Topics In Software Engineering

Assignment 02

ABDUL REHMAN , ZEESHAN SHAHID, ZIMAM AHMED
SP20-BSE-092@cuilahore.edu.pk

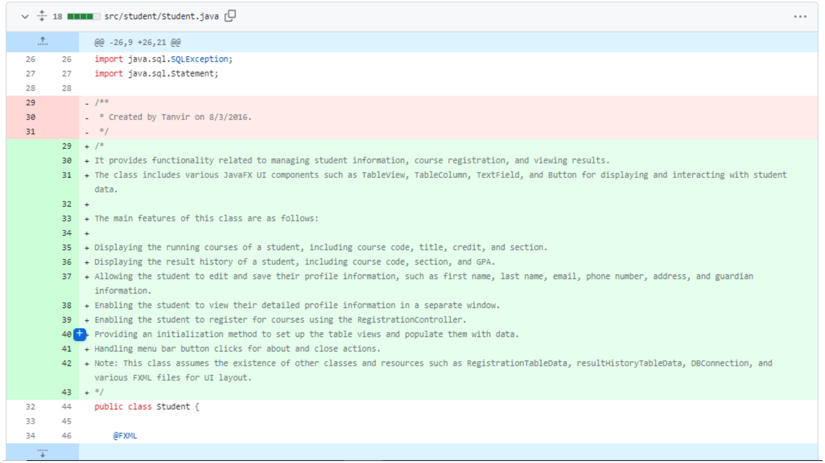
University of Comsats

June 3, 2023

Table of Contents

Table of Contents

Student Class

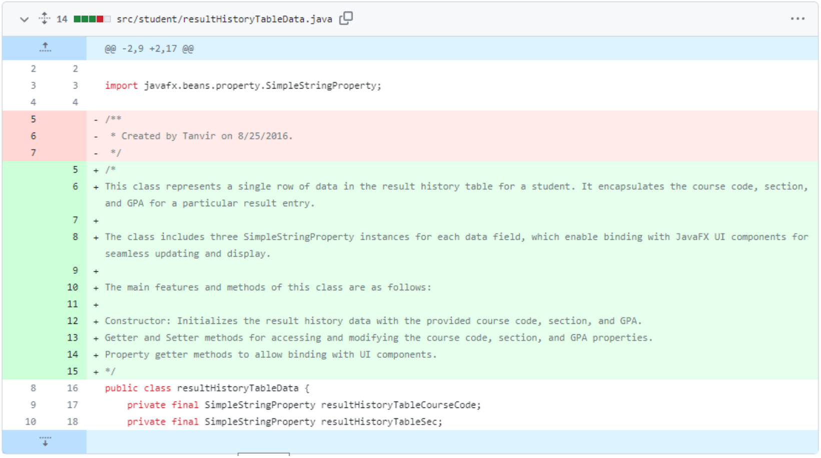


18 src/student/Student.java

```diff
@@ -26,9 +26,21 @@
 import java.sql.SQLException;
 import java.sql.Statement;
 
- /**
-  * Created by Tanvir on 8/3/2016.
-  */
+ /*
+ It provides functionality related to managing student information, course registration, and viewing results.
+ The class includes various JavaFX UI components such as TableView, TableColumn, TextField, and Button for displaying and interacting with student data.
+
+ The main features of this class are as follows:
+
+ Displaying the running courses of a student, including course code, title, credit, and section.
+ Displaying the result history of a student, including course code, section, and GPA.
+ Allowing the student to edit and save their profile information, such as first name, last name, email, phone number, address, and guardian information.
+ Enabling the student to view their detailed profile information in a separate window.
+ Enabling the student to register for courses using the RegistrationController.
+ Providing an initialization method to set up the table views and populate them with data.
+ Handling menu bar button clicks for about and close actions.
+ Note: This class assumes the existence of other classes and resources such as RegistrationTableData, resultHistoryTableData, DBConnection, and various FXML files for UI layout.
+ */
 public class Student {
 
     @FXML
```

Figure: Student Class Comments Added

resultHistoryTableData Class

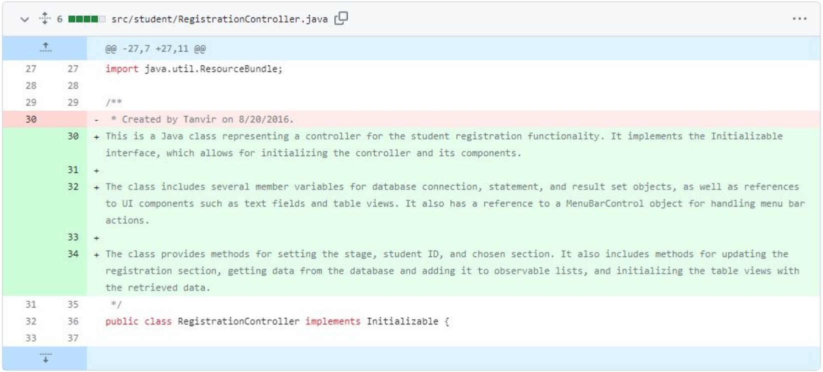


14 src/student/resultHistoryTableData.java

```
@@ -2,9 +2,17 @@

 import javafx.beans.property.SimpleStringProperty;

-/**
- * Created by Tanvir on 8/25/2016.
- */
+/*
+This class represents a single row of data in the result history table for a student. It encapsulates the course code, section, and GPA for a particular result entry.
+
+The class includes three SimpleStringProperty instances for each data field, which enable binding with JavaFX UI components for seamless updating and display.
+
+The main features and methods of this class are as follows:
+
+Constructor: Initializes the result history data with the provided course code, section, and GPA.
+Getter and Setter methods for accessing and modifying the course code, section, and GPA properties.
+Property getter methods to allow binding with UI components.
+*/
 public class resultHistoryTableData {
     private final SimpleStringProperty resultHistoryTableCourseCode;
     private final SimpleStringProperty resultHistoryTableSec;
```

Figure: ResultHistoryTableData Class Comments Added

RegistrationController Class

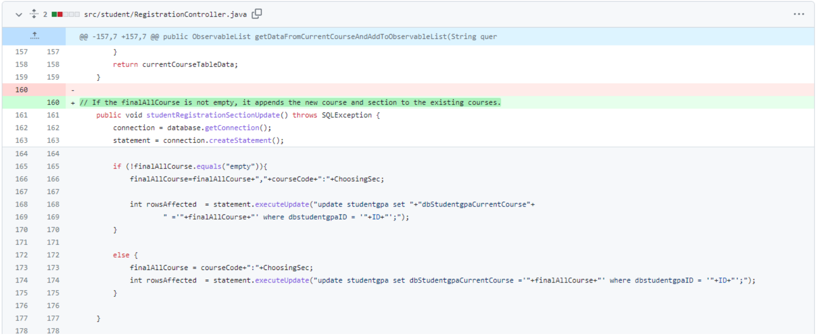


```
@@ -27,7 +27,11 @@
 import java.util.ResourceBundle;

 /**
-  * Created by Tanvir on 8/20/2016.
+ This is a Java class representing a controller for the student registration functionality. It implements the Initializable interface, which allows for initializing the controller and its components.
+
+ The class includes several member variables for database connection, statement, and result set objects, as well as references to UI components such as text fields and table views. It also has a reference to a MenuBarControl object for handling menu bar actions.
+
+ The class provides methods for setting the stage, student ID, and chosen section. It also includes methods for updating the registration section, getting data from the database and adding it to observable lists, and initializing the table views with the retrieved data.
  */
 public class RegistrationController implements Initializable {

```

Figure: RegistrationController Class Comments Added

Table of Contents

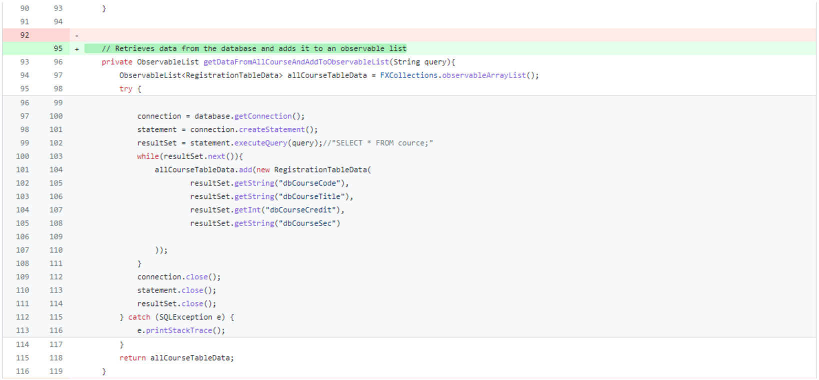
studentRegistrationSectionUpdate()



```
2 src/student/RegistrationController.java
@@ -157,7 +157,7 @@ public ObservableList getDataFromCurrentCourseAndAddToObservableList(String quer
        }
        return currentCourseTableData;
    }
-
+ // If the finalAllCourse is not empty, it appends the new course and section to the existing courses.
    public void studentRegistrationSectionUpdate() throws SQLException {
        connection = database.getConnection();
        statement = connection.createStatement();

        if (!finalAllCourse.equals("empty")){
            finalAllCourse=finalAllCourse+","+courseCode+":"+ChoosingSec;

            int rowsAffected  = statement.executeUpdate("update studentgpa set "+"dbStudentgpaCurrentCourse"+
                    " ='"+finalAllCourse+"' where dbstudentgpaID = '"+ID+"';");
        }

        else {
            finalAllCourse = courseCode+":"+ChoosingSec;
            int rowsAffected  = statement.executeUpdate("update studentgpa set dbStudentgpaCurrentCourse ='"+finalAllCourse+"' where dbstudentgpaID = '"+ID+"';");
        }

    }

```

Figure: studentRegistrationSectionUpdate() Comment Added

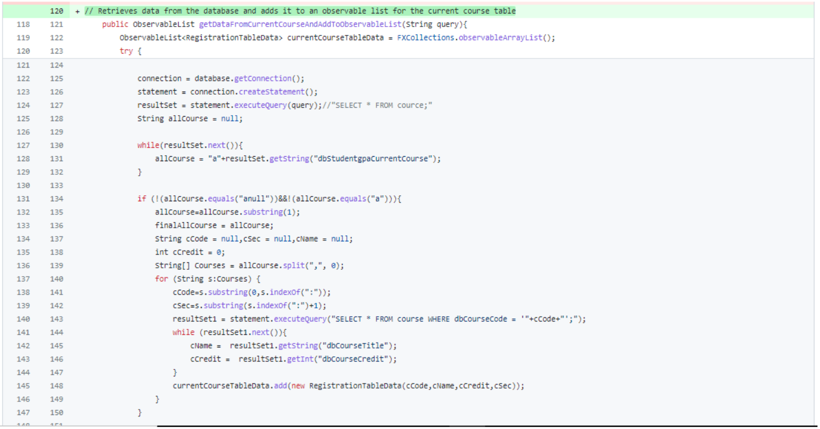
getDataFromCurrentCourseAndAddToObservableList(Strin query)



```diff
     }
 
-
+    // Retrieves data from the database and adds it to an observable list
     private ObservableList getDataFromAllCourseAndAddToObservableList(String query){
         ObservableList<RegistrationTableData> allCourseTableData = FXCollections.observableArrayList();
         try {
 
             connection = database.getConnection();
             statement = connection.createStatement();
             resultSet = statement.executeQuery(query);//"SELECT * FROM cource;"
             while(resultSet.next()){
                 allCourseTableData.add(new RegistrationTableData(
                         resultSet.getString("dbCourseCode"),
                         resultSet.getString("dbCourseTitle"),
                         resultSet.getInt("dbCourseCredit"),
                         resultSet.getString("dbCourseSec")
 
                 ));
             }
             connection.close();
             statement.close();
             resultSet.close();
         } catch (SQLException e) {
             e.printStackTrace();
         }
         return allCourseTableData;
     }
```

Figure: getDataFromCurrentCourseAndAddToObservableList(String query) Comment Added

getDataFromAllCourseAndAddToObservableList(String query)

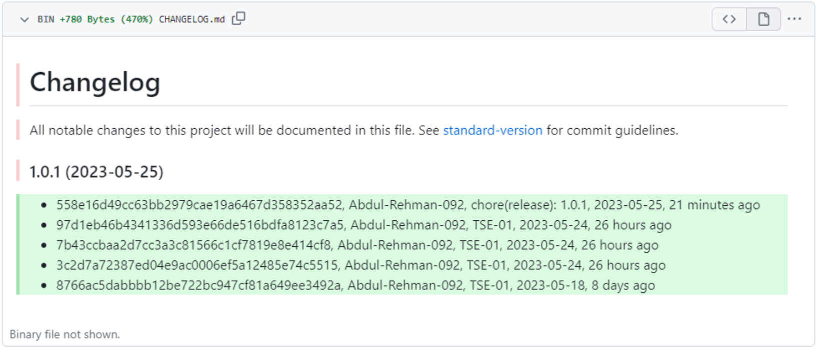


```
+ // Retrieves data from the database and adds it to an observable list for the current course table
    public ObservableList getDataFromCurrentCourseAndAddToObservableList(String query){
        ObservableList<RegistrationTableData> currentCourseTableData = FXCollections.observableArrayList();
        try {

            connection = database.getConnection();
            statement = connection.createStatement();
            resultSet = statement.executeQuery(query);//"SELECT * FROM cource;"
            String allCourse = null;

            while(resultSet.next()){
                allCourse = "a"+resultSet.getString("dbStudentgpaCurrentCourse");
            }

            if (!(allCourse.equals("anull"))&&!(allCourse.equals("a"))){
                allCourse=allCourse.substring(1);
                finalAllCourse = allCourse;
                String cCode = null,cSec = null,cName = null;
                int cCredit = 0;
                String[] Courses = allCourse.split(",", 0);
                for (String s:Courses) {
                    cCode=s.substring(0,s.indexOf(":"));
                    cSec=s.substring(s.indexOf(":")+1);
                    resultSet1 = statement.executeQuery("SELECT * FROM course WHERE dbCourseCode = '"+cCode+"';");
                    while (resultSet1.next()){
                        cName =  resultSet1.getString("dbCourseTitle");
                        cCredit =  resultSet1.getInt("dbCourseCredit");
                    }
                    currentCourseTableData.add(new RegistrationTableData(cCode,cName,cCredit,cSec));
                }
            }
```

Figure: getDataFromAllCourseAndAddToObservableList(String query) Comment Added

Table of Contents

ChangeLog File


BIN +780 Bytes (470%) CHANGELOG.md

Changelog

All notable changes to this project will be documented in this file. See standard-version for commit guidelines.

1.0.1 (2023-05-25)

- 558e16d49cc63bb2979cae19a6467d358352aa52, Abdul-Rehman-092, chore(release): 1.0.1, 2023-05-25, 21 minutes ago
- 97d1eb46b4341336d593e66de516bdfa8123c7a5, Abdul-Rehman-092, TSE-01, 2023-05-24, 26 hours ago
- 7b43ccbaa2d7cc3a3c81566c1cf7819e8e414cf8, Abdul-Rehman-092, TSE-01, 2023-05-24, 26 hours ago
- 3c2d7a72387ed04e9ac0006ef5a12485e74c5515, Abdul-Rehman-092, TSE-01, 2023-05-24, 26 hours ago
- 8766ac5dabbbb12be722bc947cf81a649ee3492a, Abdul-Rehman-092, TSE-01, 2023-05-18, 8 days ago

Binary file not shown.

Figure: Change Log Update

https://github.com/Abdul-Rehman-092/SRS.git

Thank You for Your Attention!