Law of Demeter

A method M of object O can only call methods on the following objects:

- O
- Arguments of M
- Objects created within M
- O’s direct component objects
- A global variable
Law of Demeter

Use only one dot

✅ a.method();

❌ a.b.method();

❌ a.methodB().methodC();

❌ foo = a.methodB();
  foo.methodC();
What about Builder Example?

Notification note = new Notification.Builder(mContext)
    .setContentTitle("New mail from " + sender.toString())
    .setContentText(subject)
    .setSmallIcon(R.drawable.new_mail)
    .setLargeIcon(aBitmap)
    .build();
What about Builder Example?

Each method returns the builder

```java
Notification.Builder mailNotification = new Notification.Builder(mContext);
mailNotification.setContentTitle("New mail from " + sender.toString());
mailNotification.setContentText(subject);
mailNotification.setSmallIcon(R.drawable.new_mail);
mailNotification.setLargeIcon(aBitmap);
Notification note = mailNotification.build();
```
Hinges

Business Rules

Some businesses frequently change rules/deals

   Buy two X and get third X for 1/2 price

   20 cent coffee day

Don't have time to rewrite code

Need to move business logic into data
Joel Spolsky 12 Steps
http://www.joelonsoftware.com/articles/fog0000000043.html
Joel's 12 Steps to Better Code

Do you use source control?

Can you make a build in one step?

Do you make daily builds?

Do you have a bug database?

Do you fix bugs before writing new code?

Do you have an up-to-date schedule?

Minimal data on each bug

Complete steps to reproduce the bug

Expected behavior

Observed (buggy) behavior

Who it's assigned to

Whether it has been fixed or not
Joel's 12 Steps to Better Code in Companies

Do you have a spec?

Do programmers have quiet working conditions?

Do you use the best tools money can buy?

Do you have testers?

Do new candidates write code during their interview?

Do you do hallway usability testing?

    Grab the next person that passes by in the hallway
    Force them to try to use the code you just wrote

    Learn 95% of what there is to learn about usability problems in your code
What every computer science major should know
Dr. Matt Might
University of Utah

http://matt.might.net/articles/what-cs-majors-should-know/
What should every student know to get a good job?

What should every student know to maintain lifelong employment?

What should every student know to enter graduate school?

What should every student know to benefit society?
Portfolio verse Resume

A resume says nothing of a programmer's ability

Portfolio
  Personal blog
  Projects
  Github
  Open source projects
Technical Communication

Lone wolves in computer science are an endangered species

In smaller companies, whether or not a programmer can communicate her ideas to management may make the difference between the company's success and failure
Unix Philosophy

linguistic abstraction and composition

Should be able to

- Navigate and manipulate the filesystem;
- Compose processes with pipes;
- Comfortably edit a file with emacs and vim;
- Create, modify and execute a Makefile for a software project;
- Write simple shell scripts.
Unix Philosophy

Sample tasks

Find the five folders in a given directory consuming the most space

Report duplicate MP3s (by file contents, not file name) on a computer.

Take a list of names whose first and last names have been lower-cased, and properly recapitalize them.

Find all words in English that have x as their second letter, and n as their second-to-last.

Directly route your microphone input over the network to another computer's speaker.

Replace all spaces in a filename with underscore for a given directory.

Report the last ten errant accesses to the web server coming from a specific IP address.
Systems administration

Every modern computer scientist should be able to:

- Install and administer a Linux distribution.
- Configure and compile the Linux kernel.
- Troubleshoot a connection with dig, ping and traceroute.
- Compile and configure a web server like apache.
- Compile and configure a DNS daemon like bind.
- Maintain a web site with a text editor.
- Cut and crimp a network cable.
Programming languages

Programming languages rise and fall with the solar cycle.

A programmer's career should not.

The best way to learn how to learn programming languages is to learn multiple programming languages and programming paradigms.

To truly understand programming languages, one must implement one.
Programming languages

- Racket
- C
- JavaScript
- Squeak
- Java
- Standard ML
- Prolog
- Scala
- Haskell
- C++
- Assembly
Architecture

There is no substitute for a solid understanding of computer architecture

- transistors
- gates
- adders
- muxes
- flip flops
- ALUs
- control units
- caches
- RAM
- GPU
Operating systems

Any sufficiently large program eventually becomes an operating system.

To get a better understanding of the kernel, students could:

- Print "hello world" during the boot process;
- Design their own scheduler;
- Modify the page-handling policy; and
- Create their own filesystem.
Networking

Computer scientists should have a firm understanding of the network stack and routing protocols within a network.

Every computer scientist should implement the following:
- an HTTP client and daemon;
- a DNS resolver and server; and
- a command-line SMTP mailer.

No student should ever pass an intro networking class without sniffing their instructor's Google query off wireshark.
Security

Computer scientists must be aware of the means by which a program can be compromised

At a minimum, every computer scientist needs to understand:
- social engineering
- buffer overflows
- integer overflow
- code injection vulnerabilities
- race conditions
- privilege confusion
Software testing

Software testing must be distributed throughout the entire curriculum

He uses test cases turned in by students against all other students

Students don't seem to care much about developing defensive test cases, but they unleash hell when it comes to sandbagging their classmates
Visualization

The modern world is a sea of data

The Visual Display of Quantitative Information by Tufte
Topics I left out

Databases
Artificial intelligence
Machine learning
Robotics
Graphics and simulation
Software engineering
Parallelism
User experience design
Disarmingly Forthright MSCS Advice
Nick Black

Read it