Heshawa Cooray

Part A: Conceptual Questions

Composition vs. Aggregation

- **Composition**: A strong "has-a" relationship where one object owns another. If the parent object is destroyed, the child object is also destroyed.
    - Example: A car and its engine. If the car is scrapped, the engine is gone too.
- **Aggregation**: A weaker "has-a" relationship where one object references another, but they can exist independently.
    - Example: A university and students. If the university shuts down, the students still exist.
- **Which is stronger?**
    - Composition is stronger because the child object cannot exist without the parent.

When to Use

- **Composition over inheritance**:
    - In a game, a Player has a Weapon. A weapon isn't a type of player, so composition is better than inheritance (Player contains a Weapon rather than inheriting from it).
    - Why?: It allows flexibility, players can switch weapons without affecting their identity.
- **Aggregation example**:
    - A Team and Players. A player can belong to multiple teams, and if one team disbands, the player is still available.

Differences from Inheritance

- **Inheritance** (is-a): A subclass inherits properties/methods from a parent class (e.g., Dog is a Animal).
- **Composition/Aggregation** (has-a): One class contains another class as a part (e.g., Car has an Engine).

- **Why favor composition over inheritance?**
    - It reduces tight coupling and makes code more reusable. Inheritance can lead to deep hierarchies, which are harder to maintain.

Real-World Analogy

- **Car System**:
    - **Composition**: A car has-a engine (without the car, the engine has no purpose).
    - **Aggregation**: A car has-a driver (but the driver exists independently).
    - Why does it matter?
        - Helps decide object lifecycles and dependencies in code.

Part B: Minimal Class Design

```
class Address {
public:
  string street, city;
  Address(string s, string c) : street(s), city(c) {}
};

class Person {
private:
  Address address; //Person owns the Address
public:
  Person(string s, string c) : address(s, c) {}
};
```

If a Person object is destroyed, the Address is destroyed too.

Part C: Reflection & Discussion

Ownership & Lifecycle

- **Composition**: If the parent is deleted, the child is also deleted.
- **Aggregation**: The child can still exist independently after the parent is deleted.

Advantages & Pitfalls

- **Advantage of composition**: Ensures objects that should always be together don't get accidentally separated.
- **Pitfall of wrong composition**: If you use it where aggregation makes more sense, you might force unnecessary dependencies.

  Example making a Book own an Author when an author can exist independently.

Contrast with Inheritance

- **"Has-a" vs. "Is-a"**:
  - **Inheritance** is used when an object is a specialized type of another (Dog is an Animal).
  - **Composition/Aggregation** is used when an object is a part of another (Car has an Engine).
- **Why avoid inheritance sometimes?**
  - Avoids deep class hierarchies, which can be hard to maintain.
  - More flexible, you can swap out components easily.