Object-Oriented Programming
Semester 2025-II
Workshop No. 2 — Object-Oriented Implementation for a Domotic Circuit Simulator

Eng. Carlos Andrés Sierra, M.Sc.
Lecturer
Computer Engineering
School of Engineering
Universidad Nacional de Colombia

Building on *Workshop #1* (Conceptual Design), this session focuses on translating your **conceptual design** for the Domotic Circuit Simulator into a technical implementation plan. You will refine your understanding of class structures, relationships, and the key OOP pillars (**inheritance**, **polymorphism**, **encapsulation**) to move closer to a working Python prototype.

Workshop Scope and Objectives:

- **Refinement of Conceptual Design:** Update your requirements, user stories, and CRC cards based on feedback or new insights since Workshop #1.
- **Technical Design with UML:** Produce and refine *UML class diagrams* that detail inheritance hierarchies, associations, and class responsibilities for your domotic circuit simulator.
- **OOP Principles in Action:** Demonstrate how inheritance, polymorphism, and encapsulation are applied to solve realistic scenarios in your simulator.
- **Python Implementation Plan:** Prepare initial code snippets and a directory structure for your Python-based solution.

Carlos Andrés Sierra, Computer Engineer, M.Sc. in Computer Engineering, Lecturer at Universidad Nacional de Colombia.
Any comment or concern regarding this workshop can be sent to Carlos A. Sierra at: *casierrav@unal.edu.co*.

Methodology and Deliverables:

1. **Conceptual Design Updates:**
    - Revisit Workshop #1 deliverables (requirements, CRC cards, user stories) and add any missing or revised elements.
    - Document changes or decisions that emerged from further planning or discussions.
2. **Technical Design (UML Diagrams):**
    - Create or update *class diagrams*, showing inheritance hierarchies, class members (attributes, methods), and relationships.
    - Highlight how classes implement or override methods to fulfill user requirements and integrate OOP principles.
    - Optionally, add sequence or activity diagrams to illustrate interactions between classes and the flow of data.
3. **Implementation Plan for OOP Concepts:**
    - Summarize how your code will realize **encapsulation** (private/protected/public members, getters, setters), **inheritance** (base and derived classes), and **polymorphism** (method overriding, abstract classes).
    - Outline a preliminary directory structure or package organization for your Python project.
4. **Initial Python Code Snippets:**
    - Include initial code snippets illustrating class definitions, placeholder methods, or abstract base classes in Python.
    - Clearly state how these snippets relate to your UML diagrams and design rationale.
5. **Delivery Format:**
    - Archive your updated documentation (revised conceptual design, UML diagrams, code snippets) into a single PDF.
    - Place all materials in a folder named `Workshop-2` in your project repository, referencing them in a `README.md`.

Deadline: Friday, November 7th, 2025, at 7:00 PM. Late submissions may affect your grading according to course policies.

Notes:

- All documents must be in **English**.
- Cite any relevant tutorials, articles, or external libraries that shaped your design.

- Your refined design and UML models here lay the groundwork for actual coding in future assignments. Aim for clarity and scalability.
- Focus on Python and OOP best practices suitable for second-semester engineering students.

In this workshop, you turn your conceptual plans into a technical blueprint. Focus on integrating OOP concepts rigorously, ensuring your domotic circuit simulator remains robust, maintainable, and aligned with best practices.