**Lab Sheet 1.1: Create a java program using Single Inheritance.**

**Aim:** The aim of this Java program is to demonstrate single inheritance by creating a subclass (Dog) that inherits properties and methods from a superclass (Animal). It showcases how a subclass can extend and add new functionalities while inheriting the existing ones from the superclass.

**Algorithm:**

1. Create a superclass called **Animal** with a method **eat()** that prints a message indicating that the animal eats food.
2. Create a subclass called **Dog** that extends the **Animal** class.
3. In the **Dog** class, add a new method called **bark()** that prints a message indicating that the dog barks.
4. In the **Main** class, create an instance of the **Dog** class named **myDog**.
5. Call the **eat()** method on the **myDog** object, which is inherited from the **Animal** class, to demonstrate the inheritance of the superclass method.
6. Call the **bark()** method on the **myDog** object, which is specific to the **Dog** class, to demonstrate the extension of functionality in the subclass.

**Program Explanation:**

In this program, we have a superclass **Animal** with a single method, **eat()**, which prints "The animal eats food." This represents the base class that the **Dog** class will inherit from.

The **Dog** class is a subclass of **Animal** and adds a new method **bark()**, which prints "The dog barks." This illustrates how a subclass can inherit features from its superclass and also introduce new behaviors.

In the **Main** class, we create an instance of the **Dog** class called **myDog**. We then call the **eat()** method on **myDog**, which is inherited from the **Animal** class, and it prints "The animal eats food." This demonstrates the concept of inheritance.

Next, we call the **bark()** method on **myDog**, which is specific to the **Dog** class, and it prints "The dog barks." This shows how the subclass extends the functionality of the superclass.

The program showcases the principle of single inheritance, where a class can inherit from only one superclass in Java.

**Top of Form**

**Program:**

class Animal {

void eat() {

System.out.println("The animal eats food.");

}

}

// Subclass inheriting from Animal

class Dog extends Animal {

void bark() {

System.out.println("The dog barks.");

}

}

public class Main {

public static void main(String[] args) {

// Create an instance of the Dog class

Dog myDog = new Dog();

// Call methods from both the superclass and subclass

myDog.eat(); // Inherited from Animal

myDog.bark(); // Defined in Dog

}

}

**OUTPUT**

The animal eats food.

The dog barks.

**Lab Sheet 1.2: Create a java program using Multilevel Inheritance.**

**Aim:** The aim of this Java program is to illustrate the concept of single inheritance and class hierarchy by creating a series of classes where each subclass inherits properties and methods from its parent class, and to demonstrate the use of methods at different levels of the class hierarchy.

**Algorithm:**

1. Create a base class named **Animal** with a method **eat()** that prints "Animal is eating."
2. Create a subclass named **Mammal** that inherits from the **Animal** class. In the **Mammal** class, add a new method **run()** that prints "Mammal is running."
3. Create another subclass named **Dog** that inherits from the **Mammal** class. In the **Dog** class, add a new method **bark()** that prints "Dog is barking."
4. In the **Main** class, create an instance of the **Dog** class named **myDog**.
5. Call the **eat()** method on **myDog**, which is inherited from the **Animal** class, to demonstrate the inheritance of the superclass method.
6. Call the **run()** method on **myDog**, which is inherited from the **Mammal** class, to demonstrate inheritance at multiple levels in the hierarchy.
7. Call the **bark()** method on **myDog**, which is specific to the **Dog** class, to illustrate the extension of functionality in the subclass.

**Program Explanation:**

In this program, we have a class hierarchy where each class is a subclass of the previous one. This hierarchy showcases the concept of single inheritance in Java.

1. The base class **Animal** has a method **eat()**, which prints "Animal is eating." This represents the common behavior of all animals.
2. The **Mammal** class is a subclass of **Animal** and introduces a new method **run()**, which prints "Mammal is running." This demonstrates how a subclass can inherit properties and methods from its superclass while adding its own characteristics.
3. The **Dog** class is a subclass of **Mammal** and introduces a new method **bark()**, which prints "Dog is barking." This further extends the functionality of the class hierarchy.

In the **Main** class, we create an instance of the **Dog** class named **myDog**. We then demonstrate the use of methods from each class in the hierarchy:

* **myDog.eat()** calls the **eat()** method from the **Animal** class, and it prints "Animal is eating," showing inheritance from the base class.
* **myDog.run()** calls the **run()** method from the **Mammal** class, and it prints "Mammal is running," demonstrating inheritance from the intermediate class in the hierarchy.
* **myDog.bark()** calls the **bark()** method specific to the **Dog** class, and it prints "Dog is barking," showcasing the extension of functionality in the subclass.

This program effectively illustrates the concept of class inheritance and hierarchy in Java.

**Program**

// The base class

class Animal {

void eat() {

System.out.println("Animal is eating");

}

}

// Subclass 1: Mammal, inherits from Animal

class Mammal extends Animal {

void run() {

System.out.println("Mammal is running");

}

}

// Subclass 2: Dog, inherits from Mammal

class Dog extends Mammal {

void bark() {

System.out.println("Dog is barking");

}

}

public class Main {

public static void main(String[] args) {

// Create an instance of Dog

Dog myDog = new Dog();

// Call methods from each class in the hierarchy

myDog.eat(); // Inherited from Animal

myDog.run(); // Inherited from Mammal

myDog.bark(); // Defined in Dog

}

**OUTPUT:**

Animal is eating

Mammal is running

Dog is barking