Introduction to OOAD

CSCI 4448/5448: Object-Oriented Analysis & Design

Lecture 2

Acknowledgement & Materials Copyright

- I'd like to start by acknowledging Dr. Ken Anderson
- Ken is a Professor and the Chair of the Department of Computer Science
- Ken taught OOAD on several occasions, and has graciously allowed me to use his copyrighted material for this instance of the class
- Although I will modify the materials to update and personalize this class, the original materials this class is based on are all copyrighted © Kenneth M. Anderson; the materials are used with his consent; and this use in no way challenges his copyright

Learning Objectives

- Clarify the class focus and relevance
- Determine the class fit to the student's learning goals, experience, and skills

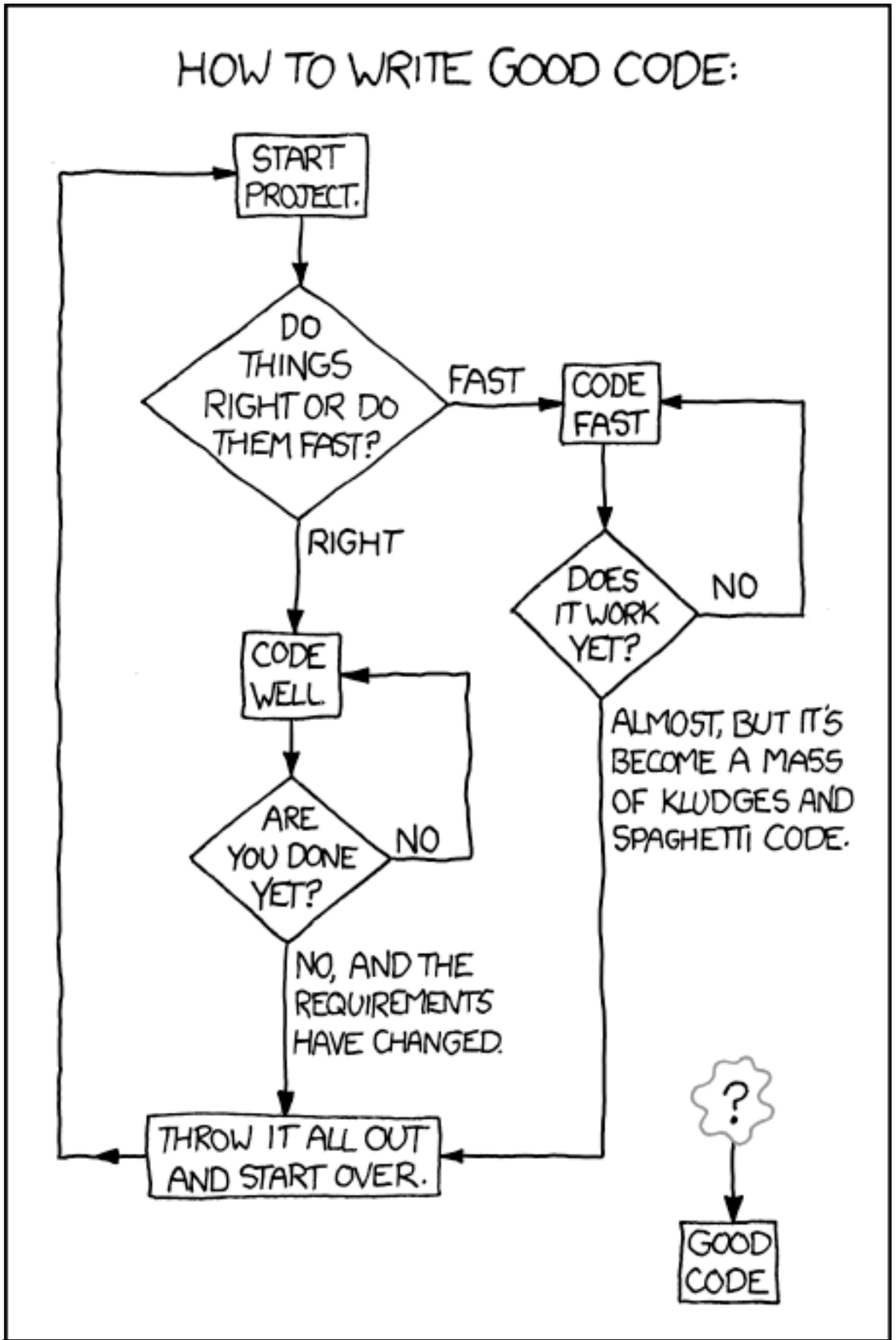
Welcome!

- Welcome to Object-Oriented Analysis and Design (aka OOAD)
  - I'm looking forward to working with you!
- 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
- Look hard at the material we cover, and we'll talk about whether this course is right for you

Good Code

This class teaches a style of software design that can help you reach the box labelled “Good Code”…

Software Design is not completely a black art… there are design techniques that lead to better results when applied in support of creative expression.



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

Prerequisites

- Programming skill is required
  - My lectures will primarily use Java examples (like the textbook)
  - I will include Python examples for most things as well as a counterpoint
  - Your class exercises will be partially coding in Java, but partially in languages of your choosing
  - I don't generally ask you to write code on exams, but I may ask for pseudo-code, so again, programming needs to be a skill you have
- Most other things you need to know, we'll review – like Git, for instance – and I will provide resources for your external review
- If you have a couple years of programming experience, Java and Python are high level languages that shouldn't be hard for you to learn, but…
- You'll do that on your own. My lessons will be focused mostly on the OO features of the languages. This is not a Java or Python programming class.

OOAD Class - Textbook

- Textbook:
  - Head First Design Patterns
  - By Bates, Sierra, Freeman, & Robson
  - O'Reilly Media
  - 10th Anniversary Edition - 2014 (updated for Java 8)
- I will be visiting most of the Java-based content in this book
- There will be other materials I will point you at as needed
- Readings will be required, and key elements will show up on quizzes and exams

- If you're going to stay in OOAD, **get the book** (or access to the book)

OOAD Class Focus

- Quick review of OO basics
- Quick visits to Git, Java, Python, UML
- Large part of class: OO patterns (right) and OO principles (next page)
- Other topics (coverage depends on pace of above)
  - Design Techniques
  - OR Mapping
  - Dependency Injection
  - Refactoring and Code Smells
  - Reflection
  - Test Driven Development
  - Software Project Management
  - Other TBD…

Patterns we will cover in detail:

- Strategy
- Observer
- Decorator
- Simple Factory, Factory, Abstract Factory
- Singleton
- Command
- Adapter, Façade
- Template
- Iterator, Composite
- State
- Proxy
- MVC and Variations

Patterns we will visit with:

- Bridge
- Builder
- Flyweight
- Interpreter
- Mediator
- Memento
- Prototype
- Visitor

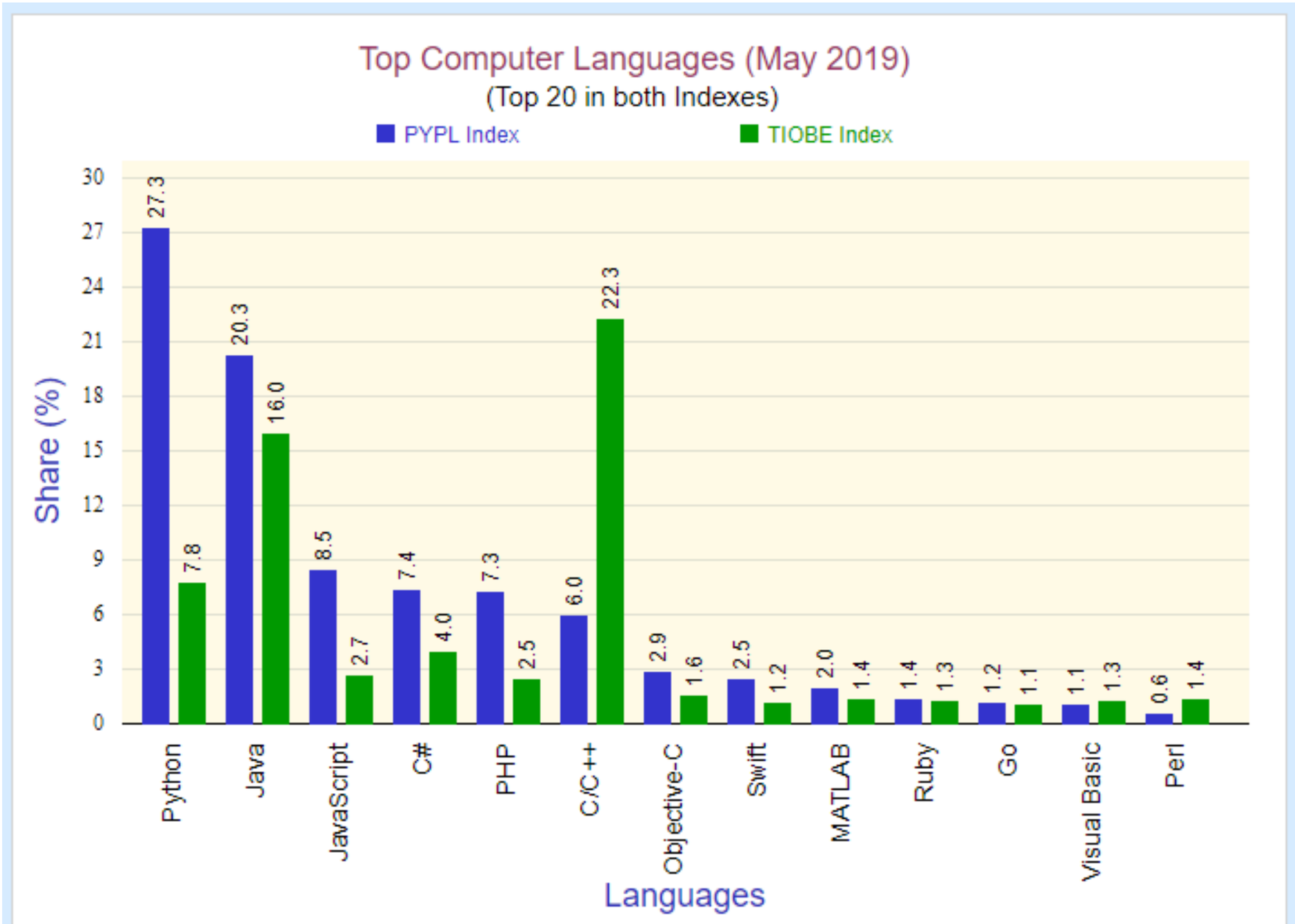
OO Principles

S
O
L
I
D

1. A class should have only one reason to change (Single Responsibility Principle)
2. Classes should be open for extension, but closed for modification (Open-Closed Principle)
3. Superclass objects should be replaceable by subclass objects without breaking functionality (Liskov Substitution Principle or LSP)
4. Clients should not depend on methods in an interface they don't use (Interface Segregation Principle)
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)

OO Relevance

- TIOBE Top 4
  - Java, C, C++, Python
- PYPL Top 4
  - Python, Java, JavaScript, C#
- Most popular languages are OO (at least in part)
- May 2019



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

Summary: Goals of the Class

- 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 should view OO software development as a software engineering process that has well-defined stages with each stage requiring specific tools and techniques
- You will also gain experience with OO programming, hopefully with languages you're interested in learning
- And you'll be better prepared for both new development and supporting legacy code