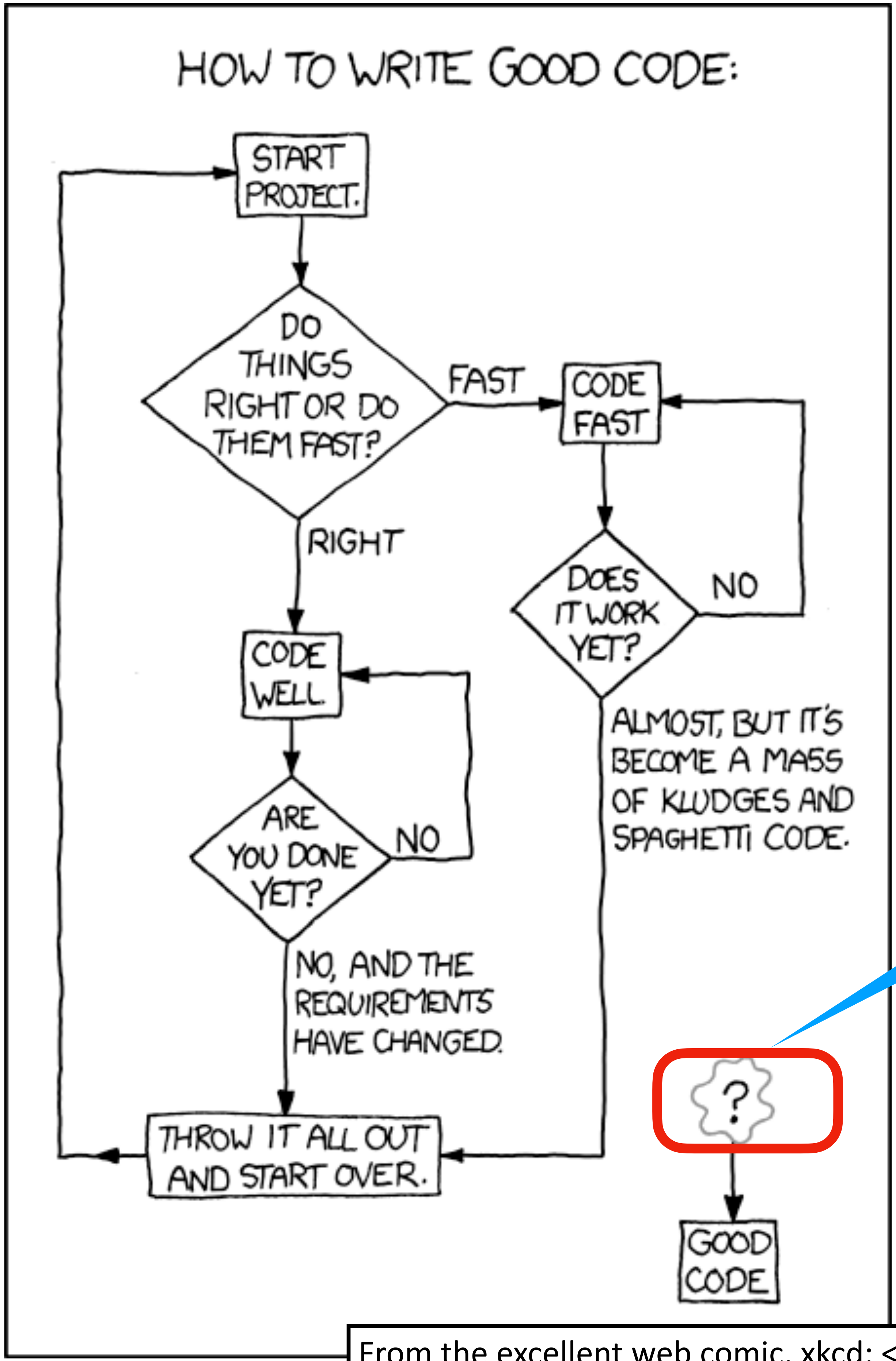
Introduction to OOAD

CSCI 4448/5448: Object-Oriented Analysis & Design

Welcome!

- This course explores Object-Oriented principles, patterns, theory, development languages, methods, processes, and related topics
- It's intended to give you a set of core design skills for use in designing and developing OO systems

To Teach You How to Be a Professional Software Engineer


HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
DOES IT WORK YET?
NO
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
RIGHT
CODE WELL.
ARE YOU DONE YET?
NO
NO, AND THE REQUIREMENTS HAVE CHANGED.
THROW IT ALL OUT AND START OVER.
?
GOOD CODE

This class!

From the excellent web comic, xkcd: <http://xkcd.com/844/>

Prerequisites

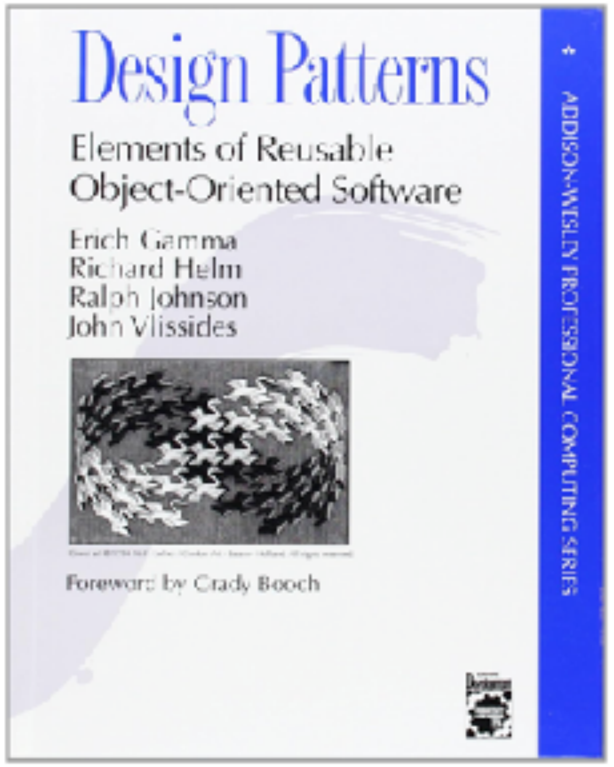
- Programming skill is required - duh
- Java will be covered, but rather quickly — all homework assignments are in Java
- Most other things you need to know, we'll review a bit – like Git and Regular Expressions, for instance – and I will provide resources for your external review

OOAD Class - Textbook

Readings will be required, and key elements will show up on quizzes and exams

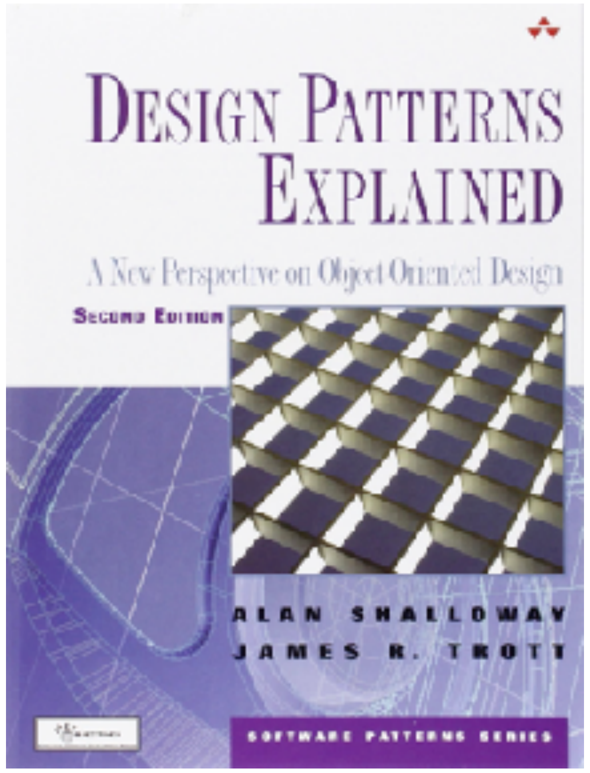
O'REILLY
Second Edition
Head First Design Patterns
Building Extensible & Maintainable Object-Oriented Software
Eric Freeman & Elisabeth Robson
with Kathy Sierra & Bert Bates
A Brain-Friendly Guide

Available in the O'Reilly e-books in the CU Library


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

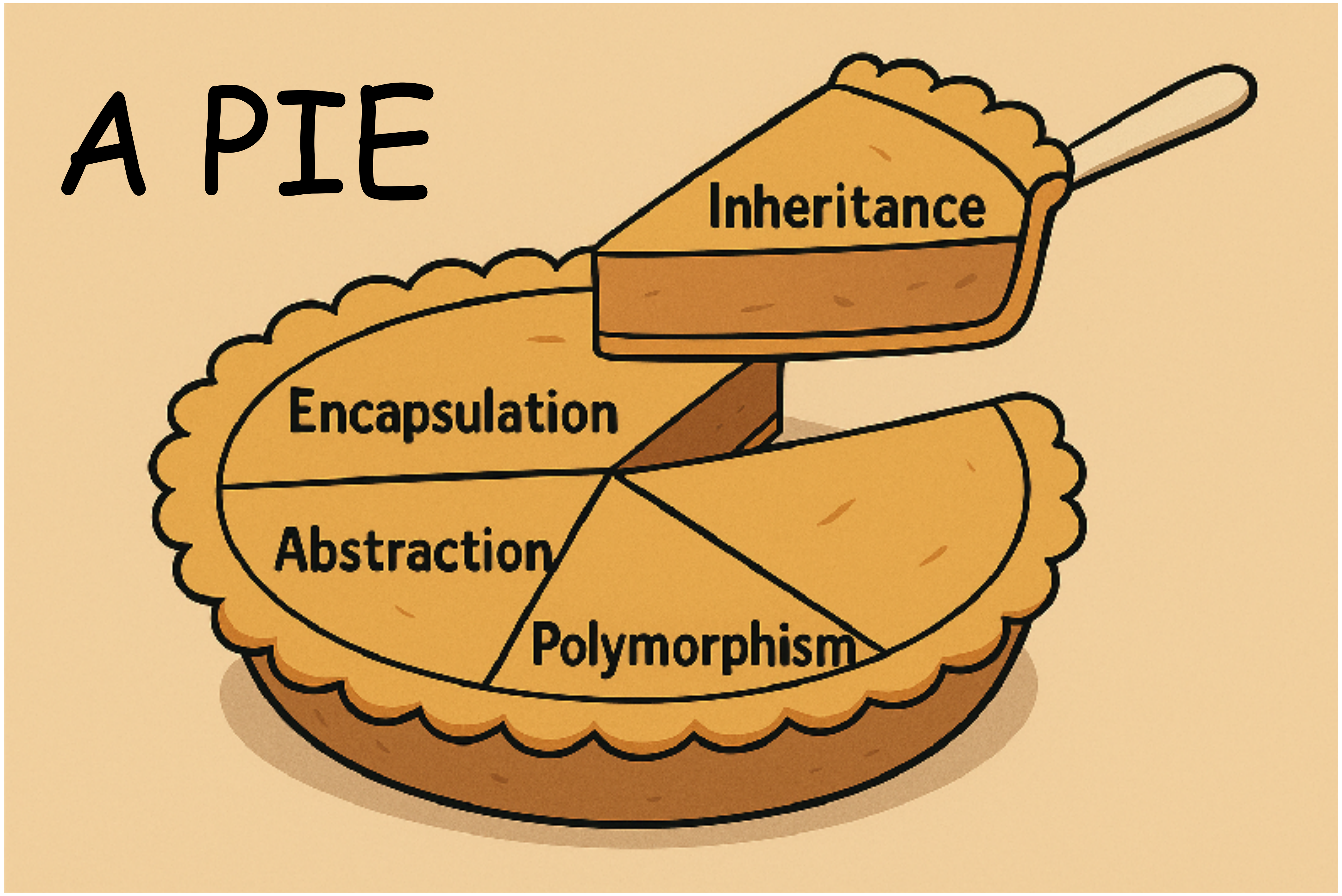
Olaf Musch
Design Patterns with Java
An Introduction
Springer

Available in Canvas

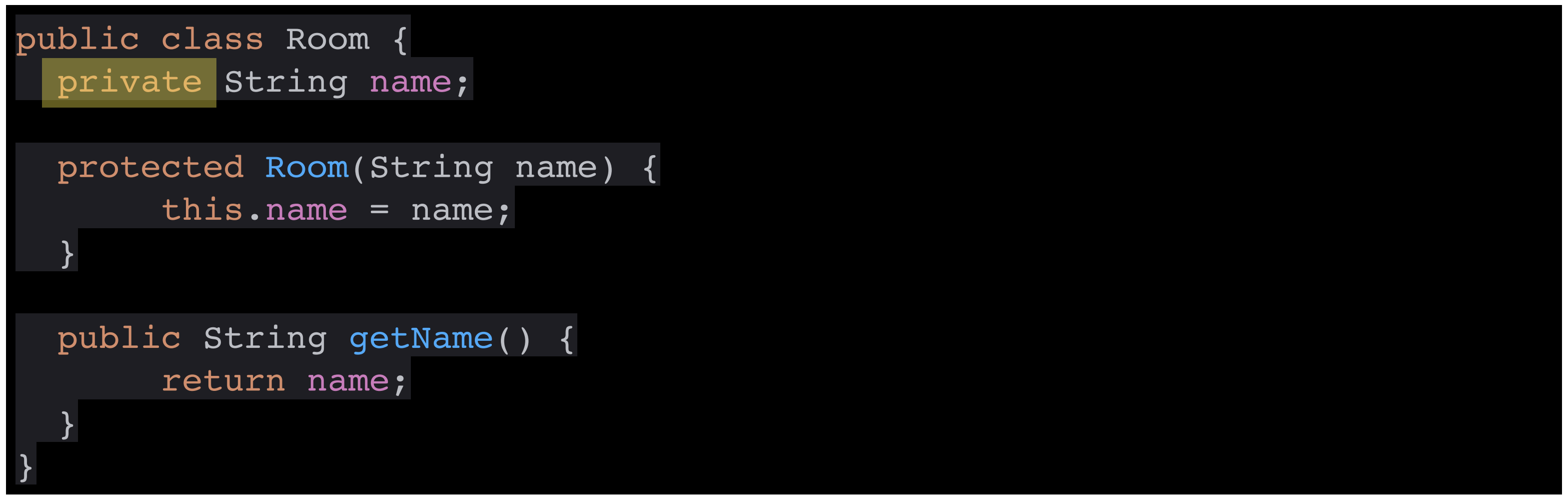

DESIGN PATTERNS EXPLAINED
A New Perspective on Object-Oriented Design
SECOND EDITION
ALAN SHALLOWAY
JAMES R. TROTT
SOFTWARE PATTERNS SERIES

Class Focus: OO Principles and Patterns

Principles First



Encapsulation

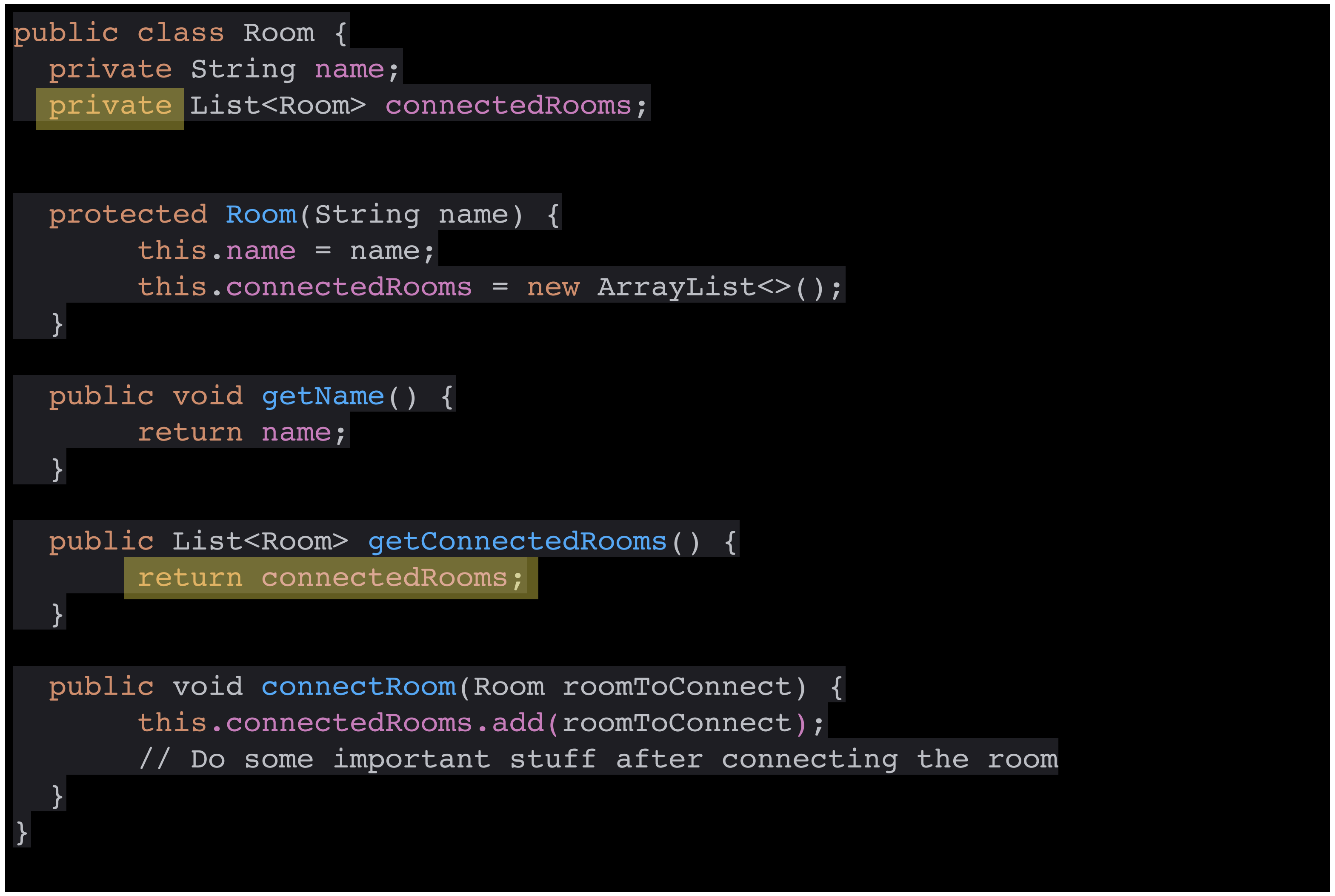


```
public class Room {
  private String name;

  protected Room(String name) {
        this.name = name;
  }

  public String getName() {
        return name;
  }
}
```

Encapsulated?



```
public class Room {
  private String name;
  private List<Room> connectedRooms;


  protected Room(String name) {
        this.name = name;
        this.connectedRooms = new ArrayList<>();
  }

  public void getName() {
        return name;
  }

  public List<Room> getConnectedRooms() {
        return connectedRooms;
  }

  public void connectRoom(Room roomToConnect) {
        this.connectedRooms.add(roomToConnect);
        // Do some important stuff after connecting the room
  }
}
```

Encapsulated?

More Principles

S 1. A class should have only one reason to change (Single Responsibility Principle)

O 2. Classes should be open for extension, but closed for modification (Open-Closed Principle)

L 3. Superclass objects should be replaceable by subclass objects without breaking functionality (Liskov Substitution Principle)

I 4. Clients should not have to implement methods in an interface they don't use (Interface Segregation Principle)

D 5. Depend on abstractions, not concrete classes (Dependency Inversion Principle)

6. Encapsulate what varies
7. Favor composition (delegation) over inheritance
8. Program to interfaces not implementations
9. Strive for loosely coupled designs between objects that interact
10. Only talk to your (immediate) friends (Law of Demeter, Principle of Least Knowledge)
11. Don't call us, we'll call you (Hollywood Principle)
12. Classes are about behavior, not specialization
13. Don't repeat yourself (DRY Principle)
14. You Aren't Going to Need It (or You Ain't Gonna Need It) (YAGNI)

Critique This Code

- Bad names
- Verbose code
-
-

```
class Character {
    int health =
DEFAULT_STARTING_HEALTH;

    void loseHealth(int loss) {
        health = health - loss;
    }

    bool isDead() {
        return health <= 0;
    }
}
```

```
class Cha {
    int health = 6;
    bool isAlive = tru

    void lh(int x) {
        health = health
        if (health <= 0
            isAlive = fa
        }
    }

    bool isDead() {
        if (!isAlive) {
            return true;
        else {
            return false
        }
    }
}
```

OO Patterns

Patterns we will cover in detail:

- Strategy
- Observer
- Decorator
- Factory
- Builder
- Singleton
- Adapter, Facade
- Template

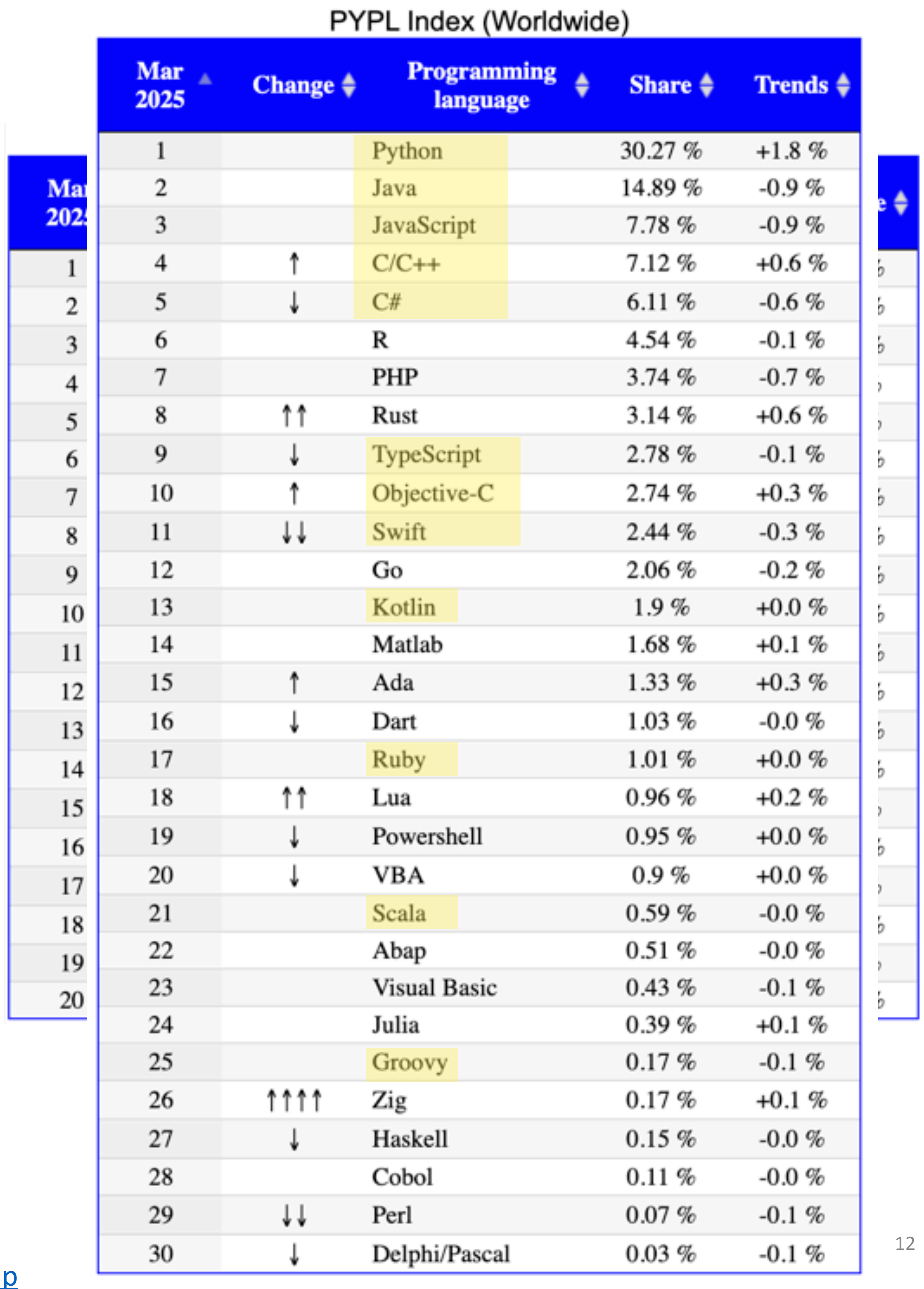
Patterns we might visit with:

- Iterator, Composite
- State
- MVC and Variations
- Command
- Bridge
- Flyweight
- Interpreter
- Mediator
- Memento
- Prototype
- Visitor

Other Topics

- Quick visits to Git, Java, UML
- Dependency Injection
- Refactoring and Code Smells
- Test Driven Development
- Behavior Driven Development
- Design Techniques
- Architecture
- Other TBD…

OO Relevance

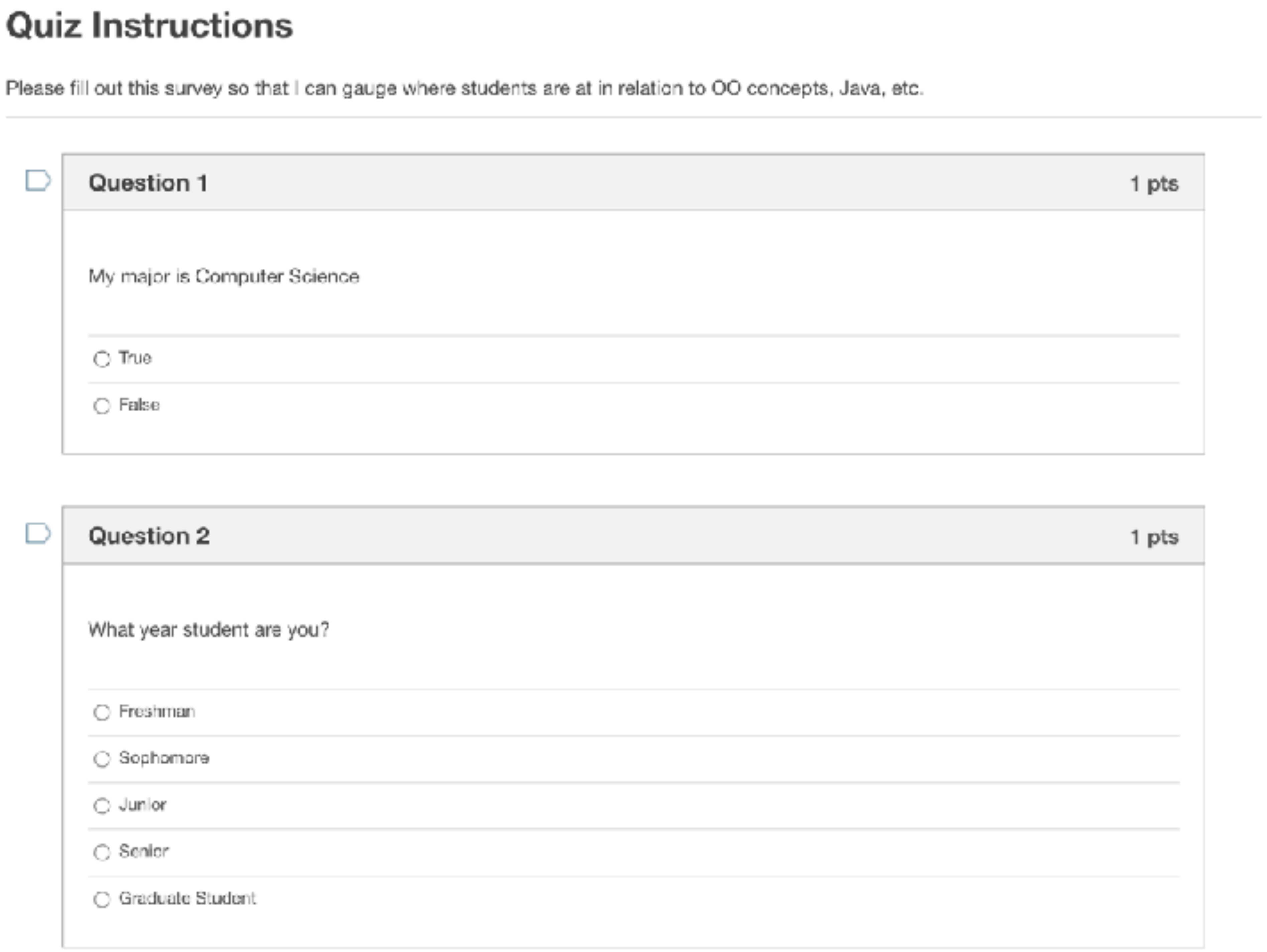


PYPL Index (Worldwide)

Mar 2025	Change	Programming language	Share	Trends
1		Python	30.27 %	+1.8 %
2		Java	14.89 %	-0.9 %
3		JavaScript	7.78 %	-0.9 %
4	↑	C/C++	7.12 %	+0.6 %
5	↓	C#	6.11 %	-0.6 %
6		R	4.54 %	-0.1 %
7		PHP	3.74 %	-0.7 %
8	↑↑	Rust	3.14 %	+0.6 %
9	↓	TypeScript	2.78 %	-0.1 %
10	↑	Objective-C	2.74 %	+0.3 %
11	↓↓	Swift	2.44 %	-0.3 %
12		Go	2.06 %	-0.2 %
13		Kotlin	1.9 %	+0.0 %
14		Matlab	1.68 %	+0.1 %
15	↑	Ada	1.33 %	+0.3 %
16	↓	Dart	1.03 %	-0.0 %
17		Ruby	1.01 %	+0.0 %
18	↑↑	Lua	0.96 %	+0.2 %
19	↓	Powershell	0.95 %	+0.0 %
20	↓	VBA	0.9 %	+0.0 %
21		Scala	0.59 %	-0.0 %
22		Abap	0.51 %	-0.0 %
23		Visual Basic	0.43 %	-0.1 %
24		Julia	0.39 %	+0.1 %
25		Groovy	0.17 %	-0.1 %
26	↑↑↑↑	Zig	0.17 %	+0.1 %
27	↓	Haskell	0.15 %	-0.0 %
28		Cobol	0.11 %	-0.0 %
29	↓↓	Perl	0.07 %	-0.1 %
30	↓	Delphi/Pascal	0.03 %	-0.1 %

http://statisticstimes.com/tech/top-computer-languages.php

Take the Experience Poll



Quiz Instructions

Please fill out this survey so that I can gauge where students are at in relation to OO concepts, Java, etc.

Question 1 **1 pts**

My major is Computer Science

- ○ True
- ○ False

Question 2 **1 pts**

What year student are you?

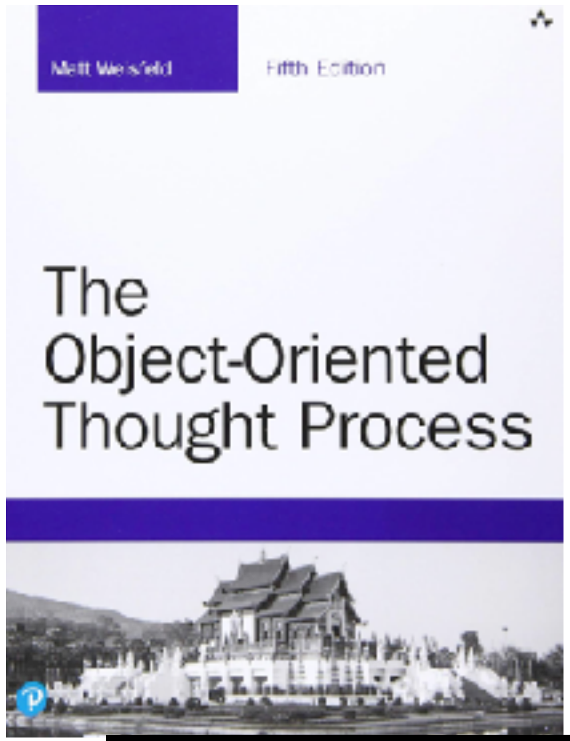
- ○ Freshman
- ○ Sophomore
- ○ Junior
- ○ Senior
- ○ Graduate Student

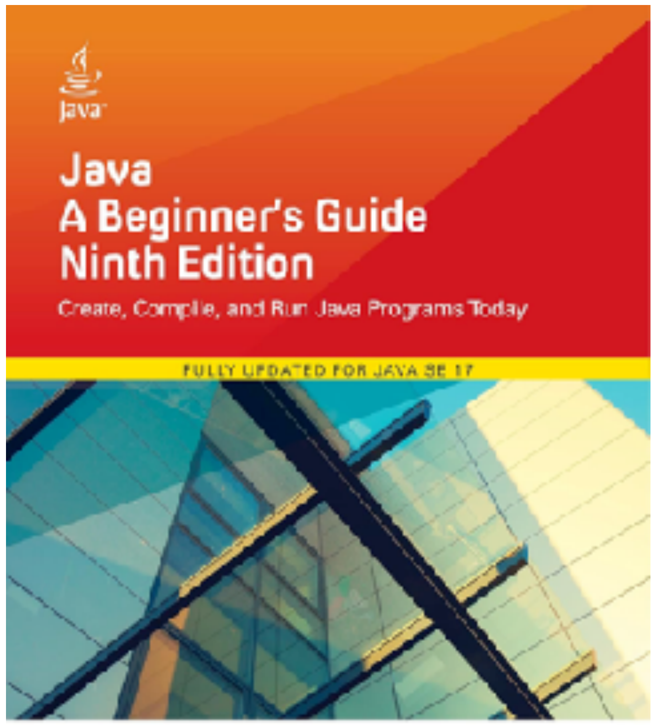
Summary: Goals of the Class

- Tools for your programming tool kit!
- Provide students with knowledge and skills in:
  - **Object-oriented concepts and patterns**
  - **OO analysis, design and implementation techniques**
  - **OO design methods (software life cycles)**
- Students will get an overview of OO software development as a software engineering process
- You will also gain experience with OO programming
- And you'll be better prepared for both new development and supporting legacy code

What should I review? (Optional)

- **OO:** We will review this, but you should know the basics of what object-oriented languages are.
- **Java:** If you haven't used Java (the main development language for the class), I'd look for an online tutorial (like https://docs.oracle.com/javase/tutorial/)
- **Git:** If you're not Git savvy, look at a tutorial like https://guides.github.com/activities/hello-world/




Java
Java A Beginner's Guide Ninth Edition
Create, Compile, and Run Java Programs Today
FULLY UPDATED FOR JAVA SE 17

Available in the O'Reilly e-books in the CU Library

Available in Canvas

- CU library site: https://libguides.colorado.edu/strategies/ebooks