CSCA08 Fall 2017

Week 10 - Encapsulation & Abstraction, OOP Design

Brian Harrington & Marzieh Ahmadzadeh

University of Toronto Scarborough

November 13 - 17, 2017

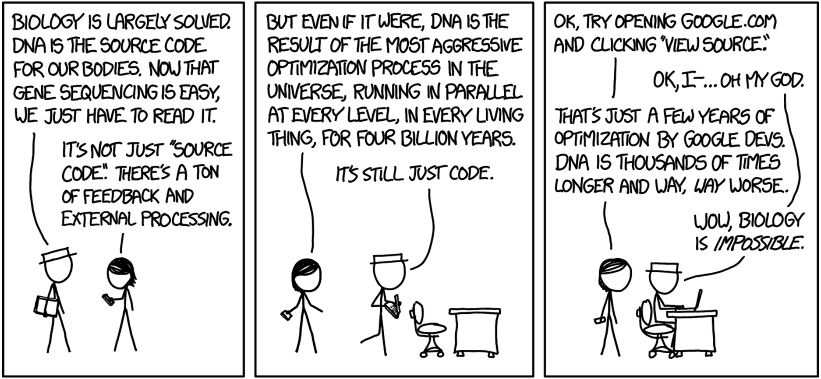
UML

- **Variables**
- Underscore in front of name means private
- Need to show type of all variables
- Include variables in the class where they can be found
- **Methods**
- Underscore in front of name means private (helper)
- Need to show type contract of all methods
- Include methods in the class where they can be found
- Include `__init__`, but (usually) not other system methods
- Don’t need to show all getter/setter methods for private variables (assumed they’re present)
- **Classes**
- Need to show all classes that you will write (don’t show built-in/imported)
- Show relationships (with name) + cardinality between classes

Encapsulation & Abstraction

- Encapsulation
  - Grouping together data and functionality into a single class
  - Work with high level objects (Events, People, Buildings) instead of low-level details (ints, floats, dicts)
  - Simplify
- Abstractions
  - Hiding implementation details from outside users/code
  - Makes it easier to change code in the future
  - Reduce later dependencies
  - Simultaneous development

Break


BIOLOGY IS LARGELY SOLVED. DNA IS THE SOURCE CODE FOR OUR BODIES. NOW THAT GENE SEQUENCING IS EASY, WE JUST HAVE TO READ IT.
IT'S NOT JUST "SOURCE CODE." THERE'S A TON OF FEEDBACK AND EXTERNAL PROCESSING.
BUT EVEN IF IT WERE, DNA IS THE RESULT OF THE MOST AGGRESSIVE OPTIMIZATION PROCESS IN THE UNIVERSE, RUNNING IN PARALLEL AT EVERY LEVEL, IN EVERY LIVING THING, FOR FOUR BILLION YEARS.
IT'S STILL JUST CODE.
OK, TRY OPENING GOOGLE.COM AND CLICKING "VIEW SOURCE."
OK, I–...OH MY GOD.
THAT'S JUST A FEW YEARS OF OPTIMIZATION BY GOOGLE DEVS. DNA IS THOUSANDS OF TIMES LONGER AND WAY, WAY WORSE.
WOW, BIOLOGY IS IMPOSSIBLE.

Abstract Data Types

- Data Type: information stored and operations that can be performed
    - We’ve seen lots of these: str, float, list, dict, etc
- Abstract Data Type: Independent of the implementation

Why ADTs?

- User doesn’t care how it works
- Other developers shouldn’t [need to] care about implementation details
- Examples:
    - dictionaries
    - lists
    - most things you interact with in the real world