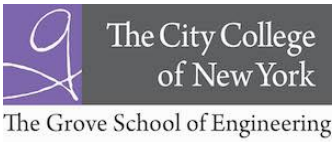


The City College of New York

The Grove School of Engineering

Computer Science

CSc 22100

Section L24139

Professor Hesham A. Auda

Project #4

Fall 2022

Project #4

1.Consider the database schema below.

Students(empID, firstName, lastName, email, gender)
Courses(courseID, courseTitle, department)
Classes(courseID, studentID, sectionNo, year, semester, grade)

The underlined attributes are the primary keys of their corresponding tables. The value of attribute gender may only be F, M, or U, referring, respectively, for female, male, or unidentified. The only letter grades allowed in the database are A, B, C, D, F, or W.

Further you are provided the class schedule for the Spring 2022 semester in file scheduleSpring2022.txt. The key to the data in scheduleSpring2022 is (courseID, sectionNo).

2. Using a Relational Database Management System (RDBMS) of your choice, your tasks are to:

A. Create and populate a Schedule table using the data provided in file scheduleSpring2022.txt.

B. Create and populate Courses and Classes tables using the data in table Schedule.

C. Create and populate Students and Classes tables using data of your own together with the data in table Schedule.

D. Using GROUP BY, Calculate and output the number of students for each letter grade in CSc 22100 [Introdcution to Database Systems] in the Spring 2022 semester.

3. Build and test a Java application that [1] connects to the database, [2] creates, [3] populates, and [4] updates the Students, Courses, Classes, and AggregateGrades tables. The application should utilize PreparedStatement objects for the execution of DDL statements and SQL queries.

A. The Java application utilizes a class StudentsDatabase which includes inner classes Schedule, Students, Courses, Classes, and AggregateGrades, corresponding, respectively, to database Tables Schedule, Students, Classes, and AggregateGrades. The constructor of class Database may be utilized to establish a connection to the RDBMS, while the constructors of the inner classes may be used to create and populate the corresponding database Tables.

B. Classes StudentsDatabase and Classes also include update methods that update, respectively, the instructor of a class and grade of a student.

C. Class Database implements interfaces StudentsDatabaseInterface and TableInterface which include constants, and abstract, and static methods that define the DDL and SQL expressions used for creating, populating, and querying the database tables.

D. Utilize the classes in Assignment 3, including HistogramAlphaBet and MyPieChart, to build and display a pie chart showing the proportion of students for each letter grade. In the pie chart:

a. Each segment has a different color;

b. Each segment has a legend showing the corresponding grades and number of students;

c. The segments for the grades are displayed in alphabetical order.

4. The report should show [1] sample input tables, [2] output table for the aggregated grades and corresponding pie chart for a sufficient amount of input data, and [3] example[s] of the use of the update function.

5. You may only use JavaFX graphics and your own classes and methods for the operations included. Further,

a. The code is applicable to canvases of variable height and width;

b. The size of the pie chart is proportional to the smallest dimension of the canvas;

c. The segments of the pie chart are filled with different colors of your choice, specified through a MyColor enum reference type.

6. Explicitly specify all the classes imported and used in your Java application.

Solution Code

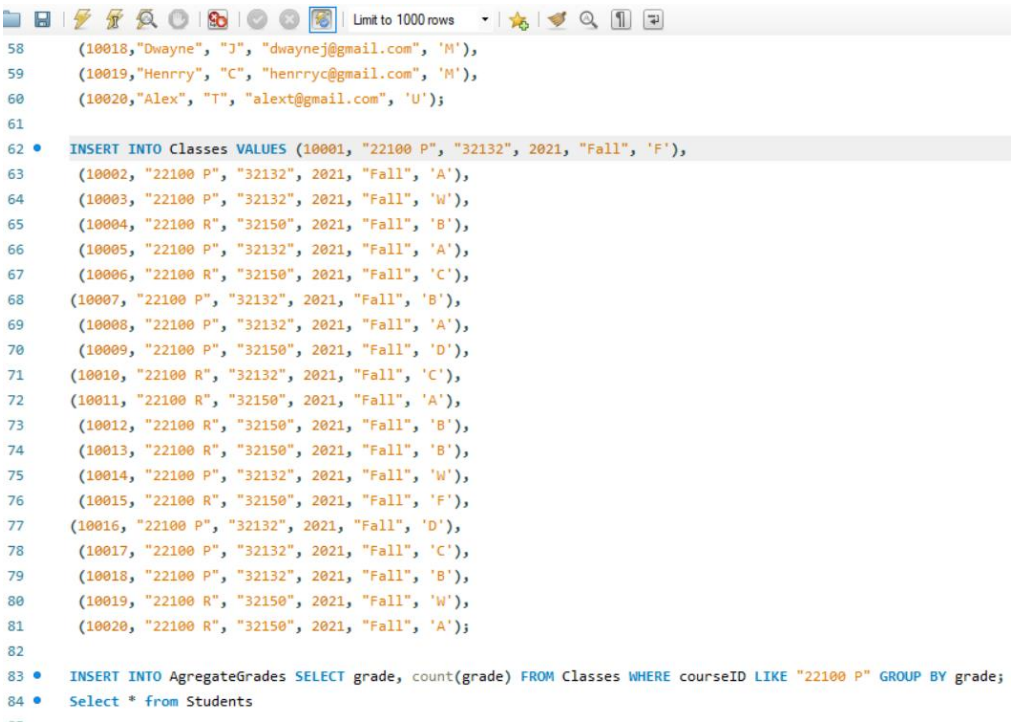
Project.sql

```sql
CREATE TABLE Schedule(
    courseId CHAR(12) NOT NULL UNIQUE,
    sectionNumber VARCHAR(8) NOT NULL UNIQUE,
    title VARCHAR(64),
    year INT,
    semester CHAR(6),
    instructor VARCHAR(24),
    department CHAR(16),
    program VARCHAR(48),
    PRIMARY KEY(courseId, sectionNumber));
CREATE TABLE Students(
    empID INT NOT NULL PRIMARY KEY,
    firstName VARCHAR(40),
    lastName VARCHAR(40),
    email CHAR(60),
    gender CHAR CHECK(gender = 'M' OR gender = 'F' OR gender = 'U'));
CREATE TABLE Courses(
    courseID CHAR(12) NOT NULL PRIMARY KEY,
    courseTitle VARCHAR(64),
    department CHAR(16));
CREATE TABLE Classes(
    empID INT REFERENCES Students(empID),
    courseID CHAR(12) REFERENCES Schedule(courseID),
    sectionNumber VARCHAR(8) REFERENCES Schedule(sectionNumber),
    year INT,
    semester CHAR(6),
    grade CHAR CHECK(grade = 'A' OR grade = 'B' OR grade = 'C' OR grade = 'D' OR grade = 'F' OR grade = 'W'),
    PRIMARY KEY(courseID, empID, sectionNumber));
```

```sql
CREATE TABLE AgregateGrades(
     grade CHAR PRIMARY KEY,
     numberStudents INT);
LOAD DATA INFILE 'C:/Program Files/MySQL/tmp/ScheduleSpring2022.txt'
INTO TABLE Schedule
FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'
IGNORE 1 lines;

INSERT INTO Courses(courseID, courseTitle, department)
SELECT courseID, title, department
FROM Schedule;

INSERT INTO Students VALUES (10001,"Jair", "R", "jairr@gmail.com", 'M'),
(10002,"Melissa", "M", "melissam@gmail.com", 'F'),
 (10003,"Peter", "D", "peterd@gmail.com", 'M'),
(10004,"Bennito", "M", "bennitom@gmail.com", 'M'),
 (10005,"Nicolas", "M", "nicolasm@gmail.com", 'U'),
 (10006,"Rafael", "C", "rafaelc@gmail.com", 'M'),
 (10007,"Freya", "A", "freyaa@gmail.com", 'F'),
 (10008,"Leonel", "M", "leonelm@gmail.com", 'M'),
 (10009,"Sergio", "A", "sergioa@gmail.com", 'M'),
 (10010,"Brithany", "F", "brithanyf@gmail.com", 'U'),
 (10011,"Marie", "C", "mariec@gmail.com", 'F'),
 (10012,"Jennifer", "L", "jenniferl@gmail.com", 'F'),
 (10013,"Megan", "F", "meganf@gmail.com", 'F'),
 (10014,"Mia", "K", "miak@gmail.com", 'F'),
 (10015,"Cristiano", "R", "cristianor@gmail.com", 'M'),
 (10016,"Ibai", "L", "ibail@gmail.com", 'U'),
 (10017,"Aurora", "R", "aurorr@gmail.com", 'M')
```



```
58      (10018,"Dwayne", "J", "dwaynej@gmail.com", 'M'),
59      (10019,"Henrry", "C", "henrryc@gmail.com", 'M'),
60      (10020,"Alex", "T", "alext@gmail.com", 'U');
61
62 •   INSERT INTO Classes VALUES (10001, "22100 P", "32132", 2021, "Fall", 'F'),
63      (10002, "22100 P", "32132", 2021, "Fall", 'A'),
64      (10003, "22100 P", "32132", 2021, "Fall", 'W'),
65      (10004, "22100 R", "32150", 2021, "Fall", 'B'),
66      (10005, "22100 P", "32132", 2021, "Fall", 'A'),
67      (10006, "22100 R", "32150", 2021, "Fall", 'C'),
68     (10007, "22100 P", "32132", 2021, "Fall", 'B'),
69      (10008, "22100 P", "32132", 2021, "Fall", 'A'),
70      (10009, "22100 P", "32150", 2021, "Fall", 'D'),
71     (10010, "22100 R", "32132", 2021, "Fall", 'C'),
72     (10011, "22100 R", "32150", 2021, "Fall", 'A'),
73      (10012, "22100 R", "32150", 2021, "Fall", 'B'),
74      (10013, "22100 R", "32150", 2021, "Fall", 'B'),
75      (10014, "22100 P", "32132", 2021, "Fall", 'W'),
76      (10015, "22100 R", "32150", 2021, "Fall", 'F'),
77     (10016, "22100 P", "32132", 2021, "Fall", 'D'),
78      (10017, "22100 P", "32132", 2021, "Fall", 'C'),
79      (10018, "22100 P", "32132", 2021, "Fall", 'B'),
80      (10019, "22100 R", "32150", 2021, "Fall", 'W'),
81      (10020, "22100 R", "32150", 2021, "Fall", 'A');
82
83 •   INSERT INTO AgregateGrades SELECT grade, count(grade) FROM Classes WHERE courseID LIKE "22100 P" GROUP BY grade;
84 •   Select * from Students
```

This sql script is used to create and populate tables Schedule, Courses, Classes, Students and AggregateGrades which is filled using GROUP BY statement from table Courses. Table Schedule is filled with data from the provided file “SpringSchedule2021.txt”, table Courses is filled with data from Schedule columns, while tables Classes and Students are filled manually using INSERT INTO statement.

HistogramAlphabet.java

```
import java.text.DecimalFormat;
import java.util.Collections;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.Map.Entry;
import java.util.stream.Collectors;
import javafx.scene.canvas.GraphicsContext;

public class HistogramAlphaBet{
    Map<Character,Integer> frequencies = new HashMap<Character, Integer>();
    Map<Character,Double> probabilities = new HashMap<Character, Double>();
    HistogramAlphaBet(String str){
        String text = str.replaceAll("[^a-zA-Z]", "").toLowerCase();
        for(int i = 0; i < text.length(); ++i){
            Character key = text.charAt(i);
            if(frequencies.containsKey(key)){
                frequencies.put(key,frequencies.get(key)+1);
```

```
            }
            else{
                frequencies.put(key,1);
            }
        }
        double totalProb = getSumOfFrequencies();
        for(Character K : frequencies.keySet()){
            probabilities.put(K, (double)frequencies.get(K)/totalProb);
        }
        probabilities = sortDownProbabilities();
    }
    HistogramAlphaBet(Map<Character, Integer> freq){
        this.frequencies = freq;
        frequencies = sortKeyOfFrequencies();
        double totalProb = getSumOfFrequencies();
        for(Character K : frequencies.keySet()){
            probabilities.put(K, (double)frequencies.get(K)/totalProb);
        }
        probabilities = sortDownProbabilities();
    }
    public int getSumOfFrequencies(){
        return frequencies.values().stream().reduce(0, Integer::sum);
    }
    public Map<Character,Integer> sortKeyOfFrequencies(){
        return frequencies.entrySet().stream()
        .sorted(Entry.comparingByValue())
        .collect(Collectors.toMap(Entry::getKey, Entry::getValue, (e1, e2) -> e1,
LinkedHashMap::new));
    }
    public Map<Character,Double> sortDownProbabilities(){
        return probabilities.entrySet().stream()
        .sorted(Collections.reverseOrder(Map.Entry.comparingByValue()))
        .collect(Collectors.toMap(Map.Entry::getKey, Map.Entry::getValue, (e1,
e2) -> e2, LinkedHashMap::new));
    }
    public void printFrequencies(){
        System.out.print(frequencies);
    }
    public void printProbabilities(){
        System.out.print(probabilities);
    }
    public class MyPieChart{
        int n;                  //number of most frequent characters
        Map<Character, Slice> pie;
        double uStartAngle;     //the start angle defined by user
```

```
        MyPoint center;         //the center of the chart
        MyColor fillers[] = MyColor.values();   //Array of colors to fill the
Slices
        double radius;          //radius of the chart
        double restProbabilities;
        DecimalFormat formatter = new DecimalFormat("0.0000");

        MyPieChart(int n, double startAngle, MyPoint center, double radius){
            pie = new HashMap<Character, Slice>();
            this.n = n;
            this.uStartAngle = startAngle;
            this.center = center;
            this.radius = radius;
            int i = 0;
            double restAngle = 0;
            double carryProb = 0;
            for(Character k : probabilities.keySet()){
                double angle = 360*probabilities.get(k);
                if(i<n){
                    pie.put(k, new Slice(center, radius, startAngle,angle,
fillers[i]));
                    startAngle += angle;
                }
                else{
                    restAngle += angle;
                    carryProb+=probabilities.get(k);
                }
                i++;
            }
            //pie.put('#', new Slice(center, radius, startAngle, restAngle,
fillers[fillers.length-1]));
            //restProbabilities = 1-carryProb;
            //restProbabilities =carryProb;
        }
        public void draw(GraphicsContext GC){
            double textheight = 20;
            for(Character key:pie.keySet()){
                if(key=='#'){
                    pie.get(key).draw(GC);
                    GC.fillRect(10,textheight+10, 25,12);
                    GC.setFill(MyColor.Black.getJavaFxColor());
                    GC.fillText("Other letters:
"+formatter.format(restProbabilities), 40, textheight+= 20);
                }
                else{
```

```
                    pie.get(key).draw(GC);
                    GC.fillRect(10,textheight + 10, 25,12);
                    GC.setFill(MyColor.Black.getJavaFxColor());
                    GC.fillText(key+": "+
formatter.format(probabilities.get(key)), 40, textheight+= 20);
                }
            }
        }
        public void printPie(){
            System.out.println(pie.keySet());//check
        }
    }
}
```

HistogramAlphabet class was a little modified for this project. A new parametrized constructor was built passing a Map instead of a String, this will contain the frequency of the grades. Furthermore, MyPieChart inner class was a little modified by commenting the lines that created a Slice for the rest of the probabilities.

The rest of the code remains the same.

Slice.java

```
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.shape.ArcType;

public class Slice{
    MyPoint center;     //central point of the circle
    double radius;      //the radius of the circle
    double startAngle, angle, endAngle; //counterclockwise
    MyColor color;
    //default constructor
    Slice(){this.radius = 1;this.startAngle=0;}
    //parametrized constructor
    Slice(MyPoint center, double radius, double startAngle, double angle, MyColor
color){
        this.center = center;
        this.radius = radius;
        this.startAngle = startAngle;
        this.angle = angle;
        this.color = color;
        this.endAngle = startAngle + angle;
    }
    //setter methods
    public void setCenter(MyPoint p){this.center = p;}
```

```
    public void setRadius(double r){this.radius = r;}
    public void setStartAngle(double startAngle){this.startAngle = startAngle;}
    public void setAngle(double angle){this.angle = angle;}
    public void setColor(MyColor color){this.color = color;}
    public void setEndAngle(double endAngle){this.endAngle = endAngle;}
    //getter methods
    public MyPoint getCenter(){return this.center;}
    public double getRadius(){return this.radius;}
    public double getStartAngle(){return this.startAngle;}
    public double getAngle(){return this.angle;}
    public MyColor getMyColor(){return this.color;}
    public double getEndAngle(){return this.endAngle;}
    //method to draw the object in a canvas using a GraphicsContext object
    public void draw(GraphicsContext GC){
        GC.setFill(this.color.getJavaFxColor());
        GC.fillArc(center.getX()-radius, center.getY()-radius,
                   radius*2, radius*2, startAngle, angle, ArcType.ROUND);
    }
    //method to get a string representation of the object
    @Override
    public String toString(){
        return "Slice\nangle extension = " + angle +
                "\nstart angle = " + startAngle +
                "\ncenter point = " + center;
    }
}
```

Slice class was not modified for the purposes of this project.

TableInterface.java

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

interface TableInterface{
    Connection getConnection(String url, String username, String password);

    static void createSchema(Connection connection, String nameSchema) throws
SQLException{
        PreparedStatement psCreateTable = connection.prepareStatement("CREATE
SCHEMA "+nameSchema);
        try{
            psCreateTable.executeUpdate();
        }
        catch(SQLException e){ System.out.println(e);}
    }
```

```
    static void dropTable(Connection connection, String nameTable) throws
SQLException{
        PreparedStatement psDropTable = connection.prepareStatement("DROP TABLE
IF EXISTS "+nameTable);
        try{
            psDropTable.executeUpdate();
        }
        catch(SQLException ex){System.out.println(ex);}
    }

    static void createTable(Connection connection, String ddlCreateTable) throws
SQLException
    {
        PreparedStatement psCreateTable =
connection.prepareStatement(ddlCreateTable);
        try
        {
            psCreateTable.executeUpdate();
        }
        catch(SQLException e){System.out.println(e);}
    }

    static String loadDataInFileTable(String nameFile, String nameTable)
    {
        return "LOAD DATA INFILE '" + nameFile + "' INTO TABLE " + nameTable +
        " FIELDS TERMINATED BY '\\t' LINES TERMINATED BY '\\n' "+
        "IGNORE 1 lines;";
    }
    static void populateTable(Connection connection, String ddlPopulateTable)
throws SQLException{
        PreparedStatement psPopulateTable =
connection.prepareStatement(ddlPopulateTable);
        try
        {
            psPopulateTable.executeUpdate();
        }
        catch(SQLException e){System.out.println(e);}
    }
}
```

createSchema(): This method is used to create the Schema of the Database.

dropTable(): This method is used to drop the tables of the connected Database.

createTable(): This method is used to create a table for the Database.

populateTable(): This method is used to insert data in the table specified.

StudentsDatabaseInterface.java

```
interface  StudentsDatabaseInterface{
    String ddlCreateTableSchedule =
    "CREATE TABLE Schedule( "+
    "courseId CHAR(12) NOT NULL UNIQUE, "+
    "sectionNumber VARCHAR(8) NOT NULL UNIQUE, "+
    "title VARCHAR(64), "+
    "year INT, "+
    "semester CHAR(6), "+
    "instructor VARCHAR(24), "+
    "department CHAR(16), "+
    "program VARCHAR(48), "+
    "PRIMARY KEY(courseId, sectionNumber));";
    String ddlCreateTableStudents =
    "CREATE TABLE Students("+
    "empID INT NOT NULL PRIMARY KEY,"+
    "firstName VARCHAR(40),"+
    "lastName VARCHAR(40),"+
    "email CHAR(60),"+
    "gender CHAR CHECK(gender = 'M' OR gender = 'F' OR gender = 'U'));";
    String ddlCreateTableCourses =
    "CREATE TABLE Courses("+
    "courseID CHAR(12) REFERENCES Schedule(courseID),"+
    "courseTitle VARCHAR(64),"+
    "department CHAR(16));";
    String ddlCreateTableClasses =
    "CREATE TABLE Classes("+
    "empID INT REFERENCES Students(empID),"+
    "courseID CHAR(12) REFERENCES Schedule(courseID),"+
    "sectionNumber VARCHAR(8) REFERENCES Schedule(sectionNumber),"+
    "year INT,"+
    "semester CHAR(6),"+
    "grade CHAR CHECK(grade = 'A' OR grade = 'B' OR grade = 'C' OR grade = 'D' OR
grade = 'F' OR grade = 'W'),"+
    "PRIMARY KEY(courseID, empID, sectionNumber));";
    String ddlCreateTableAggregateGrades =
    "CREATE TABLE AggregateGrades("+
    "grade CHAR PRIMARY KEY, "+
    "numberStudents INT);";
    String ddlInsertTableStudents =
    "INSERT INTO Students VALUES (10001,'Jair', 'R', 'jairr@gmail.com', 'M'), "+
```

```
    "(10002,'Melissa', 'M', 'melissam@gmail.com', 'F'), "+
    "(10003,'Peter', 'D', 'peterd@gmail.com', 'M'), "+
    "(10004,'Bennito', 'M', 'bennitom@gmail.com', 'M'), "+
    "(10005,'Nicolas', 'M', 'nicolasm@gmail.com', 'U'), "+
    "(10006,'Rafael','C', 'rafaelc@gmail.com', 'M'), "+
    "(10007,'Freya', 'A', 'freyaa@gmail.com', 'F'), "+
    "(10008,'Leonel', 'M', 'leonelm@gmail.com', 'M'), "+
    "(10009,'Sergio', 'A', 'sergioa@gmail.com', 'M'), "+
    "(10010,'Brithany', 'F', 'brithanyf@gmail.com', 'U'), "+
    "(10011,'Marie', 'C', 'mariec@gmail.com', 'F'), "+
    "(10012,'Jennifer', 'L', 'jenniferl@gmail.com', 'F'), "+
    "(10013,'Megan', 'F', 'meganf@gmail.com', 'F'), "+
    "(10014,'Mia', 'K', 'miak@gmail.com', 'F'), "+
    "(10015,'Cristiano', 'R', 'cristianor@gmail.com', 'M'), "+
    "(10016,'Ibai', 'L', 'ibail@gmail.com', 'U'), "+
    "(10017,'Auron', 'P', 'auronp@gmail.com', 'M'), "+
    "(10018,'Dwayne', 'J', 'dwaynej@gmail.com', 'M'), "+
    "(10019,'Henrry', 'C', 'henrryc@gmail.com', 'M'), "+
    "(10020,'Alex', 'T', 'alext@gmail.com', 'U');";
    String ddlInsertTableClasses =
    "INSERT INTO Classes VALUES (10001, '22100 P', '32132', 2021, 'Fall', 'F'),
"+
    "(10002, '22100 P', '32132', 2021, 'Fall', 'A'), "+
    "(10003, '22100 P', '32132', 2021, 'Fall', 'W'), "+
    "(10004, '22100 R', '32150', 2021, 'Fall', 'B'), "+
    "(10005, '22100 P', '32132', 2021, 'Fall', 'A'), "+
    "(10006, '22100 R', '32150', 2021, 'Fall', 'C'), "+
    "(10007, '22100 P', '32132', 2021, 'Fall', 'B'), "+
    "(10008, '22100 P', '32132', 2021, 'Fall', 'A'), "+
    "(10009, '22100 P', '32150', 2021, 'Fall', 'D'), "+
    "(10010, '22100 R', '32132', 2021, 'Fall', 'C'), "+
    "(10011, '22100 R', '32150', 2021, 'Fall', 'A'), "+
    "(10012, '22100 R', '32150', 2021, 'Fall', 'B'), "+
    "(10013, '22100 R', '32150', 2021, 'Fall', 'B'), "+
    "(10014, '22100 P', '32132', 2021, 'Fall', 'W'), "+
    "(10015, '22100 R', '32150', 2021, 'Fall', 'F'), "+
    "(10016, '22100 P', '32132', 2021, 'Fall', 'D'), "+
    "(10017, '22100 P', '32132', 2021, 'Fall', 'C'), "+
    "(10018, '22100 P', '32132', 2021, 'Fall', 'B'), "+
    "(10019, '22100 R', '32150', 2021, 'Fall', 'W'), "+
    "(10020, '22100 R', '32150', 2021, 'Fall', 'A');";
    String ddlInsertCourses =
    "INSERT INTO Courses(courseID, courseTitle, department)"+
    " SELECT courseID, title, department"+
    " FROM Schedule;";
```

```
}
```

StudentsDatabase interface contains String fields to be used in the TableInterface interface methods to pass as parameters when creating the tables and populating them in the next StudentsDatabase class.

StudentsDatabase.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.HashMap;
import java.util.Map;

class StudentsDatabase implements TableInterface, StudentsDatabaseInterface{
    String url, username, password;
    Connection connection;
    StudentsDatabase(String url, String username, String password){
        this.url = url;
        this.username = username;
        this.password = password;
        this.connection = getConnection(url, username, password);
    }
    public Connection getConnection(String url, String username, String
password){
        Connection connection = null;

        try{
            //Class.forName("com.mysql.jdbc.Driver");
            connection = DriverManager.getConnection(url, username, password);
            System.out.println("\nConnection to the databse server successful");
        }
        catch(SQLException e){System.out.println(e);}
        //catch(ClassNotFoundException c){System.out.println(c);}
        return connection;
    }

    class Schedule{
        Schedule(String nameFile) throws SQLException{
            try{
                TableInterface.dropTable(connection, "Schedule");
```

```
                TableInterface.createTable(connection, ddlCreateTableSchedule);
                TableInterface.populateTable(connection,
TableInterface.loadDataInFileTable(nameFile,"Schedule"));
            }
            catch(SQLException e){System.out.println(e);}
        }
        public void updateInstructor(String courseID, String newInstructor)
throws SQLException{
            String query = "UPDATE Schedule SET instructor = " + newInstructor +
" WHERE courseID = " + courseID;
                PreparedStatement psUpdGrade =
connection.prepareStatement(query);
                try{
                    psUpdGrade.executeUpdate();
                }catch(SQLException e){System.out.println(e);}
        }
    }
    class Students{
        Students(){
            try{
                TableInterface.dropTable(connection, "Students");

                TableInterface.createTable(connection, ddlCreateTableStudents);
                TableInterface.populateTable(connection, ddlInsertTableStudents);
            }
            catch(SQLException e){System.out.println(e);}
        }
    }
    class Courses{
        Courses(){
            try{
                TableInterface.dropTable(connection, "Courses");
                TableInterface.createTable(connection, ddlCreateTableCourses);
                TableInterface.populateTable(connection, ddlInsertCourses);
            }
            catch(SQLException e){System.out.println(e);}
        }
    }
    class Classes{
        Classes(){
            try{
                TableInterface.dropTable(connection, "Classes");

                TableInterface.createTable(connection, ddlCreateTableClasses);
                TableInterface.populateTable(connection, ddlInsertTableClasses);
```

```
            }
            catch(SQLException e){System.out.println(e);}
        }
        public void updateGrade(String studentID, String newGrade) throws
SQLException{
            String query = "UPDATE Classes SET grade = " + newGrade + " WHERE
empID = " + studentID;
            PreparedStatement psUpdGrade = connection.prepareStatement(query);
            try{
                psUpdGrade.executeUpdate();
            }catch(SQLException e){System.out.println(e);}
        }
    }
    class AggregateGrades{
        AggregateGrades(String iD){
            try{
                String ddlInsertAggregateGrade =
                "INSERT INTO AggregateGrades SELECT grade, count(grade) "+
                "FROM Classes WHERE courseID LIKE '22100 "+iD +"' GROUP BY
grade;";
                TableInterface.dropTable(connection, "AggregateGrades");
                TableInterface.createTable(connection,
ddlCreateTableAggregateGrades);
                TableInterface.populateTable(connection,
ddlInsertAggregateGrade);
            }catch(SQLException e){System.out.println(e);}
    }
        public Map<Character, Integer> getDataFromTable(){
            Map<Character, Integer> result = new HashMap<>();
            try{
                Statement st = connection.createStatement();
                ResultSet rs = st.executeQuery("SELECT * FROM AggregateGrades");
                while(rs.next()){
                    Character grade = rs.getString("grade").charAt(0);
                    Integer numberStudents = rs.getInt("numberStudents");
                    result.put(grade, numberStudents);
                }
            }catch(Exception e){System.out.println(e);}
            return result;
        }
    }
}
```

StudentsDatabase class implements both previous interfaces. This class is used to stablish connection with the Database, and has inner classes Schedule, Courses, Students, Classes and AggregateGrades.

getConnection(): This method is used to stablish a connection with the Database server system.

Schedule: This is a non-static class that uses its constructor to create and populate the table, using the file that was specified. It also has a updateInstructor() method to update the column data “instructor” from the table specifying the row by the courseID parameter.

Students: This is a non-static class that uses its constructor to create and populate the table.

Classes: This is a non-static class that uses its constructor to create and populate the table.

Courses: This is a non-static class that uses its constructor to create and populate the table. It also implements updateGrade() method to update the column data “grade” from Courses table specifying the row by the studentID parameter.

AggregateGrades: This is a non-static class that uses its constructor to create and populate the table. It also implements getDataFromTable() that returns a Map collection that contains the content in the table (the grade, and its frequency).

Project4.java

```
import java.util.HashMap;
import java.util.Map;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.Pane;
import javafx.stage.Stage;

public class Project4 extends Application{
    String namefile = "C:/Program Files/MySQL/tmp/ScheduleSpring2022.txt";
    String nameTable = "Schedule";
    String url = "jdbc:mysql://localhost:3306/project4";
    String user = "root";
    String password = "root";
    @Override
    public void start(Stage ps) throws Exception{
        StudentsDatabase db = new StudentsDatabase(url, user, password);
        MyPoint center = new MyPoint(300,250, null);

        ps.setTitle("Project 4");
        Label numberSlices = new Label("Pick between R or P 22100 course");
        Button newbtn = new Button("Submit");
```

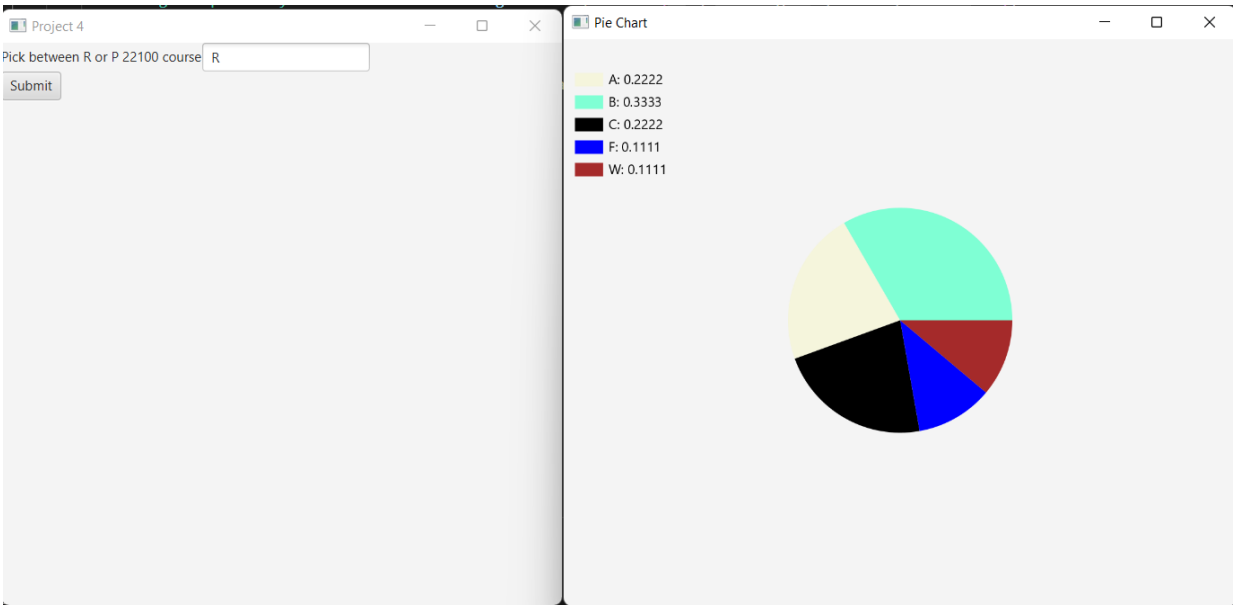
```
        GridPane root = new GridPane();
        TextField userInput1 = new TextField();
        root.addRow(1, numberSlices, userInput1);
        root.addRow(3, newbtn);
        newbtn.setOnAction(e->{
            Map<Character, Integer> grades = new HashMap<>();
            try{
                StudentsDatabase.Schedule tableSchedule = db.new
Schedule(namefile);
                StudentsDatabase.Students tableStudents = db.new Students();
                StudentsDatabase.Courses tableCourses = db.new Courses();
                StudentsDatabase.Classes tableClasses = db.new Classes();
                StudentsDatabase.AggregateGrades tableAGrades = db.new
AggregateGrades(userInput1.getText());
                grades = tableAGrades.getDataFromTable();

                //tableSchedule.updateInstructor("22100 P","Elver Galarga");
                //tableClasses.updateGrade("10001", "A");
            }
            catch(Exception ex){System.out.println(ex);}
            HistogramAlphaBet histogram = new HistogramAlphaBet(grades);
            Stage stage2 = new Stage();
            Pane p = new Pane();
            Canvas canvas = new Canvas(600,600);
            GraphicsContext GC = canvas.getGraphicsContext2D();
            HistogramAlphaBet.MyPieChart testPie = histogram.new MyPieChart(6, 0,
center, 100);
            testPie.printPie();
            testPie.draw(GC);
            p.getChildren().add(canvas);
            Scene sc = new Scene(p,600,600,MyColor.White.getJavaFxColor());
            stage2.setTitle("Pie Chart");
            stage2.setScene(sc);
            stage2.show();
        });
        Scene scene = new Scene(root, 500, 500);
        ps.setScene(scene);
        ps.show();
    }
    public static void main(String args[]){
        launch(args);
    }
}
```

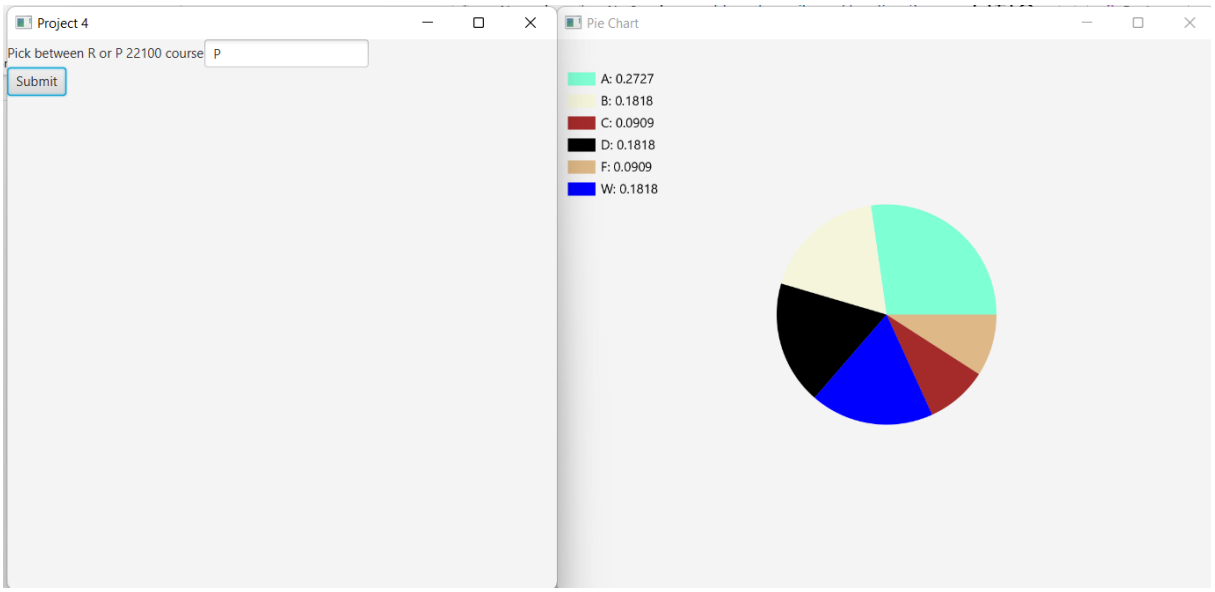
This class is application class that uses TextField and Button objects to retrieve data from the user and then, stablish the connection, create and populate the tables. Then, the frequency is obtained to create a MyPieChart object to produce the Pie Chart.

Output

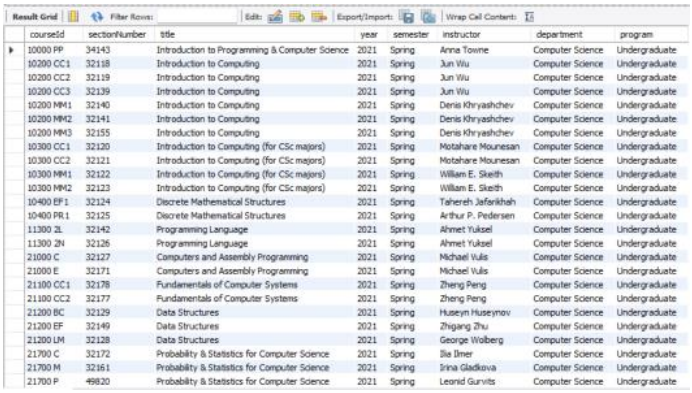
For id = R



For id = P



Schedule Table



courseId	sectionNumber	title	year	semester	instructor	department	program
10000 PP	34143	Introduction to Programming & Computer Science	2021	Spring	Anna Towne	Computer Science	Undergraduate
10200 CC1	32118	Introduction to Computing	2021	Spring	Jun Wu	Computer Science	Undergraduate
10200 CC2	32119	Introduction to Computing	2021	Spring	Jun Wu	Computer Science	Undergraduate
10200 CC3	32139	Introduction to Computing	2021	Spring	Jun Wu	Computer Science	Undergraduate
10200 MM1	32140	Introduction to Computing	2021	Spring	Denis Khryashchev	Computer Science	Undergraduate
10200 MM2	32141	Introduction to Computing	2021	Spring	Denis Khryashchev	Computer Science	Undergraduate
10200 MM3	32155	Introduction to Computing	2021	Spring	Denis Khryashchev	Computer Science	Undergraduate
10300 CC1	32120	Introduction to Computing (for CSc majors)	2021	Spring	Motahare Mounesan	Computer Science	Undergraduate
10300 CC2	32121	Introduction to Computing (for CSc majors)	2021	Spring	Motahare Mounesan	Computer Science	Undergraduate
10300 MM1	32122	Introduction to Computing (for CSc majors)	2021	Spring	William E. Skeith	Computer Science	Undergraduate
10300 MM2	32123	Introduction to Computing (for CSc majors)	2021	Spring	William E. Skeith	Computer Science	Undergraduate
10400 EF1	32124	Discrete Mathematical Structures	2021	Spring	Tahereh Jafarikhah	Computer Science	Undergraduate
10400 PR1	32125	Discrete Mathematical Structures	2021	Spring	Arthur P. Pedersen	Computer Science	Undergraduate
11300 2L	32142	Programming Language	2021	Spring	Ahmet Yuksel	Computer Science	Undergraduate
11300 2N	32126	Programming Language	2021	Spring	Ahmet Yuksel	Computer Science	Undergraduate
21000 C	32127	Computers and Assembly Programming	2021	Spring	Michael Vulis	Computer Science	Undergraduate
21000 E	32171	Computers and Assembly Programming	2021	Spring	Michael Vulis	Computer Science	Undergraduate
21100 CC1	32178	Fundamentals of Computer Systems	2021	Spring	Zheng Peng	Computer Science	Undergraduate
21100 CC2	32177	Fundamentals of Computer Systems	2021	Spring	Zheng Peng	Computer Science	Undergraduate
21200 BC	32129	Data Structures	2021	Spring	Huseyn Huseynov	Computer Science	Undergraduate
21200 EF	32149	Data Structures	2021	Spring	Zhigang Zhu	Computer Science	Undergraduate
21200 LM	32128	Data Structures	2021	Spring	George Wolberg	Computer Science	Undergraduate
21700 C	32172	Probability & Statistics for Computer Science	2021	Spring	Ilia Ilmer	Computer Science	Undergraduate
21700 M	32161	Probability & Statistics for Computer Science	2021	Spring	Irina Gladkova	Computer Science	Undergraduate
21700 P	49820	Probability & Statistics for Computer Science	2021	Spring	Leonid Gurvits	Computer Science	Undergraduate

Students Table

empID	firstName	lastName	email	gender
10001	Jair	R	jairr@gmail.com	M
10002	Melissa	M	melissam@gmail.com	F
10003	Peter	D	peterd@gmail.com	M
10004	Bennito	M	bennitom@gmail.com	M
10005	Nicolas	M	nicolasm@gmail.com	U
10006	Rafael	C	rafaelc@gmail.com	M
10007	Freya	A	freyaa@gmail.com	F
10008	Leonel	M	leonelm@gmail.com	M
10009	Sergio	A	sergioa@gmail.com	M
10010	Brithany	F	brithanyf@gmail.com	U
10011	Marie	C	mariec@gmail.com	F
10012	Jennifer	L	jenniferl@gmail.com	F
10013	Megan	F	meganf@gmail.com	F
10014	Mia	K	miak@gmail.com	F
10015	Cristiano	R	cristianor@gmail.com	M
10016	Ibai	L	ibail@gmail.com	U
10017	Auron	P	auronp@gmail.com	M
10018	Dwayne	J	dwaynej@gmail.com	M
10019	Henrry	C	henrryc@gmail.com	M
10020	Alex	T	alext@gmail.com	U
NULL	NULL	NULL	NULL	NULL

Classes Table

empID	courseID	sectionNumber	year	semester	grade
10001	22100 P	32132	2021	Fall	F
10002	22100 P	32132	2021	Fall	A
10003	22100 P	32132	2021	Fall	W
10005	22100 P	32132	2021	Fall	A
10007	22100 P	32132	2021	Fall	B
10008	22100 P	32132	2021	Fall	A
10009	22100 P	32150	2021	Fall	D
10014	22100 P	32132	2021	Fall	W
10016	22100 P	32132	2021	Fall	D
10017	22100 P	32132	2021	Fall	C
10018	22100 P	32132	2021	Fall	B
10004	22100 R	32150	2021	Fall	B
10006	22100 R	32150	2021	Fall	C
10010	22100 R	32132	2021	Fall	C
10011	22100 R	32150	2021	Fall	A
10012	22100 R	32150	2021	Fall	B
10013	22100 R	32150	2021	Fall	B
10015	22100 R	32150	2021	Fall	F
10019	22100 R	32150	2021	Fall	W
10020	22100 R	32150	2021	Fall	A
NULL	NULL	NULL	NULL	NULL	NULL

Courses Table

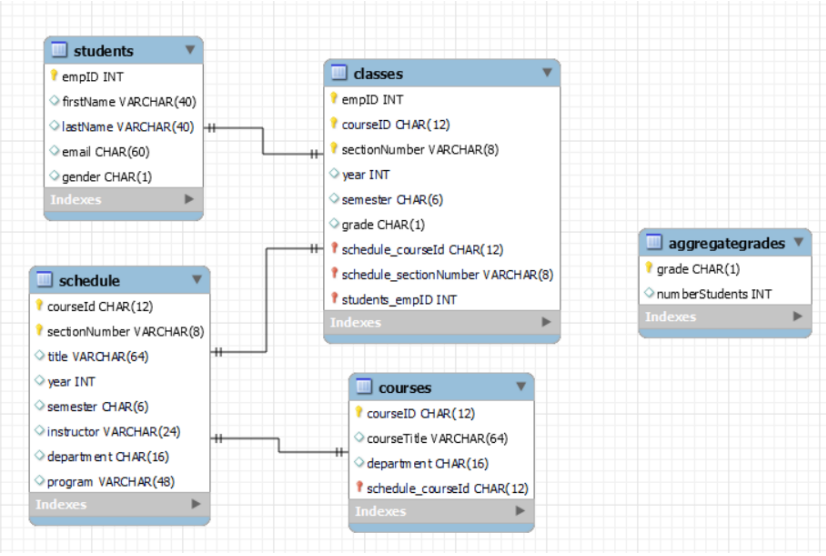
courseID	courseTitle	department
10200 MM3	Introduction to Computing	Computer Science
10300 CC1	Introduction to Computing (for CSc majors)	Computer Science
10300 CC2	Introduction to Computing (for CSc majors)	Computer Science
10300 MM1	Introduction to Computing (for CSc majors)	Computer Science
10300 MM2	Introduction to Computing (for CSc majors)	Computer Science
10400 EF1	Discrete Mathematical Structures	Computer Science
10400 PR1	Discrete Mathematical Structures	Computer Science
11300 2L	Programming Language	Computer Science
11300 2N	Programming Language	Computer Science
21000 C	Computers and Assembly Programming	Computer Science
21000 E	Computers and Assembly Programming	Computer Science
21100 CC1	Fundamentals of Computer Systems	Computer Science
21100 CC2	Fundamentals of Computer Systems	Computer Science
21200 BC	Data Structures	Computer Science
21200 EF	Data Structures	Computer Science
21200 LM	Data Structures	Computer Science
21700 C	Probability & Statistics for Computer Science	Computer Science
21700 M	Probability & Statistics for Computer Science	Computer Science
21700 P	Probability & Statistics for Computer Science	Computer Science
22000 C	Algorithms	Computer Science
22000 D	Algorithms	Computer Science
22000 M	Algorithms	Computer Science
22100 F	Software Design Laboratory	Computer Science
22100 P	Software Design Laboratory	Computer Science
22100 R	Software Design Laboratory	Computer Science

AggregateGrade Table

Result Grid | Filter Rows:

grade	numberStudents
A	3
B	2
C	1
D	2
F	1
W	2
NULL	NULL

Schema Diagram



Implemented Classes

- Class DecimalFormat
- Class Collections
- Class HashMap
- Class LinkedHashMap
- Class Map
- Class Entry
- Class Collectors
- Class GraphicsContext
- Class ArcType
- Class Application
- Class Scene
- Class Canvas
- Class Button
- Class Label
- Class TextField
- Class GridPane
- Class Pane
- Class Stage
- Class Connection
- Class DriverManager
- Class ResultSet
- Class SQLException
- Class Statement