Lab 03: Structuring and Writing Patterns

Week 3 Assignment

Student Name: _____ Date: _____

Overview

Total Points: 100 (50 points per lab)
Required Reading: *Learning JavaScript Design Patterns, 2nd Edition*, Ch. 3
Chapter 3: "Structuring and Writing Patterns" — Sections: "The Structure of a Design Pattern", "Well-Written Patterns", "Writing a Pattern".
Learning Outcome: LO1 — Understand and apply fundamental concepts of design patterns.

Key Vocabulary

- **Pattern Structure**: A design pattern should have Pattern Name, Description, Context Outline, Problem Statement, Solution, Design, Implementation, Illustrations, Examples, Corequisites, Relations, Known Usage, and Discussions (Ch. 3, "The Structure of a Design Pattern").
- **Context**: The situation or environment in which a pattern effectively responds to users' needs (Ch. 3, "The Structure of a Design Pattern").
- **Problem Statement**: A statement of the problem addressed so that we understand the pattern's intent (Ch. 3, "The Structure of a Design Pattern").
- **Solution**: A description of how the user's problem is solved in an understandable list of steps and perceptions (Ch. 3, "The Structure of a Design Pattern").
- **Well-Written Pattern**: A pattern that provides substantial reference material and evidence of why it is necessary (Ch. 3, "Well-Written Patterns").
- **GoF Style**: Formal pattern documentation style following the Gang of Four format (Ch. 3, "Writing a Pattern").
- **Pattern Transparency**: Design patterns should be entirely transparent to the end-user experience; they primarily serve developers (Ch. 3, "Writing a Pattern").

ALMAU Requirements

- All work must be completed in **OquLabs** with full-screen mode, active typing (no copy-paste), minimum 30 minutes session duration.
- Git repository structure: `Lab_03/task1/`, `Lab_03/task2/` with conventional commits.
- If AI is used, **AI_REPORT** is mandatory (see template in course materials).
- Submit a `.txt` file containing only your public GitHub repository link to OquLabs.

1. Lab 3.1: Pattern Structure Documentation (50 Points)

Objective

Write a formal description of a utility function using the pattern template from Ch. 3, "The Structure of a Design Pattern". This exercise demonstrates understanding of how patterns are structured and documented according to industry standards.

Background

Ch. 3 emphasizes that pattern authors must outline a new pattern's design, implementation, and purpose. A pattern should be presented as a rule establishing a relationship between: a context, a system of forces that arises in that context, and a configuration that allows these forces to resolve themselves.

The book specifies that a design pattern should have the following elements (with the first five being most important): Pattern Name, Description, Context Outline, Problem Statement, Solution, Design, Implementation, Illustrations, Examples, Corequisites, Relations, Known Usage, and Discussions.

Tasks

Task 1: Utility Function Selection and Analysis (15 points)

1. Select a utility function from your previous code or create a new one (e.g., data validation, formatting, API wrapper, event handler).
2. Analyze the function and identify:
   - What problem does it solve?
   - In what contexts is it used?
   - How does it solve the problem?
   - What are the key components of the solution?
3. Document your analysis in `Lab_03/task1/function_analysis.md`.

Task 2: Formal Pattern Documentation (25 points)

1. Write a complete pattern description following the structure from Ch. 3, "The Structure of a Design Pattern":
   - **Pattern Name**: Unique name representative of the pattern's purpose
   - **Description**: Brief description of what the pattern helps achieve
   - **Context Outline**: Contexts in which the pattern effectively responds to users' needs
   - **Problem Statement**: Statement of the problem addressed
   - **Solution**: Description of how the problem is solved (list of steps)
   - **Design**: Description of the pattern's design and user behavior
   - **Implementation**: Guide to how developers would implement the pattern
   - **Examples**: Implementation of the pattern in minimal form (code example)
   - **Consequences**: Benefits and trade-offs of using this pattern
2. Include a working code example in JavaScript (ES6+ syntax) demonstrating the pattern.
3. Reference Ch. 3, "The Structure of a Design Pattern" for the complete structure.

Task 3: Pattern Quality Assessment (10 points)

1. Evaluate your pattern documentation against Ch. 3, "Well-Written Patterns" criteria:
   - Does it provide substantial reference material?
   - Does it provide evidence of why it is necessary?
   - Is it transparent to the end-user experience?
   - Does it have a strong set of examples?
2. Document any improvements needed to meet "Well-Written Patterns" standards.
3. Reference Ch. 3, "Writing a Pattern" checklist: "How practical is the pattern?", "Keep best practices in mind", "Patterns should be transparent to the user".

Technical Requirements

- Use modern JavaScript syntax (ES6+)
- Code must be well-commented explaining pattern implementation
- Follow camelCase naming conventions
- Include error handling where appropriate
- Code must be executable and demonstrate the pattern working

Deliverable

Submit the following in your GitHub repository:

- `Lab_03/task1/function_analysis.md` — Function selection and analysis
- `Lab_03/task1/pattern_documentation.md` — Complete formal pattern description
- `Lab_03/task1/pattern_example.js` — Working code example
- `Lab_03/task1/quality_assessment.md` — Pattern quality evaluation
- `Lab_03/task1/README.md` — Summary and methodology

Lab 3.1 Assessment Rubric (50 points)

- **Task 1 - Function Analysis (15 pts)**:
  - Function selected and analyzed: 8 pts
  - Problem, context, solution identified: 7 pts
- **Task 2 - Pattern Documentation (25 pts)**:
  - Complete pattern description following Ch. 3 structure: 15 pts
  - Code example demonstrates pattern correctly: 7 pts
  - Accurate book references: 3 pts
- **Task 3 - Quality Assessment (10 pts)**:
  - Pattern evaluated against "Well-Written Patterns" criteria: 6 pts
  - Improvements documented: 4 pts

2. Lab 3.2: Module Pattern Documentation (GoF Style) (50 Points)

Objective

Create complete documentation for the Module Pattern following the Gang of Four (GoF) style specification as outlined in Ch. 3, “Writing a Pattern”. This exercise demonstrates mastery of formal pattern documentation.

Background

Ch. 3 explains that pattern authors should learn from others who have documented patterns well. The book references Christian Heilmann’s work on the Revealing Module pattern as an example of adapting existing patterns.

The Module Pattern is one of the most fundamental patterns in JavaScript. According to Ch. 3, a well-documented pattern should include all elements from “The Structure of a Design Pattern”, with particular emphasis on Pattern Name, Description, Context Outline, Problem Statement, and Solution.

Tasks

Task 1: Module Pattern Research (15 points)

1. Research the Module Pattern using Ch. 7, “JavaScript Design Patterns” > Section “The Module Pattern” as primary reference.
2. Document:
   - Historical context and evolution of the Module Pattern
   - Relationship to other patterns (e.g., Revealing Module Pattern)
   - Known usage in popular JavaScript libraries and frameworks
   - Variations and modern adaptations (ES6 modules)
3. Create `Lab_03/task2/module_pattern_research.md`.

Task 2: GoF-Style Pattern Documentation (25 points)

1. Write complete Module Pattern documentation following GoF style and Ch. 3 structure:
   - **Pattern Name**: Module Pattern
   - **Description**: Brief description of what the pattern helps achieve
   - **Context Outline**: Contexts where Module Pattern is effective
   - **Problem Statement**: Problems the Module Pattern addresses (e.g., global namespace pollution, encapsulation)
   - **Solution**: Step-by-step description of the Module Pattern implementation
   - **Design**: Description of the pattern’s design, including private and public members
   - **Implementation**: Guide to implementing Module Pattern in JavaScript
   - **Illustrations**: Code structure diagram or visual representation
   - **Examples**: Multiple code examples showing different Module Pattern variations
   - **Consequences**: Benefits (encapsulation, namespace management) and trade-offs
   - **Corequisites**: What other patterns may be needed to support Module Pattern usage
   - **Relations**: Patterns that resemble Module Pattern (Revealing Module, Namespace)

- **Known Usage**: Real-world examples from libraries and frameworks
- **Discussions**: Author's thoughts on Module Pattern benefits and modern alternatives

2. Reference Ch. 3, "The Structure of a Design Pattern" for complete structure.
3. Include at least three code examples:
   - Basic Module Pattern (IIFE)
   - Module Pattern with private variables
   - Modern ES6 Module equivalent

Task 3: Pattern Comparison and Evolution (10 points)

1. Compare Module Pattern with:
   - Revealing Module Pattern (Ch. 7 reference)
   - ES6 Modules (Ch. 5 reference)
   - Namespace Pattern (Ch. 11 reference)
2. Document when to use each approach.
3. Explain how Module Pattern has evolved with modern JavaScript features.
4. Reference Ch. 3, "Writing a Pattern": "Remember that originality is not key in pattern design".

Technical Requirements

- All code examples must use modern JavaScript (ES6+)
- Code must be well-commented
- Follow camelCase naming conventions
- Include working examples that can be executed
- Demonstrate both classic and modern Module Pattern implementations

Deliverable

Submit the following in your GitHub repository:

- `Lab_03/task2/module_pattern_research.md` — Research documentation
- `Lab_03/task2/module_pattern_gof.md` — Complete GoF-style pattern documentation
- `Lab_03/task2/module_examples.js` — Code examples (basic, private, ES6)
- `Lab_03/task2/pattern_comparison.md` — Comparison with related patterns
- `Lab_03/task2/README.md` — Summary and documentation approach

Lab 3.2 Assessment Rubric (50 points)

- **Task 1 - Research (15 pts)**:
  - Module Pattern researched thoroughly: 8 pts
  - Historical context and usage documented: 7 pts
- **Task 2 - GoF Documentation (25 pts)**:
  - Complete GoF-style documentation with all elements: 15 pts
  - Multiple code examples (at least 3 variations): 7 pts
  - Accurate references to Ch. 3 and Ch. 7: 3 pts
- **Task 3 - Comparison (10 pts)**:
  - Pattern comparison completed: 5 pts
  - Evolution and modern alternatives explained: 5 pts

Comprehensive Assessment Summary

Category	Points	Assessment
Lab Tasks		
Lab 3.1: Pattern Structure Documentation	50	See Lab 3.1 Rubric
Lab 3.2: Module Pattern Documentation (GoF)	50	See Lab 3.2 Rubric
Code Quality & Structure		
Code runs without errors	15	[YES] Runs / [NO] Doesn't run (-15)
Proper project structure (Lab_03/task1/, Lab_03/task2/)	10	[YES] Correct / [NO] Incorrect (-10)
Code comments and documentation	10	[YES] Present / [NO] Missing (-10)
Naming conventions (camelCase)	5	[YES] Correct / [NO] Incorrect (-5)
ALMAU Protocol Compliance		
OquLabs protocol (full-screen, active typing, duration)	10	[YES] Compliant / [NO] Violations (-20/-
Git discipline (structure, commits, messages)	10	[YES] Compliant / [NO] Violations (-10/-
AI_REPORT (if AI used)	10	[YES] Complete / [NO] Missing (-100)
Book References		
Accurate chapter and section references	10	[YES] Accurate / [NO] Inaccurate (-5)
Understanding of pattern structure and documentation	10	[YES] Demonstrated / [NO] Not shown (
TOTAL	**100**	

Submission Instructions

1. **Lab 3.1**: Submit pattern documentation in `Lab_03/task1/` folder.
2. **Lab 3.2**: Submit Module Pattern GoF documentation in `Lab_03/task2/` folder.
3. Create a public GitHub repository with the required folder structure.
4. Create a `Lab_03_submission.txt` file containing only your GitHub repository URL.
5. Upload `Lab_03_submission.txt` to OquLabs by the deadline.
6. Include your name, student ID, and date in all documentation files.

Academic Integrity

All work must be your own. You may reference the textbook and official JavaScript documentation, but code and documentation must be written by you. Plagiarism or unauthorized collaboration will result in a zero grade and academic penalties. If AI tools are used, the AI_REPORT must be completed and submitted.

References

- Osmani, A. (2023). *Learning JavaScript Design Patterns, 2nd Edition*. O'Reilly Media.
  - Chapter 3: "Structuring and Writing Patterns"
    - Section: "The Structure of a Design Pattern"
    - Section: "Well-Written Patterns"
    - Section: "Writing a Pattern"
  - Chapter 7: "JavaScript Design Patterns" > Section "The Module Pattern"