## 1. Association

**Definition:**  
Association is a general relationship between two classes where one class uses another. They are aware of each other, but **neither "owns" the other**.

* It can be **one-to-one**, **one-to-many**, **many-to-one**, or **many-to-many**.
* Objects have independent lifecycles.

**Example:**

1. class Driver {

2. name: string;

3. constructor(name: string) {

4. this.name = name;

5. }

6. }

7.

8. class Car {

9. model: string;

10. constructor(model: string) {

11. this.model = model;

12. }

13.

14. assignDriver(driver: Driver): void {

15. console.log(`${driver.name} is assigned to drive the ${this.model}`);

16. }

17. }

18.

19. const driver = new Driver("Alice");

20. const car = new Car("Toyota");

21. car.assignDriver(driver);

22.

* Driver and Car are related via **association**.
* They can exist independently.

## 2. Aggregation

**Definition:**  
Aggregation is a **special type of association** where one class contains another class, but the **contained object can exist independently** of the container.

* Think of it as a **"has-a"** relationship.
* The whole and part can exist separately.

**Example:**

1. class Department {

2. name: string;

3. constructor(name: string) {

4. this.name = name;

5. }

6. }

7.

8. class University {

9. name: string;

10. departments: Department[];

11.

12. constructor(name: string) {

13. this.name = name;

14. this.departments = [];

15. }

16.

17. addDepartment(dept: Department): void {

18. this.departments.push(dept);

19. }

20. }

21.

22. const cs = new Department("Computer Science");

23. const maths = new Department("Mathematics");

24.

25. const uni = new University("Tech University");

26. uni.addDepartment(cs);

27. uni.addDepartment(maths);

28.

* University has departments, but Department can exist without university.
* That’s **aggregation**.

## 3. Composition

**Definition:**  
Composition is a **stronger form of aggregation**. If the container is destroyed, **its parts are also destroyed**.

* Lifecycles are tied.
* It also represents a **"has-a"** relationship, but with ownership.

**Example:**

1. class Engine {

2. start(): void {

3. console.log("Engine started.");

4. }

5. }

6.

7. class Vehicle {

8. private engine: Engine;

9.

10. constructor() {

11. this.engine = new Engine(); // tightly bound

12. }

13.

14. startVehicle(): void {

15. this.engine.start();

16. }

17. }

18.

19. const vehicle = new Vehicle();

20. vehicle.startVehicle();

21.

* Vehicle owns the Engine completely.
* If Vehicle is destroyed, so is the Engine.
* That’s **a composition**.

### Summary Table:

| **Concept** | **Type** | **Ownership** | **Lifecycle Dependency** | **Example Relationship** |
| --- | --- | --- | --- | --- |
| Association | Loose | ❌ | Independent | Driver ↔ Car |
| Aggregation | Weak | ✅ (shared) | Independent | University → Dept |
| Composition | Strong | ✅ (exclusive) | Dependent | Vehicle → Engine |