C13 - Lab Experience: Aggregation and Composition

Objective:
The goal of this lab is to help you understand the concepts of **aggregation** and **composition** in object-oriented programming (OOP). You will implement and demonstrate these concepts using C++ classes. The main focus will be on how objects can "own" other objects (composition) and how objects can simply reference other objects without owning them (aggregation).

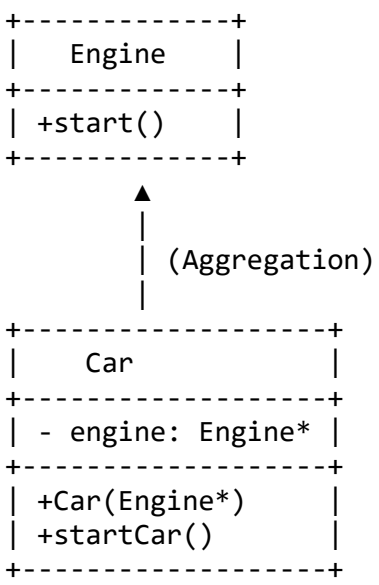
Instructions:

1. **Aggregation:**
    - Aggregation represents a **"has-a"** relationship where an object contains references to other objects, but the contained objects can exist independently. In aggregation, one object (parent) can have references to another object (child), but the child object's lifetime is not dependent on the parent object's lifetime.
    - You are required to implement the relationship between a Car and an Engine using aggregation.
    - **Key Points:**
        - A Car contains an Engine, but the Engine is created outside the Car class and passed to the Car object. This means the Car does not own the Engine.
        - The Engine can exist independently of the Car.

---

3. **Composition**
    - In **composition**, an object contains another object, but the contained object **cannot exist without** the parent object. In other words, the lifetime of the contained object is tightly bound to the lifetime of the parent object. When the parent object is destroyed, the contained object is also destroyed.
    - You are tasked with implementing the relationship between a Library and Book using **composition**.
    - **Key Points:**
        - A Library will own and manage Book objects. When the Library is destroyed, the Book objects should also be destroyed, meaning the Library controls the lifecycle of the Book objects.

Class Diagram 1 – Aggregation Style



Class Diagram 2 – Composition Style