# TASK1 DESCRIPTION

This is basically a student’s grade processing system where user is given three options out of insert, search and update the student’s data from database. User enters student’s marks and system calculates total result and grade and are then stored into database. User can also search or update this data. Whole program is completed by following steps:

**HOME SCREEN**

1. Home screen is displayed where three options are given to use. Insert Data, Search Data and Update Data. Close the program by clicking red cross button on right top corner.

**INSERT FUNCTION**

1. By clicking on insert button on home screen, insert screen is opened with insert and back button. Insert button inserts data into database and back button takes user back to home screen.
2. In insert screen, users must give input of student marks. Each input has individual validation function, e.g.: id needs to be 8 digits at least, marks needs to be between 0 -100 or else it gives error warning.
3. After passing all the validations, program checks if database already exists or not. If not then it creates a database and table, and data is inserted into it. If database exists, data is inserted into the table.
4. After inserting, all the data is displayed in dialog box.

**SEARCH FUNCTION**

1. By clicking on search button on home screen, search screen is opened with search and back button. Search button searches for data into database with different parameters, e.g.: id, marks, result, and grade. Back button takes user back to home screen.
2. In search screen, user can search for data based on student id, student name, marks, result and grade.

**UPDATE FUNCTION**

1. By clicking on update button on home screen, update screen is opened with update button and back button to update the data or to go back to home screen respectively.
2. While opening the update screen, student ids are fetched from the database and are added to combo box. User can choose student id from drop down list to edit marks of that student. Once id is chosen, all the data belonging to that id is displayed on all text fields except id and grade as it cannot be edited.
3. All the validations that applies to insert screen also applies to update text fields. E.g.: ID must be 8 digits, marks should be between 0-100.
4. Updated record is displayed in dialog box.

# TASK1 OUTPUT

|  |  |
| --- | --- |
| **Home Screen** | |
| **Test Data** | **Screenshot** |
| Starts the application and on home screen shows three options |  |
| **Insert Function** | |
| Click on insert button to go in insert screen |  |
| Input data-  Id – 11111111  Name – Vismay  Quiz – 90  Assignment 1 - 89  Assignment 2 – 85.5  Assignment 3 – 89  Exam – 90  Expected result –  Database students\_records will be created if doesn’t exist already  Table “java2” will be created  This data will be inserted into the table and will be displayed as well |  |
| Input data-  Id – 2222  Name – Gami  Quiz – 80  Assignment 1 - 89  Assignment 2 - 67  Assignment 3 – 85  Exam – 89  Expected result –  It will give error of ID as it has to be atleast 8 digits |  |
| Input data-  Id – 22222222  Name – Gami  Quiz – 110  Assignment 1 - 89  Assignment 2 - 67  Assignment 3 – 85  Exam – 89  Expected result –  It will give error for quiz marks as it has to be between 0 – 100  Note: Similarly, all the text fields check for valid inputs |  |
| **Searching** | |
| Click on search button to go in search screen |  |
| Search Term – 22222222  Expected result – it shows all the detail with id 22222222 |  |
| Search Term – DI  Expected result – it shows all the detail with grade DI |  |
| Search Term – 85.5  Expected result – it shows all the detail with marks 85.5 |  |
| Search Term – Vismay  Expected result – it shows all the detail with name vismay |  |
| Search Term – 2222ssss  Expected result – it shows error as input is not valid |  |
| Update | |
| Click on update button to go in update screen |  |
| Student id – 33333333  Expected result – it should display all the data of student with id 33333333 |  |
| Student id – 33333333  Change the name to – updated name  Expected result – it should update the data and new data should be displayed |  |
| Student id – 22222222  Change the marks to – 101  Expected result – it should display error of marks as it must be between 0-100 |  |

# TASK1 CODE

**File: GradeProcessing.java**

**import** javafx.application.Application;

**import** javafx.event.ActionEvent;

**import** javafx.event.EventHandler;

**import** javafx.geometry.Pos;

**import** javafx.scene.Scene;

**import** javafx.scene.control.Button;

**import** javafx.scene.control.Label;

**import** javafx.scene.layout.GridPane;

**import** javafx.scene.text.Font;

**import** javafx.scene.text.FontPosture;

**import** javafx.scene.text.FontWeight;

**import** javafx.stage.Stage;

**public** **class** GradeProcessing **extends** Application {

@Override

**public** **void** start(Stage homeStage)

{

// creating grid pane with default properties

GridPane gridPane = **new** GridPane();

gridPane.setAlignment(Pos.***CENTER***);

gridPane.setVgap(20);

// setting choose option label with properties

Label optionLabel = **new** Label("Choose one option");

optionLabel.setFont(Font.*font*(***STYLESHEET\_CASPIAN***, FontWeight.***BOLD***, FontPosture.***REGULAR***, 22));

// setting insert button and its properties

Button insertButton = **new** Button("INSERT RECORD");

insertButton.setAlignment(Pos.***CENTER***);

insertButton.setMinWidth(200);

insertButton.setStyle(

"-fx-background-color: linear-gradient(#61a2b1, #2A5058);" +

"-fx-background-radius: 30;" +

"-fx-background-insets: 0;"+

"-fx-text-fill: white;"+

"-fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 );"

);

// setting search button and its properties

Button searchButton = **new** Button("SEARCH RECORD");

searchButton.setAlignment(Pos.***CENTER***);

searchButton.setMinWidth(200);

searchButton.setStyle(

"-fx-background-color: linear-gradient(#61a2b1, #2A5058);" +

"-fx-background-radius: 30;" +

"-fx-background-insets: 0;"+

"-fx-text-fill: white;"+

"-fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 );"

);

// setting update button and its properties

Button updateButton = **new** Button("UPDATE RECORD");

updateButton.setAlignment(Pos.***CENTER***);

updateButton.setMinWidth(200);

updateButton.setStyle(

"-fx-background-color: linear-gradient(#61a2b1, #2A5058);" +

"-fx-background-radius: 30;" +

"-fx-background-insets: 0;"+

"-fx-text-fill: white;"+

"-fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 );"

);

// adding label and three buttons to grid pane

gridPane.add(optionLabel,0,0);

gridPane.add(insertButton,0,1);

gridPane.add(searchButton,0,2);

gridPane.add(updateButton,0,3);

// event handler to handle the click event on all the buttons

EventHandler<ActionEvent> handler = e ->

{

// gets insert button event

**if**(e.getSource() == insertButton)

{

// calls insert stage

InsertData insert = **new** InsertData();

insert.start(homeStage);

}

// gets search button event

**else** **if**(e.getSource() == searchButton)

{

// calls search stage

SearchData search = **new** SearchData();

search.start(homeStage);

}

// gets update button event

**else** **if**(e.getSource() == updateButton)

{

// calls update stage

UpdateData update = **new** UpdateData();

update.start(homeStage);

}

};

// sets all buttons on action

insertButton.setOnAction(handler);

searchButton.setOnAction(handler);

updateButton.setOnAction(handler);

// creating scene and adding grid pane to it

Scene scene = **new** Scene(gridPane, 350, 250);

homeStage.setTitle("Grade Processing System - Home");

homeStage.setScene(scene); // adds scene to home stage

homeStage.setResizable(**false**);

homeStage.show();

}

**public** **static** **void** main(String[] args)

{

*launch*(args);

}

}

**File: InsertData.java**

**import** javafx.application.Application;

**import** javafx.geometry.Insets;

**import** javafx.geometry.Pos;

**import** javafx.scene.Scene;

**import** javafx.scene.control.Alert;

**import** javafx.scene.control.Button;

**import** javafx.scene.control.Label;

**import** javafx.scene.control.TextField;

**import** javafx.scene.layout.FlowPane;

**import** javafx.scene.layout.GridPane;

**import** javafx.scene.text.Font;

**import** javafx.scene.text.FontPosture;

**import** javafx.scene.text.FontWeight;

**import** javafx.stage.Stage;

**import** java.sql.\*;

**public** **class** InsertData **extends** Application {

// initializing default database variables

Connection connection = **null**;

Statement statement = **null**;

String DATABASE\_NAME = "students\_records";

String TEST\_CONNECTION\_STRING = "jdbc:mysql://localhost:3306/?autoReconnect=true&useSSL=false";

String DATABASE\_CONNECTION\_STRING = "jdbc:mysql://localhost:3306/" + DATABASE\_NAME + "?autoReconnect=true&useSSL=false";

String CONNECTION\_ACCOUNT = "root";

String CONNECTION\_PASSWORD = "root";

String JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";

**public** **void** start(Stage insertStage)

{

// creating flow pane with default properties

FlowPane flowPane = **new** FlowPane();

flowPane.setAlignment(Pos.***CENTER***);

flowPane.setHgap(25);

flowPane.setVgap(25);

// creating grid pane with default Properties

GridPane gridPane = **new** GridPane();

gridPane.setPadding(**new** Insets(5,5,5,5));

gridPane.setHgap(30);

gridPane.setVgap(20);

// setting choose option label with properties

Label welcomeLabel = **new** Label("Enter student details");

welcomeLabel.setFont(Font.*font*(***STYLESHEET\_CASPIAN***, FontWeight.***BOLD***, FontPosture.***REGULAR***, 22));

// setting student id label and text field with default properties

Label studentIdLabel = **new** Label("Student ID (must be 8 digits)");

TextField studentIdText = **new** TextField();

// setting student name label and text field with default properties

Label studentNameLabel = **new** Label("Student Name");

TextField studentNameText = **new** TextField();

// setting student quiz marks label and text field with default properties

Label quizMarksLabel = **new** Label("Quiz Marks (Enter 0-100)");

TextField quizMarksText = **new** TextField();

// setting student assignment1 marks label and text field with default properties

Label assignment1MarksLabel = **new** Label("Assignment1 Marks (Enter 0-100)");

TextField assignment1MarksText = **new** TextField();

// setting student assignment2 marks label and text field with default properties

Label assignment2MarksLabel = **new** Label("Assignment2 Marks (Enter 0-100)");

TextField assignment2MarksText = **new** TextField();

// setting student assignment3 marks label and text field with default properties

Label assignment3MarksLabel = **new** Label("Assignment3 Marks (Enter 0-100)");

TextField assignment3MarksText = **new** TextField();

// setting student Exam marks label and text field with default properties

Label examMarksLabel = **new** Label("Exam Marks (Enter 0-100)");

TextField examMarksText = **new** TextField();

// adding all labels and text fields into grid pane

gridPane.add(welcomeLabel,0,0);

gridPane.add(studentIdLabel,0,1);

gridPane.add(studentNameLabel,0,2);

gridPane.add(quizMarksLabel,0,3);

gridPane.add(assignment1MarksLabel,0,4);

gridPane.add(assignment2MarksLabel,0,5);

gridPane.add(assignment3MarksLabel,0,6);

gridPane.add(examMarksLabel,0,7);

gridPane.add(studentIdText,1,1);

gridPane.add(studentNameText,1,2);

gridPane.add(quizMarksText,1,3);

gridPane.add(assignment1MarksText,1,4);

gridPane.add(assignment2MarksText,1,5);

gridPane.add(assignment3MarksText,1,6);

gridPane.add(examMarksText,1,7);

// setting insert and back button

Button insertButton = **new** Button("INSERT");

insertButton.setMinWidth(100);

insertButton.setStyle(

"-fx-background-color: linear-gradient(#61a2b1, #2A5058);" +

"-fx-background-radius: 30;" +

"-fx-background-insets: 0;"+

"-fx-text-fill: white;"+

"-fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 );"

);

Button backButton = **new** Button("BACK");

backButton.setMinWidth(100);

backButton.setStyle(

"-fx-background-color: linear-gradient(#61a2b1, #2A5058);" +

"-fx-background-radius: 30;" +

"-fx-background-insets: 0;"+

"-fx-text-fill: white;"+

"-fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 );"

);

// setting event handler for insert button

insertButton.setOnAction((event) ->

{

// initializing necessary variables to store user input data

**long** studentId = 0;

String studentName = "";

String grade = "";

**double** quizMarks = 0;

**double** assignmennt1Marks = 0;

**double** assignmennt2Marks = 0;

**double** assignmennt3Marks = 0;

**double** examMarks = 0;

**double** result = 0;

// using try catch to handle any error or invalid inputs

**try**

{

// getting values from text fields and converting in appropriate format

studentId = Integer.*parseInt*(studentIdText.getText());

studentName = studentNameText.getText();

quizMarks = Double.*parseDouble*(quizMarksText.getText());

assignmennt1Marks = Double.*parseDouble*(assignment1MarksText.getText());

assignmennt2Marks = Double.*parseDouble*(assignment2MarksText.getText());

assignmennt3Marks = Double.*parseDouble*(assignment3MarksText.getText());

examMarks = Double.*parseDouble*(examMarksText.getText());

// validating each input values

**if**(studentIdText.getText().length()!=8)

{

**throw** **new** Exception("ID");

}

**else** **if**(quizMarks < 0 || quizMarks > 100)

{

**throw** **new** Exception("Quiz Marks");

}

**else** **if**(assignmennt1Marks < 0 || assignmennt1Marks > 100)

{

**throw** **new** Exception("Assignment1 Marks");

}

**else** **if**(assignmennt2Marks < 0 || assignmennt2Marks > 100)

{

**throw** **new** Exception("Assignment2 Marks");

}

**else** **if**(assignmennt3Marks < 0 || assignmennt3Marks > 100)

{

**throw** **new** Exception("Assignment3 Marks");

}

**else** **if**(examMarks < 0 || examMarks > 100)

{

**throw** **new** Exception("Exam Marks");

}

// calculating total result

result =(quizMarks \* 0.05) + (assignmennt1Marks \* 0.15) + (assignmennt2Marks \* 0.2) + (assignmennt3Marks \* 0.10) + (examMarks \* 0.5);

result = Math.*round*(result\*100)/100.00;

// calculating Grades with conditions of grades

**if**(result>=85)

{

grade = "HD";

}

**else** **if**(result>=75 && result<85)

{

grade = "DI";

}

**else** **if**(result>=65 && result<75)

{

grade = "CR";

}

**else** **if**(result>=50 && result<65)

{

grade = "PS";

}

**else**

{

grade = "FL";

}

// using try catch to handle database error

**try**

{

// Register JDBC driver

Class.*forName*(JDBC\_DRIVER);

// opening a connection

System.***out***.println("Connecting to database...");

connection = DriverManager.*getConnection*(TEST\_CONNECTION\_STRING, CONNECTION\_ACCOUNT, CONNECTION\_PASSWORD);

// checking if Database already exists

String dataBaseName = DATABASE\_NAME;

String catalogs = "";

ResultSet resultSet = **null**;

**if**(connection != **null**)

{

resultSet = connection.getMetaData().getCatalogs();

}

**while**(resultSet.next())

{

catalogs = resultSet.getString(1);

**if**(dataBaseName.equals(catalogs))

{

**break**;

}

**else**

{

**continue**;

}

}

// if database is found

**if**(dataBaseName.equals(catalogs))

{

// uses database if it exists

System.***out***.println("Database Exists");

statement = connection.createStatement();

statement.executeUpdate("USE students\_records");

}

**else**

{

System.***out***.println("Database not found");

// creates a new database if it doesn't exists already

System.***out***.println("New Database is being created...");

statement = connection.createStatement();

statement.executeUpdate("CREATE DATABASE students\_records");

statement.executeUpdate("USE students\_records");

System.***out***.println("Database creation successful...");

// connects to database and creates a new table

connection = DriverManager.*getConnection*(DATABASE\_CONNECTION\_STRING, CONNECTION\_ACCOUNT, CONNECTION\_PASSWORD);

statement.executeUpdate("CREATE TABLE Java2 ("

+ "ID INT(10) NOT NULL,"

+ "StudentName VARCHAR(20),"

+ "Quiz DOUBLE(8,2),"

+ "A1 DOUBLE(8,2),"

+ "A2 DOUBLE(8,2),"

+ "A3 DOUBLE(8,2),"

+ "Exam DOUBLE(8,2),"

+ "Results DOUBLE(8,2),"

+ "Grade VARCHAR(20))"

);

System.***out***.println("Table Created");

}

// inserting user input data into table

PreparedStatement preparedStatement = connection.prepareStatement("insert into Java2 values(?,?,?,?,?,?,?,?,?)");

preparedStatement.setLong(1,studentId);

preparedStatement.setString(2,studentName);

preparedStatement.setDouble(3,quizMarks);

preparedStatement.setDouble(4,assignmennt1Marks);

preparedStatement.setDouble(5,assignmennt2Marks);

preparedStatement.setDouble(6,assignmennt3Marks);

preparedStatement.setDouble(7,examMarks);

preparedStatement.setDouble(8,result);

preparedStatement.setString(9,grade);

preparedStatement.executeUpdate();

connection.close(); // closes connection after inserting

// dialog is displayed with student details

Alert alert = **new** Alert(Alert.AlertType.***INFORMATION***);

alert.setTitle("Student Record Inserted");

alert.setHeaderText(**null**);

alert.setContentText("Record Data is:\nID = "+studentId+

"\nStudent Name = "+studentName+

"\nQuiz Marks = "+quizMarks+

"\nAssignment1 Marks = "+assignmennt1Marks+

"\nAssignment2 Marks = "+assignmennt2Marks+

"\nAssignment3 Marks = "+assignmennt3Marks+

"\nExam Marks = "+examMarks+

"\nResult = "+result+

"\nGrade = "+grade );

alert.showAndWait();

}

**catch**(Exception ex)

{

ex.printStackTrace();

}

}

// catches exceptions

**catch** (NumberFormatException e)

{

Alert alert = **new** Alert(Alert.AlertType.***ERROR***);

alert.setTitle("INPUT ERROR");

alert.setHeaderText(**null**);

alert.setContentText("Invalid Inputs! Please Try Again");

alert.showAndWait();

}

**catch**(Exception ex)

{

Alert alert = **new** Alert(Alert.AlertType.***ERROR***);

alert.setTitle("INPUT ERROR");

alert.setHeaderText(**null**);

alert.setContentText("Invalid "+ex.getMessage()+". Please Try Again");

alert.showAndWait();

}

// clearing all the text fields

studentIdText.setText("");

studentNameText.setText("");

quizMarksText.setText("");

assignment1MarksText.setText("");

assignment2MarksText.setText("");

assignment3MarksText.setText("");

examMarksText.setText("");

});

// setting event handler for back button

backButton.setOnAction((event) ->

{

GradeProcessing home = **new** GradeProcessing();

home.start(insertStage);

});

// adding grid pane and two buttons to flow pane

flowPane.getChildren().add(gridPane);

flowPane.getChildren().add(insertButton);

flowPane.getChildren().add(backButton);

// creating scene and adding flow pane to it

Scene scene = **new** Scene(flowPane, 500, 470);

insertStage.setTitle("Grade Processing System - Insert Record");

insertStage.setScene(scene); // adding scene to insert stage

insertStage.setResizable(**false**);

insertStage.show();

}

}

**File: UpdateData.java**

**import** javafx.application.Application;

**import** javafx.geometry.Insets;

**import** javafx.geometry.Pos;

**import** javafx.scene.Scene;

**import** javafx.scene.control.\*;

**import** javafx.scene.layout.FlowPane;

**import** javafx.scene.layout.GridPane;

**import** javafx.scene.text.Font;

**import** javafx.scene.text.FontPosture;

**import** javafx.scene.text.FontWeight;

**import** javafx.stage.Stage;

**import** java.sql.\*;

**public** **class** UpdateData **extends** Application {

// initializing database variables

Connection connection = **null**;

Statement statement = **null**;

String DATABASE\_NAME = "students\_records";

String DATABASE\_CONNECTION\_STRING = "jdbc:mysql://localhost:3306/" + DATABASE\_NAME + "?autoReconnect=true&useSSL=false";

String CONNECTION\_ACCOUNT = "root";

String CONNECTION\_PASSWORD = "root";

String JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";

// initializing default variables

**long** studentId = 0;

String studentName = "";

String grade = "";

**double** quizMarks = 0;

**double** assignment1Marks = 0;

**double** assignment2Marks = 0;

**double** assignment3Marks = 0;

**double** examMarks = 0;

**double** result = 0;

@Override

**public** **void** start(Stage updateStage) {

// creating flow pane with default properties

FlowPane flowPane = **new** FlowPane();

flowPane.setAlignment(Pos.***CENTER***);

flowPane.setHgap(25);

flowPane.setVgap(25);

// creating grid pane with default Properties

GridPane gridPane = **new** GridPane();

gridPane.setPadding(**new** Insets(5,5,5,5));

gridPane.setHgap(30);

gridPane.setVgap(20);

// setting choose option label with properties

Label welcomeLabel = **new** Label("Update student details");

welcomeLabel.setFont(Font.*font*(***STYLESHEET\_CASPIAN***, FontWeight.***BOLD***, FontPosture.***REGULAR***, 22));

// setting student id label and text field with default properties

Label studentIdLabel = **new** Label("Student ID (must be 8 digits)");

ComboBox<String> studentIdCheck = **new** ComboBox<String>();

studentIdCheck.setEditable(**false**); // to set ID value fixed

studentIdCheck.setMinWidth(150);

// setting student name label and text field with default properties

Label studentNameLabel = **new** Label("Student Name");

TextField studentNameText = **new** TextField();

// setting student quiz marks label and text field with default properties

Label quizMarksLabel = **new** Label("Quiz Marks (Enter 0-100)");

TextField quizMarksText = **new** TextField();

// setting student assignment1 marks label and text field with default properties

Label assignment1MarksLabel = **new** Label("Assignment1 Marks (Enter 0-100)");

TextField assignment1MarksText = **new** TextField();

// setting student assignment2 marks label and text field with default properties

Label assignment2MarksLabel = **new** Label("Assignment2 Marks (Enter 0-100)");

TextField assignment2MarksText = **new** TextField();

// setting student assignment3 marks label and text field with default properties

Label assignment3MarksLabel = **new** Label("Assignment3 Marks (Enter 0-100)");

TextField assignment3MarksText = **new** TextField();

// setting student Exam marks label and text field with default properties

Label examMarksLabel = **new** Label("Exam Marks (Enter 0-100)");

TextField examMarksText = **new** TextField();

// adding all labels and text fields into grid pane

gridPane.add(welcomeLabel,0,0);

gridPane.add(studentIdLabel,0,1);

gridPane.add(studentNameLabel,0,2);

gridPane.add(quizMarksLabel,0,3);

gridPane.add(assignment1MarksLabel,0,4);

gridPane.add(assignment2MarksLabel,0,5);

gridPane.add(assignment3MarksLabel,0,6);

gridPane.add(examMarksLabel,0,7);

gridPane.add(studentIdCheck,1,1);

gridPane.add(studentNameText,1,2);

gridPane.add(quizMarksText,1,3);

gridPane.add(assignment1MarksText,1,4);

gridPane.add(assignment2MarksText,1,5);

gridPane.add(assignment3MarksText,1,6);

gridPane.add(examMarksText,1,7);

// setting insert and back button

Button updateButton = **new** Button("UPDATE");

updateButton.setMinWidth(100);

updateButton.setStyle(

"-fx-background-color: linear-gradient(#61a2b1, #2A5058);" +

"-fx-background-radius: 30;" +

"-fx-background-insets: 0;"+

"-fx-text-fill: white;"+

"-fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 );"

);

Button backButton = **new** Button("BACK");

backButton.setMinWidth(100);

backButton.setStyle(

"-fx-background-color: linear-gradient(#61a2b1, #2A5058);" +

"-fx-background-radius: 30;" +

"-fx-background-insets: 0;"+

"-fx-text-fill: white;"+

"-fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 );"

);

// getting id data from database and setting to combo box

**try**

{

Class.*forName*(JDBC\_DRIVER);

connection=DriverManager.*getConnection*(DATABASE\_CONNECTION\_STRING, CONNECTION\_ACCOUNT, CONNECTION\_PASSWORD);

PreparedStatement preparedStatement = connection.prepareStatement("select \* from java2");

ResultSet resultSet = preparedStatement.executeQuery();

**while**(resultSet.next())

{

studentIdCheck.getItems().add((resultSet.getString(1)));

}

connection.close();

}

**catch**(Exception ex)

{

System.***out***.println("ex = "+ex);

}

// setting event handler to combo box to retrieve data according to id selected

studentIdCheck.setOnAction(e ->

{

studentId = Integer.*parseInt*(studentIdCheck.getValue().toString());

**try**

{

Class.*forName*(JDBC\_DRIVER);

connection=DriverManager.*getConnection*(DATABASE\_CONNECTION\_STRING, CONNECTION\_ACCOUNT, CONNECTION\_PASSWORD);

PreparedStatement preparedStatement = connection.prepareStatement("select \* from java2 where ID=?");

preparedStatement.setLong(1,studentId);

ResultSet resultSet = preparedStatement.executeQuery();

// getting all the data according to id and storing in respective text fields.

**while**(resultSet.next())

{

studentNameText.setText(resultSet.getString(2));

quizMarksText.setText(resultSet.getString(3));

assignment1MarksText.setText(resultSet.getString(4));

assignment2MarksText.setText(resultSet.getString(5));

assignment3MarksText.setText(resultSet.getString(6));

examMarksText.setText(resultSet.getString(7));

}

connection.close();

}

**catch**(Exception ex)

{

System.***out***.println("ex = "+ex);

}

});

// setting event handler to update button to update the data

updateButton.setOnAction((event) ->

{

**try**

{

// getting values from text fields and converting in appropriate format

studentName = studentNameText.getText();

quizMarks = Double.*parseDouble*(quizMarksText.getText());

assignment1Marks = Double.*parseDouble*(assignment1MarksText.getText());

assignment2Marks = Double.*parseDouble*(assignment2MarksText.getText());

assignment3Marks = Double.*parseDouble*(assignment3MarksText.getText());

examMarks = Double.*parseDouble*(examMarksText.getText());

// validating each input values

**if**(quizMarks < 0 || quizMarks > 100)

{

**throw** **new** Exception("Quiz Marks");

}

**else** **if**(assignment1Marks < 0 || assignment1Marks > 100)

{

**throw** **new** Exception("Assignment1 Marks");

}

**else** **if**(assignment2Marks < 0 || assignment2Marks > 100)

{

**throw** **new** Exception("Assignment2 Marks");

}

**else** **if**(assignment3Marks < 0 || assignment3Marks > 100)

{

**throw** **new** Exception("Assignment3 Marks");

}

**else** **if**(examMarks < 0 || examMarks > 100)

{

**throw** **new** Exception("Exam Marks");

}

// calculating total result

result =(quizMarks \* 0.05) + (assignment1Marks \* 0.15) + (assignment2Marks \* 0.2) + (assignment3Marks \* 0.10) + (examMarks \* 0.5);

result = Math.*round*(result\*100)/100.00;

// calculating Grades with conditions of grades

**if**(result>=85)

{

grade = "HD";

}

**else** **if**(result>=75 && result<85)

{

grade = "DI";

}

**else** **if**(result>=65 && result<75)

{

grade = "CR";

}

**else** **if**(result>=50 && result<65)

{

grade = "PS";

}

**else**

{

grade = "FL";

}

// using try catch to handle database error

**try**

{

//Register JDBC driver

Class.*forName*(JDBC\_DRIVER);

//Opening a connection

connection = DriverManager.*getConnection*(DATABASE\_CONNECTION\_STRING, CONNECTION\_ACCOUNT, CONNECTION\_PASSWORD);

statement = connection.createStatement();

statement.executeUpdate("USE students\_records");

// updating data into Database table according to student id

PreparedStatement preparedStatement = connection.prepareStatement("update java2 set "

+ "studentName=?,"

+ "quiz=?,"

+ "A1=?,"

+ "A2=?,"

+ "A3=?,"

+ "Exam=?,Results=?,Grade=? where ID=?");

preparedStatement.setString(1,studentName);

preparedStatement.setDouble(2,quizMarks);

preparedStatement.setDouble(3,assignment1Marks);

preparedStatement.setDouble(4,assignment2Marks);

preparedStatement.setDouble(5,assignment3Marks);

preparedStatement.setDouble(6,examMarks);

preparedStatement.setDouble(7,result);

preparedStatement.setString(8,grade);

preparedStatement.setLong(9,studentId);

preparedStatement.executeUpdate();

connection.close();

// dialog is displayed with updated student details

Alert alert = **new** Alert(Alert.AlertType.***INFORMATION***);

alert.setTitle("Student Record Updated");

alert.setHeaderText(**null**);

alert.setContentText("Record Data is:\nID = "+studentId+

"\nStudent Name = "+studentName+

"\nQuiz Marks = "+quizMarks+

"\nAssignment1 Marks = "+assignment1Marks+

"\nAssignment2 Marks = "+assignment2Marks+

"\nAssignment3 Marks = "+assignment3Marks+

"\nExam Marks = "+examMarks+

"\nResult = "+result+

"\nGrade = "+grade );

alert.showAndWait();

}

**catch** (Exception ex)

{

ex.printStackTrace();

}

// clearing all the text fields

studentNameText.setText("");

quizMarksText.setText("");

assignment1MarksText.setText("");

assignment2MarksText.setText("");

assignment3MarksText.setText("");

examMarksText.setText("");

studentIdCheck.setValue("0");

}

// catches exceptions

**catch** (NumberFormatException e)

{

Alert alert = **new** Alert(Alert.AlertType.***ERROR***);

alert.setTitle("INPUT ERROR");

alert.setHeaderText(**null**);

alert.setContentText("Invalid Inputs! Please Try Again");

alert.showAndWait();

studentNameText.setText("");

quizMarksText.setText("");

assignment1MarksText.setText("");

assignment2MarksText.setText("");

assignment3MarksText.setText("");

examMarksText.setText("");

}

**catch**(Exception ex)

{

Alert alert = **new** Alert(Alert.AlertType.***ERROR***);

alert.setTitle("INPUT ERROR");

alert.setHeaderText(**null**);

alert.setContentText("Invalid "+ex.getMessage()+". Please Try Again");

alert.showAndWait();

studentNameText.setText("");

quizMarksText.setText("");

assignment1MarksText.setText("");

assignment2MarksText.setText("");

assignment3MarksText.setText("");

examMarksText.setText("");

}

});

// setting event handler for back button

backButton.setOnAction((event) ->

{

GradeProcessing home = **new** GradeProcessing();

home.start(updateStage);

});

// adding grid pane and two buttons to flow pane

flowPane.getChildren().add(gridPane);

flowPane.getChildren().add(updateButton);

flowPane.getChildren().add(backButton);

// creating scene and adding flow pane to it

Scene scene = **new** Scene(flowPane, 500, 470);

updateStage.setTitle("Grade Processing System - Update Record");

updateStage.setScene(scene); // adding scene to insert stage

updateStage.setResizable(**false**);

updateStage.show();

}

}

**File: SearchData.java**

**import** javafx.application.Application;

**import** javafx.geometry.Pos;

**import** javafx.scene.Scene;

**import** javafx.scene.control.\*;

**import** javafx.scene.layout.FlowPane;

**import** javafx.stage.Stage;

**import** java.sql.\*;

**import** javafx.collections.FXCollections;

**import** javafx.collections.ObservableList;

**import** javafx.scene.control.cell.PropertyValueFactory;

**import** javafx.scene.layout.HBox;

**public** **class** SearchData **extends** Application {

// declaring obserevable arraylist to hold the student data

**private** **final** ObservableList<Student> studentRecords = FXCollections.*observableArrayList*();

// initializing database variables

Connection connection = **null**;

String DATABASE\_NAME = "students\_records";

String DATABASE\_CONNECTION\_STRING = "jdbc:mysql://localhost:3306/" + DATABASE\_NAME + "?autoReconnect=true&useSSL=false";

String CONNECTION\_ACCOUNT = "root";

String CONNECTION\_PASSWORD = "root";

String JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";

@SuppressWarnings({ "rawtypes", "unchecked" })

@Override

**public** **void** start(Stage searchStage) {

// creating flow pane with default properties

FlowPane flowPane = **new** FlowPane();

flowPane.setAlignment(Pos.***CENTER***);

flowPane.setHgap(25);

flowPane.setVgap(25);

// creating scene amnd adding flow pane to it

Scene scene = **new** Scene(flowPane, 800, 500);

// creating horizontal box with default properties

HBox hbox = **new** HBox();

hbox.setSpacing(10);

// creating welcome label

Label welcomeLabel = **new** Label("Search Item: (ID, Name, Marks, Grade)");

// creating search text field

TextField searchText = **new** TextField();

searchText.setMaxWidth(100);

// creating search button text field

Button searchButton = **new** Button("SEARCH");

searchButton.setMinWidth(100);

searchButton.setStyle(

"-fx-background-color: linear-gradient(#61a2b1, #2A5058);" +

"-fx-background-radius: 30;" +

"-fx-background-insets: 0;"+

"-fx-text-fill: white;"+

"-fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 );"

);

// creating back button

Button backButton = **new** Button("BACK");

backButton.setMinWidth(100);

backButton.setStyle(

"-fx-background-color: linear-gradient(#61a2b1, #2A5058);" +

"-fx-background-radius: 30;" +

"-fx-background-insets: 0;"+

"-fx-text-fill: white;"+

"-fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 );"

);

// adding all components to hbox

hbox.getChildren().add(welcomeLabel);

hbox.getChildren().add(searchText);

hbox.getChildren().add(searchButton);

hbox.getChildren().add(backButton);

// creating table view and setting its default properties with respect to scene

TableView tableView = **new** TableView();

tableView.setEditable(**false**);

tableView.setMaxWidth(scene.getWidth()-50);

tableView.setMaxHeight(scene.getHeight()-hbox.getHeight()-50);

// creating student id column with default properties

TableColumn studentIdColumn = **new** TableColumn("Student ID");

studentIdColumn.setMinWidth(100);

// creating student name column with default properties

TableColumn studentNameColumn = **new** TableColumn("Student Name");

studentNameColumn.setMinWidth(80);

// creating quiz marks column with default properties

TableColumn quizMarksColumn = **new** TableColumn("Quiz");

quizMarksColumn.setMinWidth(80);

// creating assignment 1 marks column with default properties

TableColumn assignment1MarksColumn = **new** TableColumn("A1");

assignment1MarksColumn.setMinWidth(75);

// creating assignment 2 marks column with default properties

TableColumn assignment2MarksColumn = **new** TableColumn("A2");

assignment2MarksColumn.setMinWidth(75);

// creating assignment 3 column with default properties

TableColumn assignment3MarksColumn = **new** TableColumn("A3");

assignment3MarksColumn.setMinWidth(75);

// creating exam marks column with default properties

TableColumn examMarksColumn = **new** TableColumn("Exam");

examMarksColumn.setMinWidth(80);

// creating result column with default properties

TableColumn resultColumn = **new** TableColumn("Results");

resultColumn.setMinWidth(80);

// creating grade column with default properties

TableColumn gradeColumn = **new** TableColumn("Grade");

gradeColumn.setMinWidth(80);

// assigning column values

studentIdColumn.setCellValueFactory(**new** PropertyValueFactory<Student, String>("studentId"));

studentNameColumn.setCellValueFactory(**new** PropertyValueFactory<Student,String>("studentName"));

quizMarksColumn.setCellValueFactory(**new** PropertyValueFactory<Student, String>("quizMarks"));

assignment1MarksColumn.setCellValueFactory(**new** PropertyValueFactory<Student, String>("assignment1Marks"));

assignment2MarksColumn.setCellValueFactory(**new** PropertyValueFactory<Student, String>("assignment2Marks"));

assignment3MarksColumn.setCellValueFactory(**new** PropertyValueFactory<Student, String>("assignment3Marks"));

examMarksColumn.setCellValueFactory(**new** PropertyValueFactory<Student, String>("examMarks"));

resultColumn.setCellValueFactory(**new** PropertyValueFactory<Student, String>("result"));

gradeColumn.setCellValueFactory(**new** PropertyValueFactory<Student, String>("grade"));

// adding all the columns to the table view

tableView.getColumns().addAll(studentNameColumn,studentIdColumn,quizMarksColumn,assignment1MarksColumn,assignment2MarksColumn,assignment3MarksColumn,examMarksColumn,resultColumn,gradeColumn);

tableView.setItems(studentRecords);

// setting event handler to search button

searchButton.setOnAction((event) ->

{

// clearing student records

studentRecords.clear();

// getting search id value

String searchItem = searchText.getText();

String query="";

// getting data from database according to search id

**try**

{

Class.*forName*(JDBC\_DRIVER);

connection = DriverManager.*getConnection*(DATABASE\_CONNECTION\_STRING, CONNECTION\_ACCOUNT, CONNECTION\_PASSWORD);

PreparedStatement preparedStatement = **null**;

// checks if search term is alphabetical and searches based on name and grade

**if**(searchItem.matches("[a-zA-Z]+"))

{

// getting data from database based on condition

query = "SELECT \* FROM java2 WHERE StudentName=? or Grade=?";

preparedStatement = connection.prepareStatement(query);

preparedStatement.setString(1,searchItem);

preparedStatement.setString(2,searchItem);

}

// checks if search term is numerical with 8 digits length and searches based on id

**else** **if**(searchItem.matches("[0-9]+") && searchItem.length()==8)

{

// getting data from database based on condition

query = "SELECT \* FROM java2 WHERE ID=?";

preparedStatement = connection.prepareStatement(query);

preparedStatement.setLong(1,Long.*parseLong*(searchItem));

}

// checks if search term is numerical with any length and searches based on marks

**else** **if**(searchItem.matches("[-+]?([0-9]\*\\.[0-9]+|[0-9]+)"))

{

// checks if marks are between 0 and 100

**if**(Double.*parseDouble*(searchItem)>=0 || Double.*parseDouble*(searchItem)<=100)

{

// getting data from database based on condition

query = "SELECT \* FROM java2 WHERE Quiz=? or A1=? or A2=? or A3=? or Exam=? or Results=?";

preparedStatement = connection.prepareStatement(query);

preparedStatement.setDouble(1,Double.*parseDouble*(searchItem));

preparedStatement.setDouble(2,Double.*parseDouble*(searchItem));

preparedStatement.setDouble(3,Double.*parseDouble*(searchItem));

preparedStatement.setDouble(4,Double.*parseDouble*(searchItem));

preparedStatement.setDouble(5,Double.*parseDouble*(searchItem));

preparedStatement.setDouble(6,Double.*parseDouble*(searchItem));

}

}

// gives input error otherwise

**else**

{

Alert alert = **new** Alert(Alert.AlertType.***ERROR***);

alert.setTitle("INPUT ERROR");

alert.setHeaderText(**null**);

alert.setContentText("Invalid Value! Please Try Again");

alert.showAndWait();

searchText.setText("");

}

// executing query

ResultSet resultSet = preparedStatement.executeQuery();

// using while loop to iterate through the records

**while**(resultSet.next())

{

// getting values from records and adding it to

**long** studentId = Long.*parseLong*(resultSet.getString(1));

String studentName = resultSet.getString(2);

**double** quizMarks = Double.*parseDouble*(resultSet.getString(3));

**double** assignment1Marks = Double.*parseDouble*(resultSet.getString(4));

**double** assignment2Marks = Double.*parseDouble*(resultSet.getString(5));

**double** assignment3Marks = Double.*parseDouble*(resultSet.getString(6));

**double** examMarks = Double.*parseDouble*(resultSet.getString(7));

**double** result = Double.*parseDouble*(resultSet.getString(8));

String grade = resultSet.getString(9);

// creates student object and add to student records list

Student student = **new** Student(studentId, studentName, quizMarks, assignment1Marks, assignment2Marks, assignment3Marks, examMarks, result, grade);

studentRecords.add(student);

}

connection.close();

}

**catch**(Exception ex)

{

ex.printStackTrace();

}

}); //Closing of Search Button Event

// setting event handler for back button

backButton.setOnAction((event) ->

{

GradeProcessing home = **new** GradeProcessing();

home.start(searchStage);

});

// adding hbox and table to flowpane

flowPane.getChildren().add(hbox);

flowPane.getChildren().add(tableView);

// setting stage

searchStage.setTitle("Grade Processing System - Search Record");

searchStage.setScene(scene); // adding scene to search stage

searchStage.setResizable(**false**);

searchStage.show();

}

}

**File: Student.java**

**public** **class** Student {

// declaring default variables

**private** String studentName;

**private** **long** studentId;

**private** **double** quizMarks;

**private** **double** assignment1Marks;

**private** **double** assignment2Marks;

**private** **double** assignment3Marks;

**private** **double** examMarks;

**private** **double** result;

**private** String grade;

// declaring constructor with arguments

**public** Student(**long** studentId,String studentName, **double** quizMarks, **double** assignment1Marks, **double** assignment2Marks, **double** assignment3Marks,**double** examMarks,**double** result, String grade)

{

**this**.studentName = studentName;

**this**.studentId = studentId;

**this**.quizMarks = quizMarks;

**this**.assignment1Marks = assignment1Marks;

**this**.assignment2Marks = assignment2Marks;

**this**.assignment3Marks = assignment3Marks;

**this**.examMarks = examMarks;

**this**.result = result;

**this**.grade = grade;

}

// get method to get student name

**public** String getStudentName() {

**return** studentName;

}

// get method to get student id

**public** **long** getStudentId() {

**return** studentId;

}

// get method to get quiz marks

**public** **double** getQuizMarks() {

**return** quizMarks;

}

// get method to get assignment 1 marks

**public** **double** getAssignment1Marks() {

**return** assignment1Marks;

}

// get method to get assignment 2 marks

**public** **double** getAssignment2Marks() {

**return** assignment2Marks;

}

// get method to get assignment 3 marks

**public** **double** getAssignment3Marks() {

**return** assignment3Marks;

}

// get method to get exam marks

**public** **double** getExamMarks() {

**return** examMarks;

}

// get method to get result

**public** **double** getResult() {

**return** result;

}

// get method to get grade

**public** String getGrade() {

**return** grade;

}

}