Australian National University

Software Design

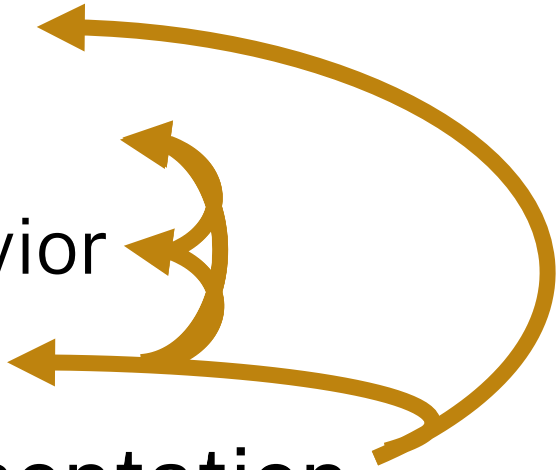
S5

Structured Programming 1110/6710

3 . 5 . 7 . 9

High-Level Steps to Working Software

1. Design
   a) Data
   b) Behavior
2. Tests
3. Implementation



Data + Behavior

What distinct things do we care about?

What aspects of them do we care about?

How many might there be?

How can we interact with those things?

What do they do?

OO: Are there any subtyping relationships?

Context Matters!

University ID

Credit Score

Genes

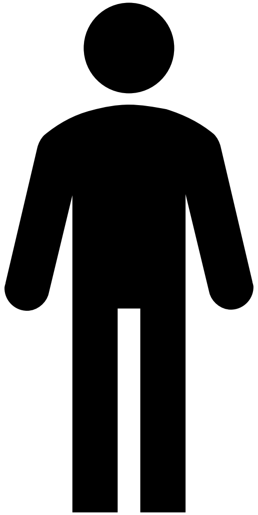
Demerit Points

Privacy Settings

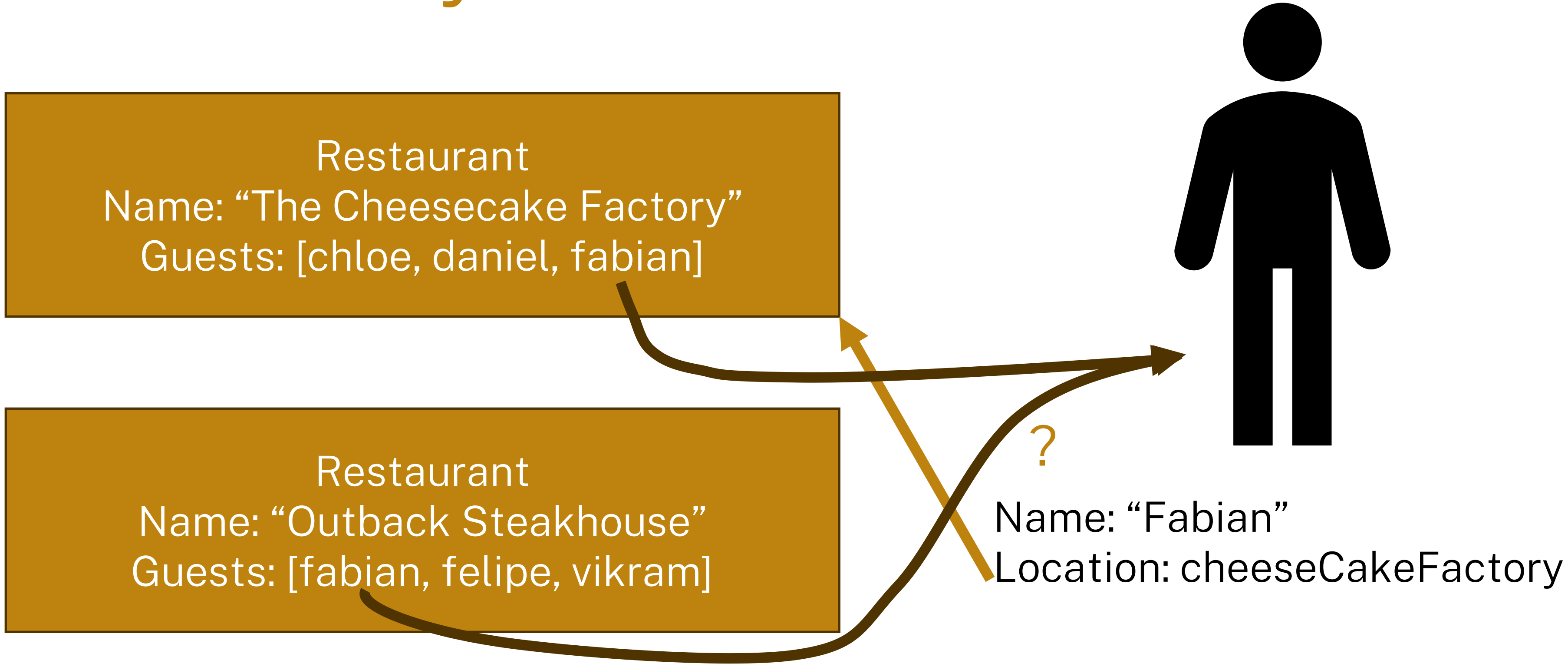
Social Security Number

Interests

Tax File Number



Consistency


Restaurant
Name: “The Cheesecake Factory”
Guests: [chloe, daniel, fabian]
Restaurant
Name: “Outback Steakhouse”
Guests: [fabian, felipe, vikram]
?
Name: “Fabian”
Location: cheeseCakeFactory

Documentation

Types and names are a great start, but do not capture everything.

Make assumptions explicit!

- Is the value of a field tied to some other value?
- Is a number non-zero/within a certain range?
- Is a reference always non-null?
- Can a String be empty?

...

Some Principles (Ousterhout)

- **Deep “modules”** (method, class, package, or module)
  - Simple interfaces* (narrow)
  - Encapsulate lots of complexity (depth)
  - General-purpose
- Prefer **simple interface** over simple implementation
- Design **errors out of existence**
- Design for **ease of reading**, not ease of writing
- Extra: Don’t Repeat Yourself (**DRY**)

* Interfaces in the broad sense, not just the Java keyword

Key Goal

SIMPLICITY!

- Avoid unnecessary complexity
- Don't expose things that do not need to be exposed
- Make code readable and maintainable by other programmers (includes future you!)