```
'''Q-1) Vehicle – example hybrid inheritance
1. Vehicle (Base Class): Represents a general vehicle with basic attributes like
make, model, and year.
2. Car (Derived from Vehicle): Represents cars, which have additional
features like the number of doors and trunk capacity.
3. Truck (Derived from Vehicle): Represents trucks, which have attributes
like cargo capacity and number of axles.
4. PickupTruck (Derived from both Car and Truck): A specific type of vehicle
that combines features of both cars (passenger-friendly) and trucks (cargo-friendly). Method – display all the features.'''
class Vehicle:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year
    def display_vehicle_info(self):
        return f"Make: {self.make},\n Model: {self.model}, \nYear: {self.year}"
class Car(Vehicle):
    def __init__(self, make, model, year, num_doors, trunk_capacity):
        Vehicle.__init__(self, make, model, year)
        self.num_doors = num_doors
        self.trunk_capacity = trunk_capacity
    def display_car_info(self):
```

```
        return f"{self.display_vehicle_info()},\n Number of
Doors: {self.num_doors}, \nTrunk Capacity:
{self.trunk_capacity} liters"
class Truck(Vehicle):
    def __init__(self, make, model, year, cargo_capacity,
num_axles):
        super().__init__(make, model, year)
        self.cargo_capacity = cargo_capacity
        self.num_axles = num_axles
    def display_truck_info(self):
        return f"{self.display_vehicle_info()},\n Cargo
Capacity: {self.cargo_capacity} tons,\n Number of Axles:
{self.num_axles}"
class PickupTruck(Car, Truck):
    def __init__(self, make, model, year, num_doors,
trunk_capacity, cargo_capacity, num_axles):
        Car.__init__(self, make, model, year, num_doors,
trunk_capacity)
        Truck.__init__(self, make, model, year,
cargo_capacity, num_axles)
    def display_all_features(self):
        return (
            f"{self.display_vehicle_info()},\n Number of Doors:
{self.num_doors}, "f"\nTrunk Capacity:
{self.trunk_capacity} liters,\nCargo Capacity:
{self.cargo_capacity} tons, "f"\nNumber of Axles:
{self.num_axles}")
pickup = PickupTruck("Toyota", "Hilux", 2023, 4, 500, 1.5, 2)
print(pickup.display_all_features())
```

```
'''Q-2) Inventory Management System [Hierarchical
inheritance]
1. Product (Base Class): Defines common attributes like
product ID, name,
and price. Method to display all the info.
2. Electronics (Derived Class): Inherits from Product and
adds attributes
like warranty period and brand. Method to display all the
info.
3. Clothing (Derived Class): Inherits from Product and adds
attributes like
size and material. Method to display all the info'''
class Product:
    def __init__(self, product_id, name, price):
        self.product_id = product_id
        self.name = name
        self.price = price
    def display_info(self):
        print(f"Product ID: {self.product_id}, Name:
{self.name}, Price: Rs.{self.price:.2f}")
class Electronics(Product):
    def __init__(self, product_id, name, price,
warranty_period, brand):
        super().__init__(product_id, name, price)
        self.warranty_period = warranty_period
        self.brand = brand
    def display_info(self):
        print (f"{super().display_info()}, Warranty Period:
{self.warranty_period} years, "f"Brand: {self.brand}")
class Clothing(Product):
```

```
    def __init__(self, product_id, name, price, size,
material):
        super().__init__(product_id, name, price)
        self.size = size
        self.material = material
    def display_info(self):
         print(f"{super().display_info()}, Size: {self.size},
Material: {self.material}")
laptop = Electronics("E001", "Laptop", 30000.00, 2, "ASUS")
shirt = Clothing("C001", "T-Shirt", 1000.00, "L", "Cotton")
print(laptop.display_info())
print(shirt.display_info())
```