Assignment on

Student Database

CSE 2112 Object Oriented Programming

Department of Computer Science and Engineering

University of Dhaka

Submitted To:
Course Instructor: Redwan Ahmed Rizvee
(Lecturer)
Department of Computer Science and Engineering
University of Dhaka

Submitted By:
Name: Papry Rahman
Roll: 1
2nd Year 1st Semester
Session:2021-2021
Submission Date: 2 April 2024

My Java program on student database follows the OOP principles as well as has some unique features, and flexibility, and is well-designed to incorporate new dimensions or modules. Here I am explaining which features and OOP principles which have been implemented successfully in my program.

- **Encapsulation:**

Encapsulation is one of the fundamental principles of object-oriented programming (OOP), including in Java. It refers to the bundling of data or methods that operate on the data into a single unit or class which may have restrictions to access some of the data or methods from outside. Here in Student class, I used some public data and also private parameters (such as “cgpa”, “totalm”, etc) which can not be accessed from outside of the class(private), but here I have designed “gettotal()”, “getGPA()”, etc methods to access or modify them. Here the internal representation of class is hidden but still, we can access them. Also, it offers to operate data as a single unit.

- **Inheritance**

Inheritance is a mechanism that allows a class to inherit properties and behaviors from another class as well as promotes code reuse and establishes relationships between classes. Here classes for every course’s grade calculation (“Security”, “AI”, “NET”, etc) are extended from the class “Course” even though all these classes have an aggregative relation with the Evaluation class. So, here are classes for every course that inherit the properties of parents as well as the “Evaluation” class so that we can re-use the methods of them and their variables.

- **Polymorphism**

Polymorphism is the ability of any data to be processed in more than one form. Here we can be acquainted with the term method overloading. In my program, while evaluating each course, I used a class named “Evaluation”, where polymorphism is used. There were many criteria to evaluate a course such as mid+final, mid+final+assignment, mid+final+assignment+attendance, etc and according to necessity, it is easy to call a constructor with the same name but different parameters.

- **Abstraction**

According to the definition, abstraction in OOPS is used to hide unnecessary information and display only necessary information to the users interacting. It is essential to represent real-world objects in a simplified manner for users to interact easily. Here abstract class “Course” has 5 extensions of each course, and it has two abstract methods(“marks()”, “total()”), which can be executed by calling any extended class with method overriding.

- **Association and Composition:**

Association represents relationships between classes, where one class is related to another class through a reference field or method parameter, and Composition is a form of association where one class contains an object of another class as a member field. In my program, I have some composite relation between one class's object with another class. In "Student" class, I have included the "Course" class as a field, the "AI" class, "Security" etc. And each extension of course class such as "AI", "ES" etc has a relation with the "Evaluation" class. Even in rank class, I have related Student's class methods. So, here the program's classes and methods are related to each other.

Features and Contents

Flexibility: The program is easy to expand, add more classes/objects/methods, evaluation criteria even different rankings where we hardly need to modify the code inside the top hierarchy.

Well Designed: The program is as far well designed to maintain a hierarchy, adding new features, as we started with student, listed them in rank class, and evaluated them in a separate class, so this design allows us to form a complete student database program.

Dynamicity and Randomness: The program allows for the dynamic creation of courses based on user input, including random generation of evaluation criteria. This feature adds flexibility and realism to the simulation, as courses can have varying evaluation methods.

Comprehensive Evaluation Criteria, Ranking Functionality: The program covers a wide range of evaluation criteria for courses, including final scores, midterm scores, assignments, and attendance. This comprehensive approach adds depth to the simulation, reflecting real-world evaluation practices in academic settings. The program includes functionality to rank and sort students based on their overall GPA or course-specific GPA. This feature provides valuable insights into student performance and allows for easy identification of high-achieving students.

Menu-Driven Interface: The program employs a menu-driven interface to interact with users, making it user-friendly and intuitive to navigate. This interface facilitates easy access to various functionalities and enhances the overall user experience.

Combination of Object-Oriented Principles: Your program effectively utilizes object-oriented principles such as encapsulation, inheritance, and polymorphism to model the university system. This structured approach results in a modular, maintainable, and extensible codebase.