# Java Assessment 3 - OOPS

Duration: 1 Hour

Instructions:

- Read each question carefully and write the code as instructed.

- Ensure your code is readable and includes comments where necessary.

- Use proper naming conventions for classes, methods, and variables.

- Test your code thoroughly before submission.

---

## Question 1: Class and Object Creation

Create a Java class `Student` with the following attributes:   
- `id` (integer)   
- `name` (String)   
- `marks` (double)   
  
\*\*Tasks\*\*:   
1. Write a constructor to initialize these attributes.   
2. Create a method `displayDetails()` to print the student's details in a readable format.   
3. Write a main method to:   
 - Create 3 student objects with sample data.   
 - Call the `displayDetails()` method for each student.

---

## Question 2: Inheritance

Create a base class `Vehicle` with the following attributes:   
- `brand` (String)   
- `speed` (double)   
  
\*\*Tasks\*\*:   
1. Add a method `displayInfo()` in `Vehicle` to print the brand and speed.   
2. Create a subclass `Car` that inherits from `Vehicle` and adds the following attribute:   
 - `fuelType` (String)   
  
3. Add a method `displayCarDetails()` in `Car` to print all details of the car, including fuel type.   
4. Write a main method to:   
 - Create an object of `Car` with sample data.   
 - Call `displayCarDetails()` to display the details.

---

## Question 3: Polymorphism

Create a base class `Shape` with a method `calculateArea()` that returns 0 by default.   
  
\*\*Tasks\*\*:   
1. Create two subclasses:   
 - `Rectangle` (with attributes `length` and `breadth`)   
 - `Circle` (with attribute `radius`)   
  
2. Override the `calculateArea()` method in each subclass to calculate the area of the respective shapes.   
3. Write a main method to:   
 - Create objects of `Rectangle` and `Circle`.   
 - Call `calculateArea()` for each and print the results.   
  
\*Formulas\*:   
- Rectangle area = length × breadth   
- Circle area = π × radius² (Use `Math.PI` for π)

---

## Question 4: Encapsulation

Create a class `BankAccount` with the following private attributes:   
- `accountNumber` (String)   
- `accountHolderName` (String)   
- `balance` (double)   
  
\*\*Tasks\*\*:   
1. Create getter and setter methods for each attribute.   
2. Add a method `deposit()` to increase the balance.   
3. Add a method `withdraw()` to decrease the balance if sufficient funds are available. Display an appropriate message if the balance is insufficient.   
4. Write a main method to:   
 - Create a `BankAccount` object.   
 - Use setters to initialize the attributes.   
 - Perform deposit and withdrawal operations.   
 - Print the updated balance after each operation.

---

## Question 5: Abstraction

Create an abstract class `Employee` with the following attributes and methods:   
- `name` (String)   
- `id` (int)   
- `calculateSalary()` (abstract method)   
  
\*\*Tasks\*\*:   
1. Create two subclasses:   
 - `FullTimeEmployee` with attributes `basicSalary` and `bonus`.   
 - `PartTimeEmployee` with attributes `hoursWorked` and `hourlyRate`.   
  
2. Implement the `calculateSalary()` method in each subclass.   
3. Write a main method to:   
 - Create objects of `FullTimeEmployee` and `PartTimeEmployee`.   
 - Print the calculated salaries for both.

---