SOFTWARE DESIGN LABORATORY
Assignment 4

Josh Miranda

Hesham Auda

Section P

13 May 2021

1. Problem

The objective of this assignment is to create a Java application that will connect to a MySQL database that will allow us to manipulate anyway we want. The tables created are Students, Courses and Classes that will be populated and created. For our assignment we are given a file containing the Spring 2021 schedule and the Courses and Classes tables will be added into the database. In the Java application the PreparedStatement is used for the DDL statements and SQL queries. Within the application there is also a way to go into the database and update the student grades. Once there is a database of grades then we can calculate and display the number of students for each letter grade in the Spring 2021 semester for the CSc 22100 class. Then with JavaFX from the last project we will input the grades into a pie chart where the data can be displayed neatly.

2. Java application developed

Class DBConnect

The first java class that will be talked about is the Class DBConnect. The Class DBConnect is where the majority of the application is created. It is responsible for connecting the database, creating the tables, populating, updating, inserting students and counting grades.

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.sql.*;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
```

Here are all the packages used for Class DBConnect. The package java.io.IOException is responsible for sending a signal that an error has occurred. The two packages java.nio.file.Files and java.nio.file.Paths are used for locating and accessing files from the computer. The java.sql package ensures that we are able to use the MySQL database. The java.util packages HashMap, List, Map and Scanner are just so that I am able to manipulate and use HashMap, List, Map and Scanner in my program.

```
public class DBConnect {
    static final String DB_URL = "jdbc:mysql://localhost:3306/studentdatabase";
    static final String USER = "root";
    static final String PASS = "207975137";
```

When connecting the MySQL database into the IDE that is being used, it will need the database URL, User and Pass.

```
public static void Conn() {
    // Open a connection
    try {
        Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
        if (conn != null) {
            System.out.println("Table created");
            String sT = "CREATE TABLE IF NOT EXISTS Students"
                    + "(empID INT UNSIGNED not NULL ,"
                    + "firstName VARCHAR(45),"
                    + "lastName VARCHAR(45),"
                    + "email VARCHAR(255),"
                    + "gender ENUM ('M', 'F', 'U'),"
                    + "PRIMARY KEY(empID))";
            PreparedStatement stmt = conn.prepareStatement(sT);
            stmt.execute(sT);
```

Then to open a connection a public static class is created called Conn() which will use a try and catch. Then a variable will be created called Connect conn which will take in DriverManager.getConnection(DB_URL, USER, PASS). This will establish a connection between the user and driver and the parameters is the SQL database that is being connected to. Underneath is a if statement that states if the connection is not null then proceed with create tables. The if statement is responsible for creating the three tables Student, Courses and Classes. At the top there is a print statement that allows us to see if the tables have been created. Next a String called sT is being created for the creation of the columns in the table. These columns include empID, firstName, lastName, email and gender, which will be populated later on. The PreparedStatement called stmt prepares the string that was just made so that it is ready when it is sent to the database. Then there is stmt.execute(sT) which sends the string that was prepared earlier to the database.

```
String cO = "CREATE TABLE IF NOT EXISTS Courses"
        + "(courseID VARCHAR(45),"
        + "courseTitle VARCHAR(200),"
        + "department VARCHAR(45),"
        + "PRIMARY KEY(courseID))";
stmt.execute(cO);

String cL = "CREATE TABLE IF NOT EXISTS Classes"
        + "(courseID VARCHAR(45),"
        + "studentID INT UNSIGNED not NULL,"
        + "sectionNo INT UNSIGNED not NULL,"
        + "year INTEGER,"
        + "semester ENUM ('fall', 'spring'),"
        + "grade ENUM ('A', 'B', 'C', 'D', 'F', 'W'),"
        + "PRIMARY KEY(courseID, studentID, sectionNo))";
stmt.execute(cL);
```

The same process that was done for the Student table was done for both Courses and Classes table.

```
new addTables.populateClass();
new addTables.populateCourse();
new addTables.populateStudent();
new addTables.updateClasses();
addTables.studentAdd();
new addTables.letterData();
```

Here is where all the functions throughout DBConnect are called. These individual functions will be discussed throughout the report.

```
public static class addTables {
    static Map<String, String[]> courses = new HashMap<>();
    static Map<String, String[]> classes = new HashMap<>();

    public static void parseData() throws IOException {
        // Test for Parsing txt
        List<String> content = Files.readAllLines(Paths.get( first: "C:\\Users\\pug\\Downloads\\Spring 2021\\CSC 22100 - P\\data.csv"));
```

This class is called Class addTables which will add the tables into the database. Below are Maps for both Courses and Classes which will be used to parse through the data. That will be done in the parseData() function. The last line of the image is where the file Spring 2021 schedule text file is being read. All the lines are being read and are added into a list.

```
for (int i = 1; i < content.size(); i++) {
    String[] s = content.get(i).split( regex: ",");

    // Courses :: Key , Title/Department :: Values
    String[] course1 = new String[]{s[2], s[6]};
    courses.put(s[0], course1);

    // Classes: courseID :: Key, studentID, sectionNo, semester, year, grade :: Values
    String[] class_array = new String[]{s[1], s[4], s[3]};
    classes.put(s[0], class_array);
}
```

In this for loop it is going through the entire text file and making it easier for the program to read. Then it will be imputed to a string either courses and class with correlating Key and Value.

```
// Classes contains: courseId, sectionNo, semester, year
String[] tempt = new String[]{};
for (Map.Entry<String, String[]> entry : classes.entrySet()) {
    //System.out.println(entry.getKey() + " " + Arrays.toString(entry.getValue()));
    tempt = entry.getValue();
    //System.out.println(tempt[0]);

}
```

The tempt string is a copy of classes which allow me to see the Value of the index. For example temp[0] will give me the first value of the section number.

```
public static class populateClass {
    String classTable = "insert into classes (courseID, studentID, sectionNo, year, semester, grade) values (?, ?, ?, ?, ?, ?)";
    Connection conn;
    {
        try {
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            PreparedStatement ps = conn.prepareStatement(classTable);
            String[] tempt = new String[]{};
            int count = 0;
            for (Map.Entry<String, String[]> entry : classes.entrySet()) {
                tempt = entry.getValue();
                ps.setString( parameterIndex: 1, entry.getKey()); //courseID
                ps.setString( parameterIndex: 2, String.valueOf(count)); //studentID
                count++;
                ps.setString( parameterIndex: 3, tempt[0]); //sectionNo
                ps.setString( parameterIndex: 4, tempt[2]); //year
                ps.setString( parameterIndex: 5, tempt[1]); //semester
                ps.setString( parameterIndex: 6,  x: null); //grade
                ps.addBatch();
            }
            ps.executeBatch();
```

The Class populateClass is where the tables will be filled for classes. String classTable contains the columns that will be imputed like courseID and studentID. Then next to it are values with question marks for placeholders. In the try and catch, we first established a connection with the database and then prepared the classTable in the PreparedStatement. Next we initialized the tempt string and count. The for loop is responsible for adding the values from the text file into arrays. Looking at the first ps.setString parameters the ‘1’

represents the first place value of the question mark as previously stated. That is the couseID, which will be populated with entry.getKey(). This is done for the rest for populating the tables. Then the batch is executed and sent to the SQL database.

```
public static class populateCourse {
    String courseTable = "insert into Courses (courseID, courseTitle, department) values (?, ?, ?)";
    Connection conn;

    {
        try {
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            PreparedStatement ps = conn.prepareStatement(courseTable);
            for (Map.Entry<String, String[]> entry : courses.entrySet()) {
                String[] tempt = entry.getValue();
                ps.setString( parameterIndex: 1, entry.getKey()); //courseID
                ps.setString( parameterIndex: 2, tempt[0]); //courseTitle
                ps.setString( parameterIndex: 3, tempt[1]); //department
                ps.addBatch();
            }
            ps.executeBatch();

        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
```

The process for populateCourse is the same as before for populateClass.

```
public static class populateStudent {
    String popStud = "REPLACE INTO Students" + "(empID, firstName, lastName, email, gender) VALUES (?, ?, ?, ?, ?)";
    Connection conn;

    {
        try {
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            PreparedStatement ps = conn.prepareStatement(popStud);
            Integer[] empID = new Integer[]{63772505, 43775455, 23772445, 28776423};
            String[] firstName = new String[]{"Muler", "Jules", "Ash", "Missy"};
            String[] lastName = new String[]{"Mayer", "Notch", "Canner", "May"};
            String[] email = new String[]{"MulerMayer@gmail.com", "JulesNotch@gmail.com", "AshCanner@gmail.com", "MissyMay@gmail.com"};
            String[] gender = new String[]{"M", "F", "U", "F"};
            for (int i = 0; i < firstName.length; i++) {
                ps.setInt( parameterIndex: 1, empID[i]); //empID
                ps.setString( parameterIndex: 2, firstName[i]); //firstName
                ps.setString( parameterIndex: 3, lastName[i]); //lastName
                ps.setString( parameterIndex: 4, email[i]); //email
                ps.setString( parameterIndex: 5, gender[i]); //gender
                ps.addBatch();
```

The class populateStudent is different from populating Course and Classes as it is static. Arrays are created for each column in the database like empID, firstName, lastName, email, gender. Within these arrays is preset data of four students. Then the for loop below reads through that preset data and inputs it into the database.

```
public static class updateStudent {
    PreparedStatement ps = null;
    Connection conn;

    {
        try {
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            String upStud = "UPDATE Students SET email= 'MaxRuby@gmail.com' WHERE emplID= 28776423";
            ps = conn.prepareStatement(upStud);
            ps.execute();

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

This is where the student can be updated in the database. This allows them to change their email.

```
public static class updateCourse {
    PreparedStatement ps = null;
    Connection conn;

    {
        try {
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            String upCour = "UPDATE Courses SET courseTitle= 'Data Structures' WHERE courseID= 32130";
            ps = conn.prepareStatement(upCour);
            ps.execute();

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

```
public static class updateClasses {
    PreparedStatement ps = null;
    Connection conn;

    {
        try {
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            String upclass = "UPDATE Classes SET grade='B' WHERE studentID= 5";
            ps = conn.prepareStatement(upclass);
            ps.execute();

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

The update classes for both Courses and Classes are the same format as the updateStudent class but courses update course title and classes update grades.

```java
public static void studentAdd() {
    Scanner myStu = new Scanner(System.in);
    System.out.print("Enter total students: ");
    int studentTotal = Integer.parseInt(myStu.nextLine());
    Connection conn;

    for (int i = 0; i < studentTotal; i++) {
        System.out.print("Enter courseID: ");
        String courseID = myStu.nextLine();
        System.out.print("Enter studentID: ");
        int studentID = Integer.parseInt(myStu.nextLine());
        System.out.print("Enter sectionNo: ");
        int sectionNo = Integer.parseInt(myStu.nextLine());
        System.out.print("Enter year: ");
        int year = Integer.parseInt(myStu.nextLine());
        System.out.print("Enter semester: ");
        String semester = myStu.nextLine();
        System.out.print("Enter grade: ");
        String grade = myStu.nextLine();
```

This class is where the user can manually add students into the database.The User is asked how many students they would like to add and they will be asked to input the empID, firstName, lastName, email and gender of the student.

```java
try {
    conn = DriverManager.getConnection(DB_URL, USER, PASS);
    PreparedStatement ps = conn.prepareStatement( sql: "INSERT Classes (courseID, studentID, sectionNo, year, semester, grade)"
            + "VALUES ('" + courseID + "','" + studentID + "','" + sectionNo + "','" + year + "','" + semester + "','" + grade + "')");
    System.out.println("Data INSERTED to Classes table");
    ps.executeUpdate();


} catch (SQLException throwables) {
    throwables.printStackTrace();
}
```

This try catch takes the User inputs from before and inserts them into the database.

```
public static class letterData {
    HashMap<Character, Float> g = new HashMap<Character, Float>();
    HashMap<Character, Float> probability_map = new HashMap<Character, Float>();
    float numStudent = 0;
    int allLetters = 0;
```

LetterData is where the grades from the database will be manipulated and used. Below are the two hashmaps g and probability map. The g hashmap is the letter grade with how many students received that grade. The probability map is the same as g but it will be the ratio of the letter grade. Underneath is num student and allLetters initialized at 0.

```
try {
    String totalGrades = "";
    Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
    PreparedStatement ps = conn.prepareStatement( sql: "SELECT * FROM CLASSES WHERE (courseID = '22100') AND (semester = 'spring')");
    ResultSet result = ps.executeQuery();

    while (result.next()) {
        String grades = result.getString( columnLabel: "grade");
        totalGrades += grades;
    }
    System.out.println(totalGrades);

    char[] letterGrades = totalGrades.toCharArray();
    for (char letter : letterGrades) {
        if (g.containsKey(letter)) {
            g.put(letter, g.get(letter) + 1);
        } else {
            g.put(letter, (float) 1);
        }
```

Inside the try and catch data from the courseID 22100 and semester Spring are being chosen from the database. The while statement goes through each row getting the total amount of grades. Then total grades will be imputed into the character array letterGrades. This will sort the letter grade so that it will count how many receive a specific grade.

```
for (Map.Entry<Character, Float> e : g.entrySet()){
    numStudent += e.getValue();
    allLetters += 1;

}
System.out.println(numStudent);
for (Map.Entry<Character, Float> e: g.entrySet()) {
    //divides hash by total
    float prob = e.getValue()/numStudent;
    //replace values in hash map with result
    probability_map.put(e.getKey(), prob);
}
System.out.println(probability_map);
```

The first for loop counts the total number of students. In the next loop it will find the probability by going through the Map and divides each element by the total number of students.

Class Main and Class HistogramAlphabet

```
DBConnect.Conn();
DBConnect.addTables.parseData();

HistogramAlphaBet.MyPieChart p = new HistogramAlphaBet.MyPieChart( x: 400, y: 10, height: 200, width: 550);
p.draw(gc);
```

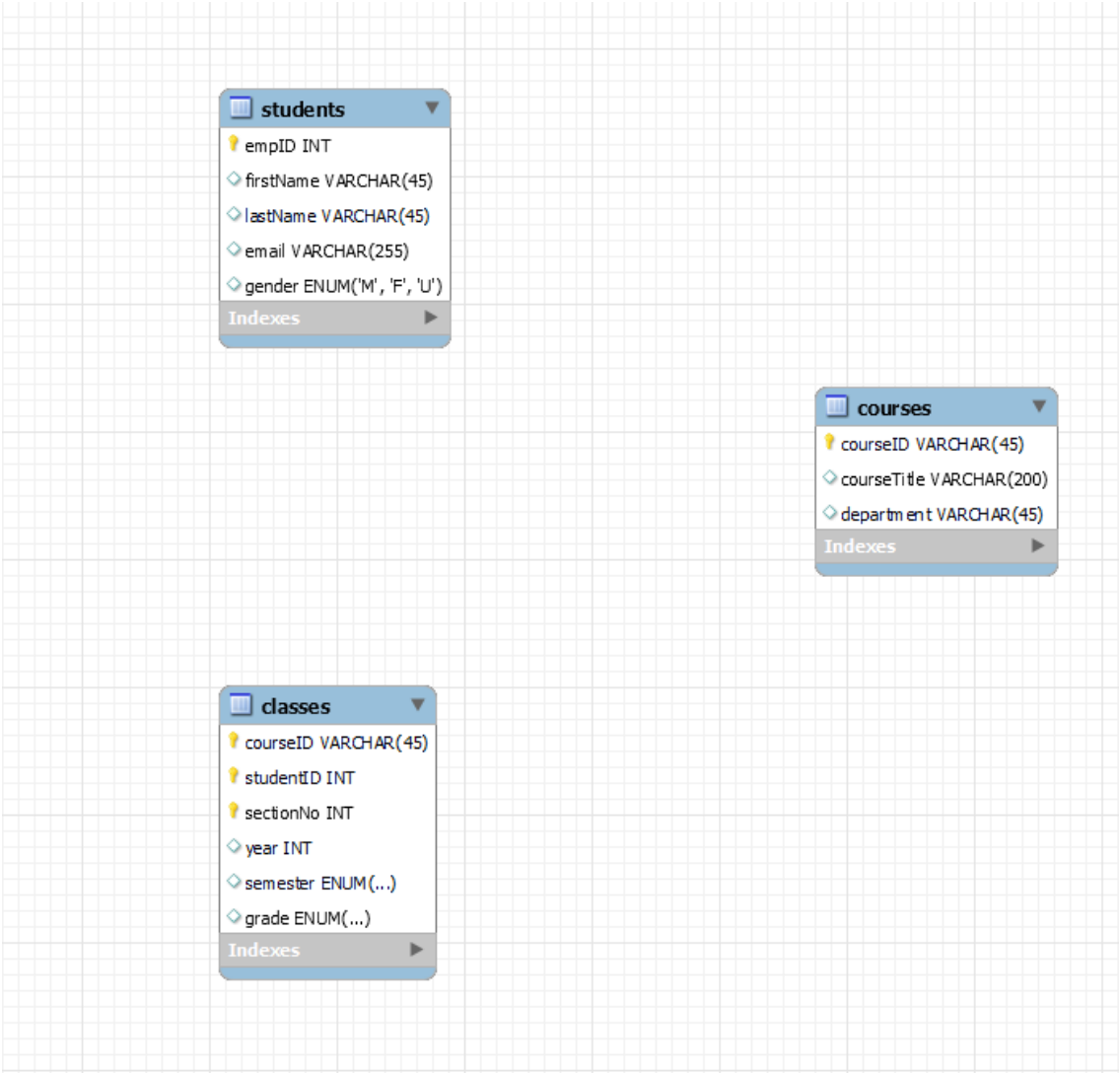
Class Main and Class HistogramAlphabet were mainly the same as the last project but there were minor changes. For main, functions were called and the parameters for the piechart removed ‘n’ slices since it was no longer needed.

```
HashMap<Character, Float> pieChartMap = new HashMap<Character, Float>();
HashMap<Character, Float> pieChart = new HashMap<Character, Float>();
```

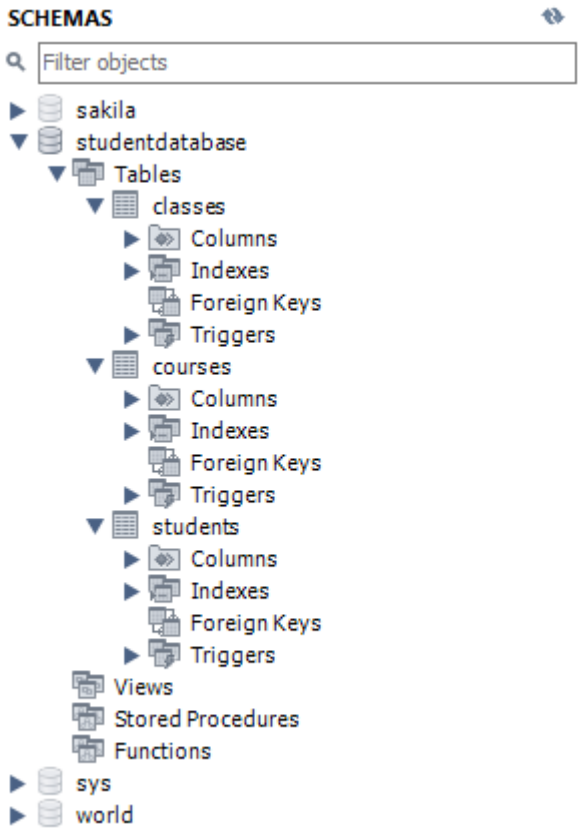
```
int u = 0;
for(Map.Entry<Character, Float> entry : this.pieChartMap.entrySet()) {
    slice_pie_chart[u].key = String.valueOf(entry.getKey());
    slice_pie_chart[u].value = String.valueOf(entry.getValue());
    u++;
    if (u == this.n){
        break;
    }
}
```

For the Class HistogramAlphabet it was also the same but replaced the old Maps with the Maps from the database.

3. Database ER diagram, Schema



This is the ER diagram for the MySQL database with the tables students, courses and classes.



Here are the Schemas for my studentdatabase that carry the three tables. There are no foreign keys and triggers within the folders.

4. Code Developed

Class Main

```
package sample;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.stage.Stage;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import java.io.IOException;


class Borderdim{
   static double height = 700;
   static double width = 1000;
}

public class Main extends Application {

   public void start(Stage primaryStage) throws
IOException {
       BorderPane root = new BorderPane();
       Canvas canvas = new
Canvas(Borderdim.width,Borderdim.height);
       Scene scene = new Scene(root,
Borderdim.width,Borderdim.height,
MyColor.WHITE.getColor());
       GraphicsContext gc =
canvas.getGraphicsContext2D();

       DBConnect.Conn();
       DBConnect.addTables.parseData();

       HistogramAlphaBet.MyPieChart p = new
```

```
HistogramAlphaBet.MyPieChart(400, 10, 200, 550);
        p.draw(gc);


        //show shape
       root.getChildren().add(canvas);
       primaryStage.setTitle("MyShape");
       primaryStage.setScene(scene);
       primaryStage.show();

    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Class DBConnect

```
package sample;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.sql.*;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

public class DBConnect {
    static final String DB_URL =
"jdbc:mysql://localhost:3306/studentdatabase";
    static final String USER = "root";
    static final String PASS = "207975137";
```

```
   public static void Conn() {
        // Open a connection
        try {
            Connection conn =
DriverManager.getConnection(DB_URL, USER, PASS);
            if (conn != null) {
                System.out.println("Table created");
                String sT = "CREATE TABLE IF NOT
EXISTS Students"
                        + "(empID INT UNSIGNED not
NULL ,"
                        + "firstName VARCHAR(45),"
                        + "lastName VARCHAR(45),"
                        + "email VARCHAR(255),"
                        + "gender ENUM ('M', 'F',
'U'),"
                        + "PRIMARY KEY(empID))";
                PreparedStatement stmt =
conn.prepareStatement(sT);
                stmt.execute(sT);

                String cO = "CREATE TABLE IF NOT
EXISTS Courses"
                        + "(courseID VARCHAR(45),"
                        + "courseTitle
VARCHAR(200),"
                        + "department VARCHAR(45),"
                        + "PRIMARY KEY(courseID))";
                stmt.execute(cO);

                String cL = "CREATE TABLE IF NOT
EXISTS Classes"
                        + "(courseID VARCHAR(45),"
                        + "studentID INT UNSIGNED
not NULL,"
                        + "sectionNo INT UNSIGNED
```

```
not NULL,"
                        + "year INTEGER,"
                        + "semester ENUM ('fall',
'spring'),"
                        + "grade ENUM ('A', 'B',
'C', 'D', 'F', 'W'),"
                        + "PRIMARY KEY(courseID,
studentID, sectionNo))";
                stmt.execute(cL);


                new addTables.populateClass();
                new addTables.populateCourse();
                new addTables.populateStudent();
                new addTables.updateClasses();
                addTables.studentAdd();
                new addTables.letterData();

            }
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
    }


    public static class addTables {
        static Map<String, String[]> courses = new
HashMap<>();
        static Map<String, String[]> classes = new
HashMap<>();

        public static void parseData() throws
IOException {
            // Test for Parsing txt
            List<String> content =
Files.readAllLines(Paths.get("C:\\Users\\pug\\Downl
oads\\Spring 2021\\CSC 22100 - P\\data.csv"));
```

```
            for (int i = 1; i < content.size(); i++)
{
                String[] s =
content.get(i).split(",");

                // Courses :: Key , Title/Department
:: Values
                String[] course1 = new
String[]{s[2], s[6]};
                courses.put(s[0], course1);

                // Classes: courseID :: Key,
studentID, sectionNo, semester, year, grade ::
Values
                String[] class_array = new
String[]{s[1], s[4], s[3]};
                classes.put(s[0], class_array);
            }
            // Classes contains: courseId,
sectionNo, semester, year
            String[] tempt = new String[]{};
            for (Map.Entry<String, String[]> entry :
classes.entrySet()) {
                //System.out.println(entry.getKey()
+ " " + Arrays.toString(entry.getValue()));
                tempt = entry.getValue();
                //System.out.println(tempt[0]);

            }
        }

        public static class populateClass {
            String classTable = "insert into classes
(courseID, studentID, sectionNo, year, semester,
grade) values (?, ?, ?, ?, ?, ?)";
            Connection conn;
            {
```

```
                try {
                    conn =
DriverManager.getConnection(DB_URL, USER, PASS);
                    PreparedStatement ps =
conn.prepareStatement(classTable);
                    String[] tempt = new String[]{};
                    int count = 0;
                    for (Map.Entry<String, String[]>
entry : classes.entrySet()) {
                        tempt = entry.getValue();
                        ps.setString(1,
entry.getKey()); //courseID
                        ps.setString(2,
String.valueOf(count)); //studentID
                        count++;
                        ps.setString(3, tempt[0]);
//sectionNo
                        ps.setString(4, tempt[2]);
//year
                        ps.setString(5, tempt[1]);
//semester
                        ps.setString(6, null);
//grade
                        ps.addBatch();
                    }
                    ps.executeBatch();

                } catch (SQLException throwables) {
                    throwables.printStackTrace();
                }
            }
        }

        public static class populateCourse {
            String courseTable = "insert into
Courses (courseID, courseTitle, department) values
(?, ?, ?)";
```

```
            Connection conn;

            {
                try {
                    conn =
DriverManager.getConnection(DB_URL, USER, PASS);
                    PreparedStatement ps =
conn.prepareStatement(courseTable);
                    for (Map.Entry<String, String[]>
entry : courses.entrySet()) {
                        String[] tempt =
entry.getValue();
                        ps.setString(1,
entry.getKey()); //courseID
                        ps.setString(2, tempt[0]);
//courseTitle
                        ps.setString(3, tempt[1]);
//department
                        ps.addBatch();
                    }
                    ps.executeBatch();

                } catch (SQLException throwables) {
                    throwables.printStackTrace();
                }
            }
        }

        public static class populateStudent {
            String popStud = "REPLACE INTO Students"
+ "(empID, firstName, lastName, email, gender)
VALUES (?, ?, ?, ?, ?)";
            Connection conn;

            {
                try {
                    conn =
```

```
DriverManager.getConnection(DB_URL, USER, PASS);
                    PreparedStatement ps =
conn.prepareStatement(popStud);
                    Integer[] empID = new
Integer[]{63772505, 43775455, 23772445, 28776423};
                    String[] firstName = new
String[]{"Muler", "Jules", "Ash", "Missy"};
                    String[] lastName = new
String[]{"Mayer", "Notch", "Canner", "May"};
                    String[] email = new
String[]{"MulerMayer@gmail.com",
"JulesNotch@gmail.com", "AshCanner@gmail.com",
"MissyMay@gmail.com"};
                    String[] gender = new
String[]{"M", "F", "U", "F"};
                    for (int i = 0; i <
firstName.length; i++) {
                        ps.setInt(1, empID[i]);
//empID
                        ps.setString(2,
firstName[i]); //firstName
                        ps.setString(3,
lastName[i]); //lastName
                        ps.setString(4, email[i]);
//email
                        ps.setString(5, gender[i]);
//gender
                        ps.addBatch();
                    }
                    ps.executeBatch();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
```

```
        public static class updateStudent {
            PreparedStatement ps = null;
            Connection conn;

            {
                try {
                    conn =
DriverManager.getConnection(DB_URL, USER, PASS);
                    String upStud = "UPDATE Students
SET email= 'MaxRuby@gmail.com' WHERE emplID=
28776423";
                    ps =
conn.prepareStatement(upStud);
                    ps.execute();

                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }

        public static class updateCourse {
            PreparedStatement ps = null;
            Connection conn;

            {
                try {
                    conn =
DriverManager.getConnection(DB_URL, USER, PASS);
                    String upCour = "UPDATE Courses
SET courseTitle= 'Data Structures' WHERE courseID=
32130";
                    ps =
conn.prepareStatement(upCour);
                    ps.execute();

                } catch (SQLException e) {
```

```
                    e.printStackTrace();
                }
            }
        }

        public static class updateClasses {
            PreparedStatement ps = null;
            Connection conn;

            {
                try {
                    conn =
DriverManager.getConnection(DB_URL, USER, PASS);
                    String upclass = "UPDATE Classes
SET grade='B' WHERE studentID= 5";
                    ps =
conn.prepareStatement(upclass);
                    ps.execute();

                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }

        public static void studentAdd() {
            Scanner myStu = new Scanner(System.in);
            System.out.print("Enter total students:
");
            int studentTotal =
Integer.parseInt(myStu.nextLine());
            Connection conn;

            for (int i = 0; i < studentTotal; i++) {
                System.out.print("Enter courseID:
");
                String courseID = myStu.nextLine();
```

```
                System.out.print("Enter studentID:
");
                int studentID =
Integer.parseInt(myStu.nextLine());
                System.out.print("Enter sectionNo:
");
                int sectionNo =
Integer.parseInt(myStu.nextLine());
                System.out.print("Enter year: ");
                int year =
Integer.parseInt(myStu.nextLine());
                System.out.print("Enter semester:
");
                String semester = myStu.nextLine();
                System.out.print("Enter grade: ");
                String grade = myStu.nextLine();


                try {
                    conn =
DriverManager.getConnection(DB_URL, USER, PASS);
                    PreparedStatement ps =
conn.prepareStatement("INSERT Classes (courseID,
studentID, sectionNo, year, semester, grade)"
                            + "VALUES ('" + courseID
+ "','" + studentID + "','" + sectionNo + "','" +
year + "','" + semester + "','" + grade + "')");
                    System.out.println("Data
INSERTED to Classes table");
                    ps.executeUpdate();


                } catch (SQLException throwables) {
                    throwables.printStackTrace();
                }
            }
        }
```

```
        public static class letterData {
            HashMap<Character, Float> g = new
HashMap<Character, Float>();
            HashMap<Character, Float>
probability_map = new HashMap<Character, Float>();
            float numStudent = 0;
            int allLetters = 0;

            {
                try {
                    String totalGrades = "";
                    Connection conn =
DriverManager.getConnection(DB_URL, USER, PASS);
                    PreparedStatement ps =
conn.prepareStatement("SELECT * FROM CLASSES WHERE
(courseID = '22100') AND (semester = 'spring')");
                    ResultSet result =
ps.executeQuery();

                    while (result.next()) {
                        String grades =
result.getString("grade");
                        totalGrades += grades;
                    }
                    System.out.println(totalGrades);

                    char[] letterGrades =
totalGrades.toCharArray();
                    for (char letter : letterGrades)
{
                        if (g.containsKey(letter)) {
                            g.put(letter,
g.get(letter) + 1);
                        } else {
                            g.put(letter, (float)
```

```
1);
                        }
                    }

                    System.out.println(g);

                    for (Map.Entry<Character, Float>
e : g.entrySet()){
                        numStudent += e.getValue();
                        allLetters += 1;

                    }
                    System.out.println(numStudent);
                    for (Map.Entry<Character, Float>
e: g.entrySet()) {
                        //divides hash by total
                        float prob =
e.getValue()/numStudent;
                        //replace values in hash map
with result

probability_map.put(e.getKey(), prob);
                    }

System.out.println(probability_map);



                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

Class HistogramAlphaBet

```
package sample;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.text.Font;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;
import static java.util.Collections.reverseOrder;

public class HistogramAlphaBet {
   String data;
   HashMap<Character, Integer> charCountMap;
   float total = 0;
   List<Map.Entry<Character, Integer>> sorted_map;
   List<Map.Entry<Character, Float>>
sorted_probabilitymap;
   HashMap<Character, Float> probability_map;

   public HistogramAlphaBet() {
       Path path =
Paths.get("D:\\Users\\pug\\IdeaProjects\\project_3\\Alice
in Wonderland.txt");
       try {
           this.data = new
String(Files.readAllBytes(path));
       } catch (IOException e) {
           e.printStackTrace();
       }
       this.data = this.data.toLowerCase();
       this.data = this.data.replaceAll("[^a-zA-Z]", "");
```

```
        this.charCountMap = new HashMap<Character,
Integer>();
        this.probability_map = new HashMap<Character,
Float>();
        char[] strArray = this.data.toCharArray();

        for (char c : strArray) {
            if (this.charCountMap.containsKey(c)) {
                this.charCountMap.put(c,
this.charCountMap.get(c) + 1);
                this.probability_map.put(c,
this.probability_map.get(c) + 1);
            }
            else {
                this.charCountMap.put(c, 1);
                this.probability_map.put(c, (float)1);
            }
        }
        //counts the total entry
        for (int entry : this.charCountMap.values()) {
            this.total += entry;
        }

        this.sorted_map = this.charCountMap.entrySet()
                .stream()

.sorted(reverseOrder(Map.Entry.comparingByValue()))
                .collect(Collectors.toList());

        for (Map.Entry<Character, Float> entry:
probability_map.entrySet()) {
            //divides hash by total
            float prob = entry.getValue()/this.total;
            //replace values in hash map with result
            probability_map.put(entry.getKey(), prob);
        }
```

```
        this.sorted_probabilitymap =
this.probability_map.entrySet()
                .stream()

.sorted(reverseOrder(Map.Entry.comparingByValue()))
                .collect(Collectors.toList());
    }

    public static class MyPieChart {
        double x, y, height, width, StartAngle;
        int n;
        float total;
        HashMap<Character, Float> pieChartMap = new
HashMap<Character, Float>();
        HashMap<Character, Float> pieChart = new
HashMap<Character, Float>();

        public MyPieChart(double x, double y, double
height, double width){
            this.x = x;
            this.y = y;
            this.height = height;
            this.width = width;
            this.StartAngle = 0;
            //this.pieChart = pieChart;


            //total is one since all the values in
piechart add to 1
            this.total = 1;
            this.pieChartMap = (HashMap<Character, Float>)
new DBConnect.addTables.letterData().probability_map;
            this.n = new
DBConnect.addTables.letterData().allLetters;
        }

        public void draw(GraphicsContext gc){
```

```
            int i = 0;
            float sum_total = 0;
            float rest_total = 0;
            Slice [] slice_pie_chart = new
Slice[this.n+1];
            MyColor [] color_slice = MyColor.values();

            float [] float_value = new float[25];
            //copy map values to float array for easy
indexing
            int p = 0;

            for(Map.Entry<Character, Float> entry :
this.pieChartMap.entrySet()){
                float_value[p] = entry.getValue();
                p++;
                if (p == 25){
                    break;
                }
            }

            while (i < n){
                sum_total += float_value[i];
                slice_pie_chart[i] = new Slice(this.x,
this.y, this.width,this.StartAngle,float_value[i]*360 ,
color_slice[i]);
                this.StartAngle += float_value[i]*360;
                i++;
                if (i >= n){
                    rest_total = this.total - sum_total;
                    //last slice
                    slice_pie_chart[this.n] = new
Slice(this.x, this.y,
this.width,this.StartAngle,rest_total*360 ,
MyColor.WHITE);
                    slice_pie_chart[this.n].key =
String.valueOf("Remaining");
```

```
                    slice_pie_chart[this.n].value =
String.valueOf(rest_total);
                    break;
                }
            }

            int u = 0;
            for(Map.Entry<Character, Float> entry :
this.pieChartMap.entrySet()) {
                slice_pie_chart[u].key =
String.valueOf(entry.getKey());
                slice_pie_chart[u].value =
String.valueOf(entry.getValue());
                u++;
                if (u == this.n){
                    break;
                }
            }

            int space = 50;
            for (int o = 0; o < slice_pie_chart.length;
o++){
                slice_pie_chart[o].draw(gc);
                gc.setFont(new Font(20));

gc.fillText(String.valueOf(slice_pie_chart[o].key), 50,
space);
                gc.fillText(slice_pie_chart[o].value, 200,
space);
                space += 20;
            }
            System.out.println(slice_pie_chart.length);
        }
    }
}
```

Class MyArc

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

public class MyArc extends MyShape {
   double x,y,radius, startAngle, centralAngle;
   public MyColor color;

   public MyArc(double x, double y, double radius,
double startAngle, double centralAngle, MyColor
color){
        super(x, y, color);
        this.x = x;
        this.y = y;
        this.radius = radius;
        this.startAngle = startAngle;
        this.centralAngle = centralAngle;
        this.color = color;
   }

   public double getRadius() {
        return this.radius;
   }
   public double getStartAngle() {
        return this.getStartAngle();
   }
   public double getCentralAngle() {
        return this.getCentralAngle();
   }
   public double getMyArea() {
        return (Math.PI * (Math.pow(radius, 2)) *
(centralAngle/360));
   }
   public MyRectangle getMyBoundingRectangle(){
```

```
        return new MyRectangle(super.getX(),
super.getY(), this.radius, this.radius,
this.color);
    }

    @Override
    public boolean pointInMyShape(MyPoint p) {
        return false;
    }

    public String toString() {
        return "Radius:" + getRadius() + "\n
Starting Angle: " + getStartAngle() +
                "\nCentral Angle " +
getCentralAngle() + "\nArc Area:" + getMyArea();
    }
    @Override
    public void draw(GraphicsContext gc) {
        gc.setFill(this.color.getColor());
        gc.strokeArc(this.x, this.y, this.radius,
this.radius,  this.startAngle, this.centralAngle,
ArcType.ROUND);
        gc.fillArc(this.x, this.y, this.radius,
this.radius, this.startAngle, this.centralAngle,
ArcType.ROUND);
    }
}
```

Class MyColor

```
package sample;
import javafx.scene.paint.Color;

public enum MyColor {
    RED(255,0,0),
    BLUE(0,0,255),
    LIME(0,255,0),
    CYAN(0,255,255),
    GREEN(0,128,0),
    GREY(128,128,128),
    MAGENTA(255,0,255),
    PURPLE(128,0,128),
    VIOLET(148,0,211),
    YELLOW(255,255,0),
    WHITE(255,255,255),
    BLACK(0,0,0),
    HOTPINK(255,105,180),
    CHOCOLATE(210, 105, 30),
    DARKSLATEGRAY(47,79,79),
    MIDNIGHTBLUE(25,25,112),
    INDIGO(75,0,130),
    AQUAMARINE(127,255,212),
    SPRINGGREEN(0,255,127),
    DARKOLIVEGREEN(85,107,47),
    ORANGERED(255,69,0),
    MAROON(128,0,0),
    ORANGE(255,165,0),
    OLIVE(128,128,0),
    DARKSEAGREEN(143,188,143);

    int Red;
    int Green;
    int Blue;
```

```
    MyColor(int Red, int Green, int Blue){
        this.Red = Red;
        this.Green = Green;
        this.Blue = Blue;
    }
    public Color getColor() {
        return Color.rgb(Red,Green,Blue);
    }
}
```

Class MyOval

```
package sample;
import javafx.scene.canvas.GraphicsContext;
import java.lang.Math;

public class MyOval extends MyShape{
   public double x, y, height, width, xCenter, yCenter,
majorAxis, minorAxis, maxX, maxY, minX, minY;
   private MyColor color;

   public MyOval (double x, double y, double w, double h, MyColor
color) {
       super(x, y, color);
       this.width = w;
       this.height = h;
       this.color = color;
       setCenter(width, height);
       setAxes();
       this.maxX = this.getX() + width;
       this.maxY = this.getY() + height;
       this.minX = this.getX();
       this.minY = this.getY();
   }
```

```
    public double getArea(){
        return (Math.PI * width/2 * height/2);
    }
    public double getPerimeter(){
        return (2 * Math.PI * Math.sqrt((Math.pow(width/2, 2) +
Math.pow(height/2, 2)) / 2));
    }
    public double getXCenter(){
        return xCenter;
    }
    public double getYCenter() {
        return yCenter;
    }
    public double getMyArea() {
        return getArea();
     }

    public void setAxes(){
        this.majorAxis = width;
        this.minorAxis = height;
    }
    private void setCenter(double width, double height) {
        this.xCenter = (width/2) + super.getX();
        this.yCenter = (height/2) + super.getY();
    }

    @Override
    public String toString(){
        return "Perimeter:" + getPerimeter() + "\nArea:" +
getArea() + "\nX:" + getXCenter() + ", Y:" + getYCenter() +
                "\nMajor Axis: " + this.majorAxis + "\nMinor Axis:
" + this.minorAxis;
    }
    @Override
    public boolean pointInMyShape(MyPoint p) {
```

```
//maxX
//maxY
        return p.getXpoint() >= getX() && p.getXpoint() <= getX()
+ width && p.getYpoint() >= getY() && p.getYpoint() <= getY() +
height;
    }
    @Override
    public MyRectangle getMyBoundingRectangle() {
        return new MyRectangle(getX(), getY(),this.majorAxis,
this.minorAxis,this.color);
    }
    @Override
    public void draw(GraphicsContext gc){
        gc.setFill(this.color.getColor());
        gc.strokeOval(getX(), getY(), width, height);
        gc.fillOval(getX(), getY(), width, height);
        gc.strokeRect(getX(), getY(), width, height);

    }


}
```

Class MyPoint

```java
package sample;


public class MyPoint {
    private double x,y;

    MyPoint(double x, double y) {
        this.x = x;
        this.y = y;
    }
    //get
    public double getXpoint(){
        return this.x;
    }
    public double getYpoint(){
        return this.y;
    }

    // set
    public void setXpoint(double x){
        this.x = x;
    }
    public void setYpoint(double y){
        this.y = y;
    }

}
```

Class MyRectangle

```
package sample;
import javafx.scene.canvas.GraphicsContext;


public class MyRectangle extends MyShape {
   public double height, width, maxX, maxY, minX,
minY;
   private MyColor color;

   public MyRectangle(double x, double y, double w,
double h, MyColor color) {
       super(x, y, color);
       this.width = w;
       this.height = h;
       this.color = color;
       this.maxX = this.getX() + width;
       this.maxY = this.getY() + height;
       this.minX = this.getX();
       this.minY = this.getY() ;
   }

   public double getWidth() {
       return this.width;
   }
   public double getHeight() {
       return this.height;
   }
   public double getPerimeter(){
       return (2*width + 2*height);
   }
   public double getArea(){
       return (getWidth() * getHeight());
   }
   public double getMyArea(){
```

```
        return getArea();
    }

    @Override
    public boolean pointInMyShape(MyPoint p) {
        return p.getXpoint() >= getX() &&
p.getXpoint() <= getX() + width && p.getYpoint() >=
getY() && p.getYpoint() <= getY() + height;
    }

    public MyRectangle getMyBoundingRectangle() {
        return new MyRectangle(this.getX(),
this.getY(), this.width, this.height, this.color);
    }

    @Override
    public String toString() {
        return "Height:" + getHeight() + "\nWidth:"
+ getWidth() + "\nPerimeter:" + getPerimeter() +
"\nArea:" + getArea();
    }

    @Override
    public void draw(GraphicsContext gc) {
        gc.setFill(this.color.getColor());
        gc.strokeRect(getX(), getY(), getWidth(),
getHeight());
        gc.fillRect(getX(), getY(), getWidth(),
getHeight());
        gc.strokeRect(getX(), getY(), getWidth(),
getHeight());

    }
}
```

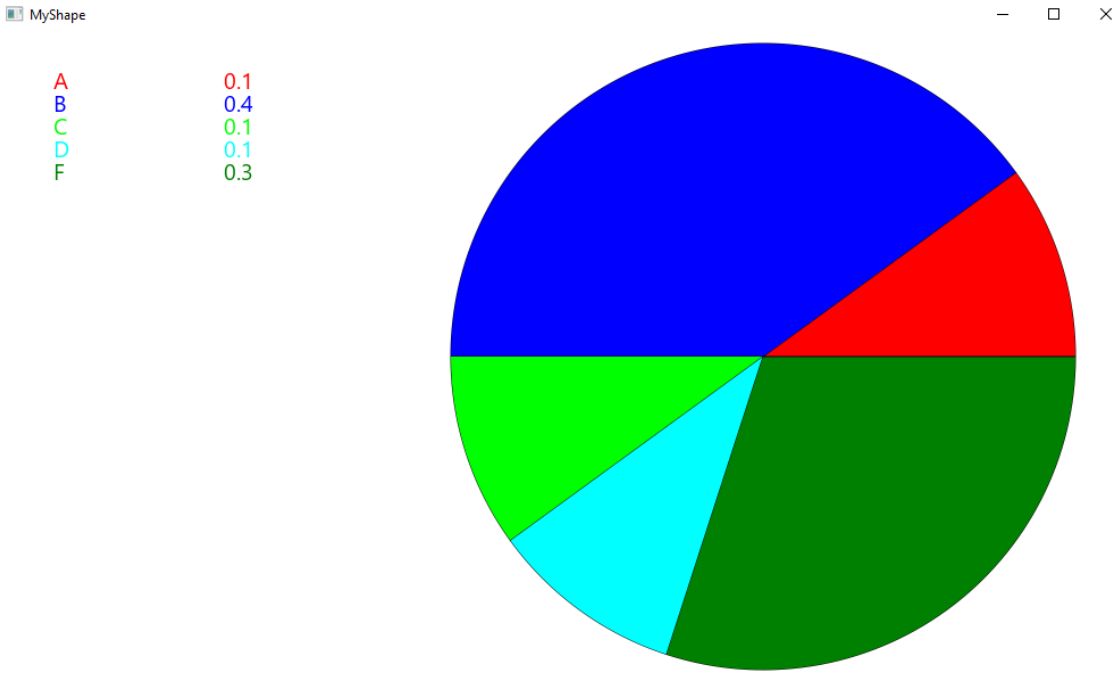
Class MyShape

```
package sample;
import javafx.scene.canvas.GraphicsContext;

abstract class MyShape implements MyShapeInterface{
   private MyPoint point;
   private MyColor color;

   public MyShape(double x, double y, MyColor
color){
        this.point = new MyPoint(x,y);
        this.color = color;
   }

   //get
   public double getX(){
        return this.point.getXpoint();
   }
   public double getY(){
        return this.point.getYpoint();
   }

   // set
   public void setX(double x){
        this.point.setXpoint(x);
   }
   public void setY(double y){
        this.point.setYpoint(y);
   }

   public String toString(){
        return "X:" + getX() + "\nY:" + getY();
   }
   abstract void draw (GraphicsContext gc);
}
```

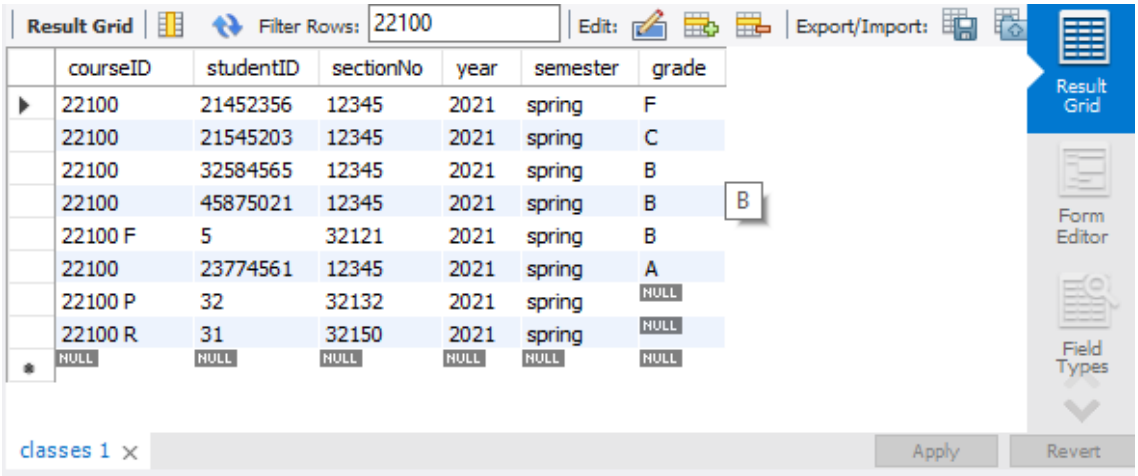
Class MyShapeInterface

```
package sample;

public interface MyShapeInterface {
   MyRectangle getMyBoundingRectangle();
   public double getMyArea();
   abstract boolean pointInMyShape(MyPoint p);

   static public String intersectMyShapes(MyShape
one, MyShape two){
       MyRectangle r1 =
one.getMyBoundingRectangle();
       MyRectangle r2 =
two.getMyBoundingRectangle();

       //x1,y1 = (r1.minX, r1.maxY)
       //x2,y2 = (r1.maxX, r1.minY)
       //x3,y3 = (r2.minX, r2.maxY)
       //x4,y4 = (r2.maxX, r2.minY)
       return "Shapes one and two do not overlap "
+ ( r2.minX > r1.maxX || r2.maxY < r1.minY ||
r1.minX > r2.maxX || r1.maxY < r2.minY );
   }
}
```

4. Outputs Produced


MyShape
A 0.1
B 0.4
C 0.1
D 0.1
F 0.3

```
Table created
Enter total students: 0
DFFCFABBBB
{A=1.0, B=4.0, C=1.0, D=1.0, F=3.0}
10.0
{A=0.1, B=0.4, C=0.1, D=0.1, F=0.3}
DFFCFABBBB
{A=1.0, B=4.0, C=1.0, D=1.0, F=3.0}
10.0
{A=0.1, B=0.4, C=0.1, D=0.1, F=0.3}
DFFCFABBBB
{A=1.0, B=4.0, C=1.0, D=1.0, F=3.0}
10.0
{A=0.1, B=0.4, C=0.1, D=0.1, F=0.3}
6
```



courseID	studentID	sectionNo	year	semester	grade
22100	21452356	12345	2021	spring	F
10235 CC	23774681	15423	2021	spring	C
22100	21545203	12345	2021	spring	C
22100	32584565	12345	2021	spring	B
22100	45875021	12345	2021	spring	B
22100 F	5	32121	2021	spring	B
22100	23774561	12345	2021	spring	A
10000 PP	22	34143	2021	spring	NULL
10200 CC1	1	32118	2021	spring	NULL
10200 CC2	0	32119	2021	spring	NULL
10200 CC3	3	32139	2021	spring	NULL
10200 MM1	49	32140	2021	spring	NULL
10200 MM2	46	32141	2021	spring	NULL
10200 MM3	52	32155	2021	spring	NULL
10300 CC1	70	32120	2021	spring	NULL
10300 CC2	71	32121	2021	spring	NULL
10300 MM1	23	32122	2021	spring	NULL
10300 MM2	24	32123	2021	spring	NULL