Object Oriented Programming (JAVA)



## Semester: Fall 2024

**Software Engineering**

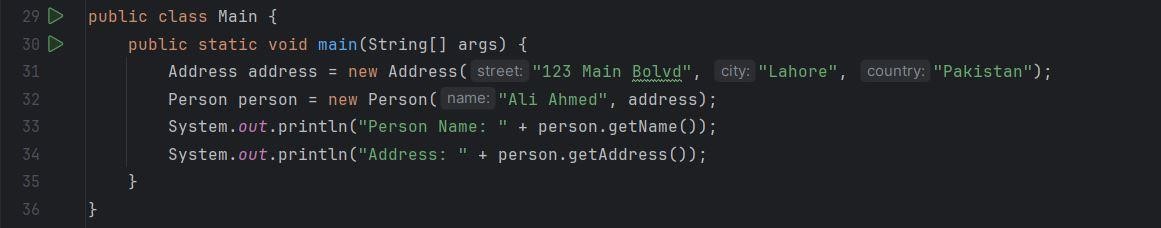
**Faculty of Information Technology UCP Lahore, Pakistan**

|  |  |
| --- | --- |
| **Lab 05** |  |
| **Topic** | Association, Aggregation and Composition |
| **Objective** | Learning the relationship between classes in terms of association, aggregation and composition |

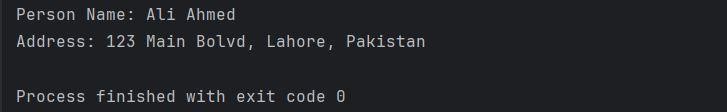
Association

In Java, association represents a relationship between two classes, where one class uses or interacts with another. Association is a general term that refers to how objects relate to one another and how they communicate within the code. It is one of the fundamental building blocks of object-oriented design.

Practice Task 1: Demonstrating Associations



Output:

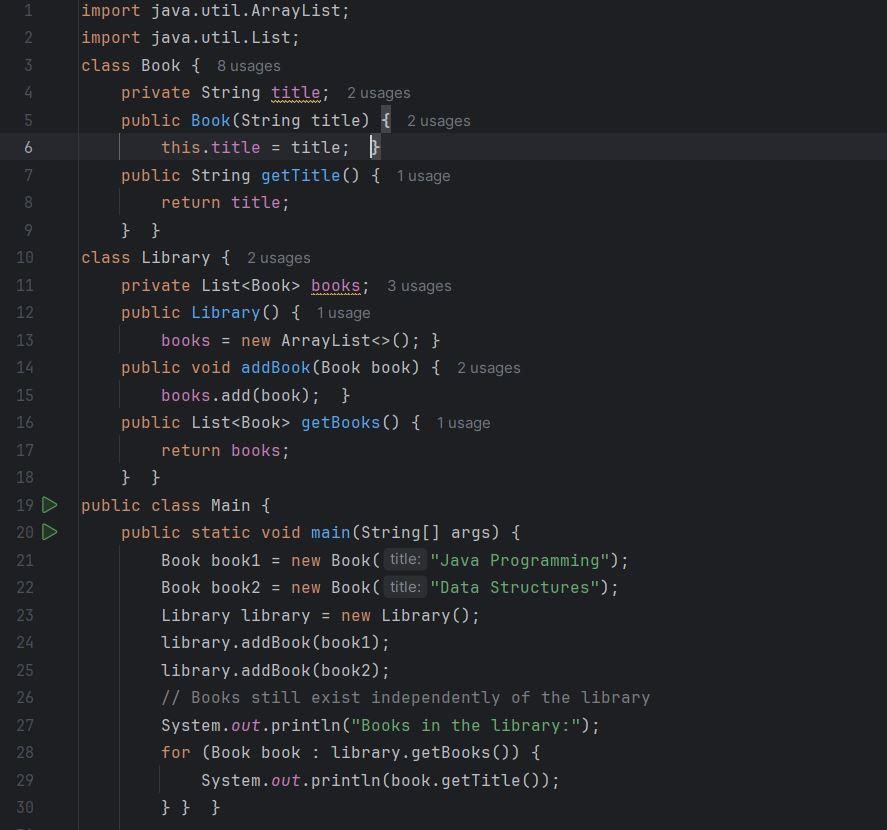


Aggregation

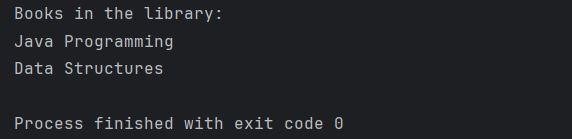
In aggregation, one class contains a reference to another class but does not "own" it, meaning that the lifecycle of the contained objects is independent of the container object.

Let's consider an example of a Library that contains Book objects. A Library can contain multiple Books, but each Book can exist independently of the Library.

Practice Task 2: Demonstrating Aggregation



Output:



Composition:

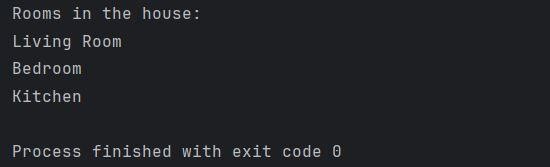
In composition, one class owns another class, and the contained object's lifecycle depends on the container. If the container object is destroyed, the contained objects are also destroyed.

Let's consider an example of a House that has Room objects. If a House is destroyed, all Room objects within it are also destroyed.

Practice Task 3: Demonstrating Composition



Output:



**Basic Practice Task**

**Hospital Management System**

A hospital needs a system to manage its data related to patients, doctors, departments, and staff. The following requirements outline the relationships:

* Patients visit the hospital and are treated by Doctors. Each Patient can be associated with one or more Doctors depending on their medical needs.
* Each Doctor has a Specialization associated with them. The hospital needs to maintain specialization details, as these can be shared by multiple doctors across different departments.
* A Hospital contains various Departments such as Cardiology, Neurology, and Pediatrics, each managed independently.
* Each Department has a group of Staff Members assigned to it.
* Each Patient has a unique MedicalRecord that contains confidential medical history and cannot exist without the patient.
* A Doctor may treat multiple Patients, but treating a patient is optional depending on appointments.

Based on these requirements, the classes and relationships can be summarized as follows:

**Classes:**  
Patient, Doctor, Specialization, Hospital, Department, StaffMember, MedicalRecord

**Relationships:**

* **Association:**
  + Patient ↔ Doctor (many-to-many)
  + Department ↔ StaffMember (one-to-many)
* **Aggregation:**
  + Hospital → Department (one-to-many)
  + Doctor → Specialization (one-to-one)
* **Composition:**
  + Patient → MedicalRecord (one-to-one)

Provide the information above, and also **design a class diagram for the given scenario**.

**Scenario Based Task**

**Task 1: Implementing Association**

**Scenario:** In a smart home system, a SmartHome is associated with multiple SmartFan objects, and each fan can be controlled by a RemoteController.

**Instructions:**

1. Create the following classes in Java:
   * SmartHome: Represents a home that manages smart devices.
   * SmartFan: Represents smart fans that can be added to a home.
   * RemoteController: Represents a controller to operate the smart fan.
2. Implement the following associations:
   * One-way association: SmartHome manages multiple SmartFan objects.
   * Two-way association: Each SmartFan can be controlled by a RemoteController.

**Expected Output:**

* A SmartHome can have multiple SmartFan objects.
* A RemoteController can control SmartFan objects through two-way association.

**Task 1.1: Class Diagram for Association**

**Objective:** Create a UML class diagram to show the associations between SmartHome, SmartFan, and RemoteController.

* One-way association between SmartHome and SmartFan.
* Two-way association between SmartFan and RemoteController.

**Task 2: Composition**

**Scenario:** A SmartFan has a Motor. The Motor is an essential part of the SmartFan, and without a SmartFan, the Motor has no independent existence.

**Instructions:**

1. Implement the following classes:
   * SmartFan: Contains a reference to a Motor.
   * Motor: Represents the motor inside the smart fan.
2. Implement the composition relationship where the SmartFan is responsible for creating and managing the Motor.

**Expected Output:**

* A SmartFan creates and manages a Motor object.
* If the SmartFan is destroyed, the Motor also ceases to exist.

**Task 3: Aggregation**

**Scenario:** A SmartHome can have multiple RemoteController objects. The RemoteController can exist independently of the SmartHome.

**Instructions:**

1. Implement the following classes:
   * SmartHome: Aggregates multiple RemoteController objects.
   * RemoteController: Exists independently of SmartHome.
2. Implement the aggregation relationship where the RemoteController is not dependent on SmartHome.

**Expected Output:**

* A SmartHome can have multiple RemoteController objects.
* The RemoteController can exist independently.

**Task 4: UML Class Diagram for Composition and Aggregation**

**Objective:** Create UML diagrams for:

* The **composition** relationship between SmartFan and Motor.
* The **aggregation** relationship between SmartHome and RemoteController.