Summary of How the Code Works

This code represents a simple transportation simulation involving vehicles, including electric and non-electric cars and trucks. It demonstrates key Object-Oriented Programming (OOP) concepts like inheritance, abstract classes, interfaces, and polymorphism. Here's a breakdown of how it works:

1. Interfaces (Polymorphism)

- Drivable Interface:
  - The Drivable interface defines a method drive(), which all implementing classes must provide.
  - This ensures that any class that implements Drivable will have the ability to "drive," though the implementation of how it "drives" can vary.
- Polymorphism:
  - Classes like Vehicle, Car, and Truck implement the Drivable interface, meaning they each have their own version of the drive() method.
  - For example, Car's drive() method specifies that it drives with passengers, while Truck's version involves carrying cargo.
  - This demonstrates polymorphism, as the same method drive() behaves differently based on the type of vehicle.

2. Abstract Classes

- ElectricVehicle (Abstract Class):
  - ElectricVehicle is an abstract class, meaning it can't be instantiated on its own and must be subclassed.
  - It contains one abstract method, charge(), which forces its subclasses to provide an implementation for charging electric vehicles.
  - It also has a concrete method getBatteryLevel() that can be used directly by subclasses. This method returns the battery percentage for electric vehicles.
- Subclasses of ElectricVehicle:
  - ElectricCar and ElectricTruck inherit from ElectricVehicle and provide their own implementations of charge(), where they set the battery level to 100% to simulate charging.

3. Inheritance

- Vehicle Base Class:

    - Vehicle is the parent class that contains properties common to all vehicles, like make, model, and year. It also defines a getInfo() method to return basic information about the vehicle and a drive() method from the Drivable interface.
- Car and Truck Subclasses:
    - Car inherits from Vehicle and adds a passengers attribute. It also overrides the getInfo() and drive() methods to reflect that a car carries passengers.
    - Truck inherits from Vehicle and adds a cargo_capacity attribute. It also overrides getInfo() and drive() to indicate the truck's cargo capacity.
- Multiple Inheritance (ElectricCar and ElectricTruck):
    - ElectricCar and ElectricTruck inherit from both ElectricVehicle (for handling the battery level and charging) and Vehicle (for basic vehicle information and drive functionality).
    - These classes demonstrate multiple inheritance, where a class inherits from two parent classes (ElectricVehicle and Vehicle) and must implement methods from both.

4. Driving Simulation

- In the test_simulation() function, different vehicle objects are created:
    - vehicle (generic vehicle),
    - car (regular car with passengers),
    - truck (regular truck with cargo),
    - electric_car (electric car with battery and passengers),
    - electric_truck (electric truck with battery and cargo).
- For each vehicle:
    - The getInfo() method is called to print the vehicle's details.
    - The drive() method is called to simulate driving the vehicle, which outputs a different message based on the vehicle type.
    - For electric vehicles, the getBatteryLevel() and charge() methods are used to manage battery levels.

Key Concepts Demonstrated:

- Inheritance: Car and Truck inherit from Vehicle, while ElectricCar and ElectricTruck inherit from both ElectricVehicle and Vehicle.
- Abstract Classes: ElectricVehicle is an abstract class with both abstract and concrete methods.
- Interfaces and Polymorphism: The Drivable interface ensures that every vehicle type can "drive," but how it drives is customized by each class.
- Multiple Inheritance: ElectricCar and ElectricTruck inherit from both a base vehicle class (Vehicle) and an abstract class (ElectricVehicle).