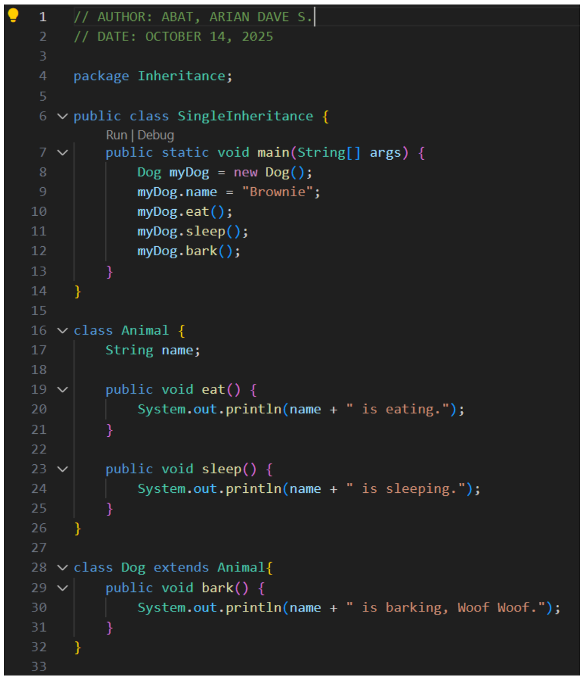
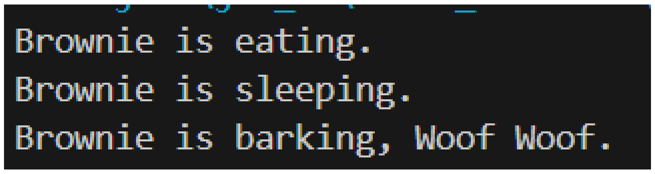
Abat, Arian Dave S. **October 14, 2025**
BSIT 2A **IT 122**

Types of Inheritance in Java

1. **Single Inheritance –** A class inherits from only one parent class, creating a simple parent-child relationship.

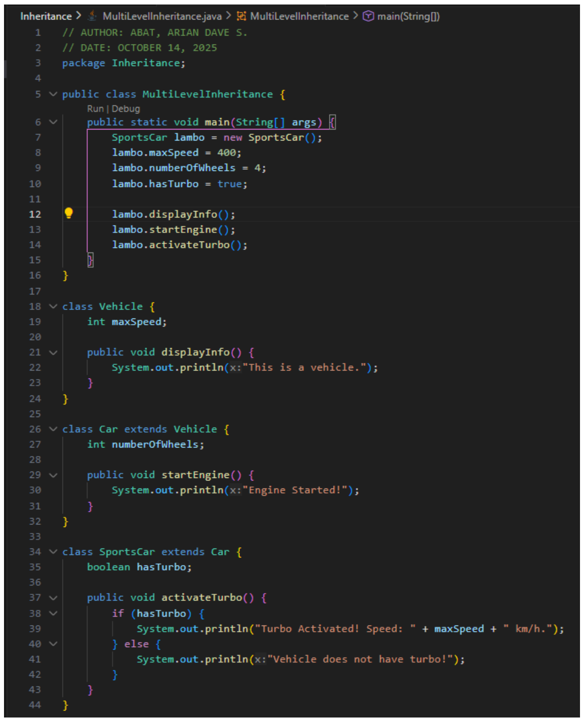


```java
// AUTHOR: ABAT, ARIAN DAVE S.
// DATE: OCTOBER 14, 2025

package Inheritance;

public class SingleInheritance {
    public static void main(String[] args) {
        Dog myDog = new Dog();
        myDog.name = "Brownie";
        myDog.eat();
        myDog.sleep();
        myDog.bark();
    }
}

class Animal {
    String name;

    public void eat() {
        System.out.println(name + " is eating.");
    }

    public void sleep() {
        System.out.println(name + " is sleeping.");
    }
}

class Dog extends Animal{
    public void bark() {
        System.out.println(name + " is barking, Woof Woof.");
    }
}

```

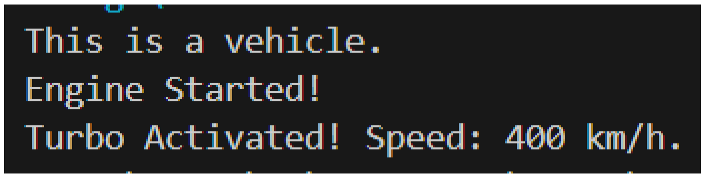


```
Brownie is eating.
Brownie is sleeping.
Brownie is barking, Woof Woof.
```

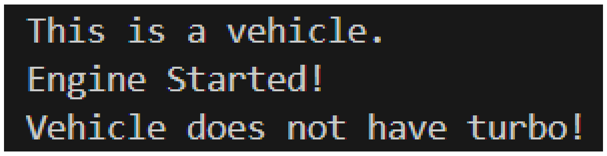
2. **Multilevel Inheritance –** A class inherits from a class that already inherits from another class, creating a chain of inheritance.


Inheritance > MultiLevelInheritance.java > MultiLevelInheritance > main(String[])
Run | Debug

```java
// AUTHOR: ABAT, ARIAN DAVE S.
// DATE: OCTOBER 14, 2025
package Inheritance;

public class MultiLevelInheritance {
    public static void main(String[] args) {
        SportsCar lambo = new SportsCar();
        lambo.maxSpeed = 400;
        lambo.numberOfWheels = 4;
        lambo.hasTurbo = true;

        lambo.displayInfo();
        lambo.startEngine();
        lambo.activateTurbo();
    }
}

class Vehicle {
    int maxSpeed;

    public void displayInfo() {
        System.out.println(x:"This is a vehicle.");
    }
}

class Car extends Vehicle {
    int numberOfWheels;

    public void startEngine() {
        System.out.println(x:"Engine Started!");
    }
}

class SportsCar extends Car {
    boolean hasTurbo;

    public void activateTurbo() {
        if (hasTurbo) {
            System.out.println("Turbo Activated! Speed: " + maxSpeed + " km/h.");
        } else {
            System.out.println(x:"Vehicle does not have turbo!");
        }
    }
}
```

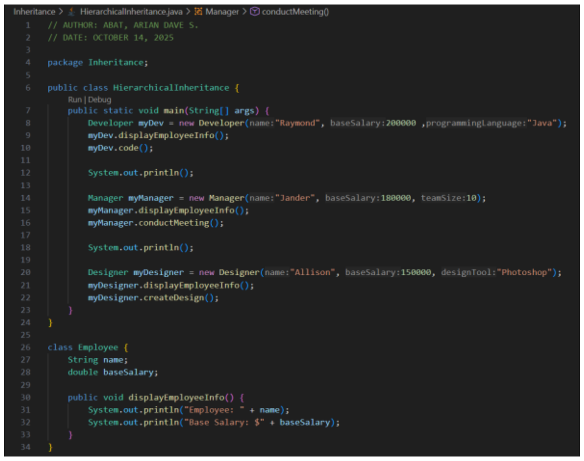


```
This is a vehicle.
Engine Started!
Turbo Activated! Speed: 400 km/h.
```

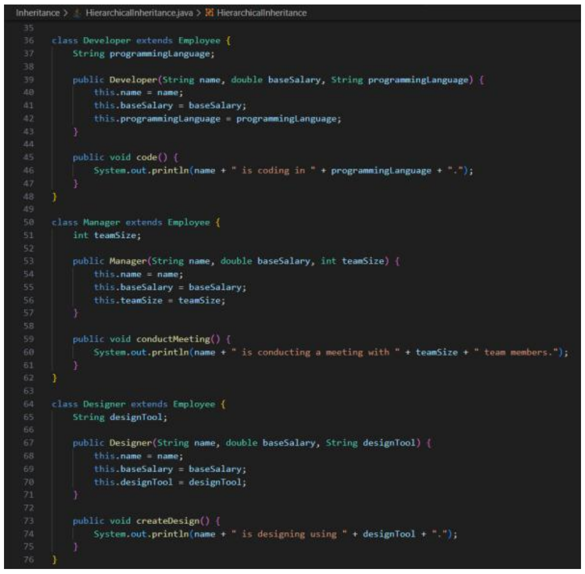


```
This is a vehicle.
Engine Started!
Vehicle does not have turbo!
```

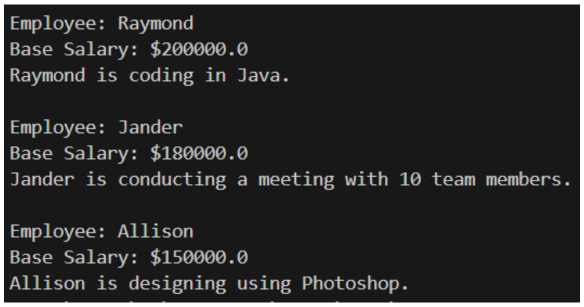
3. **Hierarchical Inheritance –** Multiple classes inherit from the same parent class, creating a tree like structure.


Inheritance > HierarchicalInheritance.java > Manager > conductMeeting()
Run | Debug

```java
// AUTHOR: ABAT, ARIAN DAVE S.
// DATE: OCTOBER 14, 2025

package Inheritance;

public class HierarchicalInheritance {
    public static void main(String[] args) {
        Developer myDev = new Developer(name:"Raymond", baseSalary:200000 ,programmingLanguage:"Java");
        myDev.displayEmployeeInfo();
        myDev.code();

        System.out.println();

        Manager myManager = new Manager(name:"Jander", baseSalary:180000, teamSize:10);
        myManager.displayEmployeeInfo();
        myManager.conductMeeting();

        System.out.println();

        Designer myDesigner = new Designer(name:"Allison", baseSalary:150000, designTool:"Photoshop");
        myDesigner.displayEmployeeInfo();
        myDesigner.createDesign();
    }
}

class Employee {
    String name;
    double baseSalary;

    public void displayEmployeeInfo() {
        System.out.println("Employee: " + name);
        System.out.println("Base Salary: $" + baseSalary);
    }
}
```

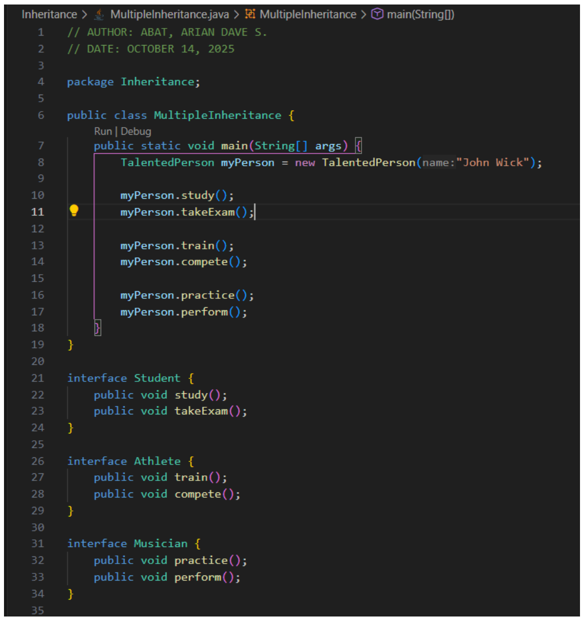

Inheritance > HierarchicalInheritance.java > HierarchicalInheritance

```java

class Developer extends Employee {
    String programmingLanguage;

    public Developer(String name, double baseSalary, String programmingLanguage) {
        this.name = name;
        this.baseSalary = baseSalary;
        this.programmingLanguage = programmingLanguage;
    }

    public void code() {
        System.out.println(name + " is coding in " + programmingLanguage + ".");
    }
}

class Manager extends Employee {
    int teamSize;

    public Manager(String name, double baseSalary, int teamSize) {
        this.name = name;
        this.baseSalary = baseSalary;
        this.teamSize = teamSize;
    }

    public void conductMeeting() {
        System.out.println(name + " is conducting a meeting with " + teamSize + " team members.");
    }
}

class Designer extends Employee {
    String designTool;

    public Designer(String name, double baseSalary, String designTool) {
        this.name = name;
        this.baseSalary = baseSalary;
        this.designTool = designTool;
    }

    public void createDesign() {
        System.out.println(name + " is designing using " + designTool + ".");
    }
}
```

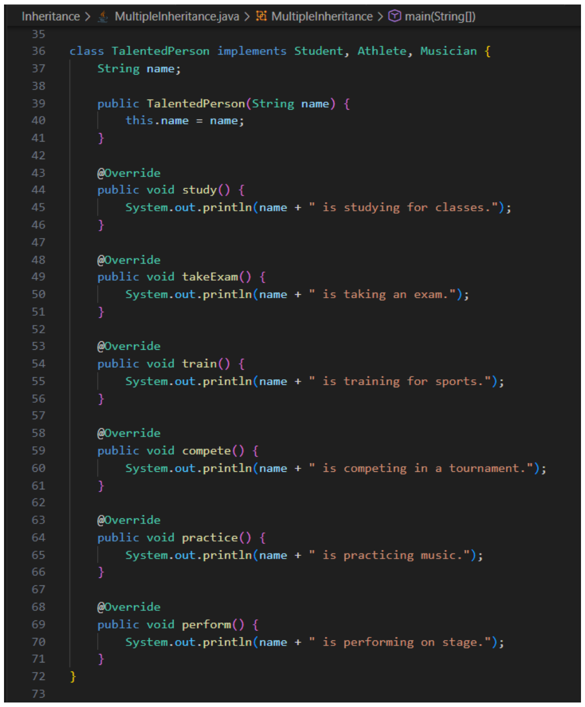


```
Employee: Raymond
Base Salary: $200000.0
Raymond is coding in Java.

Employee: Jander
Base Salary: $180000.0
Jander is conducting a meeting with 10 team members.

Employee: Allison
Base Salary: $150000.0
Allison is designing using Photoshop.
```

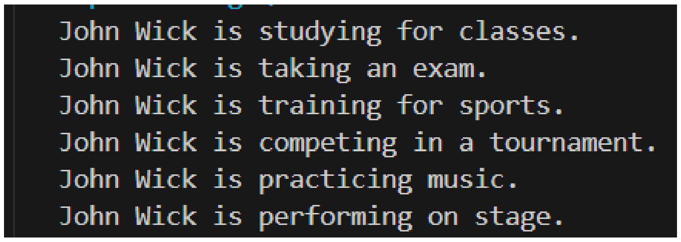
4. **Multiple Inheritance (Using Interfaces) –** A class implements multiple interfaces to inherit abstract methods from multiple sources.


Inheritance > MultipleInheritance.java > MultipleInheritance > main(String[])
Run | Debug

```java
// AUTHOR: ABAT, ARIAN DAVE S.
// DATE: OCTOBER 14, 2025

package Inheritance;

public class MultipleInheritance {
    public static void main(String[] args) {
        TalentedPerson myPerson = new TalentedPerson(name:"John Wick");

        myPerson.study();
        myPerson.takeExam();

        myPerson.train();
        myPerson.compete();

        myPerson.practice();
        myPerson.perform();
    }
}

interface Student {
    public void study();
    public void takeExam();
}

interface Athlete {
    public void train();
    public void compete();
}

interface Musician {
    public void practice();
    public void perform();
}

```

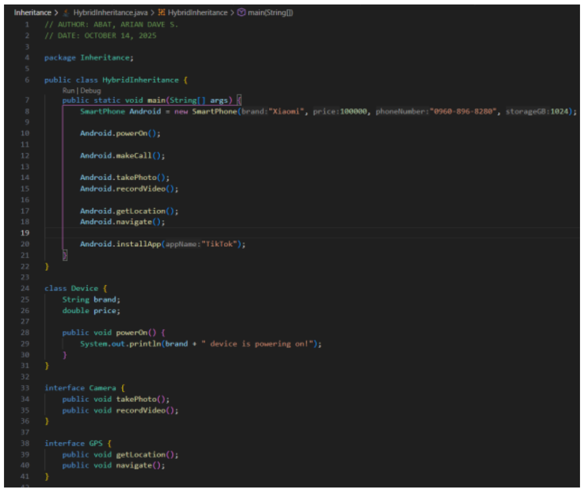

Inheritance > MultipleInheritance.java > MultipleInheritance > main(String[])

```java

class TalentedPerson implements Student, Athlete, Musician {
    String name;

    public TalentedPerson(String name) {
        this.name = name;
    }

    @Override
    public void study() {
        System.out.println(name + " is studying for classes.");
    }

    @Override
    public void takeExam() {
        System.out.println(name + " is taking an exam.");
    }

    @Override
    public void train() {
        System.out.println(name + " is training for sports.");
    }

    @Override
    public void compete() {
        System.out.println(name + " is competing in a tournament.");
    }

    @Override
    public void practice() {
        System.out.println(name + " is practicing music.");
    }

    @Override
    public void perform() {
        System.out.println(name + " is performing on stage.");
    }
}

```

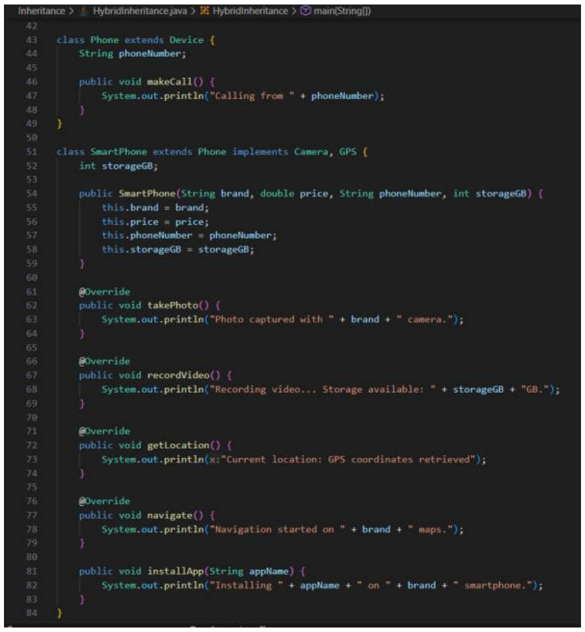


```
John Wick is studying for classes.
John Wick is taking an exam.
John Wick is training for sports.
John Wick is competing in a tournament.
John Wick is practicing music.
John Wick is performing on stage.
```

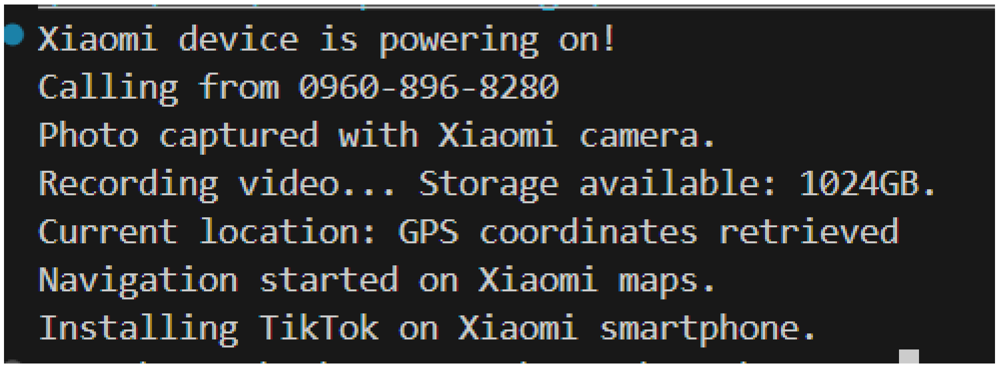
5. **Hybrid Inheritance –** A combination of two or more types of inheritance using both classes and interfaces to create complex relationships.


Inheritance > HybridInheritance.java > HybridInheritance > main(String[])
Run | Debug

```java
// AUTHOR: ABAT, ARIAN DAVE S.
// DATE: OCTOBER 14, 2025

package Inheritance;

public class HybridInheritance {
    public static void main(String[] args) {
        SmartPhone Android = new SmartPhone(brand:"Xiaomi", price:100000, phoneNumber:"0960-896-8280", storageGB:1024);

        Android.powerOn();

        Android.makeCall();

        Android.takePhoto();
        Android.recordVideo();

        Android.getLocation();
        Android.navigate();

        Android.installApp(appName:"TikTok");
    }
}

class Device {
    String brand;
    double price;

    public void powerOn() {
        System.out.println(brand + " device is powering on!");
    }
}

interface Camera {
    public void takePhoto();
    public void recordVideo();
}

interface GPS {
    public void getLocation();
    public void navigate();
}
```


Inheritance > HybridInheritance.java > HybridInheritance > main(String[])

```java

class Phone extends Device {
    String phoneNumber;

    public void makeCall() {
        System.out.println("Calling from " + phoneNumber);
    }
}

class SmartPhone extends Phone implements Camera, GPS {
    int storageGB;

    public SmartPhone(String brand, double price, String phoneNumber, int storageGB) {
        this.brand = brand;
        this.price = price;
        this.phoneNumber = phoneNumber;
        this.storageGB = storageGB;
    }

    @Override
    public void takePhoto() {
        System.out.println("Photo captured with " + brand + " camera.");
    }

    @Override
    public void recordVideo() {
        System.out.println("Recording video... Storage available: " + storageGB + "GB.");
    }

    @Override
    public void getLocation() {
        System.out.println(x:"Current location: GPS coordinates retrieved");
    }

    @Override
    public void navigate() {
        System.out.println("Navigation started on " + brand + " maps.");
    }

    public void installApp(String appName) {
        System.out.println("Installing " + appName + " on " + brand + " smartphone.");
    }
}
```



```
Xiaomi device is powering on!
Calling from 0960-896-8280
Photo captured with Xiaomi camera.
Recording video... Storage available: 1024GB.
Current location: GPS coordinates retrieved
Navigation started on Xiaomi maps.
Installing TikTok on Xiaomi smartphone.
```

Reference:

GeeksforGeeks. (2025, October 9). *Inheritance in java*. GeeksforGeeks. https://www.geeksforgeeks.org/java/inheritance-in-java/