SENG 11232 · Engineering Foundation

STUDENT PRE-LECTURE NOTES · LECTURE 08

Software Design Principles & Engineering Trade-offs

Module 3 — Engineering Process & Design

HOW TO USE	Read these notes BEFORE the lecture. Highlight terms you don't recognise. Bring your questions to class. Use the blank spaces for your own notes.

Learning Objectives

By the end of this lecture, you should be able to:

- 1. Apply abstraction, modularity, cohesion and coupling principles
- 2. Explain the SOLID principles with examples
- 3. Describe common design patterns: Singleton, Observer, Factory, Strategy
- 4. Analyse engineering trade-offs and document using ADRs
- 5. Explain technical debt and its impact on software cost

Core Design Principles

- Abstraction: expose interfaces, hide implementation details
- Modularity: partition into independent units with defined interfaces
- Separation of Concerns (SoC): each module has one reason to change
- Low Coupling: minimise dependencies between modules
- High Cohesion: elements within a module are strongly related
- DRY (Don't Repeat Yourself): one authoritative source for each piece of knowledge
- KISS (Keep It Simple): simplest solution that meets requirements

KEY	Complexity is the enemy of reliability. Every unnecessary layer of abstraction is a potential hiding place for bugs.

Your notes:

SOLID Principles & Technical Debt

SOLID (Robert Martin)

- S: Single Responsibility — one class, one reason to change
- O: Open/Closed — extend behaviour without modifying existing code
- L: Liskov Substitution — subclasses must be substitutable for base class
- I: Interface Segregation — clients depend only on methods they use

- D: Dependency Inversion — depend on abstractions, not concretions

Technical Debt

- The implied cost of rework caused by choosing a fast/easy solution now over a better long-term approach
- Types: deliberate (known shortcuts), inadvertent (design ignorance), bit rot (entropy)
- Compounds like financial debt — interest paid as slower development, more bugs

NOTE	Architecture Decision Records (ADRs): one-paragraph document recording what was decided, why, and what alternatives were rejected. Write one for every significant design decision.

Your notes:

____

____

____

Key Terms for This Lecture

Term	Your Definition (fill in during/after lecture)
Abstraction	
Coupling	
Cohesion	
DRY	
SOLID	
Single Responsibility	
Design Pattern	
Singleton	
Observer	
Technical Debt	
ADR (Architecture Decision Record)	

Self-Check Questions

Answer these after the lecture to test your understanding:

- Q1. Identify the SOLID violation in a class that reads from a database, sends emails, AND validates input.

Answer: ______________________________

- Q2. Describe the Observer pattern. Give a real-world software example where you would use it.

Answer: ______________________________

- Q3. What trade-offs exist between performance and maintainability? Give a specific example.

Answer: ______________________________

- Q4. What is technical debt? Describe two different types and explain how they arise.

Answer: ______________________________

SENG 11232 Engineering Foundation · Lecture 08: Software Design Principles & Engineering Trade-offs