CS 4110

Programming Languages & Logics

Lecture 1

Course Overview

JavaScript

```
[] + []
{} + []
[] + {}
{} + {}
```

From *Wat:*
`https://www.destroyallsoftware.com/talks/wat`

Java

```
class A {
    static int a = B.b + 1;
}

class B {
    static int b = A.a + 1;
}
```

Python

```
a = [1], 2
a[0] += [3]
```

Java and Scala

The Unsound Playground

io.livecode.ch/learn/namin/unsound

Fork me on GitHub

The Unsound Playground

We, Nada Amin and Ross Tate, broke the Java and Scala type systems!
Try it out for yourself by running the examples, which throw cast exceptions even though they contain no casts ↓
Read our paper *Java and Scala's Type Systems are Unsound* to learn how these examples work →
Come up with your own examples and use the save icon to update the URL to a permalink to your code ↑

OOPSLA '16

Distinguished Artifact Award

Which language would you like to break?
Java / Scala

Unsound.java in Java 8 (the first program we broke Java with, derived from legacy.scala)
Unsound.java in Java 9 (sometimes compilers incorrectly reject valid code)
Unsound9.java in Java 9 (we even broke Java in the future)
UnsoundSpec.java in Java 6 (incorrectly rejected unsound program that has been valid since Java 5 in 2004)
Nullless.java in Java 8 (this program has no occurence of `null`)
SingleParameters.java in Java 8 (every class and method in this program has exactly one parameter)

```
class Unsound {
  static class Constrain<A, B extends A> {}
  static class Bind<A> {
    <B extends A>
    A upcast(Constrain<A,B> constrain, B b) {
      return b;
    }
  }
  static <T,U> U coerce(T t) {
    Constrain<U,? super T> constrain = null;
    Bind<U> bind = new Bind<U>();
    return bind.upcast(constrain, t);
```

Nada Amin and Ross Tate:
`http://io.livecode.ch/learn/namin/unsound`

Design Desiderata

Question: What makes a good programming language?

Design Desiderata

Question: What makes a good programming language?

One answer: A good language is one people use.

Design Desiderata

Question: What makes a good programming language?

One answer: A good language is one people use.

Wrong! Is JavaScript bad? What's the best language?

Design Desiderata

Question: What makes a good programming language?

One answer: A good language is one people use.

Wrong! Is JavaScript bad? What's the best language?

Some good features:

- Simplicity (clean, orthogonal constructs)
- Readability (elegant syntax)
- Safety (guarantees that programs won't "go wrong")
- Modularity (support for collaboration)
- Efficiency (it's possible to write a good compiler)

Design Challenges

Unfortunately these goals almost always conflict.

- Types provide strong guarantees but restrict expressiveness.
- Safety checks eliminate errors but have a cost—either at compile time or run time.
- A language that's good for quick prototyping might not be the best for long-term development.

Design Challenges

Unfortunately these goals almost always conflict.

- Types provide strong guarantees but restrict expressiveness.
- Safety checks eliminate errors but have a cost—either at compile time or run time.
- A language that's good for quick prototyping might not be the best for long-term development.

A lot of research in programming languages is about discovering ways to gain without (too much) pain.

Language Specification

Formal Semantics: what do programs mean?

Three Approaches

- Operational
  - Models program by its execution on abstract machine
  - Useful for implementing compilers and interpreters
- Axiomatic
  - Models program by the logical formulas it obeys
  - Useful for proving program correctness
- Denotational
  - Models program literally as mathematical objects
  - Useful for theoretical foundations

Language Specification

Formal Semantics: what do programs mean?

Three Approaches

- Operational
  - Models program by its execution on abstract machine
  - Useful for implementing compilers and interpreters
- Axiomatic
  - Models program by the logical formulas it obeys
  - Useful for proving program correctness
- Denotational
  - Models program literally as mathematical objects
  - Useful for theoretical foundations

Question: Few real-world languages have a formal semantics. Why?

Formal Semantics

Too Hard?

- Real languages are complex
- Notation can gets very dense
- Sometimes requires developing new mathematics
- Not (yet?) cost-effective for everyday use

Overly General?

- Explains the behavior of a program on *every* input
- Most programmers are content knowing the behavior of their program on *this* input (or these inputs)

Okay, so who needs semantics?

Who Needs Semantics?

Unambiguous Description

- Anyone who wants to design a new feature
- Basis for most formal arguments
- Standard tool in PL research

Exhaustive Reasoning

- Sometimes have to know behavior on all inputs
- Compilers and interpreters
- Static analysis tools
- Program transformation tools
- Critical software

Course Staff

Instructor

Adrian Sampson (he/him)

Teaching Assistants

Omkar Bhalerao
Vivian Ding
Zak Kent
Megh Khaire
James Li
Stephanie Ma
Jan-Paul Ramos
Noah Rebei
Tia Vu

Prerequisites

Mathematical Maturity

- Much of this class will involve formal reasoning
- Set theory, formal proofs, induction

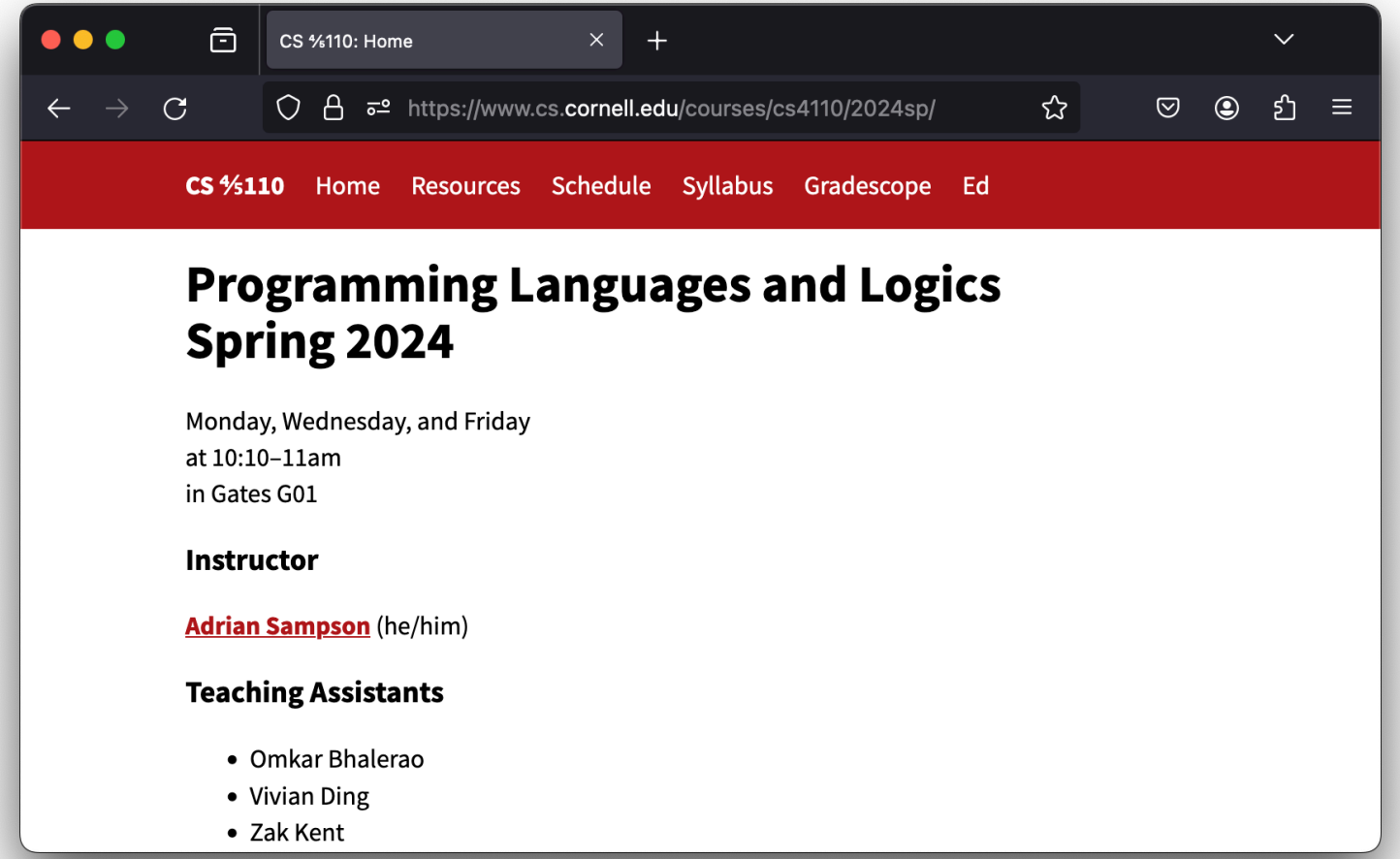
Programming Experience

- Comfortable using a functional language
- For Cornell undergrads: CS 3110 or equivalent

Interest (having fun is a goal!)

If you don't meet these prerequisites, please get in touch.

Course Website



http://www.cs.cornell.edu/courses/cs4110/2024sp/

Course Work

Homework

- 10 assignments, roughly one per week
- Can work with *at most one* partner
- Usually due on Thursday night at 11:59pm
- Automatic 24-hour extension without penalty
- Score capped at 85%
- Lowest score dropped

Course Work

Preliminary Exams (in-class)

- March 8
- April 19

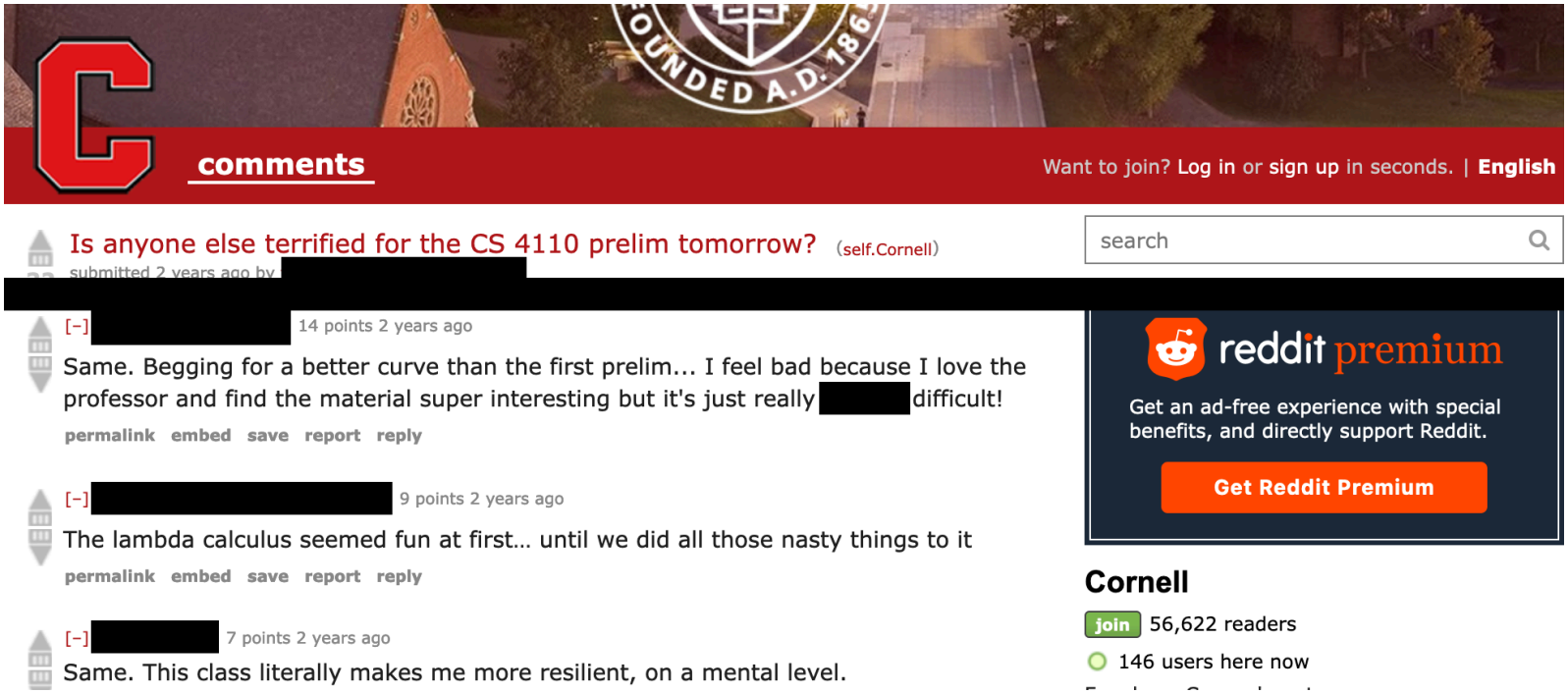
Final Exam

- Date TBD

Participation (5% of your grade)

- Introduction survey (out now!)
- Mid-semester feedback
- Course evaluation

The Difficulty You Can Expect



CS 4110 vs. CS 5110

The difference is:
CS 4110 is for undergrads (exclusively);
CS 5110 is for grad students (exclusively).

Everything else is the same, except that CS 5110 students do an extra “expanded version” of a solution after each exam.

Academic Integrity

Some simple requests:

1. You are here as members of an academic community. Conduct yourself with integrity.

2. Problem sets must be completed with your partner, and only your partner. You must *not* consult other students, alums, friends, Google, GitHub, StackExchange, Course Hero, etc.!

3. If you aren't sure what is allowed and what isn't, please ask.

Respect in Class

We hold all communication (in class & online) to a high standard for inclusiveness. It may not target anyone for harassment, and it may not exclude specific groups.

Examples:

- Do not talk over other people.
- Do not use male pronouns when you mean to refer to people of all genders.
- Avoid language that has a good chance of seeming inappropriate to others.

If anything doesn't meet these standards, contact the instructor.

Disabilities and Wellness

- I will provide accommodations to students with documented disabilities (e.g., physical, learning, psychiatric, vision, hearing, or systemic).
- If you are experiencing undue personal or academic stress at any time during the semester (or if you notice that a fellow student is), contact me, Engineering/A&S Advising, or Gannett.