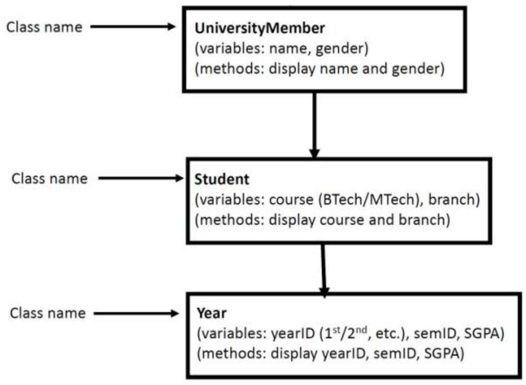
Programming with Python and Java (CS 29008) TEST-I

Solve any two questions from SECTION-A & any one question from SECTION-B. (20M)

SECTION-A (5M each)

Q1. Using a multi-level inheritance, write a Java program to implement the relationship shown in Figure-1. Also, include constructors in every class to initialize the member variables.


Class name
UniversityMember
(variables: name, gender)
(methods: display name and gender)
Class name
Student
(variables: course (BTech/MTech), branch)
(methods: display course and branch)
Class name
Year
(variables: yearID (1st/2nd, etc.), semID, SGPA)
(methods: display yearID, semID, SGPA)

Q-2. A complex number is of the form A + iB where A is the real part and B is an imaginary part of the number. Design a Java class called Complex representing the complex number with member data A and B of the number. Include constructors and member methods to perform the following:

a) to accept and display a complex number
b) to find the sum of two complex numbers
c) to find the product of two complex numbers

Q.3 Create a Java class called uniMember which has instance-variables name and gender. Within this class, create two more classes, Student with instance-variable roll number and Faculty with instance-variable employee id. Write the Java methods to enter the details (name, gender, roll number, employee id) of a student and a faculty and display the same on the console.

Q.4 Create a Java class Employee with multiple constructors: One constructor that takes name and salary. Another constructor that takes name, salary, and department. A constructor that takes all previous fields and an employee ID. Write a display() method to print employee details. Demonstrate constructor overloading by creating different employee objects.

SECTION-B (Real world Application) (10 M)

Q.5 You are asked to build an E-Commerce Order Management System using Java Inheritance.

Class Hierarchy:

1. **Base Class:** `User`
   - o Attributes: `userId`, `name`, `email`
   - o Method: `displayUserInfo()`
2. **Subclass:** `Customer` (Inherits from `User`)
   - o Additional Attributes: `customerType` (Regular/Premium)
   - o Method: `placeOrder()`
   - o Override `displayUserInfo()`
3. **Subclass:** `Admin` (Inherits from `User`)
   - o Additional Attributes: `adminLevel`
   - o Method: `manageInventory()`
   - o Override `displayUserInfo()`
4. **Abstract Class:** `Order`
   - o Attributes: `orderId`, `amount`
   - o Abstract Method: `processOrder()`
5. **Concrete Class:** `OnlineOrder` (Inherits from `Order`)
   - o Additional Attributes: `deliveryAddress`
   - o Implement `processOrder()`
6. **Concrete Class:** `StorePickupOrder` (Inherits from `Order`)
   - o Additional Attributes: `pickupLocation`
   - o Implement `processOrder()`
7. **Interface:** `Discountable`
   - o Method: `applyDiscount()`
8. **Implementing Class:** `PremiumCustomer` (Inherits `Customer` and Implements `Discountable`)
   - o Override `applyDiscount()`
   - o Apply **10% discount** for premium customers.

Tasks:

1. **Create a `User` class** with a constructor and `displayUserInfo()`.
2. **Create `Customer` and `Admin` classes** extending `User` and overriding `displayUserInfo()`.
3. **Create an abstract `Order` class** with an abstract method `processOrder()`.
4. **Implement `OnlineOrder` and `StorePickupOrder`** to extend `Order` and provide their own order processing logic.
5. **Create an interface `Discountable`** with a method `applyDiscount()`.
6. **Implement `Discountable` in `PremiumCustomer`** and apply a discount on the total order amount.
7. **Demonstrate Inheritance and Method Overriding** by creating:
   - o A **Regular Customer** placing an `OnlineOrder`.
   - o A **Premium Customer** placing a `StorePickupOrder` with a discount.
   - o An **Admin** managing inventory.

Q.6. You are asked to build a **Hotel Booking System** using **Java Constructors**.

Class Hierarchy

1. **Class `Hotel`**
    - Attributes: `hotelName`, `location`, `rating` (stars)
    - **Constructors**:
        - Default constructor initializes values to `"Unknown"` and `0 stars`
        - Parameterized constructor to set values
    - Method: `displayHotelInfo()`
2. **Class `Room`**
    - Attributes: `roomNumber`, `roomType` (Single/Double/Deluxe), `price`
    - **Constructors**:
        - Default constructor initializes room with `0`, `"Standard"`, `0.0`
        - Parameterized constructor for initialization
    - Method: `displayRoomInfo()`
3. **Class `Guest`**
    - Attributes: `guestId`, `name`, `email`
    - **Constructors**:
        - Default constructor initializes `Unknown` values
        - Parameterized constructor initializes guest details
        - **Copy Constructor** to clone an existing `Guest`
    - Method: `displayGuestInfo()`
4. **Class `Booking`**
    - Attributes: `bookingId`, `Guest`, `Room`, `Hotel`, `nightsBooked`, `totalAmount`
    - **Constructors**:
        - Default constructor initializes empty values
        - Parameterized constructor to create a booking
    - **Constructor Chaining**: The primary constructor should call the default constructor first using `this()`.
    - Method: `calculateTotalAmount()`
5. **Singleton Class `HotelManager`**
    - A singleton class to manage hotel data.
    - **Private Constructor** to prevent multiple instances.
    - **Static Method `getInstance()`** to return the only instance.

Tasks

1. **Create a `Hotel` class** with a **parameterized constructor** to initialize hotel details.
2. **Create a `Room` class** with a **default and parameterized constructor**.
3. **Create a `Guest` class** that includes a **copy constructor**.
4. **Create a `Booking` class** using **constructor chaining** to initialize booking details.
5. **Implement a `HotelManager` Singleton Class** that prevents multiple instances and prints "Hotel Manager Initialized".
6. **Demonstrate Constructor Overloading and Chaining** by:
    - Booking a hotel room for a guest.
    - Displaying all details.
    - Cloning guest data using the **copy constructor**.