COSC 3P91 – Advanced Object-Oriented Programming
Laboratory 2: UML Class Diagram

Department of Computer Science
Brock University

Winter Term

Overview

This laboratory focuses on the formalization of system requirements into a structured architectural model. While Laboratory 1 explored the transition from procedural to object-oriented code, Laboratory 2 requires students to exercise high-level design thinking using Unified Modeling Language (UML). Students will analyze a complex "Smart Home" requirement set and translate abstract relationships—such as composition, aggregation, and interface realization—into a concrete class diagram.

Learning Objectives

By completing this laboratory, students will be able to:

- Translate a text-based requirements specification into a formal UML Class Diagram.
- Distinguish between different types of associations, specifically **Composition** vs. **Aggregation**.
- Correctly model **Inheritance** (Generalization) and **Interfaces** (Realization).
- Assign appropriate multiplicities and visibility modifiers to class members.
- Identify the structural implications of abstract classes within a system.

Given Problem Description: The "SmartHome Pro" System

You are tasked with designing the core architecture for "SmartHome Pro," a centralized platform for managing automated properties.

Properties and Rooms: A `User` can own multiple `Houses`. Each `House` must have at least one `Room`. If a `House` is deleted from the system, all its associated `Rooms` are also destroyed. Each `House` contains exactly one `CentralHub` which manages system operations.

Connectivity and Interfaces: Define an interface `IConnectable` that ensures hardware can `connectToWifi()` and `getSignalStrength()`. Both `SmartDevices` and `Sensors` must implement this interface.

Devices: `SmartDevice` is an abstract class. Specific types include `Light` (with `brightnessLevel`), `Thermostat` (with `targetTemperature`), and `SmartLock` (with `pinCode`). Devices can exist independently of a `Room`; if a `Room` is deleted, devices are unassigned but not destroyed.

Automation and Logging:

- `AutomationRules` are triggered by a `Sensor` and perform actions on a `SmartDevice` (1:1:1 relationship).
- The `CentralHub` maintains an `ActivityLog` consisting of multiple `LogEntries`. Each entry contains a `timestamp` and `description`. If the hub is destroyed, the logs are also destroyed.

Analysis Tasks

Students must answer the following questions to verify their understanding of the model:

1. Which relationship best describes the link between `House` and `Room`?
2. How should the relationship between `Room` and `SmartDevice` be represented to show that devices survive room deletion?
3. If `SmartDevice` is abstract, can it be instantiated directly?
4. What notation is used to represent the `IConnectable` implementation?
5. What is the multiplicity between `House` and `Room`?
6. Where should the `startRecording()` method be placed if only `SecurityCamera` uses it?
7. Is the link between `CentralHub` and `LogEntry` Aggregation or Composition?
8. Which OOP principle is applied when creating `Admin` and `Guest` from `User`?
9. How are interface names typically formatted in UML (e.g., `<<interface>>`)?
10. How many objects are involved in a single execution of an `AutomationRule`?

Required Task

Students must produce a comprehensive UML Class Diagram based on the SmartHome Pro description. The solution must:

1. Use standard UML 2.0 notation.
2. Clearly label all associations (composition, aggregation, generalization, and realization).
3. Include multiplicities at all association ends.
4. Specify attributes and methods for each class, including appropriate visibility (+/-).

Key Takeaways

This laboratory emphasizes that structural design is the foundation of robust OOP. By identifying the lifecycle of objects (Composition vs. Aggregation) and defining clear contracts (Interfaces), students learn to create systems that are scalable and maintainable before a single line of code is written.