Today’s topics

Industry Practice
Software Engineering

Upcoming
The Killer Robot

Reading
Great Ideas, Chapters 7

Engineering a Program

- Programming in the Large
  - Not the toy programs we are writing and demonstrating
- What do we require of Software Products for Commercial or Industrial Use?
  - Robustness
  - Testing
  - Documentation
  - Customer Support
    - User Friendliness
    - Help Desk / 800 Number
  - Training
  - Follow on Products
    - Dealing with Competing Products
    - “Creeping Featurism”

What Makes a Successful Product?

- Class Experiences
  - Good
    - ...
    - ...
  - Bad
    - ...
    - ...
- Scenario: “Bad” Product looks “Good”
- Scenario: “Good” Product looks “Bad”
- Which is really the good product?
- Which will succeed?

Program Life Cycle

- Define the Product
- Developing the Program Specifications
- Designing the System Structure
- Coding the System (small part!)
- Testing the Code
- Revision
- Documentation
- Delivery and Training
- Maintenance and Upgrade
Program Life Cycle

Understanding Problem / Specification
- Communicating with the Customer
  - Often customer doesn’t understand
  - Capabilities of computer systems
  - Limitations of computer systems
- Possible role of user documentation
- Specification languages
  - Can be very technical and involved
  - In some cases more trouble than they are worth

Program Life Cycle

Implementation Strategies
- Top Down Implementation
  - Stubs
    - Can test many parts in the absence of other parts
  - Output First
    - Allows you to “see” what the program is doing
- Test as you go
  1. Make it Run
  2. Make it Right
  3. Make it Fast
- Always have a “running” program

Program Life Cycle

Design Strategies/Models
- Classical Waterfall Model
  - Everything moves forward at steady pace
  - Little customer involvement in design
  - Revision possible
    - Late in the game
    - At great expense
- Rapid Prototyping Model
  - Build Prototype quickly
  - Get customer involved
  - Then:
    - May scrap prototype and start over
    - If prototype good enough, may build on it

Program Life Cycle

Debugging (dealing with Defects)
- Testing
- Can only show presence, not absence of bugs
- Design for Testability
  - Modular
  - Hierarchical

Correctness
- Proofs
  - Formal Definition of Specs
  - Formal definition language
- Very hard
- Used in life-critical applications
- What is a correct Graphical User Interface (GUI)?
Program Life Cycle

- **Documentation**
  - On-line documentation
  - Comments on comments in code
    - Program header
    - Block header
    - Function (method) specifications
      - Pre-conditions
      - Post-conditions
    - Variable descriptions
  - Java Docs
    - Available for Java API (see class web page)
    - Can produce for your own code as well

Aspects of Software Engineering

- **Psychology of Programming** has its effects
  - Design by Committee (good or bad?)
  - Communications problems
  - Interaction
  - Creator Independence
  - 2nd System Syndrome

- **Organizational Schemes**
  - Chief Programmer Team (Harlan Mills)
    - NY Time Morgue Project
    - Surgical Team Model
    - Assume that some people are 100 time better than others when programming

Net Productivity:
15 LINES OF CODE / DAY

Rather:
15 LINES OF CODE / DAY

Aspects of Software Engineering

- **Nitty-Gritty Practical Problems**
  - Back-ups
  - Revision Control System
  - Backwards Compatibility
  - Staff Turnover
  - Pleasant Environment
    - Silicon Valley
    - SAS

- **Programming Tools (CASE)**
  - E.g., IDE’s such as Eclipse
  - . . .