Key Concepts

- Overall goal: reduce complexity
  - Dependencies
  - Obscurity

- Good design is an investment
  - Tactical vs. strategic programming

- Complexity is incremental: zero tolerance

- Abstraction: find simple ways to think about complicated things

- Information hiding
  - Interface vs. implementation

- Classes should be deep

- General-purpose classes are deeper

- Different layers should have different abstractions

- Pull complexity downward, push specialization upward

- Comments should describe things that aren’t obvious from the code

- Comments are at a different level of precision than code

- Names matter!

- Define errors out of existence

- Code should be obvious
Red Flags

- Shallow classes
- Information leakage
  - Dependencies
  - Conjoined methods/classes
- Temporal decomposition
- Pass-through methods
- Code duplication
- Special cases
- Inconsistencies

- Comment duplicates code
- Implementation contaminates interface documentation
- Documentation has to be long to be complete
- Vague names
- Code is not obvious
Workloads

- If Project 1 was 1.0 units of work, how many units were
  - Project 2:
  - Project 3: