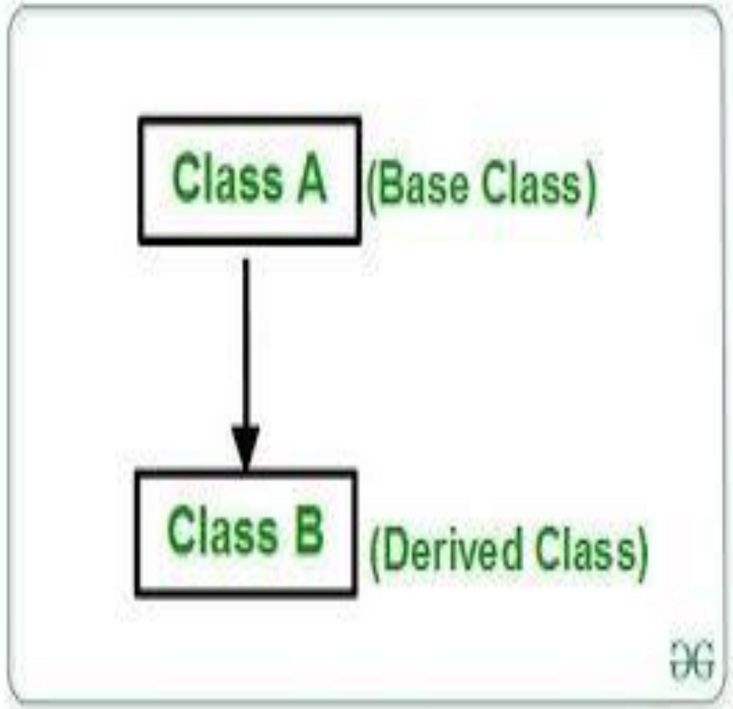
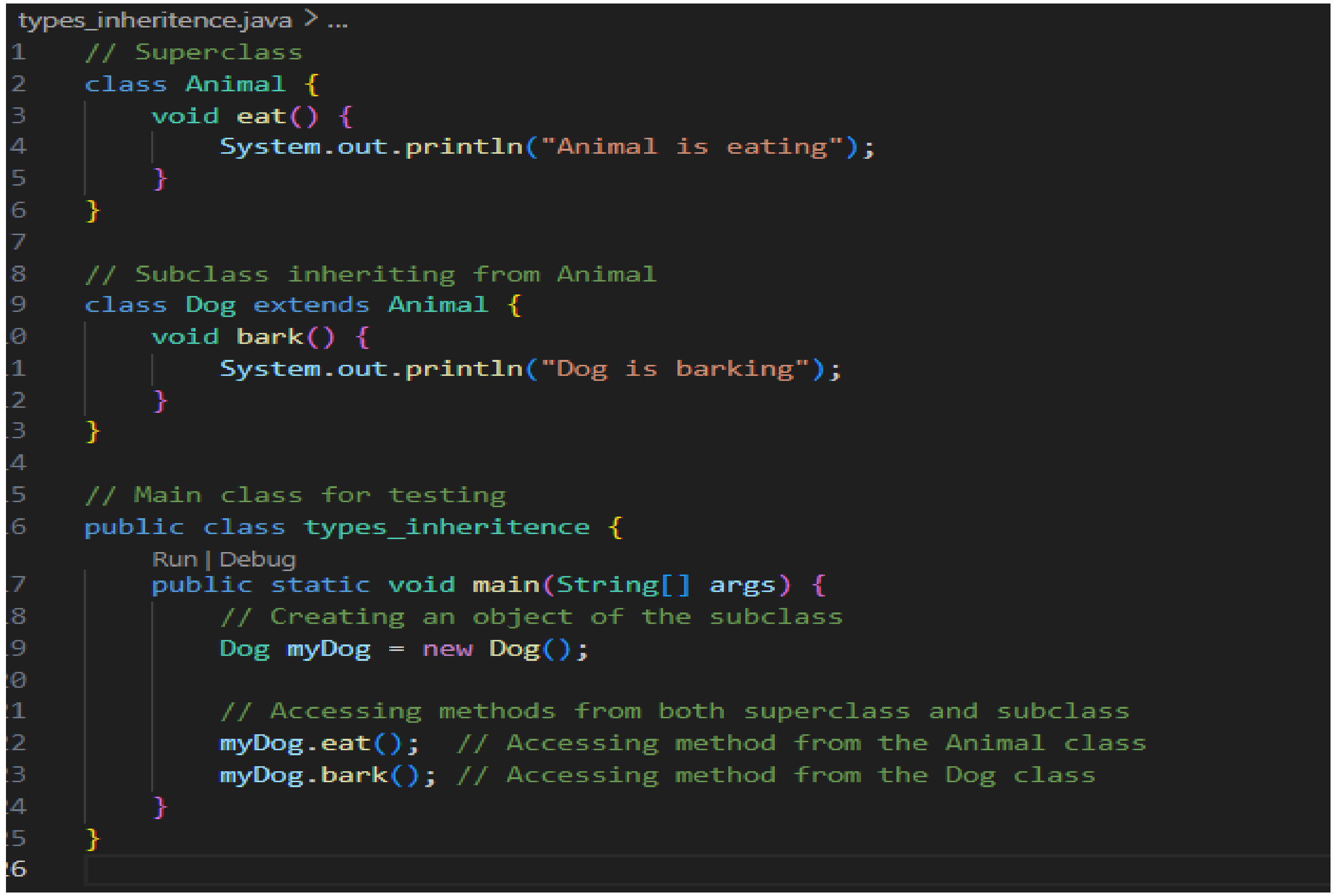
Types of Inheritance

Single Inheritance

- Single inheritance refers to a scenario where a class inherits from only one superclass. It's a simple form of inheritance, where a derived class (subclass) inherits the properties and behaviors of a single base class (superclass).

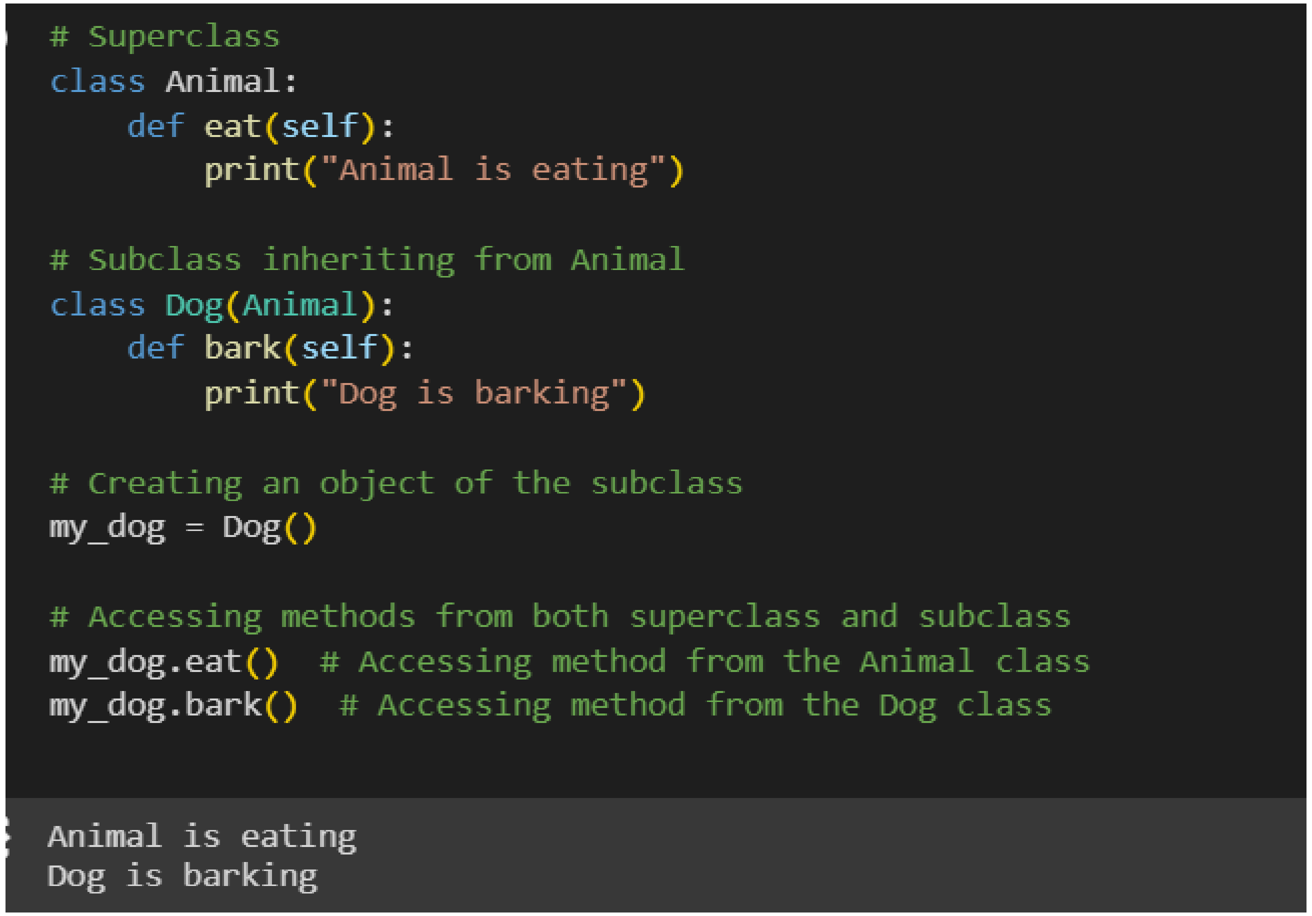

Class A
(Base Class)
Class B
(Derived Class)

In Java


types_inheritence.java > ...

```java
// Superclass
class Animal {
    void eat() {
        System.out.println("Animal is eating");
    }
}

// Subclass inheriting from Animal
class Dog extends Animal {
    void bark() {
        System.out.println("Dog is barking");
    }
}

// Main class for testing
public class types_inheritence {
    Run | Debug
    public static void main(String[] args) {
        // Creating an object of the subclass
        Dog myDog = new Dog();

        // Accessing methods from both superclass and subclass
        myDog.eat();  // Accessing method from the Animal class
        myDog.bark(); // Accessing method from the Dog class
    }
}
```

In Python

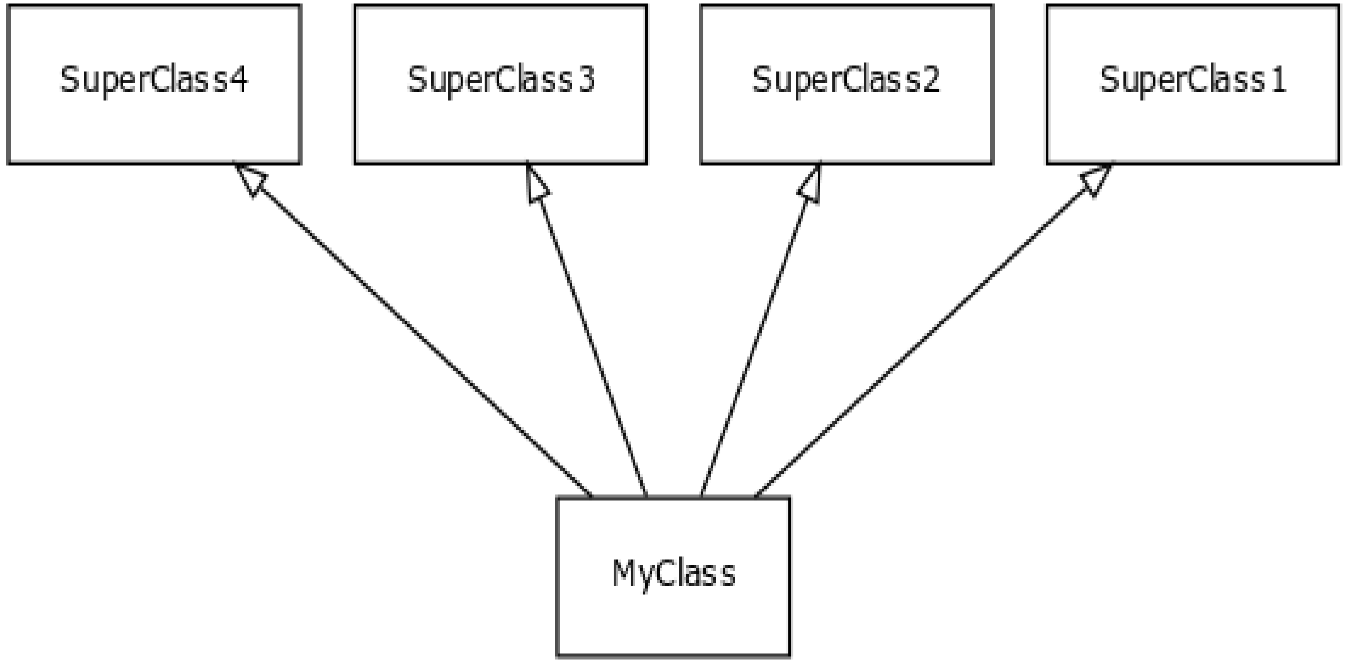


```python
# Superclass
class Animal:
    def eat(self):
        print("Animal is eating")

# Subclass inheriting from Animal
class Dog(Animal):
    def bark(self):
        print("Dog is barking")

# Creating an object of the subclass
my_dog = Dog()

# Accessing methods from both superclass and subclass
my_dog.eat()  # Accessing method from the Animal class
my_dog.bark()  # Accessing method from the Dog class
```

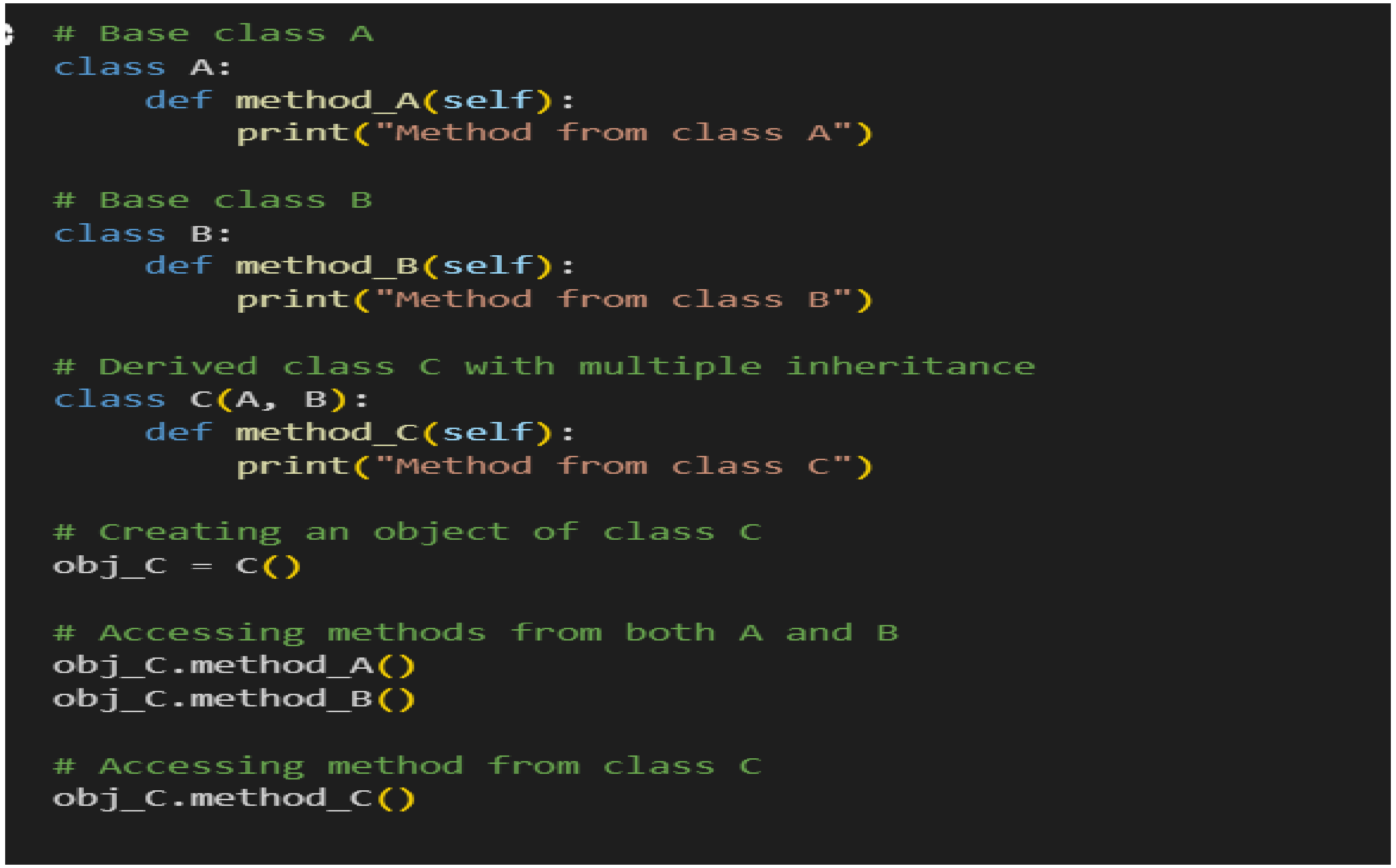
```
Animal is eating
Dog is barking
```

Multiple Inheritance

- Multiple inheritance refers to a situation in object-oriented programming where a class can inherit attributes and methods from more than one parent class. In other words, a class can be derived from more than one base classes, and it inherits features from all of them.
- Note: Java Does not support Multiple Inheritance. Why?


SuperClass4
SuperClass3
SuperClass2
SuperClass1
MyClass

In Python



```python
# Base class A
class A:
    def method_A(self):
        print("Method from class A")

# Base class B
class B:
    def method_B(self):
        print("Method from class B")

# Derived class C with multiple inheritance
class C(A, B):
    def method_C(self):
        print("Method from class C")

# Creating an object of class C
obj_C = C()

# Accessing methods from both A and B
obj_C.method_A()
obj_C.method_B()

# Accessing method from class C
obj_C.method_C()
```

Multilevel Inheritance

- Multilevel inheritance is a type of inheritance in which a derived class inherits from a base class, and then another class inherits from this derived class, forming a chain of inheritance.

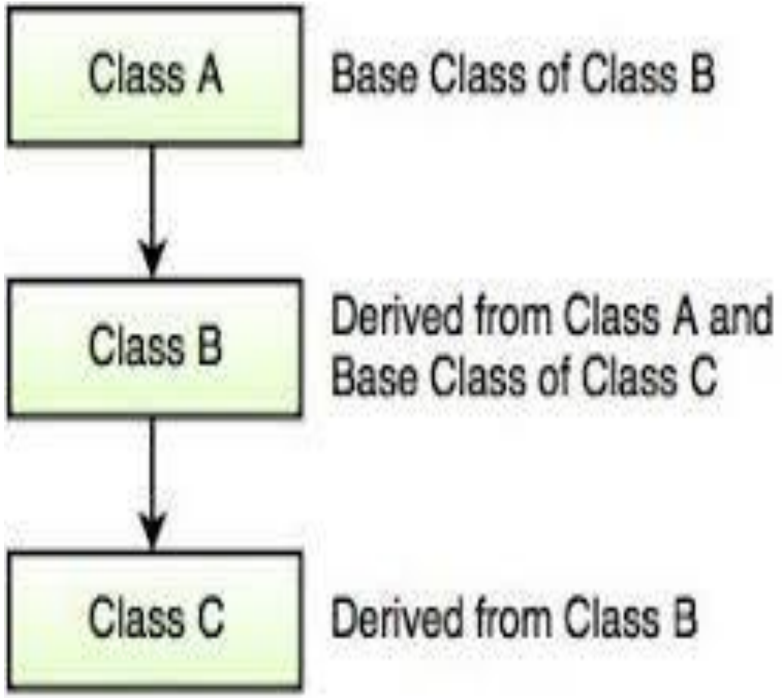
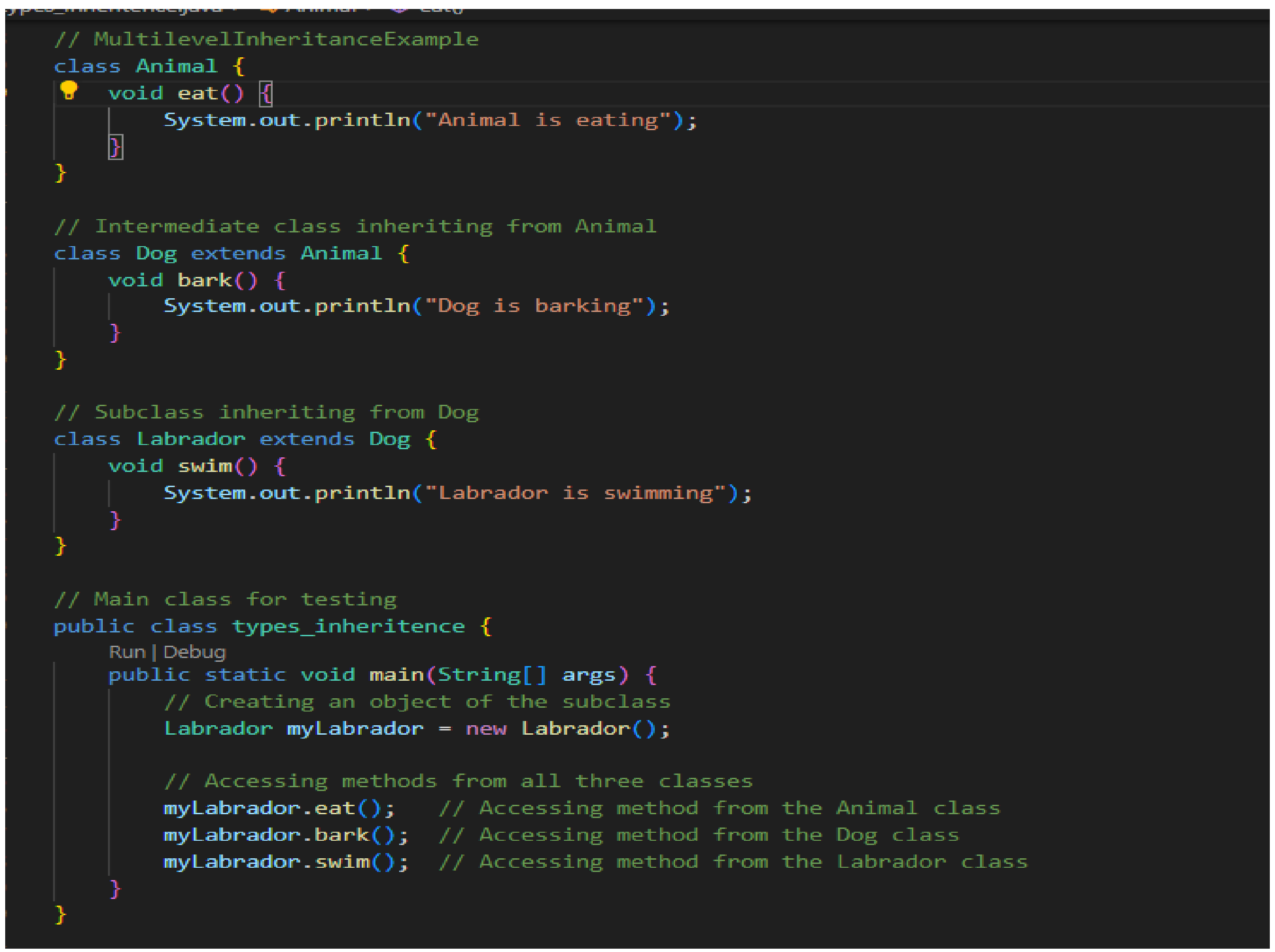


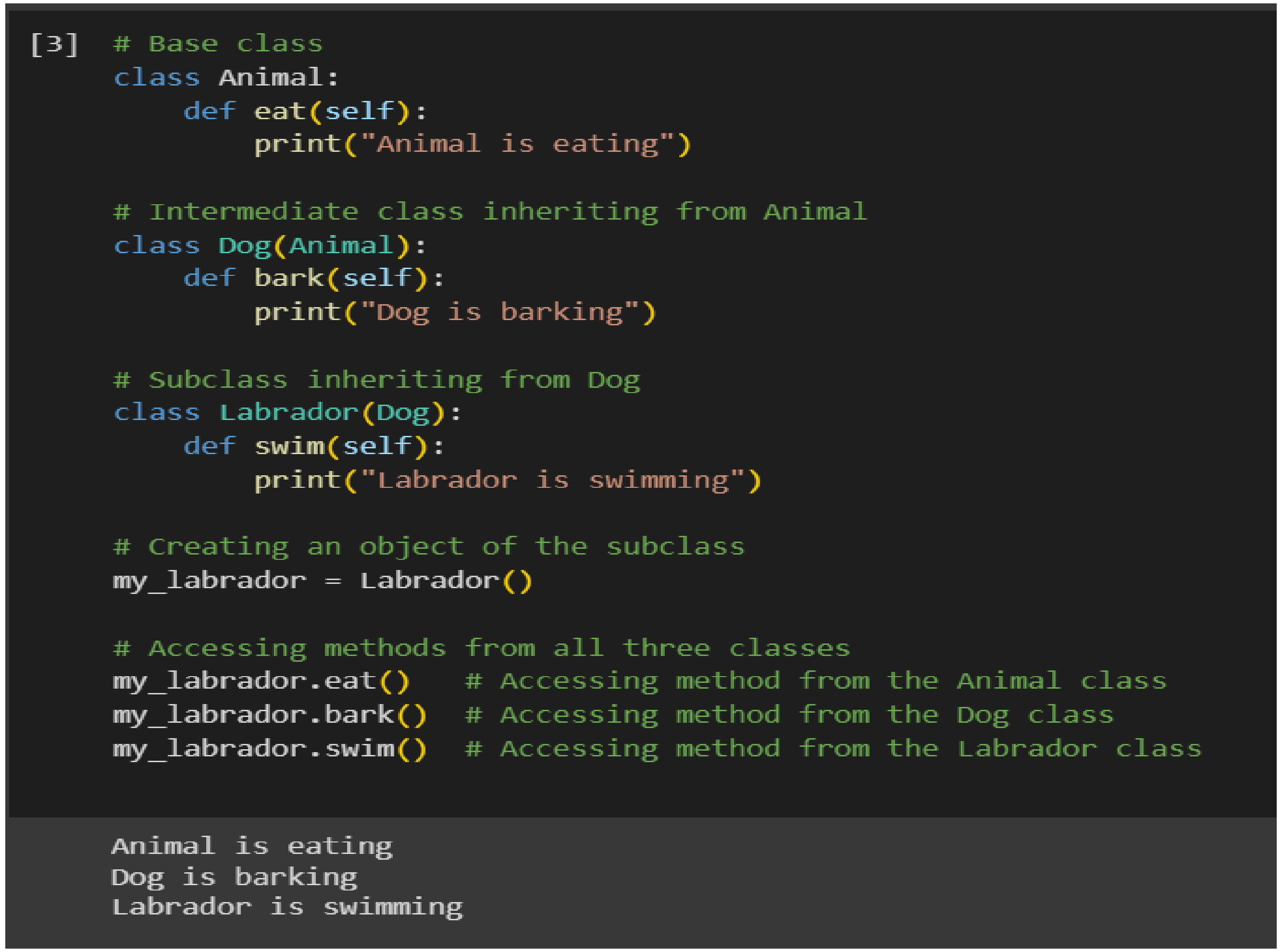
Fig. Multilevel Inheritance

In Java



```java
// MultilevelInheritanceExample
class Animal {
    void eat() {
        System.out.println("Animal is eating");
    }
}

// Intermediate class inheriting from Animal
class Dog extends Animal {
    void bark() {
        System.out.println("Dog is barking");
    }
}

// Subclass inheriting from Dog
class Labrador extends Dog {
    void swim() {
        System.out.println("Labrador is swimming");
    }
}

// Main class for testing
public class types_inheritence {
    public static void main(String[] args) {
        // Creating an object of the subclass
        Labrador myLabrador = new Labrador();

        // Accessing methods from all three classes
        myLabrador.eat();   // Accessing method from the Animal class
        myLabrador.bark();  // Accessing method from the Dog class
        myLabrador.swim();  // Accessing method from the Labrador class
    }
}
```

In Python

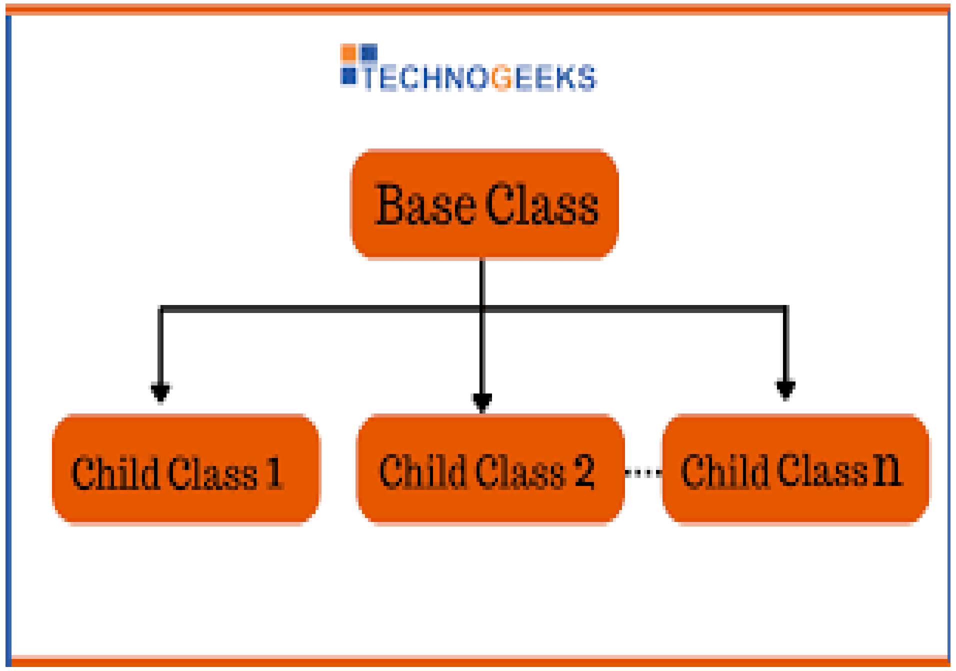


```
[3] # Base class
    class Animal:
        def eat(self):
            print("Animal is eating")

    # Intermediate class inheriting from Animal
    class Dog(Animal):
        def bark(self):
            print("Dog is barking")

    # Subclass inheriting from Dog
    class Labrador(Dog):
        def swim(self):
            print("Labrador is swimming")

    # Creating an object of the subclass
    my_labrador = Labrador()

    # Accessing methods from all three classes
    my_labrador.eat()   # Accessing method from the Animal class
    my_labrador.bark()  # Accessing method from the Dog class
    my_labrador.swim()  # Accessing method from the Labrador class
```

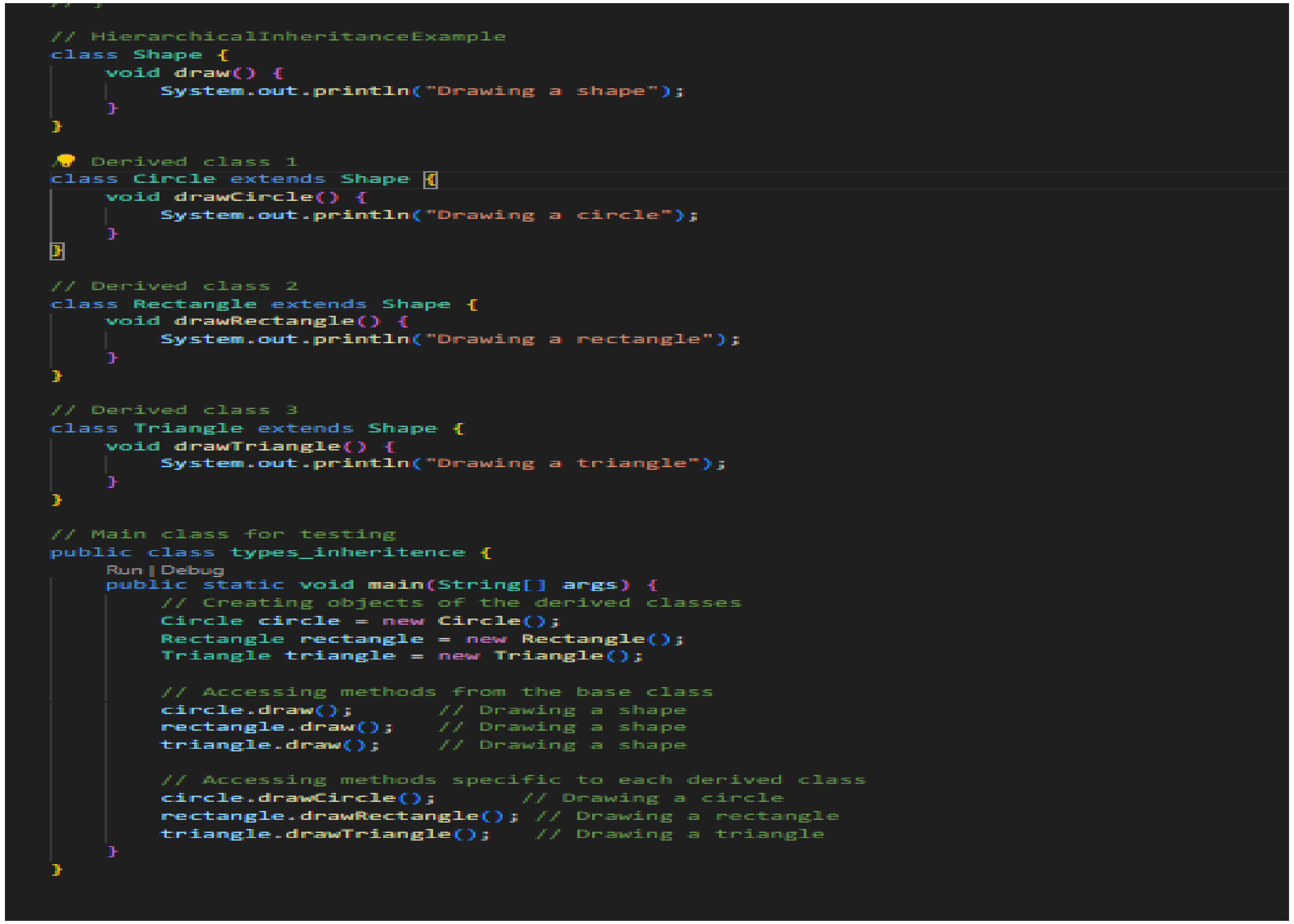
```
Animal is eating
Dog is barking
Labrador is swimming
```

Hierarchical Inheritance

- Hierarchical inheritance is a type of inheritance in which a single base class is inherited by multiple derived classes. Each derived class becomes a base class for other classes.

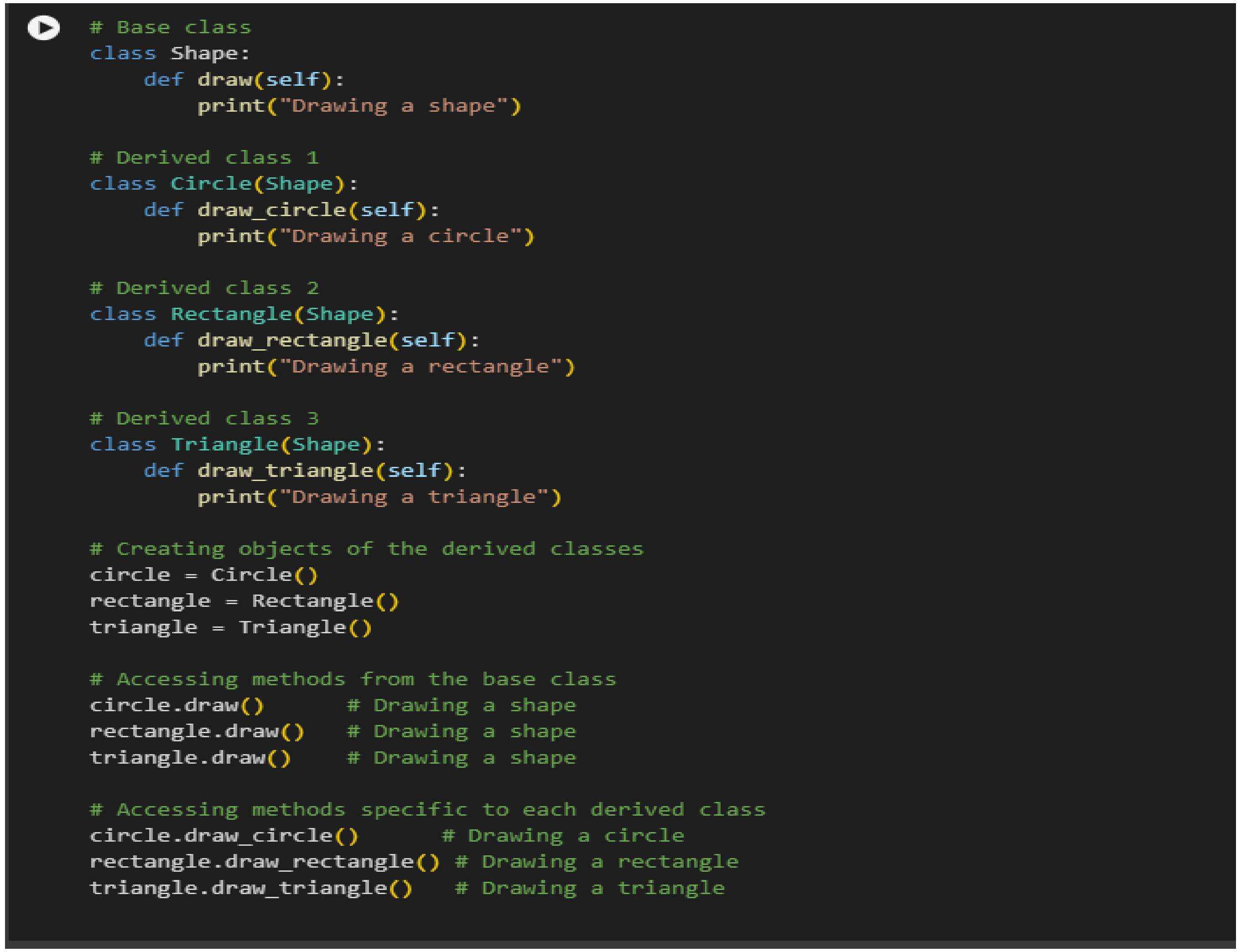

TECHNOGEEKS
Base Class
Child Class 1
Child Class 2
Child Class n

In Java



```java
// HierarchicalInheritanceExample
class Shape {
    void draw() {
        System.out.println("Drawing a shape");
    }
}

/ Derived class 1
class Circle extends Shape {
    void drawCircle() {
        System.out.println("Drawing a circle");
    }
}

// Derived class 2
class Rectangle extends Shape {
    void drawRectangle() {
        System.out.println("Drawing a rectangle");
    }
}

// Derived class 3
class Triangle extends Shape {
    void drawTriangle() {
        System.out.println("Drawing a triangle");
    }
}

// Main class for testing
public class types_inheritence {
    Run | Debug
    public static void main(String[] args) {
        // Creating objects of the derived classes
        Circle circle = new Circle();
        Rectangle rectangle = new Rectangle();
        Triangle triangle = new Triangle();

        // Accessing methods from the base class
        circle.draw();       // Drawing a shape
        rectangle.draw();    // Drawing a shape
        triangle.draw();     // Drawing a shape

        // Accessing methods specific to each derived class
        circle.drawCircle();        // Drawing a circle
        rectangle.drawRectangle(); // Drawing a rectangle
        triangle.drawTriangle();    // Drawing a triangle
    }
}
```

In Python



```python
# Base class
class Shape:
    def draw(self):
        print("Drawing a shape")

# Derived class 1
class Circle(Shape):
    def draw_circle(self):
        print("Drawing a circle")

# Derived class 2
class Rectangle(Shape):
    def draw_rectangle(self):
        print("Drawing a rectangle")

# Derived class 3
class Triangle(Shape):
    def draw_triangle(self):
        print("Drawing a triangle")

# Creating objects of the derived classes
circle = Circle()
rectangle = Rectangle()
triangle = Triangle()

# Accessing methods from the base class
circle.draw()      # Drawing a shape
rectangle.draw()   # Drawing a shape
triangle.draw()    # Drawing a shape

# Accessing methods specific to each derived class
circle.draw_circle()      # Drawing a circle
rectangle.draw_rectangle() # Drawing a rectangle
triangle.draw_triangle()   # Drawing a triangle
```

Task……!

- Hierarchical Inheritance:
  - **Task: Shape Hierarchy**
    - Create a base class Shape with a method draw().
    - Implement two derived classes, Circle and Rectangle, inheriting from Shape.
    - Create objects of both derived classes and invoke the draw() method.
  - **Task: Employee Hierarchy**
    - Create a base class Employee with attributes name and salary.
    - Implement two derived classes, Manager and Developer, inheriting from Employee.
    - Include specific attributes for each derived class.
    - Create objects of both derived classes and display their information.

- Single Inheritance:
  - **Task: Animal Hierarchy**
    - Create a base class Animal with a method makeSound().
    - Implement a derived class Dog that inherits from Animal.
    - Add a specific method bark() to the Dog class.
    - Create an object of the Dog class and call both makeSound() and bark().
- Hierarchical and Multilevel Inheritance:
  - **Task: School Hierarchy**
    - Create a base class Person with attributes name and age.
    - Implement two derived classes, Student and Teacher, inheriting from Person.
    - Further, implement a class TeachingAssistant that inherits from Teacher.
    - Create objects of each class and display relevant information.
- Multilevel Inheritance:
  - **Task: Vehicle Hierarchy**
    - Create a base class Vehicle with attributes brand and year.
    - Implement a derived class Car that inherits from Vehicle.
    - Further implement a class SportsCar that inherits from Car.
    - Create an object of the SportsCar class and display its brand, year, and any additional attributes.