KENYATTA UNIVERSITY

SCHOOL OF PURE AND APPLIED SCIENCE

UNIT CODE: SMA 391

UNIT TITLE: OBJECT ORIENTED PROGRAMMING

GROUP 14 PROJECT ASSIGNMENT

UNIVERSITY EXAM GRADING SYSTEM

Group Members

Registration Number	**Name**
I163/1397/2023	Ruth Chemtai
I163/4484/2023	Auther Kachuodho
I163/7000/2023	Evy Jerono
I163/7001/2023	Phabian Otieno
I163/7011/2023	Samuel Wachara
I163/7023/2023	Henry Otieno
I163/7027/2023	Samuel Mkalla
I163/7038/2023	Augo Shiltone
I163/8066/2023	Brian Adams
I163/6960/2023	Rubia Lewis

1. Introduction

The University Exam Grading System is a Java-based console application designed to automate the process of recording student marks, computing grades, and persisting academic records in a PostgreSQL database. The project demonstrates core Object-Oriented Programming (OOP) principles and enterprise-level practices, including abstraction, inheritance, polymorphism, encapsulation, database integration (JDBC), input validation and robust error handling.

The system is suitable for university-level examinations and is structured to be extensible, maintainable and secure.

2. Algorithms Used

a. Student Registration Algorithm (Pseudocode)

```
try {
    System.out.println("=== UNIVERSITY EXAM GRADING SYSTEM ===");

    // --- Read Student ID as String
    System.out.print("Enter Student ID: ");
    String id = scanner.nextLine();
```

b. Grade Computation Algorithm

```
class StandardGrading implements GradingPolicy {

    @Override
    public String calculateGrade(double marks) {
        if (marks >= 70) return "A";
        else if (marks >= 60) return "B";
        else if (marks >= 50) return "C";
        else if (marks >= 40) return "D";
        else return "E";
    }
}
```

3. Application Design

a. **Database table creation -** The following table was created in PostgreSQL database, with a table called Students

```sql
DROP TABLE IF EXISTS students;
DELETE FROM students;

CREATE TABLE students (
    id VARCHAR(25) PRIMARY KEY,
    name VARCHAR(100),
    course VARCHAR(100),
    marks DOUBLE PRECISION,
    grade VARCHAR(2),
    department VARCHAR(50),
    school VARCHAR(50)
);


SELECT * FROM students;
```

b. **Student input form -** This is just an example of a sample student with his results that we used.

```
C:\Users\Administrator\Documents\UniversityExamGradingSystem-Folder>java -cp ".;lib/postgresql-42.7.8.jar" UniversityExamGradingSystem
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
=== UNIVERSITY EXAM GRADING SYSTEM ===
Enter Student ID: I163/7027/2023
Enter Student Name: Samuel Mkalla
Enter Marks: 73

--- STUDENT DETAILS ---
Student ID : I163/7027/2023
Name       : Samuel Mkalla
Course     : Mathematics
Marks      : 73.0
Grade      : A
Department : Mathematics Department
School     : School of Pure and Applied Science
✓ Student record saved to database.
```

c. **Grade output display -** The data base output contains a record of the student information such as Student ID, Name, Course, Marks, Grade, Department and School.

	id [PK] character varying (25)	name character varying (100)	course character varying (100)	marks double precision	grade character varying (2)	department character varying (50)	school character varying (50)
1	J17/2689/2021	Mercy Wambui	Computer Science	43	D	Computer Science Departm...	School of Pure and Applied Science
2	J17/7038/2023	Aisha Mwangi	Computer Science	22	E	Computer Science Departm...	School of Pure and Applied Science
3	E125/7372/2023	Abigael Mushiri	Education	66	B	Education Department	School of Education
4	I163/7011/2023	Samuel wachara	Mathematics	72	A	Mathematics Department	School of Pure and Applied Science
5	I163/7000/2023	Evy Jerono	Mathematics	59	C	Mathematics Department	School of Pure and Applied Science
6	I163/7023/2023	Henry Otieno	Mathematics	85	A	Mathematics Department	School of Pure and Applied Science
7	I163/8066/2023	Brian Adams	Mathematics	62	B	Mathematics Department	School of Pure and Applied Science
8	I163/1397/2023	Ruth chemtai	Mathematics	76	A	Mathematics Department	School of Pure and Applied Science
9	J17/4636/2020	Faith Njoroge	Computer Science	26	E	Computer Science Departm...	School of Pure and Applied Science
10	I163/7001/2023	Phabian Otieno	Mathematics	43	D	Mathematics Department	School of Pure and Applied Science
11	J17/9000/2023	Daniel Otieno	Computer Science	66	B	Computer Science Departm...	School of Pure and Applied Science
12	I163/7038/2023	Augo Shiltone	Mathematics	79	A	Mathematics Department	School of Pure and Applied Science
13	E125/4790/2019	Grace Atieno	Education	46	D	Education Department	School of Education
14	E125/6960/2023	Rubia Lewis	Education	10	E	Education Department	School of Education
15	I163/7027/2023	Samuel Mkalla	Mathematics	73	A	Mathematics Department	School of Pure and Applied Science
16	I163/4484/2023	Auther Kachuodho	Mathematics	88	A	Mathematics Department	School of Pure and Applied Science

4. Full Java Code

a. Abstract Class (Abstraction)

Used for creating an abstract class Person, so as to store the student information such as Student ID.

```java
// ================ ABSTRACTION ================
abstract class Person {
    protected String id;  // String ID now
    protected String name;

    public Person(String id, String name) {
        this.id = id;
        this.name = name;
    }

    public abstract void displayInfo();
}
```

b. Student Class (Inheritance & Encapsulation)

The concept was integrated so as to create multiple classes, which depends on the main class Student.

```java
// ================ INHERITANCE + ENCAPSULATION ================
class Student extends Person {
    private String course;
    private double marks;

    public Student(String id, String name, String course, double marks) {
        super(id, name);
        this.course = course;
        setMarks(marks);
    }

    public double getMarks() {
        return marks;
    }

    public void setMarks(double marks) {
        if (marks < 0 || marks > 100) {
            throw new IllegalArgumentException("Marks must be between 0 and 100.");
        }
        this.marks = marks;
    }

    public String getCourse() {
        return course;
    }

    @Override
    public void displayInfo() {
        System.out.println("Student ID : " + id);
        System.out.println("Name       : " + name);
        System.out.println("Course     : " + course);
        System.out.println("Marks      : " + marks);
    }
}
```

c. Grading Interface (Polymorphism)

```
// ================ POLYMORPHISM ================
interface GradingPolicy {
    String calculateGrade(double marks);
}
```

d. Grade Calculator Implementation

```
class StandardGrading implements GradingPolicy {

    @Override
    public String calculateGrade(double marks) {
        if (marks >= 70) return "A";
        else if (marks >= 60) return "B";
        else if (marks >= 50) return "C";
        else if (marks >= 40) return "D";
        else return "E";
    }
}
```

e. PostgreSQL Database Connection (JDBC) and Data Access Object (DAO)

```
// ================ DATABASE INTEGRATION ================
class DatabaseManager {

    private static final String URL =
            "jdbc:postgresql://localhost:5432/UniversityExamGradingSystem";
    private static final String USER = "postgres";
    private static final String PASSWORD = "Phabiansharish@254";

    public static void saveStudent(Student student, String grade, String department, String school) {

        String sql = "INSERT INTO students(id, name, course, marks, grade, department, school) " +
                     "VALUES (?, ?, ?, ?, ?, ?, ?)";

        try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);
             PreparedStatement stmt = conn.prepareStatement(sql)) {

            stmt.setString(1, student.id);
            stmt.setString(2, student.name);
            stmt.setString(3, student.getCourse());
            stmt.setDouble(4, student.getMarks());
            stmt.setString(5, grade);
            stmt.setString(6, department);
            stmt.setString(7, school);

            stmt.executeUpdate();
            System.out.println("✓ Student record saved to database.");

        } catch (SQLException e) {
            System.err.println("✗ Database Error: " + e.getMessage());
        }
    }
}
```

f. Main Application Class

```java
public class UniversityExamGradingSystem {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        GradingPolicy gradingPolicy = new StandardGrading();

        try {
            System.out.println("=== UNIVERSITY EXAM GRADING SYSTEM ===");

            // --- Read Student ID as String
            System.out.print("Enter Student ID: ");
            String id = scanner.nextLine();

            // Determine course and department automatically
            String course;
            String department;
            char firstChar = Character.toUpperCase(id.charAt(0));

            switch (firstChar) {
                case 'J':
                    course = "Computer Science";
                    department = "Computer Science Department";
                    break;
                case 'I':
                    course = "Mathematics";
                    department = "Mathematics Department";
                    break;
                case 'E':
                    course = "Education";
                    department = "Education Department";
                    break;
                default:
                    System.out.print("Enter Course (manual): ");
                    course = scanner.nextLine();
                    System.out.print("Enter Department (manual): ");
                    department = scanner.nextLine();
                    break;
            }
```

g. Error Handling

```java
public void setMarks(double marks) {
    if (marks < 0 || marks > 100) {
        throw new IllegalArgumentException("Marks must be between 0 and 100.");
    }
    this.marks = marks;
}
```

The following show an Error if the Marks that is input is not in the range of 0 to 100

```
C:\Users\Administrator\Documents\UniversityExamGradingSystem-Folder>java -cp ".;lib/postgresql-42.7.8.jar" UniversityExamGradingSystem
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
=== UNIVERSITY EXAM GRADING SYSTEM ===
Enter Student ID:  I163/6960/2023
Enter Course (manual): Mathematics and Computer Science
Enter Department (manual): Mathematics
Enter Student Name: Barnard Odiek
Enter Marks: a10
❌ Invalid input type. Please enter correct values.
```

7. Limitations and Solutions

- **Database Connectivity Issues:** Resolved by correctly configuring PostgreSQL JDBC driver and connection string.
- **Input Validation Errors:** Addressed using exception handling and defensive programming.
- **Code Organization:** Solved by adopting DAO and layered architecture.

8. Conclusion

The University Exam Grading System successfully demonstrates core Java programming concepts, OOP principles, and database integration. The project provides a strong foundation for building scalable academic management systems and can be extended with GUIs, authentication, and reporting modules.

9. References

i. Oracle Java Documentation
ii. PostgreSQL Official Documentation
iii. Head First Java – O'Reilly
iv. JDBC API Guide