Final Project

Leji Li

1. The Problem

This project is to use java application to connect to the database and get the GPAs information of the CSc 22100 in the spring 2020 semester, and then display the information using the PieChart in Exercise 3.

In this project, the java application should be able to interact with the database, including:

- Creates the table **Student, Course** and **Classes**, if those tables are not exists.
- Inserts data to those three tables.
- Gets the information from the database, more specifically, get the number of studnets enrolled in CSC 22100 in the Spring 202 semester for each letter grade.

After getting the number of students for each letter grade, the java application should be able to display those grades in a piechart. The piechart should:

- Has different color for each segment.
- Has the corresponding GPA and number of students as the legend.
- Shows the GPAs in alphabetical order.

2. Solution Methods

Create a database:

To let the java application connect to the database, a database should be created in advance. In this project, MariaDb was used. Login to the database management system, use

```
CREATE DATABASE exercise_student;
```

to create the database **exercise_student.**

DBConnector:

This class is used to maintain the connection between the java application and the database, as well as executing SQL statements to insert data to the database and get data from the database.

initialize: this method is used to create the three tables if they are not exists.

```
String studentTable = "CREATE TABLE IF NOT EXISTS Students(" +
 "studentID INT UNSIGNED NOT NULL," +
 "firstName VARCHAR(255) NOT NULL," +
 "lastName VARCHAR(255) NOT NULL," +
 "email VARCHAR(255)," +
 "sex ENUM('F', 'M')," +
 "PRIMARY KEY(studentID)" +
 ")";
String coursesTable = "CREATE TABLE IF NOT EXISTS Courses(" +
  "courseID VARCHAR(10) NOT NULL," +
  "courseTitle VARCHAR(255) NOT NULL," +
  "department VARCHAR(255) NOT NULL," +
  "PRIMARY KEY(courseID)" +
  ")";
 String classesTable = "CREATE TABLE IF NOT EXISTS Classes(" +
  "courseID VARCHAR(10) NOT NULL," +
  "studentID INT UNSIGNED NOT NULL," +
  "section INT UNSIGNED NOT NULL," +
  "year INT UNSIGNED NOT NULL," +
  "semester ENUM('Spring', 'Summer', 'Fall', 'Winter')," +
  "GPA ENUM('A', 'B', 'C', 'D', 'F', 'W')," +
  "PRIMARY KEY (courseID, studentID, section)," +
  "FOREIGN KEY (courseID) REFERENCES Courses (courseID)," +
  "FOREIGN KEY (studentID) REFERENCES Students (studentID)" +
  ")";
```

The "CREATE TABLE IF NOT EXISTS" will not cause crash when those tables are already exists.

insert2DB: this function is used to execute the pass-in SQL statement and handle the exceptions.

insertStudent, insertCourse, insertClazz: these three methods will prepare the SQL statements and pass to the *insert2DB* so that the corresponding data can be inserted to the correct table in the database.

getClassCount: this method is used to check whether the sample data was put into the database or not. Beacuse the initial data will have 18 records in the Class table, if there's no record in the Class table, that can consider that the data initialzation step has not been proccessed.

getClassesByCourse: this method accepts the courseID, the year and semester and return the GPAs information.

```
String s = "SELECT GPA AS Grade, COUNT(*) AS Val FROM classes " +
 "WHERE courseID = '"+ courseID +"' " +
 "AND year = "+ year + " " +
 "AND semester = '" + semester.getVal() +"' " +
 "GROUP BY GPA";
```

The "COUNT(*)" combines with the "GROUP BY GPA" gives the number of students for each letter grade of the given class.

close: close the connection between the java application and the database.

RandomDataGenerator:

This class is used to insert sample data into the database. This class provides a convenient way to generate data for the piechart demonstration.

initializeData: this method creates the fixed initial data and insert them to the database.

randomStudent: this method will create a student object with random name, studentID and sex.

randomGPA: this method will create a random letter grade.

classSampleData: this method will insert records using the given class information with random-generated student using *randomStudent* and random GPA using *randomGPA.*

Sex, GPA, Semester:

Enum classes.

MyPieChart:

This class takes the map and draws the piechart of the map in GraphicsContext.

3. Code Developed

Main.java

```java
package com.demo;

import javafx.application.Application;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
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.layout.BorderPane;
import javafx.scene.layout.Pane;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

import java.util.Iterator;
import java.util.Map;

public class Main extends Application {
 private DBConnector connector = null;
 private Map<Character, Integer> chartData = null;
 private double pieWidth = 500;
 private double pieHeight = 400;
 private Pane center = null;
 private VBox right = null;
 private Label sideMessage = null;

 public static void main(String[] args) {
 // write your code here
 launch(args);
 }

 @Override
 public void start(Stage primaryStage) throws Exception {
 connector = new DBConnector("root", "321478965");
 RandomDataGenerator generator = new RandomDataGenerator(connector);

 if(connector.getClassCount() == 0){
 generator.initializeData();
 }
 chartData = connector.getClassesByCourse("CSC22100", 2020,
Semester.Spring);

 primaryStage.setTitle("GPA Chart");
 BorderPane borderPane = new BorderPane();
 center = new Pane();

 center.widthProperty().addListener(new ChangeListener<Number>() {
 @Override
 public void changed(ObservableValue<? extends Number> observable, Number o
ldValue, Number newValue) {
 pieWidth = newValue.doubleValue();
 updateCenter(pieWidth, pieHeight);
```

```java
}
});

center.heightProperty().addListener(new ChangeListener<Number>() {
@Override
public void changed(ObservableValue<? extends Number> observable, Number oldValue, Number newValue) {
pieHeight = newValue.doubleValue();
updateCenter(pieWidth, pieHeight);
}
});

Label topHint = new Label("Click the button \n to add more random grade");
Button click = new Button("Click");
// add more students into this class
click.setOnAction(new EventHandler<ActionEvent>() {
@Override
public void handle(ActionEvent event) {
generator.classSampleData("CSC22100", 42264, 2020, Semester.Spring, 20);
chartData = connector.getClassesByCourse("CSC22100", 2020, Semester.Spring);
updateCenter(pieWidth, pieHeight);
updateSideMessage();
}
});

right = new VBox();
sideMessage = new Label();
right.getChildren().addAll(topHint, click, sideMessage);
right.setAlignment(Pos.CENTER_LEFT);
right.setSpacing(10);
right.setPadding(new Insets(10));

updateSideMessage();
updateCenter(pieWidth, pieHeight);
borderPane.setCenter(center);
borderPane.setRight(right);
```

```java

primaryStage.setScene(new Scene(borderPane));
primaryStage.show();
}

/**
* when close the application, shutdown the database connection
* @throws Exception
*/
 @Override
 public void stop() throws Exception {
 super.stop();
 if(null != connector){
 connector.close();
 }
 }

 /**
 * update the central view of the borderpane
 * @param width new width of the center pie chart
 * @param height new height of the center pie chart
 */
 private void updateCenter(double width, double height){
 Canvas canvas = new Canvas(width, height);
 double r = 0.6 * Math.min(width, height) / 2;
 MyPieChart chart = new MyPieChart(width/2, height/2, r);
 chart.setData(chartData);
 GraphicsContext gc = canvas.getGraphicsContext2D();
 chart.draw(gc);
 if(null == center){
 center = new Pane();
 }
 center.getChildren().clear();
 center.getChildren().add(canvas);
 }

 /**
```

```java
* display the number of students enrolled in this class for each letter grade.
*/
private void updateSideMessage(){
int[] counts = new int[6];
Iterator iterator = chartData.entrySet().iterator();
int sum = 0;
while (iterator.hasNext()){
Map.Entry<Character,Integer> entry = (Map.Entry<Character, Integer>)iterator.next();
if(entry.getKey() == 'F'){
counts[4] = entry.getValue();
} else if(entry.getKey() == 'W'){
counts[5] = entry.getValue();
} else{
counts[entry.getKey() - 'A'] = entry.getValue();
}
sum += entry.getValue();
}
if(null == sideMessage){
sideMessage = new Label();
}
String s = "Total Students: " + sum + "\n";
for(int i = 0; i < 4; i++){
s += (char)(i + 'A') + ": " + counts[i] + "\n";
}
s += "F: " + counts[4] + "\n";
s += "W: " + counts[5];
sideMessage.setText(s);
}
}
```

DBConnector.java

```java
package com.demo;

import javafx.util.Pair;
import org.omg.CORBA.INTERNAL;

import java.sql.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;

public class DBConnector {

static final String JDBC_DRIVER = "org.mariadb.jdbc.Driver";
static final String DB_URL = "jdbc:mariadb://localhost:3306/exercise_stude
nt";
private Connection conn = null;

public DBConnector(String userName, String psw){
try{
Class.forName(JDBC_DRIVER);
conn =DriverManager.getConnection(DB_URL, userName, psw);
System.out.println("Connected to DB");
} catch (Exception e){
e.toString();
}
boolean initialized = initialize();
if(initialized){
System.out.println("DB initialized!");
} else{
System.out.println("DB cannot be initialized");
}
}

/**
* create the tables if not exist
* @return
*/
private boolean initialize() {
if(null == conn){
System.out.println("Connection Error! Cannot initialize Database!");
return false;
}
String studentTable = "CREATE TABLE IF NOT EXISTS Students(" +
"studentID INT UNSIGNED NOT NULL," +
"firstName VARCHAR(255) NOT NULL," +
"lastName VARCHAR(255) NOT NULL," +
"email VARCHAR(255)," +
"sex ENUM('F', 'M')," +
"PRIMARY KEY(studentID)" +
")";
String coursesTable = "CREATE TABLE IF NOT EXISTS Courses(" +
"courseID VARCHAR(10) NOT NULL," +
"courseTitle VARCHAR(255) NOT NULL," +
"department VARCHAR(255) NOT NULL," +
"PRIMARY KEY(courseID)" +
")";
String classesTable = "CREATE TABLE IF NOT EXISTS Classes(" +
"courseID VARCHAR(10) NOT NULL," +
"studentID INT UNSIGNED NOT NULL," +
"section INT UNSIGNED NOT NULL," +
"year INT UNSIGNED NOT NULL," +
"semester ENUM('Spring', 'Summer', 'Fall', 'Winter')," +
"GPA ENUM('A', 'B', 'C', 'D', 'F', 'W')," +
"PRIMARY KEY (courseID, studentID, section)," +
"FOREIGN KEY (courseID) REFERENCES Courses (courseID)," +
"FOREIGN KEY (studentID) REFERENCES Students (studentID)" +
")";
try {
Statement statement = conn.createStatement();
statement.execute(studentTable);
statement.execute(coursesTable);
statement.execute(classesTable);
} catch (SQLException throwables) {
System.out.println("Initialize failed!");
System.out.println(throwables.toString());
return false;
}
return true;
}

```

```java
/**
* insert a student to database
* @param student
* @return
*/
public boolean insertStudent(Student student){
String s = "INSERT Students" +
"(studentID, firstName, lastName, email, sex)" +
" VALUES (" +
"'" + student.getID() + "'," +
"'" + student.getFirstName() + "'," +
"'" + student.getLastName() + "'," +
"'" + student.getEmail() + "'," +
"'" + student.getSex().getVal()+ "'" +
")";
return insert2DB(s,
"Cannot add a student!",
"Add student failed!");
}

 /**
 * insert a course to database
 * @param course
 * @return
 */
 public boolean insertCourse(Course course){
 String s = "INSERT Courses (courseID, courseTitle, department) VALUES ("
+
 "'" + course.getID() + "'," +
 "'" + course.getTitle() + "'," +
 "'" + course.getDepartment() + "'" +
 ")";
 return insert2DB(s,
 "Cannot add a course!",
 "Add course failed!");
 }

 /**
```

```
* insert a class to database
* @param clazz
* @return
*/
public boolean insertClass(Clazz clazz){
String s = "INSERT Classes (courseID, studentID, section, year, semester, GPA) VALUES (" +
"'" + clazz.getCourseID() + "'," +
"'" + clazz.getStudentID() + "'," +
"'" + clazz.getSection() + "'," +
"'" + clazz.getYear() + "'," +
"'" + clazz.getSemester().getVal() + "'," +
"'" + clazz.getGPA().getVal() + "'" +
")";
return insert2DB(s,
"Cannot add a class!",
"Add class failed!");
}

/**
* execute the given sql
* @param sql
* @param NoConnectionWarning the warning message when the connection is empty
* @param failAddWarning
* @return
*/
private boolean insert2DB(String sql, String NoConnectionWarning, String failAddWarning){
if(null == conn){
System.out.println("Connection Error! " + NoConnectionWarning);
return false;
}
try {
Statement statement = conn.createStatement();
statement.execute(sql);
} catch (SQLException throwables) {
System.out.println(failAddWarning);
```

```java
System.out.println(throwables.toString());
return false;
}
return true;
}

/**
* get the count of records in the Classes table
* @return
*/
public int getClassCount(){
if(null == conn){
System.out.println("Connection Empty! Cannot get count");
return -1;
}
int count = -1;
String s = "SELECT COUNT(*) AS Val FROM Classes";
try {
Statement statement = conn.createStatement();
ResultSet resultSet = statement.executeQuery(s);
while (resultSet.next()){
count = resultSet.getInt("Val");
}
} catch (SQLException throwables) {
System.out.println(throwables.toString());
return -1;
}
return count;
}

/**
* get the GPA count of a given class
* @param courseID the course ID of the class
* @param year the year of the class
* @param semester the semester of the class
* @return an map that contains the GPAs count
*/
```

```
public Map<Character, Integer> getClassesByCourse(String courseID, int year, Semester semester){
if(null == conn){
System.out.println("Connection Error!");
return null;
}
String s = "SELECT GPA AS Grade, COUNT(*) AS Val FROM classes " +
"WHERE courseID = '"+ courseID +"' " +
"AND year = "+ year + " " +
"AND semester = '" + semester.getVal() +"' " +
"GROUP BY GPA";
Map<Character, Integer> map = new HashMap<Character, Integer>(6);
try {
Statement statement = conn.createStatement();
ResultSet resultSet = statement.executeQuery(s);
while (resultSet.next()){
char grade = resultSet.getString("Grade").charAt(0);
int val = resultSet.getInt("Val");
map.put(grade, val);
System.out.println(grade + " -> " + val);
}
} catch (SQLException throwables) {
System.out.println("Query Error!");
System.out.println(throwables.toString());
return null;
}
return map;
}

/**
* close the database connection
*/
public void close(){
if(null != conn){
try {
conn.close();
} catch (SQLException throwables) {
throwables.toString();
```

```
    }
    }
    }
  }
```

RandomDataGenerator.java

```
package com.demo;

import java.sql.Connection;
import java.util.ArrayList;
import java.util.Random;

public class RandomDataGenerator {
 private DBConnector connector = null;

  public RandomDataGenerator(DBConnector connector){
  this.connector = connector;
  }

  /**
  * This function will insert random-generated students to the database,
  * as well as adding that student to the given course
  * @param courseID the course of the class that need add more students
  * @param section the section of the class
  * @param year the year of the class
  * @param semester the semester of the class
  * @param repeat the amount of randomly generated student
  */
  public void classSampleData(String courseID, int section, int year, Semest
er semester, int repeat){
  int count = 0;
  while(count < repeat){
  Student ranS = randomStudent();
  if(connector.insertStudent(ranS)){
  connector.insertClass(new Clazz(courseID, ranS.getID(), section, year, sem
ester, getGPA()));
  count++;
  }
  }
```

```java
}

/**
* generate a student randomly.
* The student ID, name and sex are being picked up randomly
* @return randomly generated student
*/
private Student randomStudent(){
Random random = new Random();
String first = "";
String last = "";
for(int i = 0; i < 5; i++){
first += (char)(random.nextInt(26) + 'a');
}
for(int i = 0; i < 3; i++){
last += (char)(random.nextInt(26) + 'a');
}
int id = random.nextInt(89999999) + 10000000;
Sex[] sexes = Sex.values();
return new Student(id, first, last,sexes[random.nextInt(1)]);
}

/**
* putting the initial data to the database
*/
public void initializeData(){
ArrayList<Student> students = new ArrayList<Student>();
students.add(new Student(12345678, "Leji", "Li", "leji@email.com",
Sex.M));
students.add(new Student(15978634, "Kara", "Chen", "kara@email.com",
Sex.F));
students.add(new Student(32641287, "Jiayi", "Li", "jiayi@email.com",
Sex.F));
students.add(new Student(98732164, "Ceci", "Ao", "cci@email.com", Sex.F));
students.add(new Student(80204672, "Erik", "Hu", "erik@email.com",
Sex.M));
students.add(new Student(74123690, "Yubo", "Liang", "erik@email.com",
Sex.M));

```

```java

ArrayList<Course> courses = new ArrayList<Course>(3);
courses.add(new Course("CSC22100", "Software Design Laboratory", "Computer Science"));
courses.add(new Course("CSC11300", "Programming Language", "Computer Science"));
courses.add(new Course("CSC22000", "Algorithms", "Computer Science"));

int[] sections = {42264, 42255, 25696};
int[] years = {2020, 2020, 2019};
Semester[] semesters = {Semester.Spring, Semester.Spring, Semester.Fall};
ArrayList<Clazz> clazzes = new ArrayList<Clazz>();
for (int i = 0; i < students.size(); i++) {
for (int j = 0; j < courses.size(); j++) {
clazzes.add(new Clazz(courses.get(j).getID(), students.get(i).getID(), sections[j], years[j], semesters[j], getGPA()));
}
}

for(Student s: students){
connector.insertStudent(s);
}
for(Course c: courses){
connector.insertCourse(c);
}
for(Clazz c: clazzes){
connector.insertClass(c);
}
}

/**
* randomly generate a GPA in letter form
* @return a random GPA in letter
*/
private GPA getGPA(){
GPA[] GPAs = GPA.values();
Random random = new Random();
 return GPAs[random.nextInt(GPAs.length)];
 }
}
```

Student.java

```java
package com.demo;

public class Student {
 private int ID;
 private String firstName;
 private String lastName;
 private String email;
 private Sex sex;

  public Student(int ID, String firstName, String lastName, String email, Sex sex) {
  this.ID = ID;
  this.firstName = firstName;
  this.lastName = lastName;
  this.email = email;
  this.sex = sex;
  }

  public Student(int ID, String firstName, String lastName, Sex sex) {
  this(ID, firstName, lastName, "", sex);
  }

  public int getID() {
  return ID;
  }

  public void setID(int ID) {
  this.ID = ID;
  }

  public String getFirstName() {
  return firstName;
  }

  public void setFirstName(String firstName) {
```

```
this.firstName = firstName;
}

public String getLastName() {
return lastName;
}

public void setLastName(String lastName) {
this.lastName = lastName;
}

public String getEmail() {
return email;
}

public void setEmail(String email) {
this.email = email;
}

public Sex getSex() {
return sex;
}

public void setSex(Sex sex) {
this.sex = sex;
}
}
```

Course.java

```
package com.demo;

public class Course {
 private String ID;
 private String title;
 private String department;

 public Course(String ID, String title, String department) {
 this.ID = ID;
 this.title = title;
```

```java
this.department = department;
}

public String getID() {
return ID;
}

public void setID(String ID) {
this.ID = ID;
}

public String getTitle() {
return title;
}

public void setTitle(String title) {
this.title = title;
}

public String getDepartment() {
return department;
}

public void setDepartment(String department) {
this.department = department;
}
}
```

Clazz.java

```java
package com.demo;

public class Clazz {
 private String courseID;
 private int studentID;
 private int section;
 private int year;
 private Semester semester;
 private GPA GPA;

```

```
public Clazz(String courseID, int studentID, int section, int year, Semester semester, GPA GPA) {
this.courseID = courseID;
this.studentID = studentID;
this.section = section;
this.year = year;
this.semester = semester;
this.GPA = GPA;
}

public String getCourseID() {
return courseID;
}

public void setCourseID(String courseID) {
this.courseID = courseID;
}

public int getStudentID() {
return studentID;
}

public void setStudentID(int studentID) {
this.studentID = studentID;
}

public int getSection() {
return section;
}

public void setSection(int section) {
this.section = section;
}

public int getYear() {
return year;
}

```

```java
public void setYear(int year) {
this.year = year;
}

public Semester getSemester() {
return semester;
}

public void setSemester(Semester semester) {
this.semester = semester;
}

public GPA getGPA() {
return GPA;
}

public void setGPA(GPA GPA) {
this.GPA = GPA;
}
}
```

Semester.java

```java
package com.demo;

public enum Semester{
 Spring("Spring"),
 Summer("Summer"),
 Fall("Fall"),
 Winter("Winter");

 private String val;
 private Semester(String s){
 val = s;
 }
 public String getVal(){
 return val;
 }
}
```

Sex.java

```java
package com.demo;

public enum Sex{
 F('F'),
 M('M');

 private char val;
 private Sex(){
 this('F');
  }
  private Sex(char c){
  val = c;
  }

  public char getVal() {
  return val;
  }
 }
```

GPA.java

```java
package com.demo;

public enum GPA {
 A('A'),
 B('B'),
 C('C'),
 D('D'),
 F('F'),
 W('W');

  private char val;
  private GPA(char gpa){
  val = gpa;
  }
  public char getVal(){
  return val;
  }
}
```

MyPieChart.java

```java
package com.demo;

import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;
import javafx.scene.shape.ArcType;
import javafx.scene.text.Font;
import javafx.scene.text.Text;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Map;

public class MyPieChart extends MyShape {
 private Map<Character, Integer> chartData;
 private double r;
 private static List<Color> colorList = new ArrayList<Color>();

 public MyPieChart(double x, double y, double r, int n, Map<Character, Integer> chartData){
 super(x, y);
 this.r = r;
 setData(chartData);
 for (int i = 0; i < 6; i++) {
 colorList.add(MyColor.randomColor());
 }
 }

 public MyPieChart(double x, double y, double r){
 this(x, y, r, 3, null);
 }

 public void setData(Map<Character, Integer> data){
 this.chartData = data;
 }

```

```java
/**
* this function is to find how wide a string should occupies in a certain font
* @param font some kind of font
* @param text the target string
* @return the width and height of the text will takes under the given font
* [0] is the width
* [1] is the height
*/
private double[] getTextWidth(Font font, String text){
Text helper = new Text(text);
helper.setFont(font);
helper.setWrappingWidth(0);
helper.setLineSpacing(0);
// prefWidth pass-in -1 because node has null content-bias
double w = helper.prefWidth(-1);
helper.setWrappingWidth((int)Math.ceil(w));
return new double[]{
Math.ceil(helper.getLayoutBounds().getWidth()),
Math.ceil(helper.getLayoutBounds().getHeight())
};
}

@Override
public void draw(GraphicsContext gc) {
if(chartData == null || chartData.size() == 0){
String hint = "No data to display";
Font font = new Font(16);
double[] size = getTextWidth(font, hint);
gc.setFont(font);
gc.setFill(MyColor.Black.toFXPaintColor());
gc.fillText(hint, getX() - size[0]/2, getY() - size[1]/2);
return;
}
Iterator iterator;
int count = 0;
int sum = 0;
iterator = chartData.entrySet().iterator();
```

```java
while (iterator.hasNext()){
Map.Entry<Character, Integer> entry = (Map.Entry<Character, Integer>)iterator.next();
sum += entry.getValue();
}
// the sector is a part of a circle
// fillArc works like the fillOval
// it takes the top left corner of the bounding box
// arc width and arc height are the radius of the circle
// startAngle is the angle between the x axis and the right side of the sector, counterclockwise
// arcExtent is the angle of the sector
double startingX = this.getX() - this.r; // x of the top left corner
double startingY = this.getY() - this.r; // y of the top left corner
double sectorAngle = 0d; // arcExtent
double angleShift = 90d; // startAngle
double midAngle = 0d; // the angle of middle of a sector, use to locate the label
double labelX = 0d, labelY = 0d; // the starting point of a label
Font font = new Font(12); // use to unify the font of the label and find the length
iterator = chartData.entrySet().iterator(); // draw the sectors
while(iterator.hasNext()){
Map.Entry<Character, Integer> entry = (Map.Entry<Character, Integer>)iterator.next();
// all angles passed in to the fillArc method should be in degree
sectorAngle = 360d * (double)entry.getValue() / (double)sum;
gc.setFill(colorList.get(count));
gc.fillArc(startingX, startingY, this.r * 2, this.r * 2, angleShift, sectorAngle, ArcType.ROUND);

// prepare for the label of a sector
String labelText = entry.getKey() + (": " + entry.getValue());

 // find the location of the label of a sector
 // it should be placed to the middle of its sector
 // r*1.1 makes the label has the padding to the piechart
 midAngle = (angleShift + sectorAngle / 2);
 double xShift = this.r * 1.1 * Math.cos(Math.toRadians(midAngle));
labelX = this.getX() + xShift;
labelY = this.getY() - this.r *1.1 * Math.sin(Math.toRadians(midAngle));

// handle the spacing
// since the width of the label is not constant
// we need to find out the label width
// and to avoid the label overlapping with the piechart or going out of the window
// we should take the minimum of them, and set this value as the maximum of the label
double labelWidth = getTextWidth(font, labelText)[0];
if(xShift < 0){
// if the label is in the left side of the piechart
// the starting point should be the middle point of the sector arc minus the label width
// r * 0.006 is the margin of the window
labelWidth = Math.min(labelWidth, labelX - this.r * 0.06);
labelX -= labelWidth;
} else {
labelWidth = Math.min(labelWidth, (2*getX() - labelX) - this.r * 0.06);
}

gc.setFill(MyColor.Black.toFXPaintColor());
gc.setFont(font);
gc.fillText(labelText, labelX, labelY, labelWidth);

// accumulate the starting angle for the next sector
angleShift += sectorAngle;
count++;
}
}
}
```

MyShape.java

```java
package com.demo;

import javafx.scene.canvas.GraphicsContext;

public abstract class MyShape {
```

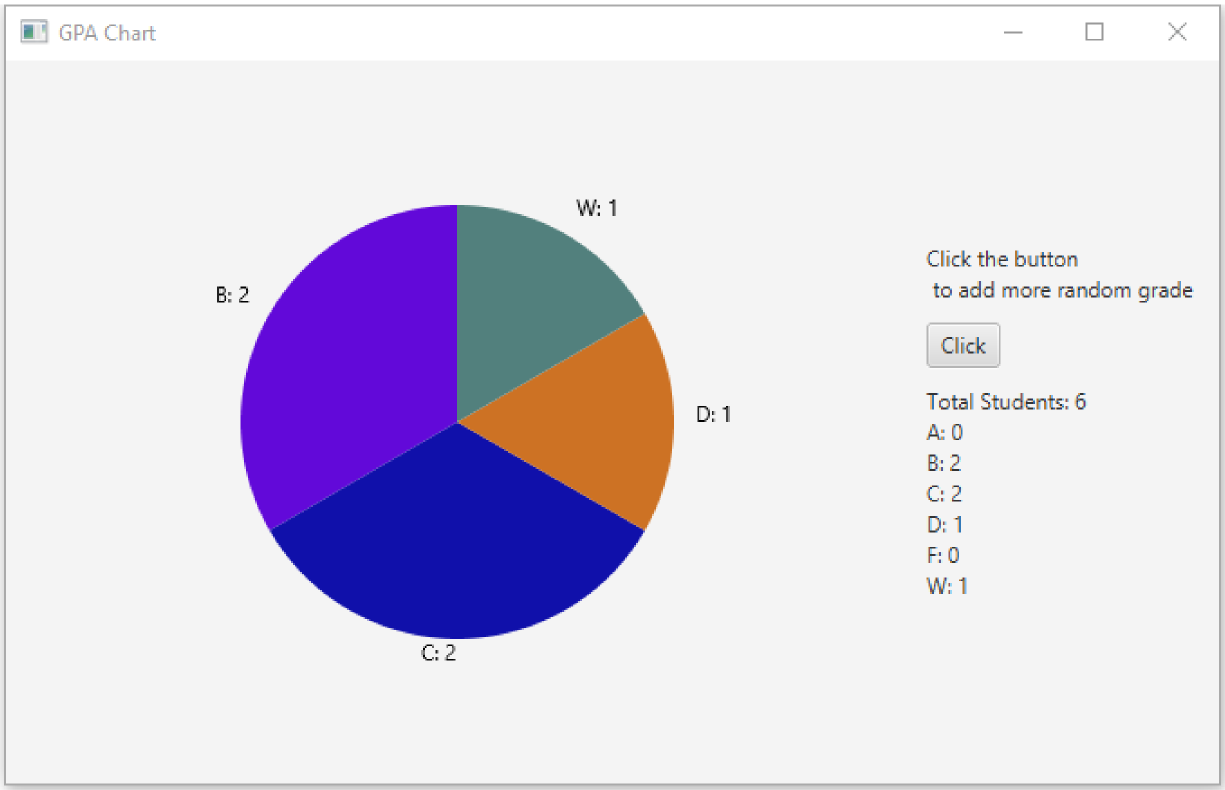
```java
private double x;
private double y;
private MyColor color;

 public MyShape(double x, double y, MyColor color){
 this.x = x;
 this.y = y;
 this.color = color;
 }
 public MyShape(double x, double y){
 this(x, y, MyColor.Black);
 }
 public MyShape(MyColor color){
 this(0, 0, color);
 }

 public MyShape(){
 this(0, 0);
 }

 public double getX() {
 return x;
 }

 public void setX(double x) {
 this.x = x;
 }

 public double getY() {
 return y;
 }

 public void setY(double y) {
 this.y = y;
 }

 public MyColor getColor() {
 return color;
```

```java
  }

  public void setColor(MyColor color) {
  this.color = color;
  }

  public abstract void draw(GraphicsContext gc);

  @Override
  public String toString(){
  return "Class MyShapeis the hierarchy’s superclass and inherits the Java c
lass Object. An\n" +
  "implementation of the class defines a point (x, y) and the color of the s
hape. ";
  }
}
```

MyColor.java

```java
package com.demo;

import javafx.scene.paint.Color;

public enum MyColor {
 FireBrick(178,34,34),
 LightPink(255,182,193),
 OliveDrab(85,107,47),
 MediumAquamarine(0,250,154),
 Turquoise(64,224,208),
 RoyalBlue(65,105,225),
 White(255,255,255),
 Black(0,0,0),
 Gray(128,128,128),
 LightGray(211,211,211),
 Yellow( 255,255,0);

 private int r, g, b;

 private MyColor(){
 this(0, 0,0);
 }

 private MyColor(int r, int g, int b){
 setColor(r, g, b);
 }

 public Color toFXPaintColor(){
 return Color.rgb(r, g, b);
 }

 public void setColor(int r, int g, int b){
 this.r = r;
 this.g = g;
 this.b = b;
 }
 public MyColor getColor(){
 return this;
 }

 public static Color randomColor(){
 int r = (int)(Math.random() * 256);
 int g = (int)(Math.random() * 256);
 int b = (int)(Math.random() * 256);

 return Color.rgb(r, g, b);
 }
 }
```

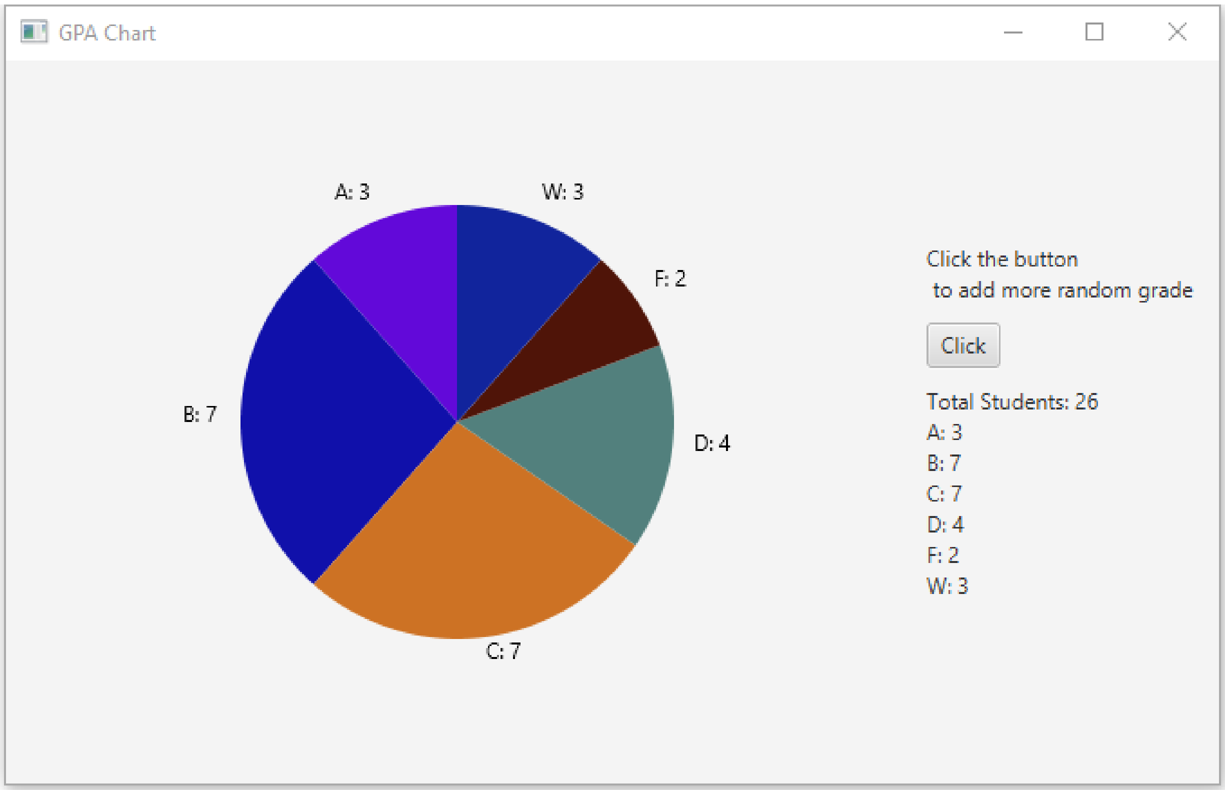
4. Outputs



Click the button
to add more random grade

Total Students: 6
A: 0
B: 2
C: 2
D: 1
F: 0
W: 1

Initially, the Classes table only has 6 records for the CSC 22100 in 2020 Spring semester, and no student has the A grade and F grade. The grade A and F will be shown in the list, but not in the piechart.



Click the button
to add more random grade

Total Students: 26
A: 3
B: 7
C: 7
D: 4
F: 2
W: 3

After clicking the button, 20 random class record about the CSC 22100 in 2020 Spring semester were insert into the database. The piechart and the list updated.

5. DDL and SQL

DDL for creating the database

```
CREATE DATABASE exercise_student
```

DDL for creating tables

```
CREATE TABLE IF NOT EXISTS Classes (
  courseID VARCHAR(10) NOT NULL,
  studentID INT(10) UNSIGNED NOT NULL,
  section INT(10) UNSIGNED NOT NULL,
  year INT(10) UNSIGNED NOT NULL,
  semester ENUM('Spring','Summer','Fall','Winter'),
  GPA ENUM('A','B','C','D','F','W'),
  PRIMARY KEY (courseID,studentID,section),
  FOREIGN KEY (courseID) REFERENCES Courses (courseID),
 FOREIGN KEY (studentID) REFERENCES Students (studentID)
)

CREATE TABLE IF NOT EXISTS Courses (
  courseID VARCHAR(10) NOT NULL,
  courseTitle VARCHAR(255) NOT NULL,
  department VARCHAR(255) NOT NULL,
  PRIMARY KEY (courseID)
)

CREATE TABLE IF NOT EXISTS Students (
  studentID INT(10) UNSIGNED NOT NULL,
  firstName VARCHAR(255) NOT NULL,
  lastName VARCHAR(255) NOT NULL,
  email VARCHAR(255),
  sex ENUM('F','M'),
  PRIMARY KEY (studentID)
)
```

SQL for inserting courses

```
INSERT INTO Courses (courseID, courseTitle, department) VALUES
```

```
    ('CSC11300', 'Programming Language', 'Computer Science'),
    ('CSC22000', 'Algorithms', 'Computer Science'),
    ('CSC22100', 'Software Design Laboratory', 'Computer Science')
```

SQL for inserting students

```
INSERT INTO Students (studentID, firstName, lastName, email, sex) VALUES
    (12345678, 'Leji', 'Li', 'leji@email.com', 'M'),
 (15978634, 'Kara', 'Chen', 'kara@email.com', 'F'),
    (32641287, 'Jiayi', 'Li', 'jiayi@email.com', 'F'),
    (98732164, 'Ceci', 'Ao', 'cci@email.com', 'F'),
    (80204672, 'Erik', 'Hu', 'erik@email.com', 'M'),
    (74123690, 'Yubo', 'Liang', 'erik@email.com', 'M'),
    (14108576, 'ykfol', 'csd', '', 'F'),
    (29853027, 'uibei', 'fjn', '', 'F'),
    (34708176, 'pjwdw', 'xtu', '', 'F'),
    (36087094, 'bsciu', 'ivw', '', 'F'),
    (46550981, 'klfyt', 'dva', '', 'F'),
    (46894722, 'tqpwj', 'fav', '', 'F'),
    (47691681, 'thqsi', 'qot', '', 'F'),
    (49748254, 'fdyft', 'dnk', '', 'F'),
    (50846640, 'lelfp', 'nem', '', 'F'),
    (60212116, 'ggsjm', 'xah', '', 'F'),
    (62601011, 'dnmtt', 'qvl', '', 'F'),
    (63933274, 'tinyw', 'vcw', '', 'F'),
    (68993771, 'plyri', 'flq', '', 'F'),
    (69588427, 'yeehi', 'tlt', '', 'F'),
    (77905042, 'aekff', 'qbd', '', 'F'),
    (80594626, 'ebnro', 'vvt', '', 'F'),
    (90782236, 'zuvxp', 'awf', '', 'F'),
    (90856787, 'orqmd', 'mpe', '', 'F'),
    (94539762, 'khfqg', 'obz', '', 'F'),
    (94811579, 'ilapo', 'lfu', '', 'F');
```

SQL for inserting classes

```
INSERT INTO classes (courseID, studentID, section, year, semester, GPA) VALU
ES
    ('CSC11300', 12345678, 42255, 2020, 'Spring', 'D'),
    ('CSC11300', 15978634, 42255, 2020, 'Spring', 'D'),
    ('CSC11300', 32641287, 42255, 2020, 'Spring', 'W'),
```

```sql
    ('CSC11300', 74123690, 42255, 2020, 'Spring', 'W'),
    ('CSC11300', 80204672, 42255, 2020, 'Spring', 'B'),
    ('CSC11300', 98732164, 42255, 2020, 'Spring', 'A'),
    ('CSC22000', 12345678, 25696, 2019, 'Fall', 'C'),
    ('CSC22000', 15978634, 25696, 2019, 'Fall', 'W'),
('CSC22000', 32641287, 25696, 2019, 'Fall', 'B'),
    ('CSC22000', 74123690, 25696, 2019, 'Fall', 'D'),
    ('CSC22000', 80204672, 25696, 2019, 'Fall', 'F'),
    ('CSC22000', 98732164, 25696, 2019, 'Fall', 'B'),
    ('CSC22100', 12345678, 42264, 2020, 'Spring', 'C'),
    ('CSC22100', 14108576, 42264, 2020, 'Spring', 'A'),
    ('CSC22100', 15978634, 42264, 2020, 'Spring', 'B'),
    ('CSC22100', 29853027, 42264, 2020, 'Spring', 'B'),
    ('CSC22100', 32641287, 42264, 2020, 'Spring', 'B'),
    ('CSC22100', 34708176, 42264, 2020, 'Spring', 'C'),
    ('CSC22100', 36087094, 42264, 2020, 'Spring', 'W'),
    ('CSC22100', 46550981, 42264, 2020, 'Spring', 'W'),
    ('CSC22100', 46894722, 42264, 2020, 'Spring', 'C'),
    ('CSC22100', 47691681, 42264, 2020, 'Spring', 'D'),
    ('CSC22100', 49748254, 42264, 2020, 'Spring', 'A'),
    ('CSC22100', 50846640, 42264, 2020, 'Spring', 'B'),
    ('CSC22100', 60212116, 42264, 2020, 'Spring', 'B'),
    ('CSC22100', 62601011, 42264, 2020, 'Spring', 'D'),
    ('CSC22100', 63933274, 42264, 2020, 'Spring', 'C'),
    ('CSC22100', 68993771, 42264, 2020, 'Spring', 'F'),
    ('CSC22100', 69588427, 42264, 2020, 'Spring', 'B'),
    ('CSC22100', 74123690, 42264, 2020, 'Spring', 'W'),
    ('CSC22100', 77905042, 42264, 2020, 'Spring', 'D'),
    ('CSC22100', 80204672, 42264, 2020, 'Spring', 'C'),
    ('CSC22100', 80594626, 42264, 2020, 'Spring', 'C'),
    ('CSC22100', 90782236, 42264, 2020, 'Spring', 'C'),
    ('CSC22100', 90856787, 42264, 2020, 'Spring', 'B'),
    ('CSC22100', 94539762, 42264, 2020, 'Spring', 'A'),
    ('CSC22100', 94811579, 42264, 2020, 'Spring', 'F'),
    ('CSC22100', 98732164, 42264, 2020, 'Spring', 'D');
```

SQL for querying the number of class records

```sql
SELECT COUNT(*) AS Val FROM Classes
```

SQL for querying the amoung of students for each letter grade of CSC 22100 in 2020 Spring semester

```sql
SELECT GPA AS Grade, COUNT(*) AS Val FROM classes
WHERE courseID = 'CSC22100'
AND year = 2020
AND semester = 'Spring'
GROUP BY GPA
```