Coding and Refactoring

Outline

- Software Implementation Techniques
  - Coding practices
- Refactoring

Software Implementation Techniques: Coding practices

- **Best coding practices** are a set of informal rules that the software development community has learned over time which can help improve the quality of software
- Many computer programs remain in use for far longer than the original authors ever envisaged (sometimes 40 years or more) so any rules need to facilitate both initial development and subsequent maintenance and enhancement by people other than the original authors.
- In Ninety-ninety rule, Tim Cargill is credited with this explanation as to why programming projects often run late: "The first 90% of the code accounts for the first 90% of the development time. The remaining 10% of the code accounts for the other 90% of the development time."
- The size of a project or program has a significant effect on error rates, programmer productivity, and the amount of management needed
  - Maintainability.
  - Dependability.
  - Efficiency.
  - Usability.

Refactoring

- Refactoring—a construction technique that is also a method for design optimization.
- Refactoring is a reorganization technique that simplifies the design (or code) of a component without changing its function or behavior.
- Refactoring improves the internal structure of a design (or source code) without changing its external functionality or behavior.
- Examines redundancy, unused design elements, inefficient or unnecessary algorithms, poorly constructed or inappropriate data structures, or any other design failure.
- For example, a first design iteration might yield a component that exhibits low cohesion (i.e., it performs three functions that have only limited relationship to one another). After careful consideration, you may decide that the component should be refactored into three separate components, each exhibiting high cohesion.