

Wrap-Up



EECS3311: Software Design
Fall 2017

CHEN-WEI WANG

What You Learned



- **Design Principles:**
 - **Abstraction** [contracts, architecture, math models]
Think *above the code level*
 - Information Hiding
 - Single Choice Principle
 - Open-Closed Principle
 - Uniform Access Principle
- **Design Patterns:**
 - Singleton
 - Iterator
 - State
 - Composite
 - Visitor
 - Observer
 - Event-Driven Design
 - Undo/Redo, Command
 - Model-View-Controller

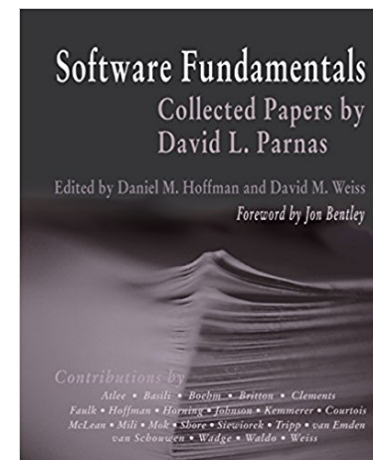
[lab 4]
[project]

Beyond this course... (1)



- How do I program in a language not supporting **DbC** natively?
 - Document your **contracts** (e.g., JavaDoc)
 - But, it's critical to ensure (manually) that contracts are *in sync* with your latest implementations.
 - Incorporate contracts into your Unit and Regression **tests**
- How do I program in a language without a **math library**?
 - Again, before diving into coding, always start by **thinking above the code level**.
 - Plan ahead how you intend for your system to behaviour at runtime, in terms of interactions among **mathematical objects**.
A **mathematical relation**, a formal model of the **graph data structure**, suffices to cover all the common problems.
 - Use efficient data structures to support the math operations.
 - Document your code with **contracts** specified in terms of the math models.
 - Test!

Beyond this course... (2)



- *Software fundamentals: collected papers by David L. Parnas*
- Design Techniques:
 - Tabular Expressions
 - Information Hiding