CSCI 334:
Principles of Programming Languages

Lecture 21: Packages/OOP

Instructor: Dan Barowy

Williams

Topics

Packages

Object-Oriented Programming

Your to-dos

1. Study for optional quiz **on Thursday**.
2. Lab 10 (project checkpoint #2), **due Monday, May 13** (group project).

Final project timeline

1. ~~Minimally working version (Lab 9), **due Mon 4/29**~~
2. Mostly working version (Lab 10), **due Mon 5/13**
3. Project + video presentation, **due Mon 5/20** (last day of exams)

Ward Prize nomination deadline: **May 6**

Language Package Framework

A **language package framework** is a repository of **software libraries**, together with a **distribution mechanism**, to simplify finding and incorporating **third-party software** into your code.

A good package framework makes using a language more **productive** and **fun**!

5

Popular Package Frameworks

Java: Maven

.NET: NuGet

Python: pip

Ruby: gem

Rust: crates

Go: cargo

Tons more!

6

https://www.nuget.org/

7

(code)

8

Programming in the small

9

10

11

Programming in the large

12

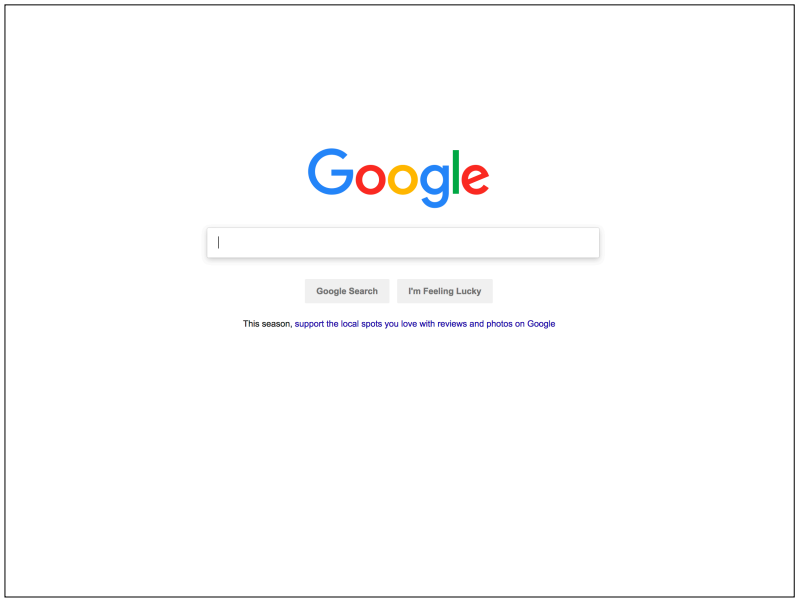
13

14

15

TensorFlow

16



17

What languages?

18

Java

19

C++

20

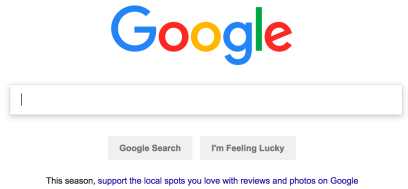
Ruby

21

TensorFlow

C++, Python

22



Java

23

Object-Oriented Programming

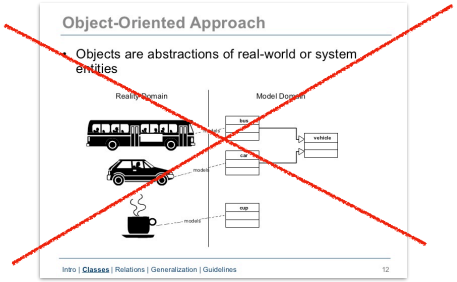
24

Object-Oriented Programming

- OOP is both a **language design** and a **way of programming** (OO design).
- OOP is possibly the **most impactful** development in the history of programming languages.

What OOP is Not

- Many, many instructors introduce OOP as a way of "simulating the world."



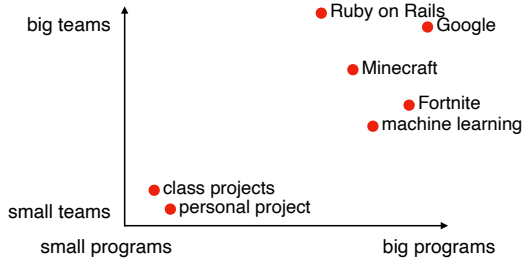
- This view **entirely misses the point** of OOP!

"Object-oriented programming is a **solution** to **complexity**."

—Dan Ingalls, inventor of the SmallTalk language

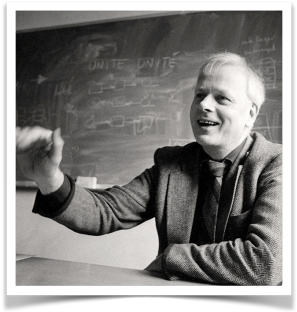
What OOP is

- Object-oriented programming is actually about **scalability**.
- The original motivation was motivated by two questions:
  - How do we manage **big codebases**?
  - How do big teams of programers **collaborate effectively**?



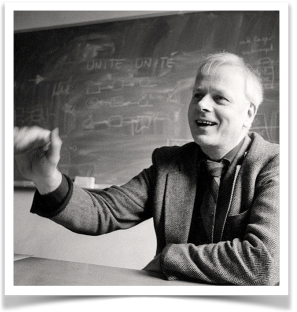
History

- First language recognizable as OO: Simula-67.
- Developed by Kristen Nygaard and others at the Norwegian Computing Center.
- Grew out of frustrations using ALGOL.
- Original plan was to add an “object” library, inspired by C.A.R. Hoare’s “record classes”.
- It was eventually realized that a fundamentally different way of structuring a program was possible; Simula became its own language.



History

- But Simula-67 was not the most influential OO language.
- That language was…



Smalltalk

Alan Kay
Essentially invented the laptop/tablet (“Dynabook”)

Turing Award

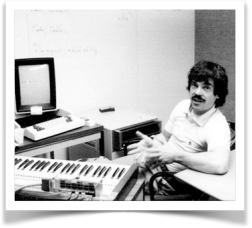
Dan Ingalls
Essentially invented object oriented programming

Grace Murray Hopper Award

Adele Goldberg
Essentially invented graphical user interfaces

ACM Software Systems Award

Smalltalk

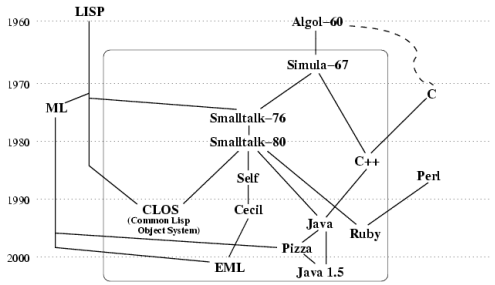


- First mainstream OO success: Smalltalk
- Developed by Alan Kay, Dan Ingalls, and Adele Goldberg at Xerox PARC and later Apple Computer.
- Used to implement major components of the groundbreaking Xerox Alto computer: OS, compiler, GUI, applications.
- Highly influential. E.g., C++, Java, Ruby, etc.

Smalltalk

One of the things they showed me was object orienting programming […] within you know ten minutes it was obvious to me that all computers would work like this some day.

Smalltalk



OK, really, what is OO?

Object-oriented programming is composed primarily of four key language features:

1. Abstraction
2. Dynamic dispatch
3. Subtyping
4. Inheritance

Purpose: polymorphism at scale

OK, really, what is OO?

Object-oriented programming is composed primarily of four key language features:

1. Abstraction
2. **Dynamic dispatch**
3. Subtyping
4. Inheritance

This is the killer feature.

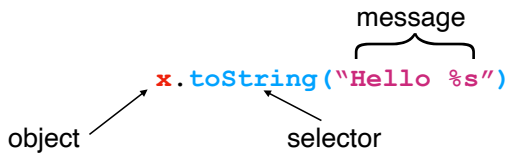
Purpose: polymorphism at scale

Dynamic Dispatch

(the secret to understanding how Java, Python, Ruby, etc. work)

Dynamic Dispatch

- **Dynamic dispatch** is the OO mechanism for **polymorphism**.
- Functions ("**methods**") are always bound to an object (or class)
- A method is called ("**dispatched**") by sending a "**message**" to the "**selector**" of an object.



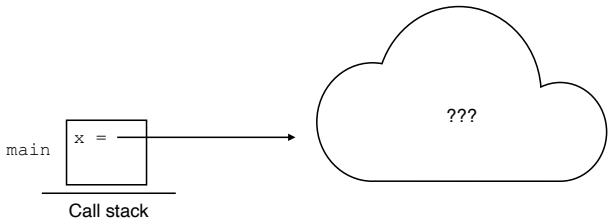
Dynamic Dispatch

- Suppose we have:

```
class Number {
  int value;

  public Number(int v) {
    value = v;
  }

  public int getValue() {
    return value;
  }

  public String squee() {
    return "squee!";
  }
}
```

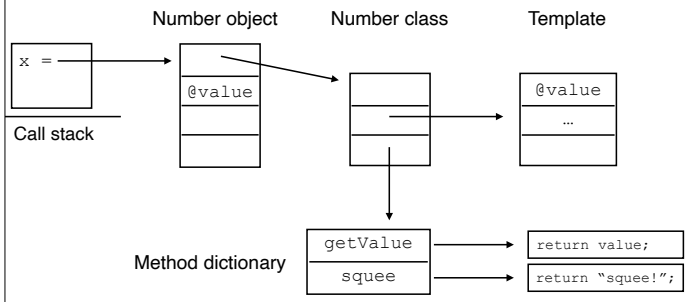

Dynamic Dispatch

- `x` is a `Number`.
- How does an object work?

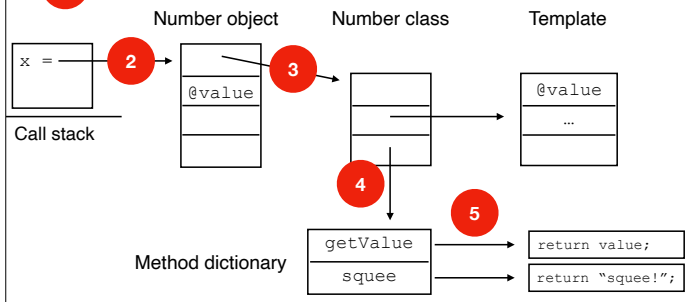


Dynamic Dispatch

- Dynamic dispatch is an **algorithm** for finding the **implementation** of a given **selector** (i.e., method).



1. Call `x.getValue`
2. `x.getValue` message dispatched to `x`
3. `x.getValue` message forwarded to `Number`
4. `x.getValue` message lookup in method dictionary
5. `x.getValue` executed.



"Object-oriented programming is a **solution** to **complexity**"

Recap & Next Class

This lecture:

Packages

OOP

Next lecture:

SCS / How to give a good talk