

**ASSIGNMENT # 2**

**COMSATS UNIVERSITY ISLAMABAD SAHIWAL CAMPUS**

**HABIB UR REHMAN**

FA23-BCS-251

SECTION: E

**Sir Ali Sher Kashif**

**SUBMITTED TO**

Object Oriented Programming

**Question 1**

Scenario:

You are hired as a software developer for a company that manages employee payroll systems. The company wants to ensure that sensitive employee information such as salary and bank account details is not exposed directly to external classes. The system should allow users to update their name and position but only authorized personnel should be able to modify the salary.

Task:

Designaclass named Employee that stores the following details: employeeID, name,

position, salary, and bankAccountNumber.

Ensurethat sensitive information (salary and bankAccountNumber) cannot be

directly accessed or modified by external classes.

Provideappropriate public methods to:

* Changetheemployee's name and position.
* Updatesalary, but ensure that this can only be done through a special method, say updateSalary(), that simulates authorized access.

Implement the class in Java and explain how the concept of encapsulation ensures data

protection in this scenario.

**Solution:**

class Employee {

private String employeeID;

private String name;

private String position;

private double salary;

private String bankAccountNumber;

public Employee(String employeeID, String name, String position, double salary, String bankAccountNumber) {

this.employeeID = employeeID;

this.name = name;

this.position = position;

this.salary = salary;

this.bankAccountNumber = bankAccountNumber;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPosition() {

return position;

}

public void setPosition(String position) {

this.position = position;

}

public void updateSalary(double newSalary, boolean authorized) {

if (authorized) {

this.salary = newSalary;

} else {

System.out.println("Unauthorized access to modify salary.");

}

}

public void displayEmployeeDetails() {

System.out.println("Employee ID: " + employeeID);

System.out.println("Name: " + name);

System.out.println("Position: " + position);

}

}

public class Main {

public static void main(String[] args) {

Employee emp1 = new Employee("001", "Ahmad Kha", "Software Developer", 50000, "1234");

emp1.displayEmployeeDetails();

emp1.setName("Ahmad Khan");

emp1.setPosition("Senior Developer");

emp1.updateSalary(55000, true);

emp1.displayEmployeeDetails();

}

}

**Question 2:**

**Scenario:**

You are tasked with designing a software system for a university to manage different types of users such as Students, Professors, and Administrators. The university wants to keep track of general information like userID, name, and email, but each type of user has specific attributes:

* Students have amajor and yearOfStudy.
* Professors have a department and researchArea.
* Administrators have a role and officeLocation.

Task:

* Createabaseclass UniversityUser with the general attributes userID, name, and email.
* Derivethree classes: Student, Professor, and Administrator from UniversityUser.
  + Studentshould have additional attributes: major and yearOfStudy.
  + Professor should have additional attributes: department and researchArea.
  + Administrator should have additional attributes: role and officeLocation.
* ImplementamethoddisplayDetails() in each class to print the user’s details, including their specific attributes.

**Solution:**

class UniversityUser {

protected String userID;

protected String name;

protected String email;

public UniversityUser(String userID, String name, String email) {

this.userID = userID;

this.name = name;

this.email = email;

}

public void displayDetails() {

System.out.println("User ID: " + userID);

System.out.println("Name: " + name);

System.out.println("Email: " + email);

}

}

class Student extends UniversityUser {

private String major;

private int yearOfStudy;

public Student(String userID, String name, String email, String major, int yearOfStudy) {

super(userID, name, email);

this.major = major;

this.yearOfStudy = yearOfStudy;

}

@Override

public void displayDetails() {

super.displayDetails();

System.out.println("Major: " + major);

System.out.println("Year of Study: " + yearOfStudy);

}

}

class Professor extends UniversityUser {

private String department;

private String researchArea;

public Professor(String userID, String name, String email, String department, String researchArea) {

super(userID, name, email);

this.department = department;

this.researchArea = researchArea;

}

@Override

public void displayDetails() {

super.displayDetails();

System.out.println("Department: " + department);

System.out.println("Research Area: " + researchArea);

}

}

class Administrator extends UniversityUser {

private String role;

private String officeLocation;

public Administrator(String userID, String name, String email, String role, String officeLocation) {

super(userID, name, email);

this.role = role;

this.officeLocation = officeLocation;

}

@Override

public void displayDetails() {

super.displayDetails();

System.out.println("Role: " + role);

System.out.println("Office Location: " + officeLocation);

}

}

public class Main {

public static void main(String[] args) {

//student

Student student = new Student("S001", "Tariq", "tariq@gmail.com", "Computer Science", 3);

student.displayDetails();

// professor

Professor professor = new Professor("PS001", "Dr. Sajid", "Sajid@gmail.com", "Physics", "Mechanics");

professor.displayDetails();

// administrator

Administrator admin = new Administrator("A001", "Muzammil", "muzammil@gmail.com", "System Admin", "Room 202");

admin.displayDetails();

}

}