the cells of the original JTable cannot contain button components, we encapsulate two different custom JTables, OneJButtonTablePanel and MultiJButtonsTablePanel according to the needs of the system. Since the system has many reusable forms: JLabel on the left and TextField on the right, in order to improve our code efficiency, we encapsulated TextFieldWithLabel to simplify code redundancy. RadioButtonPanel is also for the same purpose.

There are descriptions of the various custom components and interface classes.

Class Name	Description
TextFieldWithLabel	A JPanel contains a JLabel and JTextField serves as a custom JTextField.
RadioButtonPanel	A JPanel contains JRadioButton serves as a custom JRadioButton.
MultiJButtonsTablePanel	A JPanel contains a JTable, and two buttons serves as a custom JTable.
OneJButtonTablePanel	A JPanel contains a JTable, and one button serves as a custom JTable.
CallBack	An interface for the communication between different components and pages.

2.3 UML of Entity.

To implement the function of this system, we created two kinds of entities, user entity and course entity. The UserEntry and CourseEntry include the main information obtained from the database that needs to be displayed on the system.

This is the UML of these two kinds of entities.

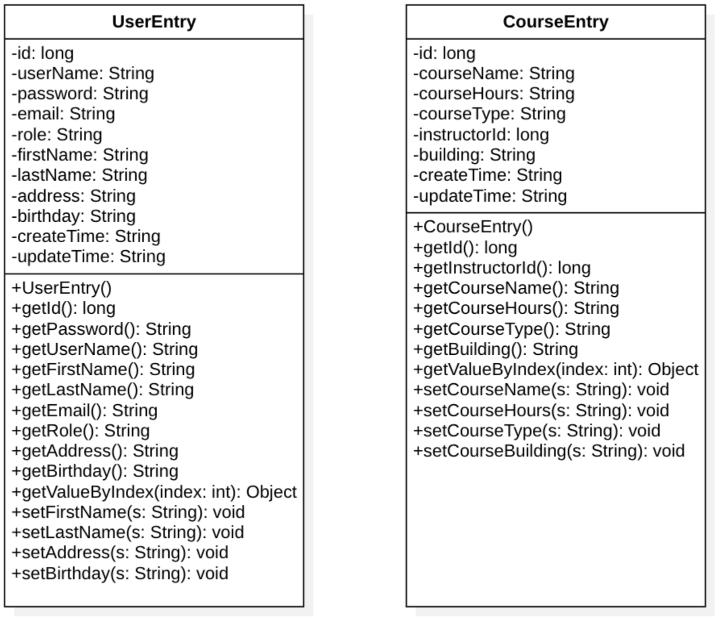

Figure 5: The UML of the UserEntry and CourseEntry.

The descriptions of these two entities.

Class Name	Description
UserEntry	This class encapsulates key information about a user and the corresponding methods for getting and setting that information.
CourseEntry	This class encapsulates key information about a course and the corresponding methods for getting and setting that information.

2.4 UML of Database related.

In the course registration system, we use the SQLite library to build the database functionality of the system. To better communicate with the database, we created the SqliteDatabase class as a bridge between the java program and the database. The SqliteDatabase class contains a number of methods, such as getUser, getCourse, createUser, createCourse, which are similar to the API.

The following shows the UML of SqliteDatabase.

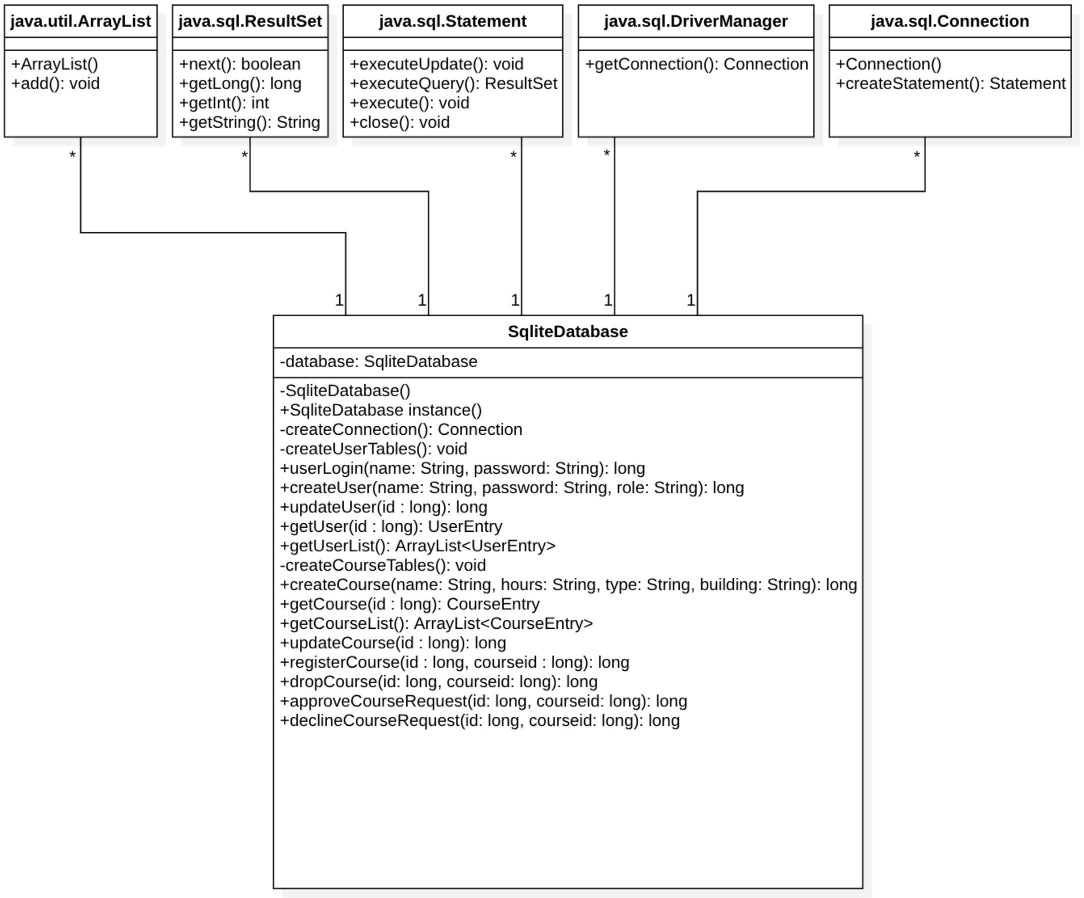
java.util.ArrayList
+ArrayList()
+add(): void
java.sql.ResultSet
+next(): boolean
+getLong(): long
+getInt(): int
+getString(): String
java.sql.Statement
+executeUpdate(): void
+executeQuery(): ResultSet
+execute(): void
+close(): void
java.sql.DriverManager
+getConnection(): Connection
java.sql.Connection
+Connection()
+createStatement(): Statement
*
*
*
*
*
1
1
1
1
1
SqliteDatabase
-database: SqliteDatabase
-SqliteDatabase()
+SqliteDatabase instance()
-createConnection(): Connection
-createUserTables(): void
+userLogin(name: String, password: String): long
+createUser(name: String, password: String, role: String): long
+updateUser(id : long): long
+getUser(id : long): UserEntry
+getUserList(): ArrayList<UserEntry>
-createCourseTables(): void
+createCourse(name: String, hours: String, type: String, building: String): long
+getCourse(id : long): CourseEntry
+getCourseList(): ArrayList<CourseEntry>
+updateCourse(id : long): long
+registerCourse(id : long, courseid : long): long
+dropCourse(id: long, courseid: long): long
+approveCourseRequest(id: long, courseid: long): long
+declineCourseRequest(id: long, courseid: long): long

Figure 6: The UML of SqliteDatabase.

There are some important methods of the SqliteDatabase class.

Method Name	Detail
createConnection	Create connection between the program and database.
createUser	Create a user data and insert to the User table.
getUser	Get a user data from the User table, return an instance of the UserEntry class.
getUserList	Get a list of user data from the User table, return a list of instances of the UserEntry class.
updateUser	Update a user data to the User table.
createCourse	Create a course data and insert to the Course table.
getCourse	Get a course data from the Course table, return an instance of the CourseEntry class.
getCourseList	Get a list of course data from the Course table, return a list of instances of the CourseEntry class.
updateCourse	Update a course data to the Course table.
dropCourse	Delete a data from the Enrolled table.
registerCourse	Add a data to the Requested table.
approveCourseRequest	Move a data from the Requested table to the Enrolled table.
declineCourseRequest	Delete a data from the Requested table.