Composition

🎯 **Direct Challenges**

1. Create a class Engine and use it in class Car.

```
class Engine {
    void start() {
        System.out.println("Engine started.");
    }
}

class Car {
    Engine engine;

    Car() {
        engine = new Engine();
    }

    void startCar() {
        engine.start();
        System.out.println("Car is ready to go!");
    }
}

public class Main {
    public static void main(String[] args) {
        Car c = new Car();
        c.startCar();
    }
}
```

Output:

```
Engine started.
Car is ready to go!
```

2. Use composition to build a Computer with Processor, RAM, and HardDrive objects.

```
class Processor {
    String brand = "Intel";
}

class RAM {
    int size = 16; // in GB
}

class HardDrive {
    int capacity = 512; // in GB
}

class Computer {
    Processor processor = new Processor();
    RAM ram = new RAM();
    HardDrive hdd = new HardDrive();

    void showSpecs() {
        System.out.println("Processor: " + processor.brand);
        System.out.println("RAM: " + ram.size + "GB");
        System.out.println("Hard Drive: " + hdd.capacity + "GB");
    }
}

public class Main {
    public static void main(String[] args) {
        Computer pc = new Computer();
        pc.showSpecs();
    }
}
```

Output:

```
Processor: Intel
RAM: 16GB
Hard Drive: 512GB
```

3. Demonstrate "has-a" relationship using class Library with Book objects.

```
class Book {
    String title;

    Book(String title) {
        this.title = title;
    }
}

class Library {
    Book[] books;

    Library() {
        books = new Book[] {
            new Book("Java Programming"),
            new Book("Data Structures"),
            new Book("OOP Concepts")
        };
    }

    void displayBooks() {
        for (Book b : books) {
            System.out.println("Book: " + b.title);
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Library lib = new Library();
        lib.displayBooks();
    }
}
```

Output:

```
Book: Java Programming
Book: Data Structures
Book: OOP Concepts
```

🧩 Scenario-Based Challenges

1. Model a Student having an Address and IDCard as composed objects.

```
class Address {
    String city = "Hyderabad";
    String pin = "500001";
}

class IDCard {
    String idNumber = "S12345";
}

class Student {
    String name = "Sreevani";
    Address address = new Address();
    IDCard idCard = new IDCard();

    void showDetails() {
        System.out.println("Name: " + name);
        System.out.println("City: " + address.city);
        System.out.println("PIN: " + address.pin);
        System.out.println("ID Number: " + idCard.idNumber);
    }
}

public class Main {
    public static void main(String[] args) {
        Student s = new Student();
        s.showDetails();
    }
}
```

Output:

```
Name: Sreevani

City: Hyderabad

PIN: 500001

ID Number: S12345
```

2. Create a House class that has Room and Kitchen as components.

```java
class Room {
    int number = 2;
}

class Kitchen {
    boolean modular = true;
}

class House {
    Room room = new Room();
    Kitchen kitchen = new Kitchen();

    void showHouseDetails() {
        System.out.println("Rooms: " + room.number);
        System.out.println("Modular Kitchen: " + (kitchen.modular ? "Yes" : "No"));
    }
}

public class Main {
    public static void main(String[] args) {
        House h = new House();
        h.showHouseDetails();
    }
}
```

Output:

```
Rooms: 2
Modular Kitchen: Yes
```