Software Architectures

From Design Patterns to Enterprise Architecture

Introduction and Basic Principles

Why be a software architect?

Why be a software architect?
Construction

- When constructing a system...
  - How do you decide what is a good idea?
  - How do you keep it scalable?
  - How do you keep up with change?

Why be a software architect?
Leadership

- When recruiting a team...
  - How do you know if someone writes good code?
  - How do you pick the right people?
- When leading a team...
  - What do you set as a standard for good practice?
  - How do you evaluate people's work?

Why be a software architect?
Evaluation

- You will often get into a situation where you are asked to assess the quality of software...
  - Your company acquires another company
  - You are getting code from a subcontractor
  - You are trying to find a good software library
  - You have to tell other team members what they are doing wrong...
- How are you doing this?

Discussion

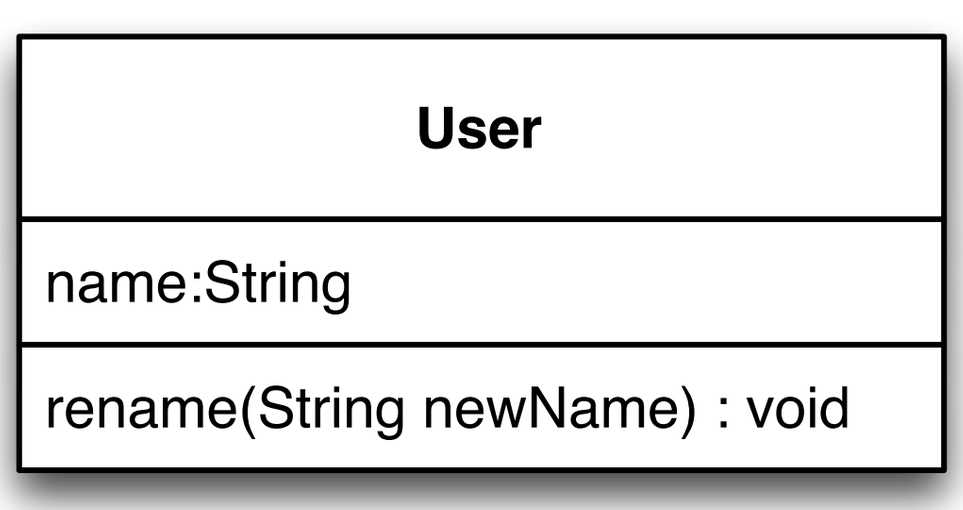
Carreer patterns

Guiding principles

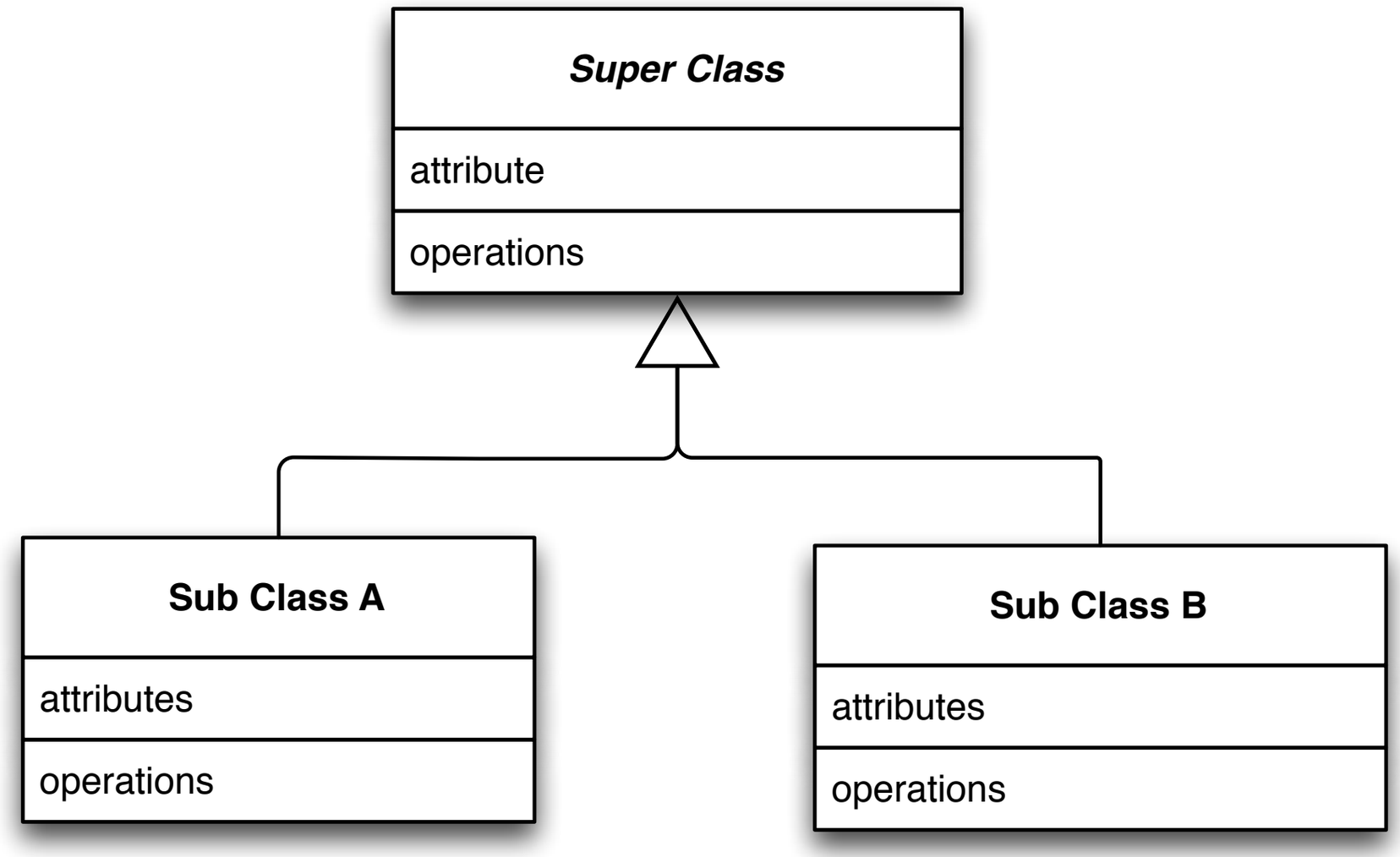
Short review on object oriented-principles

- Let's quickly review basic object-oriented principles
- Do you know all of them?
  - Classes
  - Objects
  - Methods
  - Inheritance
  - Subtype Polymorphism

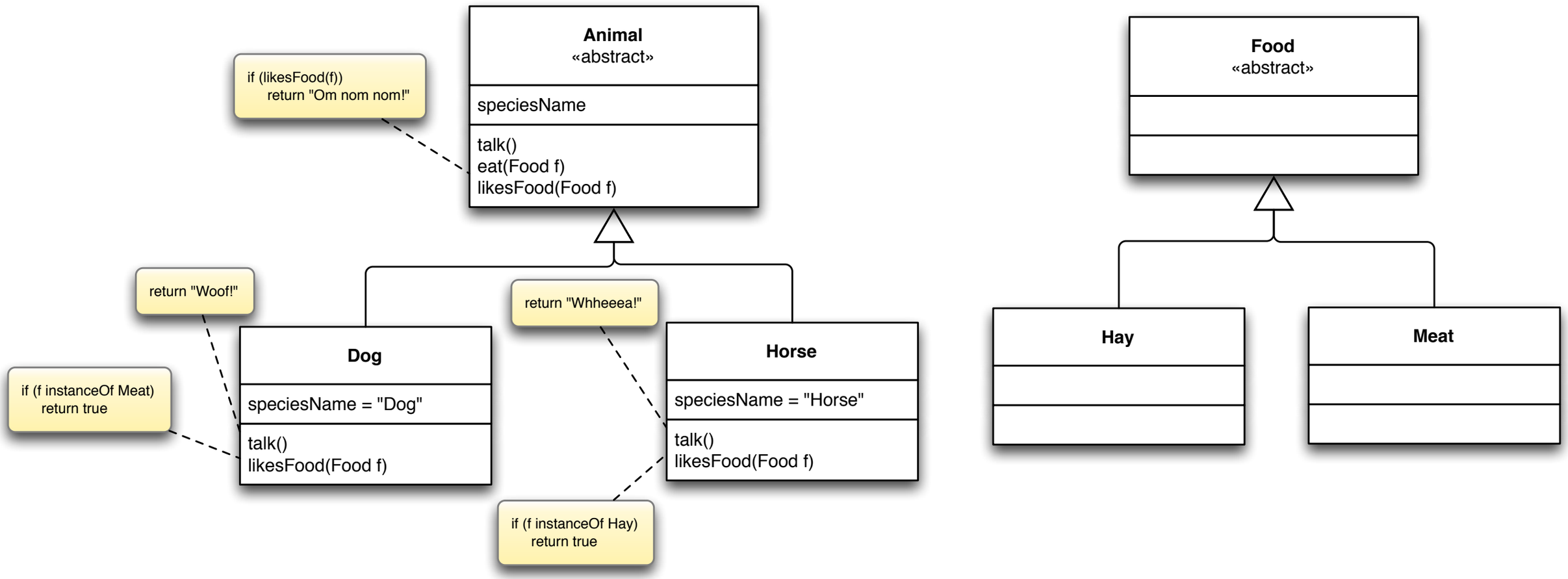
Classes, Objects & Methods



Inheritance



Subtype Polymorphism



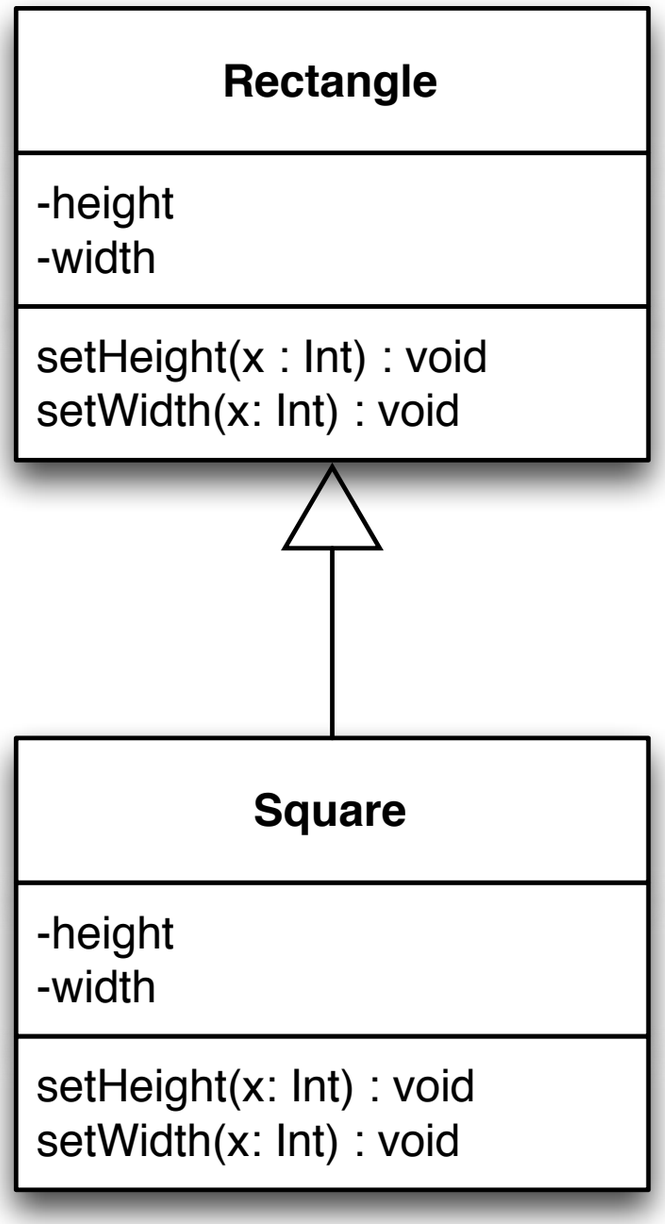
- So, this is your toolbox... but, how should you use it?

Low Coupling
High Cohesion

- Coupling:
  a measure of the “connectedness” of two components of a system
- Cohesion:
  a measure of the degree that elements of a component belong together
- Prefer low or loose coupling and strive for high cohesion

Liskov Substitution Principle

- Every objects of a class T may be replaceable with objects of a class S which is a subclass of T ... substitution
- That means that each overridden method must adhere to the same guarantees as the super method
- Good in theory, hard in practice



Open-Closed Principle

- Open for extension
- Closed for modification

- Altering behavior without modifying source code
- Reacting to changing requirements

Single Responsibility Principle

- We postpone this one until next week

Discussion

Questions & examples

One step back:
A look at software quality

Quality Criteria (ISO 9126)

- Correctness
- Robustness
- Extendibility
- Reusability
- Compatibility
- Efficiency
- Portability
- Ease of use
- Functionality

Quality Criteria (ISO 9126)

- Correctness
- Robustness
- Extendibility
- Reusability
- Compatibility
- Efficiency
- Portability
- Ease of use
- Functionality

Most of them can be achieved by following design principles!

Testing for correctness

- Fortunately, you can test for correctness
- Unit tests for classes or sets of classes
- Integration tests for components and their interaction
- System test for testing complete systems

Testing other quality criteria

- How do you test for
  - Robustness?
  - Extendibility?
  - Reusability?
  - Compatibility?
  - Efficiency?
  - Portability?
  - Ease of use?
  - Functionality?

Discussion

How are we doing this?

Testing for other quality criteria

Criteria	Test
Robustness	Simulated system failures, Wrong entries, Test failure handling, Test for system recovery
Extendibility	Make extensions
Reusability	Reuse parts or complete systems
Compatibility	Check import/export, file formats, interfaces
Efficiency	Test for large data volumes or large amounts of request
Portability	Test of (all) target platforms
Ease of use	User acceptance tests, test for learnability
Functionality	Manual or automated system tests

Wrap up

- Now you know why the contents of this lecture are important
- If you didn't know all the object-oriented concepts, now you should
- You learned some of the basic principles of object-oriented software development
- We reviewed software quality and how we can check a system for it