

# Implementing Encapsulation and Abstract Classes & Methods in Java

---

## Project Abstract

The purpose of this project is to demonstrate the use of Encapsulation and Abstract Classes and Methods in Java, which are fundamental principles in object-oriented programming (OOP). Encapsulation ensures that an object's internal state is protected from unauthorized access and modification, while abstract classes and methods provide a way to define a common structure for derived classes to implement. In this project, we will focus on how to implement encapsulation.

## Tasks Overview

### Task 1: Implement Encapsulation and Abstract Methods

Objective: Create an abstract class with encapsulated fields and abstract methods, and then create subclasses to implement the abstract methods.

Detailed Description: In this task, you will create an abstract `Animal` class that demonstrates Encapsulation and includes an abstract method `speak()`. The `Animal` class will have a private field `name` (demonstrating encapsulation), and you will provide public getter and setter methods to access and modify the `name` field. You will then create two subclasses (`Dog` and `Cat`) that extend `Animal` and provide their own implementation of the `speak()` method.

- Steps:
  1. Create a class `Animal`:
    - Define a private field `name` and provide public getter and setter methods to access and modify the `name`.
    - Define an abstract method `speak()` to be implemented by subclasses.
  2. Create a class `Dog`:
    - Inherit from the `Animal` class.
    - Implement the `speak()` method to print a message specific to the dog as `getName() + " barks."`.
  3. Create a class `Cat`:
    - Inherit from the `Animal` class.
    - Implement the `speak()` method to print a message specific to the cat as `getName() + " meows."`.

## Task 2: Demonstrate Encapsulation and Abstract Method Usage in the Main Method

Objective: Demonstrate the usage of Encapsulation and Abstract Methods in the main() method to show how encapsulated fields are accessed and abstract methods are implemented in subclasses.

Detailed Description: In this task, you will create objects of the Dog and Cat classes and use encapsulation to get and set the name. You will also demonstrate how abstract methods are implemented in subclasses.

- Steps:
4. In the main() method:
    - Create an instance of the Dog class as dog and Cat class as cat with default value as "Buddy" and "Whiskers" respectively.
    - Then call the speak() method for both to show the overridden behavior.
    - Update dog name by its setter method as "Charlie"..
    - Access the private name field using the getter method and print the name of the animal as "Dog's new name is: " + dog.getName().

### Execution Steps to Follow:

1. All actions like build, compile, running application, running test cases will be through Command Terminal.
2. To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) □ Terminal □New Terminal.
3. This editor Auto Saves the code.
4. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
5. To run your project use command:  
**sudo JAVA\_HOME=\$JAVA\_HOME /usr/share/maven/bin/mvn compile exec:java -Dexec.mainClass="com.yaksha.assignment.EncapsulationAbstractClassAssignment"**

**\*If it asks for the password, provide password : pass@word1**

6. To test your project test cases, use the command

```
sudo JAVA_HOME=$JAVA_HOME /usr/share/maven/bin/mvn test
```

**\*If it asks for the password, provide password : pass@word1**