How to design/write programs

- Start with a small, working program
  - Don't write 10's, 100's, 1000's of lines before compiling
  - Compile, test, implement, repeat

- Add features to an already working program
  - A program designed to do nothing is a known entity
  - No reason to solve entire problem in one fell-swoop
  - Rare at first (if ever) to get things right the first time

- Refactor again and again and …
  - Change design without changing functionality
  - No new features, how do we ensure this works?

Make code readable

- You will need to read your program tomorrow, next month,…
  - What's the purpose of this variable? Why this loop?
  - Write comments, "notes to myself and other programmers"

- Specify methods using javadoc/other commenting styles
  - All parameters described
  - Purpose of method described
  - Return value described
  - Precondition: what must be true for method to work as intended
- Methods don't check preconditions
  - Caller ensures integrity, not callee (exceptions?)

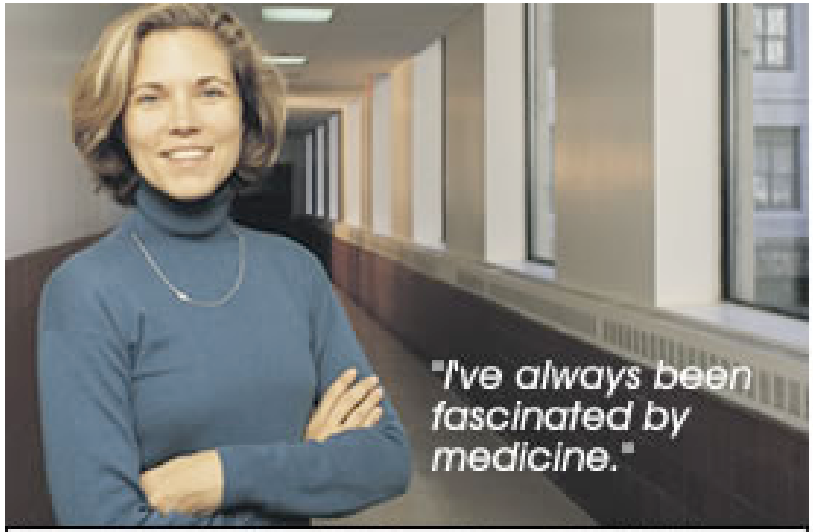
Our programs and "real data"

- How is the protein-finding program similar to real programs
  - Eukaryotes, prokaryotes
  - Differences? HGP?

- What are differences between this, BLAST/other programs?
  - Real data comes from real organisms
  - How does it get into the computer?

- Why do we start with a model of a real program?
  - Why do scientists model things in general?
  - Where do we go after the model?

Terry Gaasterland



Terry Gaasterland is a bioinformatics scientist at The Rockefeller University in New York City.

- Duke Russian/CPS major 1984
  - Her far-reaching goal, and that of other computer scientists who concentrate on biological mysteries, is to get computers to apply logic to biology, which by nature is incredibly complicated. It will take many years, she says, but ultimately future versions of computer programs like MAGPIE will swallow huge amounts of genetic data and spit out predictions about how biology works.