Object-Oriented Programming
Semester 2025-III
Workshop No. 3 — Object-Oriented SOLID Principles

Eng. Carlos Andrés Sierra, M.Sc.
Full-time Adjunct Professor
Computer Engineering Program
School of Engineering
Universidad Distrital Francisco José de Caldas

Building on **Workshop #1** (Object-Oriented Design) and **Workshop #2** (Object-Oriented Implementation), this session focuses on incorporating **SOLID principles** into both your conceptual plan and technical model. You will refine your simple transactional application by applying each principle (Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, and Dependency Inversion) to ensure a robust, maintainable software design.

Workshop Scope and Objectives:

- **Conceptual & Technical Design Updates:** Revisit your class diagrams, use cases, and requirements to incorporate SOLID concepts.
- **Implementation of SOLID Principles:** Show explicit examples of how your classes and interactions follow these principles.
- **Refined UML Models:** Adapt your UML (class diagrams, sequence diagrams if applicable) to illustrate the impact of SOLID-driven changes.

Carlos Andrés Sierra, Computer Engineer, M.Sc. in Computer Engineering, Titular Professor at Universidad Distrital Francisco José de Caldas.

Any comment or concern regarding this workshop can be sent to Carlos A. Sierra at: *cavirguezs@udistrital.edu.co*.

Methodology and Deliverables:

1. **Revisiting Requirements & Design:**
   - Update your functional and non-functional requirements if new insights emerged while integrating SOLID.
   - Reflect any changes in user stories and CRC cards from Workshop #1 and #2.
2. **Enhanced UML Diagrams:**
   - Highlight modifications to your class designs, emphasizing new interfaces, abstract classes, or design patterns that ensure SOLID compliance.
   - Use sequence diagrams or additional notes to showcase how these principles improve the system's transaction flows.
3. **SOLID-Focused Implementation:**
   - Demonstrate how Single Responsibility is achieved by splitting or reorganizing classes with multiple roles.
   - Implement Open/Closed by extending behaviors without modifying base classes unnecessarily.
   - Show adherence to Liskov Substitution when creating subclasses or interfaces.
   - Apply Interface Segregation by ensuring no class is forced to implement methods it does not need.
   - Illustrate Dependency Inversion by injecting dependencies or referencing abstractions rather than concrete classes.
4. **Work in Progress Code & Documentation:**
   - Include short examples or snippets that illustrate updated class designs, relevant interfaces, or refined inheritance hierarchies.
   - Provide a brief rationale or commentary for each SOLID-related change.
5. **Delivery Format:**
   - Consolidate revised documentation, diagrams, and code snippets into a single PDF.
   - Code snippets should be in Java (or another OOP language) and formatted for clarity.
   - Full code is not required; focus on the most relevant parts that illustrate your SOLID-driven changes.
   - Place all materials in a folder named `Workshop-3` in your project repository and add a `README.md` noting how to review or run the updated designs.

Deadline: Friday, November 7th, 2025, 16:00. Submissions will be reviewed promptly.

Notes:

- Keep your deliverables in **English**.
- Cite or reference any articles, tutorials, or patterns that shaped your SOLID-driven redesign.
- Prioritize clarity and the practical application of SOLID to ensure a flexible, maintainable codebase as your transactional application evolves.
- Deliveries are incremental: build on your previous workshops and ensure each submission shows progress and refinement.
- Consider including a brief reflection (1-2 paragraphs) describing challenges faced and decisions made during this phase.

This workshop cements your grasp on SOLID principles, bridging them with your growing OOP application architecture. Use these techniques to yield a reliable, testable, and extensible design. Good luck!