The Last Lecture

Lessons Learned

- From my experience…and now, from yours.
- Both technical lessons and people lessons.

Risk

- Mitigating risk is perhaps the #1 job of a software engineer.
- A *programmer* gets the job done.
- A *software engineer* says "maybe I shouldn't do this right now."
  - Or chooses a less risky approach.
- How many of you pushed a "safe" change, only to find you broke something?

Reducing Risk

- Make the change in the caller, not the callee (there may be many callers)
  - Or: make the change in the callee (the bug may occur in many contexts)
- Add asserts (someone else might violate your invariant)
- Add a wrapper rather than rewriting the module
- Defer the change until later
- Ask an expert; get more reviewers

Murphy's Law, Applied to Software

```
if (condition)
    doStuff();
```

```
if (condition)
    doStuff();
    alsoPrint();
```

oops!

Lesson: always use curly braces!

```
if (condition) {
    doStuff();
}
```

Project Management

- If your estimates are longer than a day, subdivide the task.
- Ask someone more senior to help you estimate.
  - Their estimate will probably exceed yours. Then, double their estimate because they could do it faster than you could!
- "I'm upset that you finished early!" — said no manager ever.

Scope Creep

- Scope creep kills projects because it prevents them from ever finishing.
- Someone has to be empowered to say "no" or at least "not yet."

Working With People

- Hanlon's razor: "Never attribute to malice that which is adequately explained by ignorance."
  - Assume good intent.
- Ask a trusted colleague for advice about your situation.
- In my experience, usually serious disagreements are a result of differing assumptions.
  - If you can identify the assumptions, you'll feel a lot better.
- Most arguments are not over things that really matter. Are you bike shedding?

Handling Meetings

- Meetings start and end at agreed-upon times.
- Discussion will expand into the available time.
- "Let's take that offline."
- Bring an agenda. Someone should run the meeting.

Morale

- Crunch time happens…but if it happens all the time, morale takes a hit.
  - Eventually, people leave.
- What do most software engineers want?
  - Money?
  - Yes, some. But more importantly: a sense of purpose. Feeling empowered. Being trusted. Flexibility.

Maintaining Morale

- Provide stability
  - Stable hours, stable expectations
- Release products everyone is proud of
- Work-life balance (and boundaries)

Management Cares About…

- Schedule
- Features
- Quality
- Cost
- Risk

- Long-term product strategy (how does *your* feature fit in?)
- Lawyers
- Hopefully, you!
- Whether you showered & used deodorant before your interview

- Management doesn't care about…
  - Your fancy algorithm (make a recommendation)
    - Or exactly how it works
  - Small amounts of money (don't sweat $100 for a fancy mouse)
  - Your disagreement with your colleague over emacs vs. vi (work it out, bring a proposal)
  - What shirt you wore to the interview (within limits)

Hiring

- Perhaps actually the most important thing you'll do.
- Making a hiring mistake can be very expensive.
- Usually (in my experience), people are themselves in interviews.
  - (story time)

Hiring Techniques

- Write the job description first, including required and desired skills
  - Reduces bias (less "just doesn't seem like they fit in here…")
- Assemble a *diverse* interview team. Meet; divide up responsibilities.
- Meet afterward.