Fall 2022 | CSC 22100 | Assignment 4

Alhamza Muswara

Problem/Task:

[1] The full code of the Java application developed,

[2] The database ER diagram

[3] The DDL and SQL statements used (separately from the Java code), and

[4] The outputs produced for the tasks indicated

[2] Solution Methods

Class Database:

```
package application;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class Database implements StudentsDatabaseInterface {
	Connection conn = null;

	public String loadfile(String filename, String table) {
		try {
			Statement stmt = conn.createStatement();
			String sql = "SELECT * FROM" + table + "INTO OUTFILE
'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/"
					+ filename + ".txt'";
			stmt.executeUpdate(sql);
			System.out.println("Created schedule table in given
database...");
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return filename;
	}

	Database() {

		try {

			Class.forName("com.mysql.cj.jdbc.Driver");
```

```
                    conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/db", "root",
"nrio&nroK)><i340mi8");
                    Statement stmt = conn.createStatement();

                    String sql = "SET GLOBAL local_infile = 'On'";
                    stmt.executeUpdate(sql);

                    System.out.println("Connected to database db...");
                    System.out.println("local_infile = 'ON'");
             } catch (Exception exception) {
                    System.out.println(exception);
             }
      }

      class Schedule implements TableInterface {
             Schedule(String filename) {
                    try {
                           Statement stmt = conn.createStatement();
                           String sql = "CREATE TABLE Schedule(CourseID
CHAR(40),SectionNo INT,Title CHAR(100),yr INT,Semester CHAR(10),Instructor
CHAR(30),Department CHAR(30),Program CHAR(50))";
                           stmt.executeUpdate(sql);
                           sql = "LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server
8.0/Uploads/" + filename
                                        + ".txt' INTO TABLE schedule";
                           stmt.executeUpdate(sql);
                           System.out.println("Created schedule table in given
database...");
                    } catch (SQLException e) {
                           e.printStackTrace();
                    }

             }

             @Override
             public String write(String filename, String column) {
                    // TODO Auto-generated method stub
                    try {
                           Statement stmt = conn.createStatement();
                           String sql = "SELECT " + column + " FROM Schedule"
                                        + "INTO OUTFILE 'C:/ProgramData/MySQL/MySQL
Server 8.0/Uploads/" + filename + ".txt'";
                           stmt.executeUpdate(sql);
                    } catch (SQLException e) {
                           e.printStackTrace();
                    }
                    return filename;
             }

             public void insertRow(String v1, String v2, String v3, String v4, String
v5, String v6, String v7, String v8) {
                    try {
                           Statement stmt = conn.createStatement();
```

```
                        String sql = "INSERT INTO schedule (
CourseID,SectionNo,Title,yr , Semester,Instructor,Department,Program )"
                                    + "    VALUES ('" + v1 + "'," + v2 + ",'" + v3
+ "'," + v4 + ",'" + v5 + "','" + v6 + "','" + v7
                                    + "','" + v8 + ")";
                        stmt.executeUpdate(sql);
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }

            }

            @Override
            public void delRow(String column, String item) {
                  try {
                        Statement stmt = conn.createStatement();
                        String sql = "DELETE FROM Schedule WHERE" + column + " = "
+ "'" + item + "'";
                        stmt.executeUpdate(sql);
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }

            }

      }

      class Students implements TableInterface {
            Students(String filename) {
                  try {
                        Statement stmt = conn.createStatement();
                        String sql = "CREATE TABLE Students(Name CHAR(40),
StudentID CHAR(40), email CHAR(40));";
                        stmt.executeUpdate(sql);
                        System.out.println("Created student table in given
database...");
                        sql = "LOAD DATA LOCAL INFILE 'C:/ProgramData/MySQL/MySQL
Server 8.0/Uploads/" + filename + ".txt'"
                                    + " INTO TABLE db.Students";
                        stmt.executeUpdate(sql);
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }

            }

            public void insertRow(String v1, String v2, String v3, String v4) {
                  try {
                        Statement stmt = conn.createStatement();
                        String sql = "INSERT INTO students (name ,id , age, email
)" + " VALUES ('" + v1 + "'," + v2 + ","
                                    + v3 + ",'" + v4 + "')";
                        stmt.executeUpdate(sql);
                  } catch (SQLException e) {
                        e.printStackTrace();
```

```
			}

		}

		@Override
		public String write(String filename, String column) {
			try {
				Statement stmt = conn.createStatement();
				String sql = "SELECT " + column
						+ " FROM students INTO OUTFILE
'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/" + filename
						+ ".txt'";
				stmt.executeUpdate(sql);
			} catch (SQLException e) {
				e.printStackTrace();
			}
			return filename;

		}

		@Override
		public void delRow(String column, String item) {
			try {
				Statement stmt = conn.createStatement();
				String sql = "DELETE FROM students WHERE" + column + " = "
+ "'" + item + "'";
				stmt.executeUpdate(sql);
			} catch (SQLException e) {
				e.printStackTrace();
			}

		}

	}

	class Courses implements TableInterface {
		Courses(String filename) {
			try {
				Statement stmt = conn.createStatement();
				String sql = "CREATE TABLE courses(CourseID CHAR(40)," +
"SectionNo INT," + "Title CHAR(100));";
				stmt.executeUpdate(sql);
				System.out.println("Created courses table in given
database...");
				sql = "LOAD DATA LOCAL INFILE 'C:/ProgramData/MySQL/MySQL
Server 8.0/Uploads/" + filename + ".txt'"
						+ "INTO TABLE db.courses";
				stmt.executeUpdate(sql);
			} catch (SQLException e) {
				e.printStackTrace();
			}
		}

		public void insertRow(String ID, String section, String title, String
v1, String v2, String v3) {
```

```
                try {
                        Statement stmt = conn.createStatement();
                        String sql = "INSERT INTO courses ( CourseID ,SectionNO ,
Title)" + "  VALUES ('" + v1 + "'," + v2
                                        + ",'" + v3 + "')";
                        stmt.executeUpdate(sql);
                } catch (SQLException e) {
                        e.printStackTrace();
                }

        }

        @Override
        public String write(String filename, String column) {
                try {
                        Statement stmt = conn.createStatement();
                        String sql = "SELECT " + column
                                        + " FROM Courses INTO OUTFILE
'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/" + filename
                                        + ".txt'";
                        stmt.executeUpdate(sql);
                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return filename;
        }

        @Override
        public void delRow(String column, String item) {
                try {
                        Statement stmt = conn.createStatement();
                        String sql = "DELETE FROM courses WHERE" + column + " = "
+ "'" + item + "'";
                        stmt.executeUpdate(sql);
                } catch (SQLException e) {
                        e.printStackTrace();
                }

        }
    }

    class Classes implements TableInterface {
        Classes(String filename) {
                try {
                        Statement stmt = conn.createStatement();
                        String sql = "CREATE TABLE classes(Title CHAR(40),yr
INT,Semester CHAR(40), Department CHAR(50), Program CHAR(50))";
                        stmt.executeUpdate(sql);
                        System.out.println("Created classes table in given
database...");
                        sql = "LOAD DATA LOCAL INFILE 'C:/ProgramData/MySQL/MySQL
Server 8.0/Uploads/" + filename + ".txt'"
                                        + " INTO TABLE db.classes";
                        stmt.executeUpdate(sql);
                } catch (SQLException e) {
```

```
                        e.printStackTrace();
                }

        }

        public void delRow(String column, String item) {
                try {
                        Statement stmt = conn.createStatement();
                        String sql = "DELETE FROM classes WHERE " + column + " = "
+ "'" + item + "'";
                        stmt.executeUpdate(sql);
                } catch (SQLException e) {
                        e.printStackTrace();
                }

        }

        @Override
        public String write(String filename, String column) {
                try {
                        Statement stmt = conn.createStatement();
                        String sql = "SELECT " + column + " FROM classes "
                                        + "INTO OUTFILE 'C:/ProgramData/MySQL/MySQL
Server 8.0/Uploads/" + filename + ".txt'";
                        stmt.executeQuery(sql);
                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return filename;
        }

        public void insertRow(String v1, String v2, String v3, String v4, String
v5, String v6) {
                // TODO Auto-generated method stub
                try {
                        Statement stmt = conn.createStatement();
                        String sql = "INSERT INTO classes (Title,
yr,Semester,Instructor, Department , Program )      VALUES ('"
                                        + v1 + "'," + v2 + ",'" + v3 + "','" + v4 +
"','" + v5 + "')";
                        stmt.executeUpdate(sql);
                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }

}

class AggregateGrades {
        AggregateGrades(String filename) {
                try {
                        Statement stmt = conn.createStatement();
                        String sql = "CREATE TABLE AggregateGrades( grades
CHAR(1),name CHAR(40),ID INT,CourseID CHAR(40), SectionNo CHAR(10),Title CHAR(100))";
                        stmt.executeUpdate(sql);
```

```
                        System.out.println("Created AggregateGrades table in given
database...");
                        sql = "LOAD DATA LOCAL INFILE 'C:/ProgramData/MySQL/MySQL
Server 8.0/Uploads/" + filename + ".txt'"
                                        + "INTO TABLE db.AggregateGrades";
                        stmt.executeUpdate(sql);
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
            }

            public void insertRow(String v1, String v2, String v3, String v4, String
v5, String v6) {
                  try {
                        Statement stmt = conn.createStatement();
                        String sql = "INSERT INTO AggregateGrades (grades ,name
,id ,CourseID , SectionNo,Title) VALUES ('" + v1
                                        + "','" + v2 + "'," + v3 + ",'" + v4 + "'," +
v5 + ",'" + v6 + "')";
                        stmt.executeUpdate(sql);
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }

            }

            public String write(String filename, String column) {
                  try {
                        Statement stmt = conn.createStatement();
                        String sql = "SELECT " + column
                                        + " FROM AggregateGrades INTO OUTFILE
'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/"
                                        + filename + ".txt'";
                        stmt.executeQuery(sql);
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }

                  return filename;
            }

            public void delRow(String column, String item) {
                  try {
                        Statement stmt = conn.createStatement();
                        String sql = "DELETE FROM AggregateGrades WHERE " + column
+ " = " + "'" + item + "'";
                        stmt.executeUpdate(sql);
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }

            }
      }

}
```

StudentsDatabaseInterface

```
package application;

public interface StudentsDatabaseInterface {
	String loadfile(String filename, String table);
}
```

TableInterface

```
package application;

public interface TableInterface {
	String write(String filename, String column);
	void delRow(String column, String item);

}
```

[3] Codes Developed

Classes Imported

```
import java.lang.Math;
import java.util.ArrayList;
import javafx.application.Application;
import javafx.scene.Group;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.shape.ArcType;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import java.io.File;
import java.util.Scanner;
import java.text.DecimalFormat;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;


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

Class Main:

```
package application;

import java.lang.Math;
import java.util.ArrayList;
import javafx.application.Application;
import javafx.scene.Group;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.shape.ArcType;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Main extends Application {

	@Override
	public void start(Stage primaryStage) throws Exception {
		Database db= new Database();
		//Database.AggregateGrades grades =db.new
AggregateGrades("AggregateGrades");
		//grades.insertRow("A", "Jacob", "45789321", "10200CC1", "32118",
"IntroductiontoComputing");
		//String file=grades.write("gradeoutput","grades");
		JFrame g = new JFrame();
		String num = JOptionPane.showInputDialog(g, "Enter number of the highest
appearing letters in file");
		int n = Integer.parseInt(num);
		primaryStage.setTitle("Grades");
		Group root = new Group();
		Canvas canvas = new Canvas(700, 700);
		GraphicsContext gc = canvas.getGraphicsContext2D();
		gc.setLineWidth(1);
		HistogramAlphaBet f = new HistogramAlphaBet();
		f.setFreq("C:/ProgramData/MySQL/MySQL Server
8.0/Uploads/gradeoutput.txt");
		f.setFract();
		HistogramAlphaBet.MyPieChart chart = f.new MyPieChart();
		chart.getSlices(n);
		chart.draw(gc);
		root.getChildren().add(canvas);
		primaryStage.setScene(new Scene(root));
		primaryStage.show();
	}
}
```

Class HistogramAlphaBet:

```
package application;

import java.util.*;
import java.util.Map;
import java.util.Map.Entry;

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

import java.io.File; // Import the File class
import java.util.Scanner; // Import the Scanner class to read text files
import java.text.DecimalFormat;

public class HistogramAlphaBet {
	Map<Character, Integer> characterFreq = new HashMap<Character, Integer>();
	Map<Character, Double> characterFract = new HashMap<Character, Double>();
	String finalS = "";
	Integer sum = 0;

	class MyPieChart {
		Map<Character, Slice> slices = new HashMap<Character, Slice>();
		int topN = 0;
		double startAngle = 0;
		Slice s;

		void getSlices(int n) {
			Map.Entry<Character, Double> entryWithMaxPercentage = null;
			for (MyColor color : MyColor.values()) {
				//
				entryWithMaxPercentage = null;
				for (Entry<Character, Double> entry :
characterFract.entrySet()) {
					if (entryWithMaxPercentage == null
							||
entry.getValue().compareTo(entryWithMaxPercentage.getValue()) > 0) {
						entryWithMaxPercentage = entry;
					}
				}
				//
System.out.println(entryWithMaxPercentage.getValue()*sum);
				System.out.println(entryWithMaxPercentage.getKey());

				characterFract.remove(entryWithMaxPercentage.getKey());
				s = new Slice(300, 300, 300, 300, startAngle, startAngle +
(360 * entryWithMaxPercentage.getValue()),
						ArcType.ROUND);
```

```
                        s.setFill(color);
                        slices.put(entryWithMaxPercentage.getKey(), s);
                        startAngle += (360 * entryWithMaxPercentage.getValue());
                        topN++;
                        if (topN == n || characterFract.size()==0) {
                              s = new Slice(300, 300, 300, 300, startAngle, 360,
ArcType.ROUND);
                              s.setFill(MyColor.AliceBlue);
                              slices.put('*', s);
                              break;
                        }
                  }

            }

            void draw(GraphicsContext gc) {
                  DecimalFormat f = new DecimalFormat("#.###");
                  Double pointSlice = (double) 0;
                  for (Entry<Character, Slice> entry : slices.entrySet()) {
                        pointSlice = entry.getValue().endAngle -
(entry.getValue().angleSlice / 2);
                        entry.getValue().draw(gc);
                        System.out.println(entry.getValue().angleSlice / 360);
                        System.out.println(entry.getKey());
                        if (pointSlice <= 90) {
                              gc.strokeText(entry.getKey() + ", " +
f.format(entry.getValue().angleSlice / 360),
                                          (150 *
Math.cos(Math.toRadians(pointSlice))) + 450,
                                          -(150 *
Math.sin(Math.toRadians(pointSlice))) + 450);

                        } else if (pointSlice <= 180) {
                              gc.strokeText(entry.getKey() + ", " +
f.format(entry.getValue().angleSlice / 360),
                                          -(150 *
Math.sin(Math.toRadians(pointSlice - 90))) + 450 - 30,
                                          -(150 *
Math.cos(Math.toRadians(pointSlice - 90))) + 450 - 20);

                        }

                        else if (pointSlice <= 270) {
                              gc.strokeText(entry.getKey() + ", " +
f.format(entry.getValue().angleSlice / 360),
                                          -(150 *
Math.cos(Math.toRadians(pointSlice - 180))) + 450 - 20,
                                          (150 *
Math.sin(Math.toRadians(pointSlice - 180))) + 450 + 20);

                        } else if (pointSlice <= 360) {
                              gc.strokeText(entry.getKey() + ", " +
f.format(entry.getValue().angleSlice / 360),
                                          (150 *
Math.sin(Math.toRadians(pointSlice - 270))) + 450 + 10,
```

```
                                                (150 *
Math.cos(Math.toRadians(pointSlice - 270))) + 450 + 10);

                              }
                        }

                  }
            }

            public void setFreq(String filename) throws Exception {


                  File myObj = new File(filename);
                  Scanner myReader = new Scanner(myObj);
                  while (myReader.hasNextLine()) {
                        finalS += myReader.nextLine().replaceAll("[^a-zA-Z]",
"").toLowerCase();
                  }
                  myReader.close();
                  for (int i = 0; i < finalS.length(); i++) {
                        if (characterFreq.containsKey(finalS.charAt(i))) {
                              characterFreq.put(finalS.charAt(i),
characterFreq.get(finalS.charAt(i)) + 1);
                        } else {
                              characterFreq.put(finalS.charAt(i), 1);
                        }
                        // Print current character finalS.charAt(i)

                  }
            }

            public void setFract() {
                  for (Entry<Character, Integer> entry : characterFreq.entrySet()) {
                        sum += entry.getValue();
                  }

                  for (Entry<Character, Integer> entry : characterFreq.entrySet()) {
                        characterFract.put(entry.getKey(), (double) entry.getValue() /
sum);
                  }

            }

            public void printFinal() {
                  System.out.println(finalS);
            }
}
```

Class MyArc:

```
package application;

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

public class MyArc extends MyShape {
	MyPoint p1, p2;
	private double width;
	private double height;
	private double startAngle;
	private double arcExtent;
	private ArcType closuretype;

	MyArc(double x, double y, double w, double h, double angle1, double angle2,
ArcType closure) {
		p.setX(x);
		p.setY(y);
		this.width = w;
		this.height = h;
		this.startAngle = angle1;
		this.arcExtent = angle2 - angle1;
		this.closuretype = closure;
	}

	// approximate of arc length
	double length() {
		double r1 = (width / 2) * (height / 2) / (Math.sqrt(
				Math.pow((width / 2) * Math.sin(startAngle), 2) +
Math.pow((width / 2) * Math.sin(startAngle), 2)));
		double r2 = (width / 2) * (height / 2) / (Math.sqrt(Math.pow((width / 2)
* Math.sin(startAngle + arcExtent), 2)
				+ Math.pow((width / 2) * Math.sin(startAngle + arcExtent),
2)));
		double averageR = (r1 + r2) / 2;
		return ((Math.PI) / 180) * arcExtent * averageR;
	}

	@Override
	double getPerimeter() {
		return length();
	}

	@Override
	double getArea() {
		return 1;
	}

	@Override
```

```
	public String toString() {
		return "Arc[x=" + p.getX() + ", y=" + p.getY() + ", width=" + width + ",
height=" + height + ", fill="
				+ fill.toString() + ", length=" + (2 * width + 2 * height)
+ "]";
	}

	@Override
	void draw(GraphicsContext gc) {
		gc.strokeArc(p.getX(), p.getY(), width, height, startAngle, arcExtent,
closuretype);
		gc.setFill(fill.myColor());
		gc.fillArc(p.getX(), p.getY(), width, height, startAngle, arcExtent,
closuretype);
	}

	@Override
	double getWidth() {
		return width;
	}

	@Override
	double getHeight() {
		return height;
	}

	@Override
	public boolean pointInMyShape(double x, double y) {
		if ((Math.pow((x - p.getX() + (width / 2)), 2) / (Math.pow(width / 2,
2)))
				+ (Math.pow((y - p.getY() + (height / 2)), 2) /
(Math.pow(height / 2, 2))) <= 1) {
			if (x > p.getX() + (width / 2) && y < p.getY() - (height / 2)) {
				if (Math.atan(((p.getY() + (height / 2)) - y) / (x -
(p.getX() + (width / 2)))) >= startAngle
						&& Math.atan(((p.getY() + (height / 2)) - y)
/ (x - (p.getX() + (width / 2)))) <= startAngle
								+ arcExtent) {
					return true;
				}
			}
			if (x < p.getX() + (width / 2) && y < p.getY() - (height / 2)) {
				if (Math.atan(((p.getX() + (width / 2)) - x) / ((p.getY()
+ (height / 2)) - y)) + 90 >= startAngle
						&& Math.atan((p.getX() + (width / 2) - x) /
((p.getY() + (height / 2)) - y)) + 90 <= startAngle
								+ arcExtent) {
					return true;
				}
			}
			if (x < p.getX() + (width / 2) && y > p.getY() - (height / 2)) {
				if (Math.atan((y - (p.getY() + (height / 2))) / ((p.getX()
+ (width / 2) - x))) + 180 >= startAngle
						&& Math.atan((y - (p.getY() + (height / 2)))
/ ((p.getX() + (width / 2) - x)))
```

```
                                                        + 180 <= startAngle + arcExtent)
{
                                return true;
                        }
                }

                if (x > p.getX() + (width / 2) && y > p.getY() - (height / 2)) {
                        if (Math.atan((x - (p.getX() + (width / 2))) / (y -
(p.getY() + (height / 2)))) + 270 >= startAngle
                                        && Math.atan(x - (p.getX() + (width / 2)) /
(y - (p.getY() + (height / 2)))) + 270 <= startAngle
                                                        + arcExtent) {
                                return true;
                        }
                }
                if (x == p.getX() + (width / 2) && y < p.getY() - (height / 2)) {
                        if (90 >= startAngle && 90 <= startAngle + arcExtent) {
                                return true;
                        }
                }
                if (x == p.getX() + (width / 2) && y > p.getY() - (height / 2)) {
                        if (270 >= startAngle && 270 <= startAngle + arcExtent) {
                                return true;
                        }
                }
                if (x > p.getX() + (width / 2) && y == p.getY() - (height / 2)) {
                        if (0 >= startAngle && 0 <= startAngle + arcExtent) {
                                return true;
                        }
                }
                if (x < p.getX() + (width / 2) && y == p.getY() - (height / 2)) {
                        if (180 >= startAngle && 180 <= startAngle + arcExtent) {
                                return true;
                        }
                }

        }
        return false;
    }

    @Override
    public MyRectangle getMyBoundingRectangle() {
            MyRectangle S1 = new MyRectangle(p.getX(), p.getY(), getWidth(),
getHeight());
            return S1;
    }

}
```

Class Circle:

```
package application;

public class MyCircle extends MyOval {

	MyCircle(double x, double y, double w, double h) {
		super(x, y, w, w);

	}
	@Override
	public String toString() {
		return "MyCircle";
	}

	public boolean pointInMyShape(double x, double y) {

		if (Math.sqrt(Math.pow((x - (p.getX() + (getWidth() / 2))), 2)
				+ Math.pow((y - (p.getX() + (getWidth() / 2))), 2)) <=
(getWidth() / 2)) {
			return true;
		}
		return false;

	}

}
```

Class MyColor:

```
package application;

import javafx.scene.paint.Color;

public enum MyColor {
	RED(255, 0, 0, 1),
	GREEN(0, 255, 0, 1),
	BLUE(0, 0, 255, 1),
	DARKKHAKI(189, 183, 107, 1),
	DARKORANGE(255, 140, 0, 1),
	DARKORCHID(99,32,204,1),
	FUCHSIA(255, 0, 255, 1),
	GOLD(255, 215, 0, 1),
	GoldenRod(218, 165, 32, 1),
	Lavender(230, 230, 250, 1),
	LightBlue(173, 216, 230, 1),
	LightCoral(240, 128, 128, 1),
	MediumAquaMarine(102, 205, 170, 1),
	Olive(128, 128, 0, 1),
	PaleGreen(152, 251, 152, 1),
	PALEVIOLETRED(219, 112, 147,1),
	YELLOWGREEN(154, 205, 50,1),
	LIGHTSALMON(255, 160, 122,1),
	LIGHTGOLDENRODYELLOW(250, 250, 210,1),
	DEEPPINK(255, 20, 147,1),
	DARKSALMON(233, 150, 122,1),
	CYAN(0, 255, 255,1),
	CORNFLOWERBLUE(100, 149, 237,1),
	BEIGE(245, 245, 220,1),
	AZURE(240, 255, 255,1),
	AQUAMARINE(127, 255, 212,1),
	AliceBlue(240, 248, 255, 1),
	;

	private int red;
	private int green;
	private int blue;
	private double opacity;

	MyColor(int red, int green, int blue, double opacity) {
		this.red = red;
		this.blue = blue;
		this.green = green;
		this.opacity = opacity;
	}

	public Color myColor() {
		return Color.rgb(red, green, blue, opacity);
	}
```

```
	public String toString() {
		return String.format("#%02x%02x%02x", red, green, blue);
	}
}
```

Class MyOval:

```
package application;

import javafx.scene.canvas.GraphicsContext;

public class MyOval extends MyShape {
	private double width = 0;
	private double height = 0;

	// constructor
	MyOval(double x, double y, double w, double h) {
		p.setX(x);
		p.setY(y);
		this.width = w;
		this.height = h;
	}

	@Override
	double getWidth() {
		return width;
	}

	@Override
	double getHeight() {
		return height;
	}

	// area, perimeter
	@Override
	double getArea() {
		return (Math.PI * (width / 2) * (height / 2));
	}

	@Override
	double getPerimeter() {
		return (2 * Math.PI * (Math.sqrt((Math.pow(height / 2, 2) +
Math.pow(width / 2, 2)) / 2)));
	}

	// getX, getY, getA, getB

	// getX(center)
	double getCenterX() {
		return p.getX() + (width / 2);
	}

	// getY,(center)
	double getCenterY() {
		return p.getY() + (height / 2);
	}
```

```
	// toString
	@Override
	public String toString() {
		return "MyOval";
	}

	@Override
	void draw(GraphicsContext gc) {
		gc.strokeOval(p.getX(), p.getY(), this.width, this.height);
		gc.setFill(fill.myColor());
		gc.fillOval(p.getX(), p.getY(), width, height);
	}

	@Override
	public boolean pointInMyShape(double x, double y) {
		if ((Math.pow((x - p.getX() + (width / 2)),2) / (Math.pow(width / 2,
2)))
				+ (Math.pow((y - p.getY() + (height / 2)), 2) / (Math.pow(
height / 2, 2))) <= 1) {
			return true;
		}
		return false;
	}

	@Override
	public MyRectangle getMyBoundingRectangle() {
		MyRectangle S1= new
MyRectangle(p.getX(),p.getY(),getWidth(),getHeight());
		return S1;
	}
}
```

Class MyPoint:

```
package application;

public class MyPoint {
	private double x;
	private double y;

	MyPoint(double x, double y) {
	    this.x=x;
	    this.y=y;
	}

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

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

	double getX() {
```

```
            return x;
        }
        double getY() {
            return y;
        }

        void shiftX(double x) {
            this.x+=x;
        }
        void shiftY(double y) {
            this.y+=y;
        }
        public String toString() {
            return "MyPoint [x = "+x+", y = "+y+"]";
        }


        //distance from P(x,y) to some other point x2,y2
        void getDistance(double x2, double y2){
            System.out.println(Math.sqrt(Math.pow(x2-this.x,2)+Math.pow(y2-
this.y,2)));
        }
        //angle of line from the +x counterclockwise
        void getAngle(double x2, double y2) {
              double m=((y2-this.y)/(x2-this.x));
              System.out.println(Math.atan(m));
        }

}
```

Class MyRectangle:

```
package application;

import javafx.scene.canvas.GraphicsContext;

public class MyRectangle extends MyShape {
        private double width;
        private double height ;

        MyRectangle(double x, double y, double w, double h) {
            this.width = w;
            this.height = h;
            p.setX(x);
            p.setY(y);
        }

        @Override
        double getWidth() {
            return width;
        }

        @Override
        double getHeight() {
            return height;
        }
```

```
	@Override
	public String toString() {
		return "MyRectangle";
	}

	@Override
	double getArea() {
		// TODO
		return (width * height);
	}

	@Override
	double getPerimeter() {
		// TODO
		return (2 * width + 2 * height);
	}

	@Override
	void draw(GraphicsContext gc) {
		// TODO
		gc.strokeRect(p.getX(), p.getY(), this.width, this.height);
		gc.setFill(fill.myColor());
		gc.fillRect(p.getX(), p.getY(), width, height);
	}

	public boolean pointInMyShape(double x, double y) {

		if (x >= p.getX() && x <= p.getX() + this.width && y >= p.getY() && y <=
p.getY() + this.height) {
			return true;
		}
		return false;

	}


	@Override
	public MyRectangle getMyBoundingRectangle() {
		MyRectangle S1= new
MyRectangle(p.getX(),p.getY(),getWidth(),getHeight());
		return S1;
	}

}


/*
if (x >= p.getX() && x <= p.getX() + this.width && y >= p.getY() && y <= p.getY() +
this.height) {
			return true;
		}
*/
```

Class MyShape:

```
package application;

import javafx.scene.canvas.GraphicsContext;
import java.lang.Math;

public abstract class MyShape implements MyShapeInterface {
	// MyPoint,MyColor
	MyPoint p = new MyPoint(0, 0);
	MyColor fill = MyColor.BLUE;

	// area, parameter

	abstract double getArea();

	void draw() {

	}

	abstract double getWidth();

	abstract double getHeight();

	abstract double getPerimeter();

	// toString

	public String toString() {
		return "";
	}

	// draw
	abstract void draw(GraphicsContext gc);

	public abstract boolean pointInMyShape(double x, double y);

	public abstract MyRectangle getMyBoundingRectangle();

}
```

Interface MyShapeInterface:

```
package application;

import java.util.ArrayList;


import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;


public interface MyShapeInterface {

	abstract MyRectangle getMyBoundingRectangle();

	static MyRectangle intersectMyRectangles(MyRectangle R1, MyRectangle R2) {
		double x = 0, y = 0, w = 0, h = 0;
		// if rectangle has area 0, no overlap
		if (R1.getWidth() == 0 || R1.getHeight() == 0 || R2.getWidth() == 0 ||
R2.getHeight() == 0)
			return null;

		// If one rectangle is on left side of other
		if (R1.p.getX() > R2.p.getX() + R2.getWidth() || R2.p.getX() >
R1.p.getX() + R1.getWidth()) {
			return null;
		}

		// If one rectangle is above other
		if (R1.p.getY() > R2.p.getY() + R2.getHeight() || R2.p.getY() >
R1.p.getY() + R1.getHeight()) {
			return null;
		}

		if (R1.p.getY() >= R2.p.getY())
			y = R1.p.getY();
		else
			y = R2.p.getY();

		if (R1.p.getX() >= R2.p.getX())
			x = R1.p.getX();
		else
			x = R2.p.getX();

		if ((R1.p.getY() + R1.getHeight()) >= (R2.p.getY() + R2.getHeight()))
			h = R2.p.getY() + R2.getHeight() - y;
		else
			h = R1.p.getY() + R1.getHeight() - y;

		if ((R1.p.getX() + R1.getWidth()) >= (R2.p.getX() + R2.getWidth()))
```

```
                w = R2.p.getX() + R2.getWidth() - x;
            else
                w = R1.p.getX() + R1.getWidth() - x;

            MyRectangle R3 = new MyRectangle(x, y, w, h);
            return R3;

        }

        static boolean similarObjects(MyShape S1, MyShape S2) {
            if (S1.toString() == S2.toString() && S1.getWidth() == S2.getWidth() &&
S1.getHeight() == S2.getHeight()) {
                // System.out.print(S1.toString());
                return true;
            }

            // System.out.print(S1.toString());
            return false;
        }

        abstract boolean pointInMyShape(double x, double y);

        static ArrayList<MyPoint> intersectMyShapes(MyRectangle R, MyShape S1, MyShape
S2) {
            ArrayList<MyPoint> List = new ArrayList<MyPoint>();

            for (double i = R.p.getY(); i <= R.p.getY() + R.getHeight(); i++) {
                for (double j = R.p.getX(); j <= R.p.getX() + R.getWidth(); j++)
{
                    if (S1.pointInMyShape(j, i) && S2.pointInMyShape(j, i)) {
                        List.add(new MyPoint(j, i));
                        // System.out.print(i+" "+j+"\n");
                    }
                }
            }

            return List;
        }

        static Canvas drawIntersectMyShapes(MyRectangle R, ArrayList<MyPoint> L,
MyColor color) {

            Canvas canvas = new Canvas(R.p.getX() + R.getWidth(), R.p.getY() +
R.getHeight());
            GraphicsContext gc = canvas.getGraphicsContext2D();
            gc.setFill(color.myColor());
            for (int i = 0; i < L.size() - 1; i++) {
                gc.fillRect(L.get(i).getX(), L.get(i).getY(), 1, 1);
            }
            return canvas;
        }
}
```

Class Slice:

```
package application;

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


public class Slice {
	MyArc slice;
	Double angleSlice;
	Double startAngle;
	Double endAngle;

	Slice(double x, double y, double w, double h, double angle1, double angle2,
ArcType closure) {
		slice = new MyArc(x, y, w, h, angle1, angle2, closure);
		this.endAngle=angle2;
		this.angleSlice=angle2-angle1;
	}

	void setFill(MyColor c) {
		slice.fill = c;
	}

	@Override
	public String toString() {
		return "Slice";
	}

	void draw(GraphicsContext gc) {
		slice.draw(gc);
	}
}
```

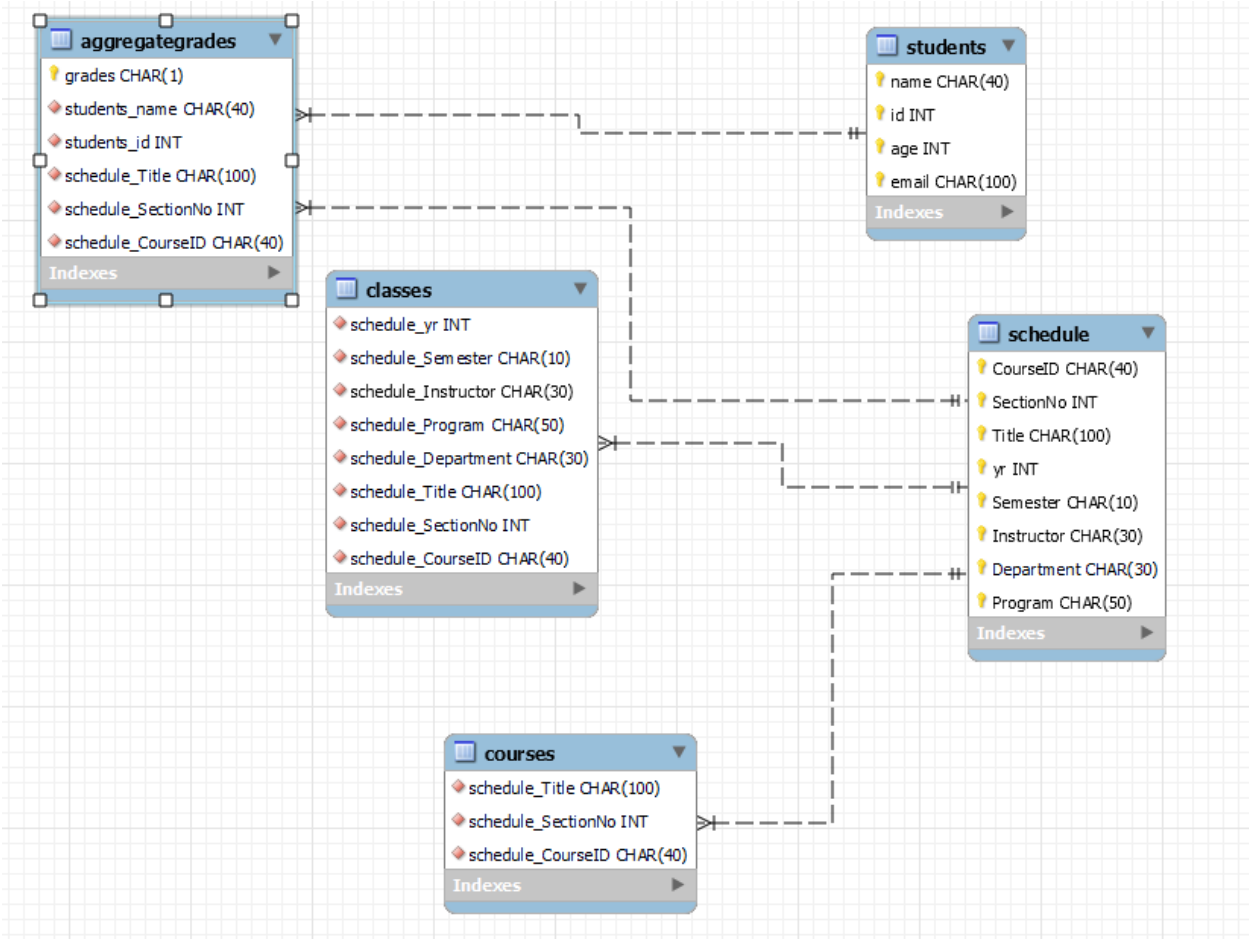
Interface `StudentsDatabaseInterface`:

```
package application;

public interface StudentsDatabaseInterface {
	String loadfile(String filename, String table);
}
```

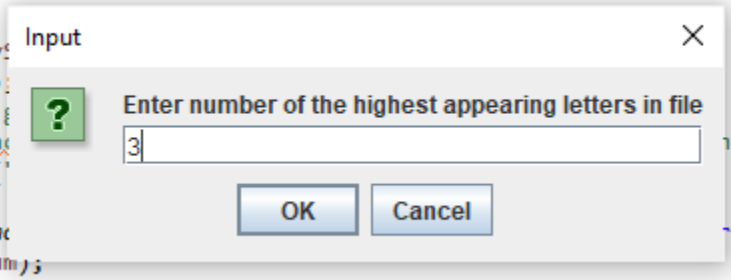
Interface TableInterface:

```
package application;

public interface TableInterface {
	String write(String filename, String column);
	void delRow(String column, String item);

}
```

OUTPUT and Other[4]:

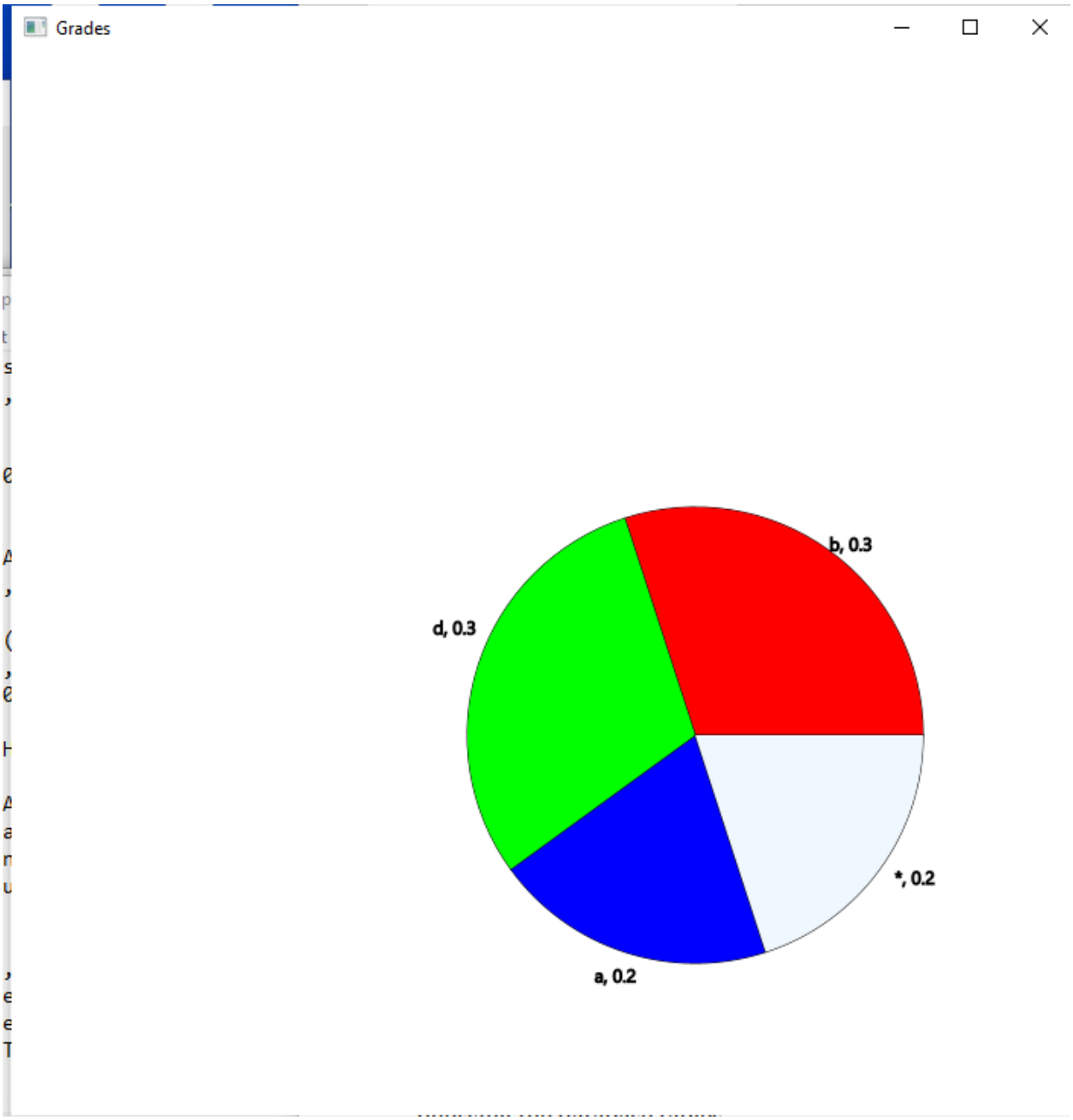



```
lication {
```




```
rimary
base()
rades
", "Ja
write("
();
ane.sh
Int(num);
("Grades");
```

```
ntoComputi
ring lette
```


Grades
b, 0.3
d, 0.3
*, 0.2
a, 0.2

SQL Code: (after creating database)[3]*

```
use db;
CREATE TABLE Schedule(
CourseID CHAR(40),
SectionNo INT,
 Title CHAR(100),
  yr INT,
Semester CHAR(10),
 Instructor CHAR(30),
Department CHAR(30),
Program CHAR(50)

);
 LOAD DATA LOCAL INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/ScheduleSpring2022.txt'
INTO TABLE db.schedule;


CREATE TABLE courses(
CourseID CHAR(40),
SectionNo INT,
Title CHAR(100)
);


INSERT INTO courses (CourseID,SectionNo,Title  )
SELECT CourseID,SectionNo,Title
FROM schedule;
```

```
select * from courses;



INSERT INTO classes (Title,yr,Semester,    Instructor,Department,Program )
SELECT Title,yr,Semester,Instructor,Department,Program
FROM schedule;

select * from classes;

CREATE TABLE students(
name CHAR(40),
id INT,
age INT,
email CHAR(100)
);

LOAD DATA LOCAL INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/students.txt' INTO TABLE
db.students;



CREATE TABLE AggregateGrades(
grades CHAR(40),
name CHAR(40),
id CHAR(40),
CourseID CHAR(40),
SectionNo INT,
Title CHAR(100),
);
```

```
LOAD DATA LOCAL INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/AggregateGrades1.txt'
INTO TABLE db.AggregateGrades;



select * from aggregategrades;

SELECT grades,Count(id)

FROM aggregategrades

GROUP BY grades

ORDER BY COUNT(CourseID) DESC;
```