Implementing Single Inheritance in Java

Project Abstract

The purpose of this project is to demonstrate how to implement single inheritance in Java, where a subclass inherits properties and methods from a superclass. Single inheritance allows a class to inherit the behavior of another class, making the code reusable and more maintainable.

This project focuses:

1. Understand the concept of inheritance in Java, where one class (subclass) extends another (superclass).
2. Define a subclass that inherits fields and methods from a superclass.
3. Override methods from the superclass in the subclass to provide specific behavior.

Tasks Overview

Task 1: Define a Class with Single Inheritance

Objective: Create a class that demonstrates single inheritance by extending another class.

Detailed Description: In this task, you will create a class called Animal with a field String field (species) and a method (speak). Then, you will create a subclass called Dog that inherits from Animal. The Dog class will override the speak method to provide specific behavior for dogs.

- Steps:

1. Create a class called Animal.
2. Define a field called species of type String and initialize it with a default value as "Unknown species".
3. Define a method called speak() that prints a generic sound for animals as "The animal makes a sound.".
4. Create a class called Dog that extends Animal.
5. In the Dog class, override the speak() method to print "The dog barks."

Task 2: Use the Inherited Behavior in the Main Method

Objective: Demonstrate how inheritance works by using objects of the subclass Dog in the main method.

Detailed Description: In this task, you will instantiate a Dog object in the main method. The object will inherit the species field and speak() method from the Animal class. You will

access the species field, print its value, and call the speak() method to see how method overriding works.

- Steps:

6. In the main method, create an instance of the Dog class as dog.
7. Print the species field to show that the field is inherited from the Animal class.
8. Call the speak() method to see the method overridden in the Dog class.

Note: In the `speak()` method, You need to print the message in the following format:

```
Species: <species>
<Dog-specific speak message>
```

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. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**
5. **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.**
6. **To run your project use command:**
   mvn compile exec:java
   -Dexec.mainClass="com.yaksha.assignment.SingleInheritanceAssignment"
7. **To test your project test cases, use the command**
   mvn test

8. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**