CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

PROJECT REPORT
ON

Student Information System

Program Name: BCA

Subject Name/Code: JAVA PROGRAMMING LAB (22CAP-352)

Submitted by:

Name: Sangay Dorji Tamang

UID:22BCA10918

Section:22BCA-10A

Submitted to:

Name: Mr.Suman Acharya

Designation: Asst.prof

ABSTRACT

The Student Information System is a Java mini project that has been designed to demonstrate the practical implementation of Object-Oriented Programming (OOP) principles in a real-world situation. This system allows users to enter and maintain academic information pertaining to a graduate-level student, such as personal info, departmental association, academic record, and thesis subject. The system also maintains course information related to the student.

The major objective of this project is to offer practical exposure to fundamental OOP concepts like class design, multilevel inheritance (Person → Student → GraduateStudent), method and constructor overloading, and the utilization of interfaces for modular functionality such as grade calculation. The project also uses static variables to maintain the count of the total number of students instantiated during runtime.

The system communicates with users through the console through the use of Java's Scanner class, which renders it easy and efficient to utilize for the display of dynamic input of data, object creation, and result calculation. Through the project, the students are able to acquire first-hand knowledge about OOP while also learning the way to structure and design an application that can scale and remain modular. This project provides an excellent foundation to more advanced systems like student portal management or academic databases.

Introduction

Handling student information is a basic need in schools and educational institutions to monitor academic record, enrollment status, or other personal details. Conventionally, this is large data and needs manual intervention. With the emergence of software applications and object-oriented programming, systems can be simplified, made error-free, and more scalable.

The Student Information System project is a basic console-based Java application that mimics the handling of a student's academic record. It enables the user to input pertinent student details, including name, age, department, marks, and thesis topic, as well as related course information. The system subsequently processes this input to compute and display the student's academic grade according to a pre-defined grading system.

This project mainly addresses the usage of Object-Oriented Programming (OOP) concepts. It illustrates how classes can be employed to represent actual-world entities, how inheritance establishes a rational hierarchy among them, and how interfaces may be used to provide extra functionalities such as computing grade. Additionally, it uses constructor and method overloading to demonstrate how several means of object initialization can be addressed, and employs static variables to store common data like the total number of students.

The system is well-organized for future improvements and serves as a teaching aid to learn OOP principles in Java. It serves as a basis for more sophisticated systems, for example, complete student management software with database connection and graphical user interfaces.

Technique

This project uses the following **Object-Oriented Programming (OOP)** principles and techniques:

1. **Class and Object**:
    - Person, Student, GraduateStudent, and Course classes represent different entities.
2. **Constructor Overloading**:
    - Multiple constructors for different use cases in Student and Person classes.
3. **Method Overloading**:
    - displayInfo() method is overloaded to show optional department info.
4. **Multilevel Inheritance**:
    - Person → Student → GraduateStudent.
5. **Interface Implementation**:
    - Grading interface is implemented in the Student class to calculate grades based on marks.
6. **Static Variables and Methods**:
    - studentCount static variable tracks the total number of students.
7. **User Input with Scanner**:
    - Collects data for the course and the graduate student at runtime.

Source Code:

```
import java.util.Scanner;


interface Grading {
    String calculateGrade(double marks);
}


class Person {
    String name;
    int age;

    Person() {}

    Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    void displayInfo() {
        System.out.println("Name: " + name + ", Age: " + age);
    }
}


class Student extends Person implements Grading {
    int studentId;
    static int studentCount = 0;

    Student() {
        super();
        this.studentId = ++studentCount;
    }

    Student(String name, int age) {
        super(name, age);
        this.studentId = ++studentCount;
    }
```

```
    void displayInfo(String dept) {
        System.out.println("Name: " + name + ", Age: " + age + ", Dept: " + dept);
    }

    @Override
    public String calculateGrade(double marks) {
        if (marks >= 90) return "A";
        else if (marks >= 75) return "B";
        else if (marks >= 60) return "C";
        else if (marks >= 50) return "D";
        else return "F";
    }

    void displayStudentInfo() {
        super.displayInfo();
        System.out.println("Student ID: " + studentId);
    }
}


class GraduateStudent extends Student {
    String thesisTopic;

    GraduateStudent(String name, int age, String thesisTopic) {
        super(name, age);
        this.thesisTopic = thesisTopic;
    }

    void displayGraduateInfo(String dept, double marks) {
        displayStudentInfo();
        displayInfo(dept);
        System.out.println("Thesis Topic: " + thesisTopic);
        System.out.println("Grade: " + calculateGrade(marks));
    }
}


class Course {
    String courseName;
    int courseCode;
```

```
    Course(String courseName, int courseCode) {
        this.courseName = courseName;
        this.courseCode = courseCode;
    }

    void displayCourse() {
        System.out.println("Course: " + courseName + " | Code: " + courseCode);
    }
}


public class Main{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);


        System.out.print("Enter course name: ");
        String courseName = sc.nextLine();
        System.out.print("Enter course code: ");
        int courseCode = sc.nextInt();
        sc.nextLine();
        Course course = new Course(courseName, courseCode);
        course.displayCourse();


        System.out.print("Enter student name: ");
        String name = sc.nextLine();
        System.out.print("Enter student age: ");
        int age = sc.nextInt();
        sc.nextLine();
        System.out.print("Enter department: ");
        String dept = sc.nextLine();
        System.out.print("Enter marks: ");
        double marks = sc.nextDouble();
        sc.nextLine();
        System.out.print("Enter thesis topic: ");
        String thesis = sc.nextLine();

        GraduateStudent student = new GraduateStudent(name, age, thesis);
        student.displayGraduateInfo(dept, marks);
```

```
        System.out.println("Total Students: " + Student.studentCount);

        sc.close();
    }
}
```

Formula

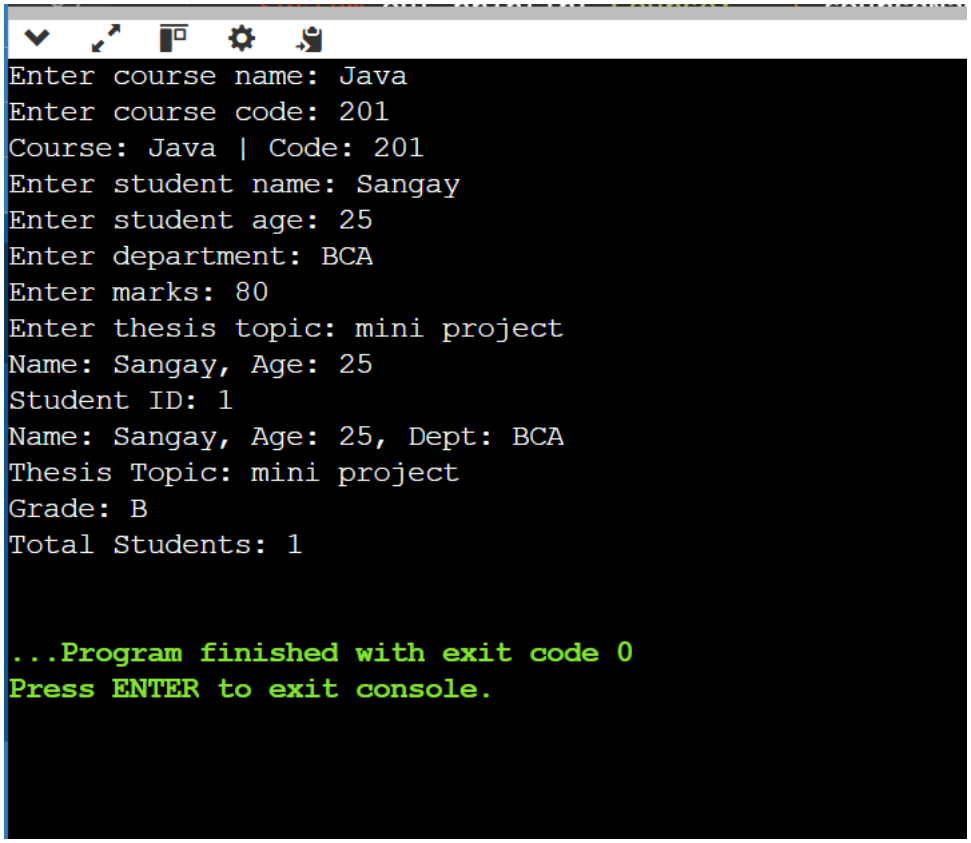
The **Grade Calculation** is performed using the following logic:

Java Logic (Pseudocode):

```
if (marks >= 90) return "A";
else if (marks >= 75) return "B";
else if (marks >= 60) return "C";
else if (marks >= 50) return "D";
else return "F";
```

Marks Range	Grade
90 – 100	A
75 – 89	B
60 – 74	C
50 – 59	D
Below 50	F

Result and Analysis



Analysis:

- The system correctly takes user input and represents it using class structures.
- The grading logic functions as expected.
- Static tracking of student count is effective.
- Inheritance and interfaces provide modular, reusable code.

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

SUMMARY

The Student Information System mini project effectively illustrates the real-world application of Object-Oriented Programming (OOP) principles in Java to simulate and handle student-related information. Through the simulation of creating a graduate student profile with properties such as name, age, department, marks, and thesis title, the system offers an organized and interactive mechanism for gathering and processing user data. It further incorporates course details and calculates the student's grade dynamically based on input marks.

Major OOP concepts—like class and object declaration, constructor and method overloading, multilevel inheritance, interface implementation, and static variables—are efficiently incorporated into the system to develop an elastic, reusable, and scalable framework. The application of Scanner for runtime user input makes the program more realistic and interactive by allowing dynamic data handling.

This project is an introductory learning project for students learning Java programming and OOP design patterns. It is easy yet strong enough to be extended into more intricate systems with functionalities such as multiple student records, data persistence with databases, or even graphical user interfaces (GUI).

In summary, the Student Information System not only achieves its core goals but also lays a solid foundation for further development and learning in software design and development.