CS-1201 Object Oriented Programming

Aggregation and Composition

Arbish Akram

Department of Computer Science
Government College University

Association

- Association represents a relationship between two classes, where one class uses or interacts with another.
- Each class can exist independently of the other.
- Association is a generalized concept of relationships, encompassing both Composition and Aggregation.
- Aggregation is a specific type of association. It represents a directional relationship where one object 'has-a' another object.
- In aggregation, each object has its own lifecycle, but there is ownership, similar to a parent-child relationship. A child object cannot belong to multiple parent objects simultaneously.
- For example, a `Player` can exist independently of a `Team` and can be part of different teams over time.

Composition and Aggregation

Composition implies ownership

- if the university disappears then all of its departments disappear
- a university is a composition of departments

Aggregation does not imply ownership

- if a department disappears then the instructors do not disappear
- a department is an aggregation of professors

Aggregation

```
1  class Engine {
2      public:
3          void start() {
4              cout << "Engine started." << endl;
5          }
6  };
7  class Car {
8      private:
9          Engine engine; // Aggregation: Car "has-a" Engine,
10         // but the Engine can still be created separately
11     public:
12         // Initialize the car with an existing Engine
13         Car(const Engine& eng) : engine(eng) {}
14         void startCar() {
15             engine.start(); // Use the engine to start the car
16             cout << "Car is running." << endl;
17         }
18 };
19 int main() {
20     Engine engine; // Create an Engine object independently
21     Car car(engine); // Create a Car and associate it with the existing Engine
22     car.startCar(); // Start the car using the Engine
23     return 0;
24 }
```

Composition

- Represents a "whole-part" relationship where the part cannot exist without the whole.
- If the whole is destroyed, the parts are also destroyed.

Composition

```
1  class Processor {
2      public:
3          void process() {
4              cout << "Processing data." << endl;
5          }
6  };
7  class Computer {
8      private:
9          Processor processor; // Composition: Computer "has-a" Processor,
10         // and the Processor cannot exist without the Computer
11     public:
12         Computer() {} // Constructor
13         void startProcessing() {
14             processor.process(); // Use the processor to perform a task
15             cout << "Computer is processing data." << endl;
16         }
17 };
18 int main() {
19     Computer computer; // Create a Computer object
20     computer.startProcessing(); // Start the processing using the Computer's
21     //Processor
22     return 0;
23 }
```

Summary

- Association: A relationship where classes use each other.
- Composition: A strong relationship where the contained objects cannot exist independently of the container.
- Aggregation: A weaker relationship where the contained objects can exist independently of the container.