System software

Software development

Software development

- Metaphors
  - writing code
    - like writing a letter, one-person activity, code reuse, simple projects, throw-away code
  - growing a system, incremental development
    - small steps, design, test, code piece by piece, each version should work (may contain dummy methods)
  - building or constructing software
    - like building a house or skyscraper (small and big projects)
    - under- and over-engineering

Software development

- General guidelines for solving (math) problems
  - How to solve it?, Polya, 1957

1. understand the problem
2. devise a plan
3. carry out the plan
4. look back

Software development

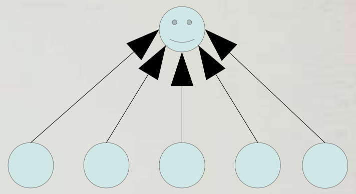
- Software design
  - conception, invention, contrivance of a scheme for turning a software **specification** into operational **software**

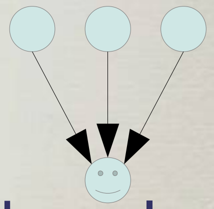
Software development

- Desirable characteristics of a design
  - minimal complexity
    - Occam's razor, use abstractions, avoid being too clever
  - ease of maintenance
    - design should be self-explanatory, intuitive
  - loose coupling
    - interfaces and connections between modules should be minimal
  - extensibility
    - possibility of enhancing the system without violating the underlying structure

Software development

- Desirable characteristics of a design
  - reusability
    - ability to reuse parts of the system
  - high fan-in
    - having a high number of classes that use a given class, good use of utility classes and functions
  - low-to-medium fan-out
    - a given class uses low-to-medium number of other classes





Software development

- Desirable characteristics of a design
  - portability
    - the system is easy to move to another environment
  - leanness
    - the system has no extra parts
    - Voltaire: the book is finished NOT when nothing more can be added but when nothing more can be taken away
  - stratification
    - try to keep the levels of decomposition stratified so that you can view at any level and get consistent view
    - layered design

Software development

- Levels of design
    - software system
    - subsystems or packages
    - classes within packages
    - data and methods within classes
    - internal method design

Software development

- Design heuristics
  - find real-world objects
    - identify objects and their attributes and operations
    - identify interactions of the object with other objects
  - form consistent abstractions
    - high-level view
    - base classes are more abstract than derived ones
  - encapsulate implementation details
    - at particular level of detail you ignore other levels
  - information hiding
    - based on encapsulation, modularity, abstraction

Software development

- Design heuristics
  - use inheritace when it simplifies the design
    - determine common properties of objects, avoid duplication of code
  - identify areas likely to change
    - indentify and design for isolation of such areas
  - keep coupling loose
    - keep number of connections between modules low
    - keep low visibility of data
  - look for common design patterns
    - use ready-made solutions to problems

Software development

- Software development tools (for SIC/XE)
  - version control system
    - subversion
  - programming language
    - Java, C, C++, Rust, …
  - other tools
    - IDE, make, scripting, …

Project: simulator

- virtual machine
  - registers, memory, devices
  - execution support
- user interface
  - GUI or TUI
  - various views of various parts of the machine
  - control of the execution