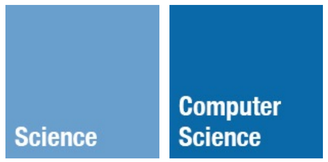
Preliminaries



CS 340: Programming Paradigms and Patterns
Michael Lee <lee@iit.edu>

Agenda

- Administrivia
  - Websites, References, Grading, etc.
- What is “PPP”?
- Why Haskell?
- Why take CS 340?

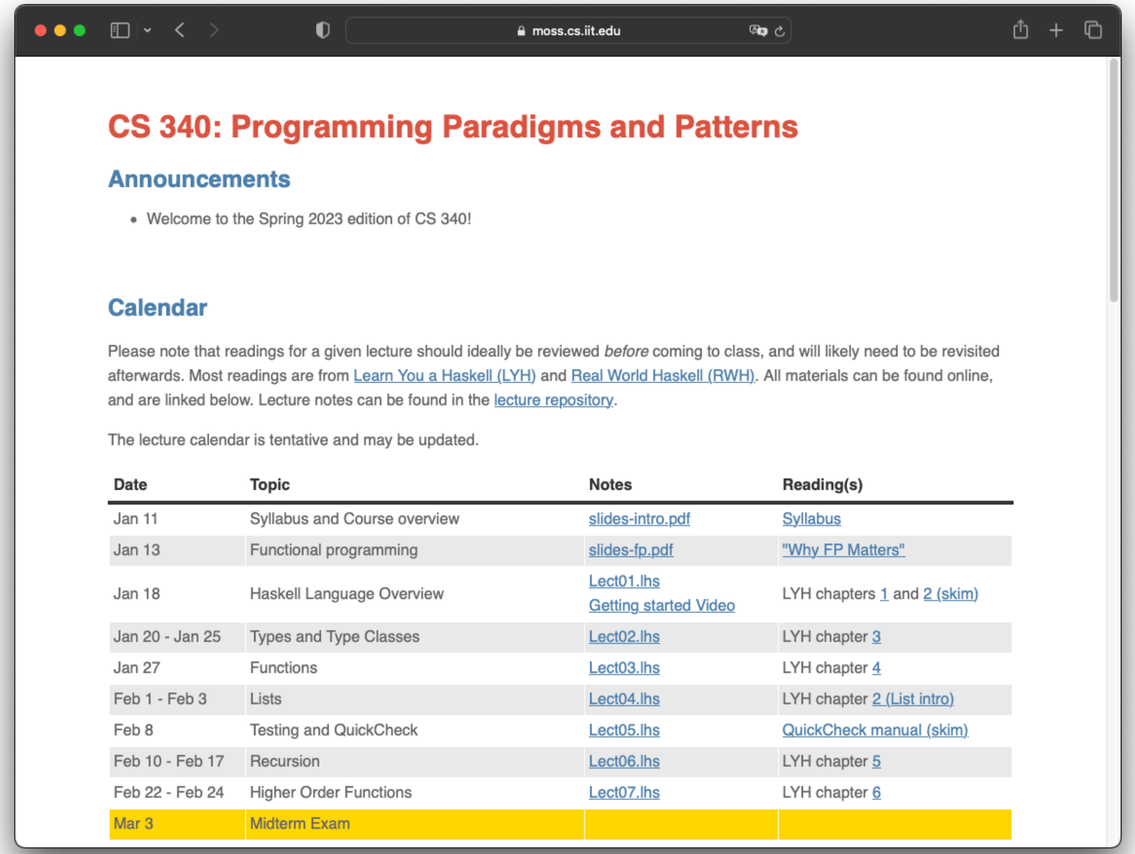
§ Administrivia

Michael Lee

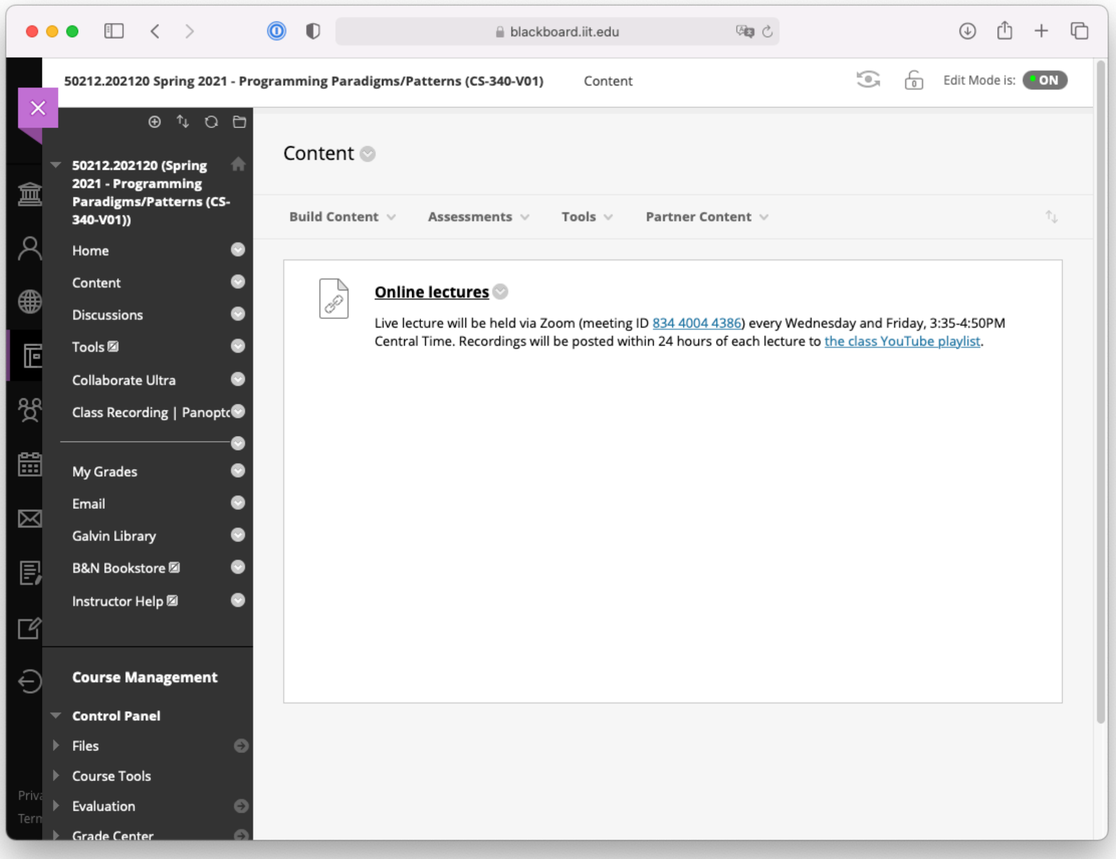
- Email: lee@iit.edu
- Homepage: http://moss.cs.iit.edu
- Office: SB 226C
- Hours: Tue & Thu 10AM-12PM over Zoom
  (by appointment only!)

Prerequisites

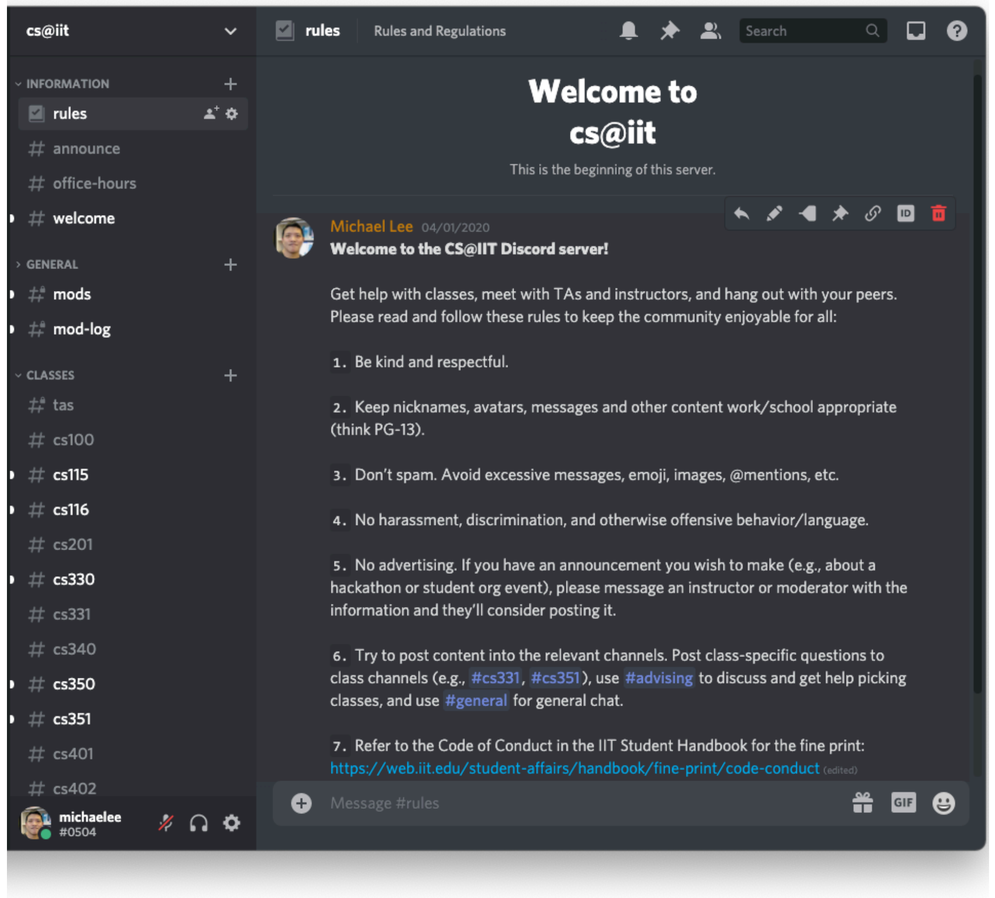
- I assume you are …
  - fluent in some programming language
    - familiar with procedural & OO programming
  - comfortable with development processes:
    - compilation, debugging, testing



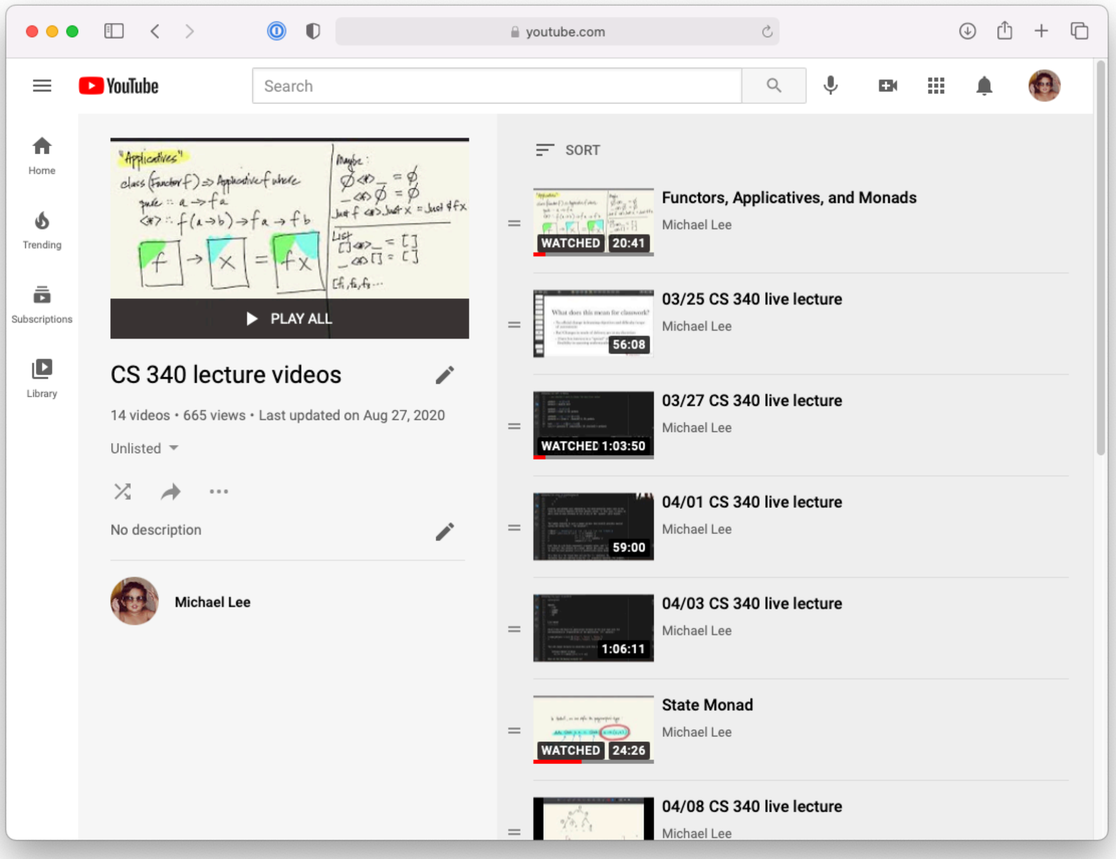
Course website: http://moss.cs.iit.edu/cs340



Blackboard: http://blackboard.iit.edu



Discord: TA office hours, class discussion, and Q/A (invite on course website)



Video playlist (on YouTube)

Primary text: “Learn You a Haskell for Great Good!”

References

- Miran Lipovača, **Learn You a Haskell for Great Good!**
- Graham Hutton, **Programming in Haskell**
- O'Sullivan, Stewart, Goerzen, **Real World Haskell**

Grading

- 50% Machine Problems
  - 4-6 Haskell programming assignments
- 25% Midterm Exam
- 25% Final Exam (Cumulative)

Late Policy

- 7-day late pool, distributed however you like across labs (a day at a time)
- If you're out of late days, late submissions will not be accepted!

Exams

- Midterm tentatively scheduled for **March 8**
- Scores may be linearly scaled so that median/mean (whichever lower) is 75%

A: ≥ 90%

B: 80-89%

C: 70-79%

D: 60-69%

E: < 60%

§ “Programming Paradigms and Patterns”

Paradigm

- Model for how a program (in some language) is organized, expressed, or executed
- A given paradigm typically imposes some syntactic/semantic conventions or limits on programs
- E.g., procedural, imperative, object-oriented, functional
- We will be focusing on the **functional** paradigm

Why Functional?

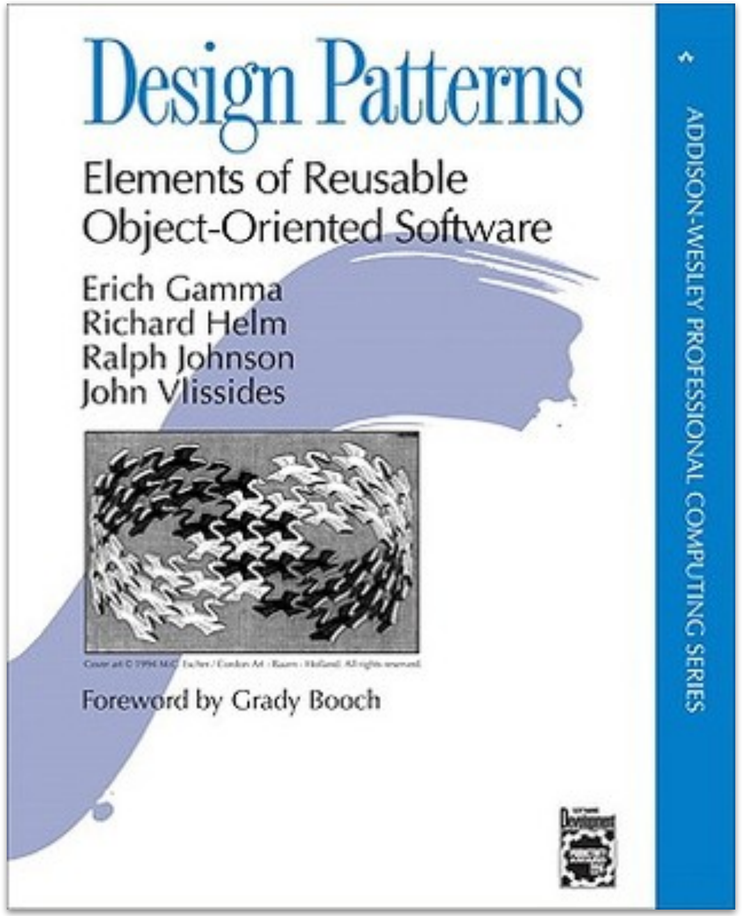
- Substantively different from the imperative paradigm, which is likely your "native" model
  - E.g., no state mutations → *referential transparency*
  - Arguably easier to reason about rigorously in some contexts (and many other purported benefits)
- You'll read a paper on this for Friday!

Pattern

- A reusable template for solving a common class of problem(s)
- May be paradigm/language specific, and typically as abstract as possible to encourage reuse

E.g., Imperative & OOP patterns

- Loops/Iterators for array, list, or collection traversal
- Encapsulation with setter/getter methods
- Singleton & Factory patterns
- Observer pattern, aka Publish/Subscribe



"Gang of Four" book

Our focus: Functional patterns

- Structural and Generative recursion
- Higher order functions
- Functors and Monads
- Monoids and Foldables
- Etc.

Haskell

- Our functional language of choice: **Haskell**
  - **Pure**: purely functional; side-effects are isolated
  - **Strongly typed**: types are checked/enforced at compile time
  - **Lazy**: expressions aren't evaluated until absolutely necessary
- Likely very different from another language you've used!

Why Haskell?

- It's fun, surprising, and powerful!
- Learning a (different) new language gives you an entirely new way to think about and tackle problems
  - Valuable, even if you don't actually code the solution up in said language

A Taste of Haskell

```haskell
fibs = 0 : 1 : zipWith (+) fibs (tail fibs)

primes = filterPrime [2..]
  where filterPrime (p:xs) =
          p : filterPrime [x | x <- xs, x `mod` p /= 0]

quicksort :: Ord a => [a] -> [a]
quicksort []     = []
quicksort (p:xs) = (quicksort lesser) ++ [p] ++ (quicksort greater)
    where lesser  = filter (< p) xs
          greater = filter (>= p) xs
```

Why take CS 340?

- You **love to program**
- You **love programming languages**
- You are frustrated with languages you currently know
- You want to learn new ways to reason about programming
  - Which will help in later classes and your career

Topics (not exhaustive)

- Functional programming
- Haskell Types and Typeclasses (like OOP on steroids)
- Higher Order Functions
- Functors and Monads
- Automated Property-Based Testing
- Concurrency and Software Transactional Memory

For Friday

- Read Hughes's "Why Functional Programming Matters" (at least sections 1 & 2, if you can get further, great!)
- Start reading "Learn You a Haskell"