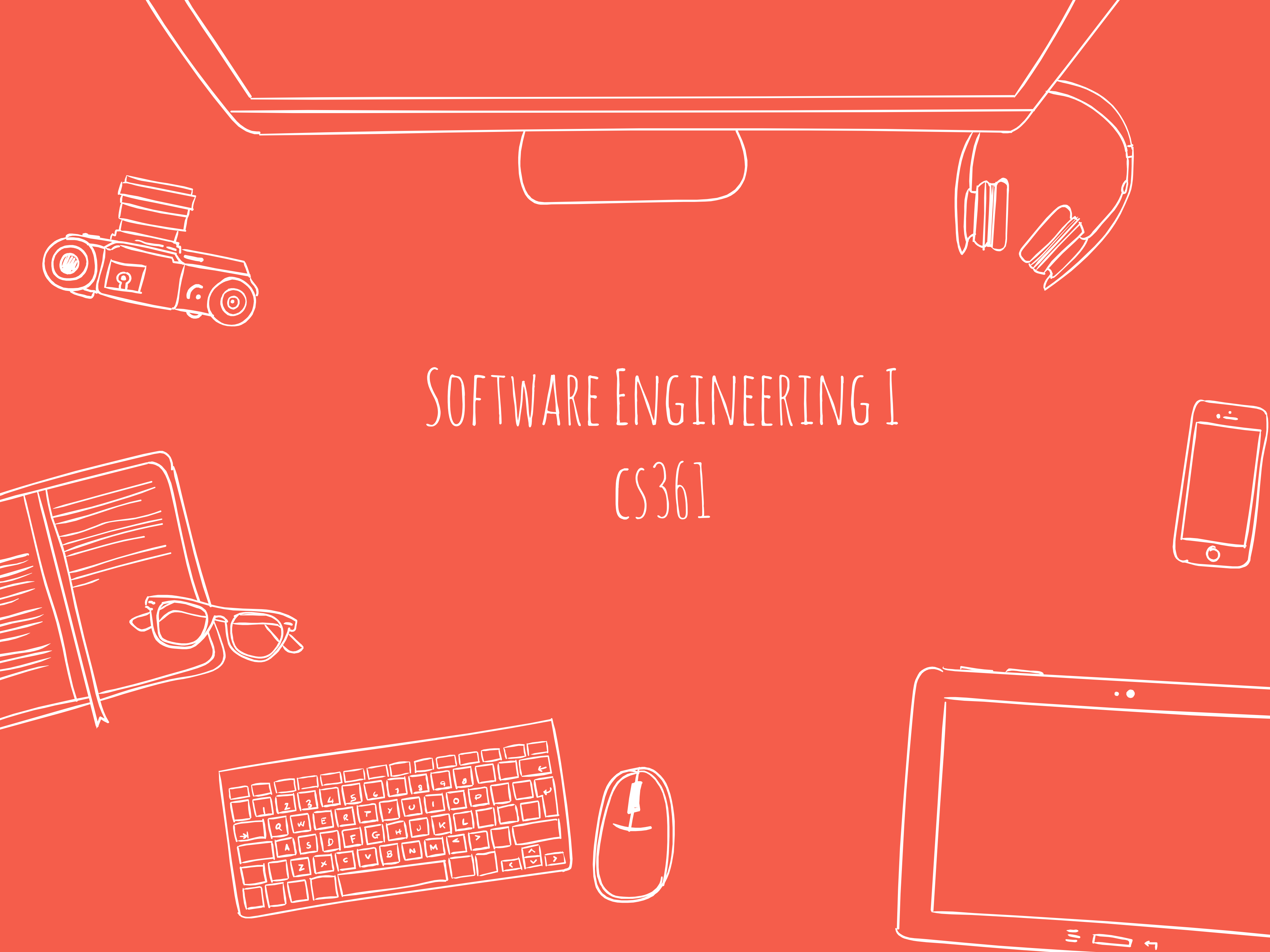


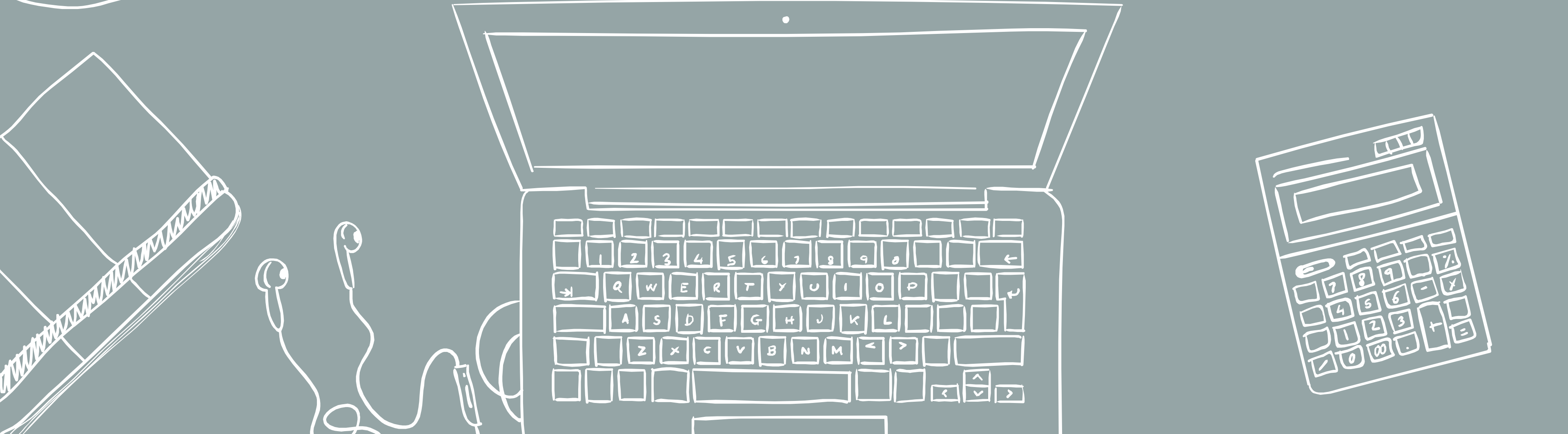
Software Engineering I

CS361

Announcements

- ✖ Writing Assignment 3
- ✖ http://web.engr.oregonstate.edu/~hiltonm/classes/cs361/WritingAssignments/WritingAssignment3.pdf

Class Diagrams



Attribution

Much of this material inspired by a great slides from Steve Easterbrook, available here: http://www.cs.toronto.edu/~sme/CSC340F/slides/11-objects.pdf

also some material taken from here: http://web.stanford.edu/class/cs193j/LectureThree.pdf

Object Terminology Review

Any entity which mirrors the existence of a real world entity is an Object.

Examples:

Object Terminology Review

Any entity which mirrors the existence of a real world entity is an Object.

Examples:

Person, Student, Car, Playing Card, etc.

Object Terminology Review

Objects Contain:

-attributes (variables)

-functionality (methods)

Objects can have properties or be acted upon

Object Terminology Review

A description of an Object is called a class

Examples:

Encapsulation

- Objects allow data and functionality to be bundled together.
- Additionally, access to the data may be restricted to some of the objects components

Inheritance

- Allows one Class to automatically “assume” the attributes of another class
- Defines an “is a” relationship for classes

Polymorphism

- ✖ The ability to send the same message (call a method) to an Object, without knowing how the receiver (Object) will implement the message.

Building an Object Oriented Model

Our model should:

represent people, things and concepts

show connections and interactions

show enough detail to evaluate designs

maintain value after design phase

Object Oriented Analysis

Background

–Model the requirements in terms of objects and services

Motivation

–OO is (claimed to be) more ‘natural’

–OO emphasizes importance of well-defined interfaces between objects.

Nearly Anything can be an object...

Nearly Anything can be an object...

- ✖**External Entities** e.g. people, devices, other systems.
- ✖**Things** e.g. reports, displays, signals, etc
- ✖**Occurrences or Events** e.g. transfer of resources, a control actions, etc
- ✖**Roles** people who interact with the system
- ✖**Organizational Unites** e.g. division, group, team, etc
- ✖**Places** e.g. manufacturing floor, loading dock, game board, etc
- ✖**Structures** e.g. sensors, computers, etc

THINGS THAT SHOULD NOT BE AN OBJECT

✖Procedures: e.g. print, draw, deal, etc

✖Attributes: e.g. blue, 50Mb, etc

Classes

A class describes a group of objects with:

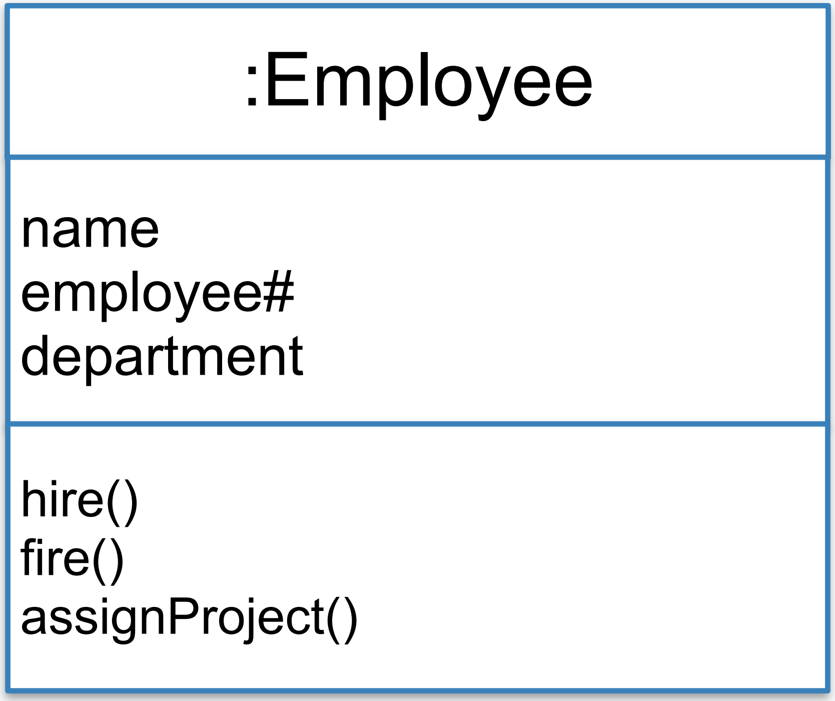
similar properties (attributes)
common behavior (operations)
common relationships
common meaning

Example Class:

employee:
has a name,
employee#,
department

an employee is
hired,
fired;

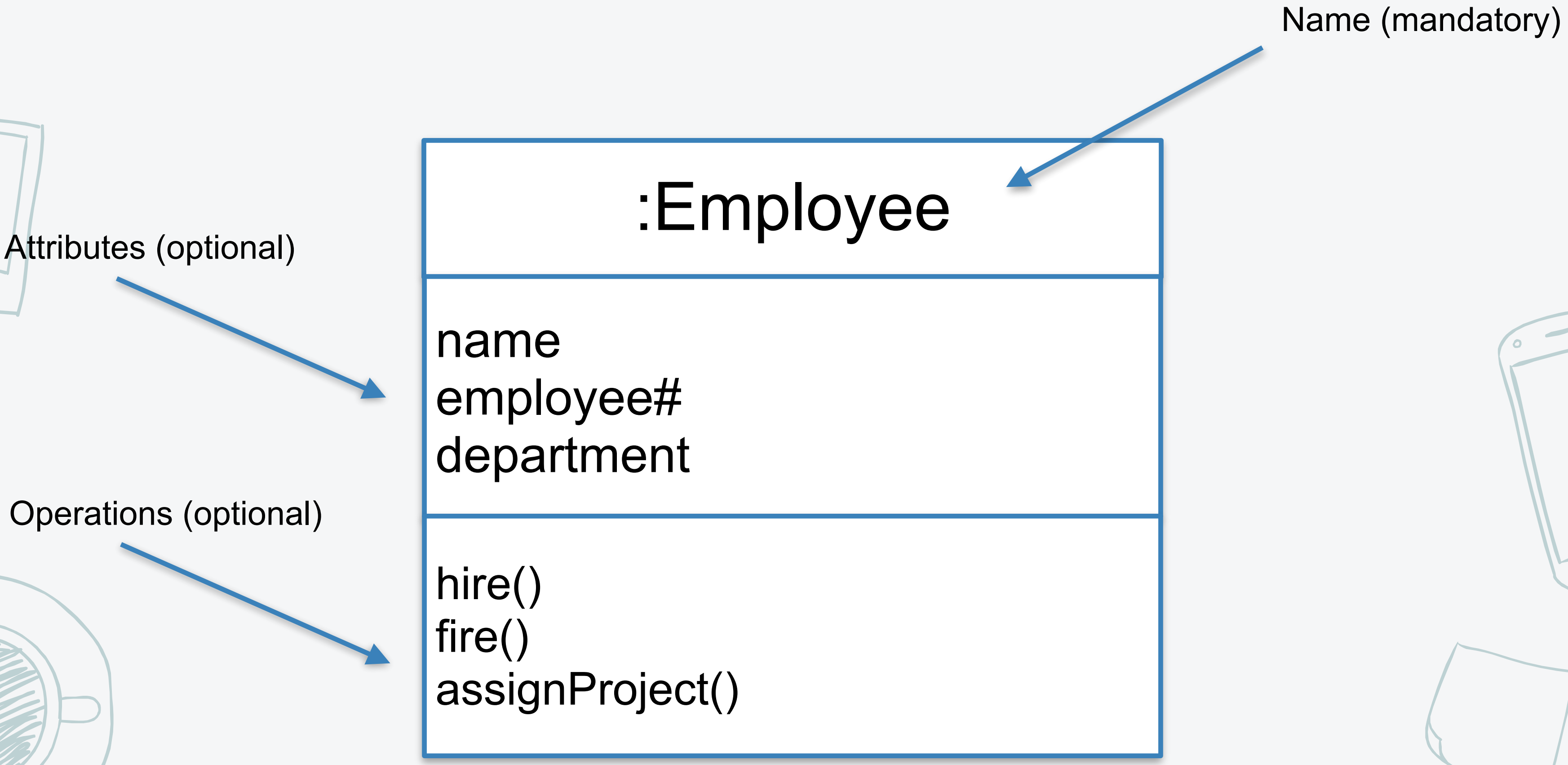
an employee works in one or more projects



:Employee

name
employee#
department

hire()
fire()
assignProject()



Name (mandatory)

:Employee

Attributes (optional)

name
employee#
department

Operations (optional)

hire()
fire()
assignProject()

How to find classes

- ✖ Look for nouns in user stories

- ✖ Review background information

- ✖ It’s better to start with too many and discard later

Selecting Classes

Discard classes for concepts which:

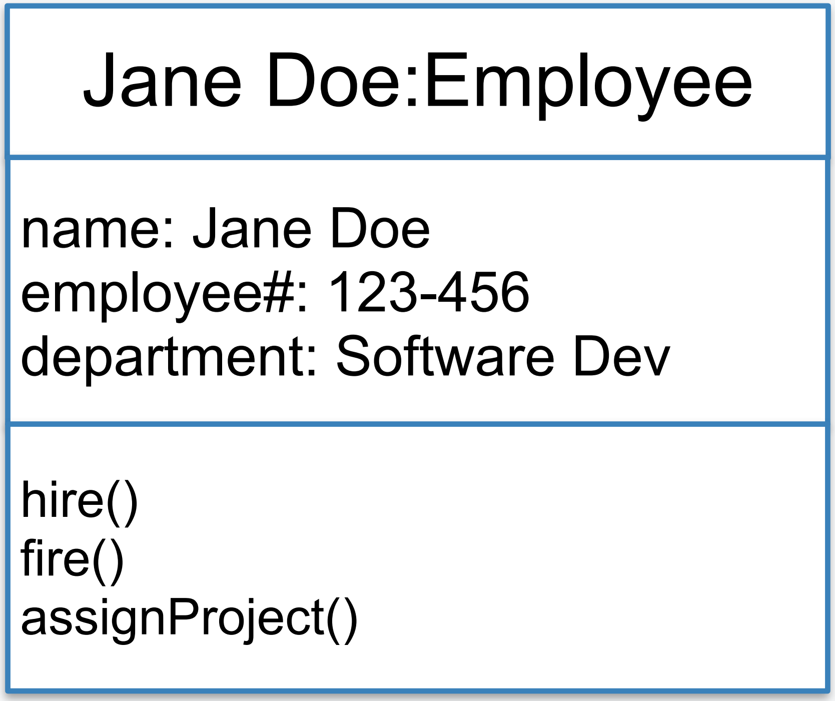
- Are beyond the scope of the analysis
- Refer to the system as a whole
- Duplicate other classes
- External entities should not be included as classes

Cold & Yourdon’s critera

- **Retained information**: will the system need to remember info about this class?
- **Needed Services:** Do these objects have identifiable operations that change values
- **Multiple Attributes:** A single attribute class may be an attribute
- **Common Attributes and Operations:** Does the class share attributes and operations will all of its objects

Objects vs Classes

The instances of a class are called objects.



Jane Doe:Employee

name: Jane Doe
employee#: 123-456
department: Software Dev

hire()
fire()
assignProject()

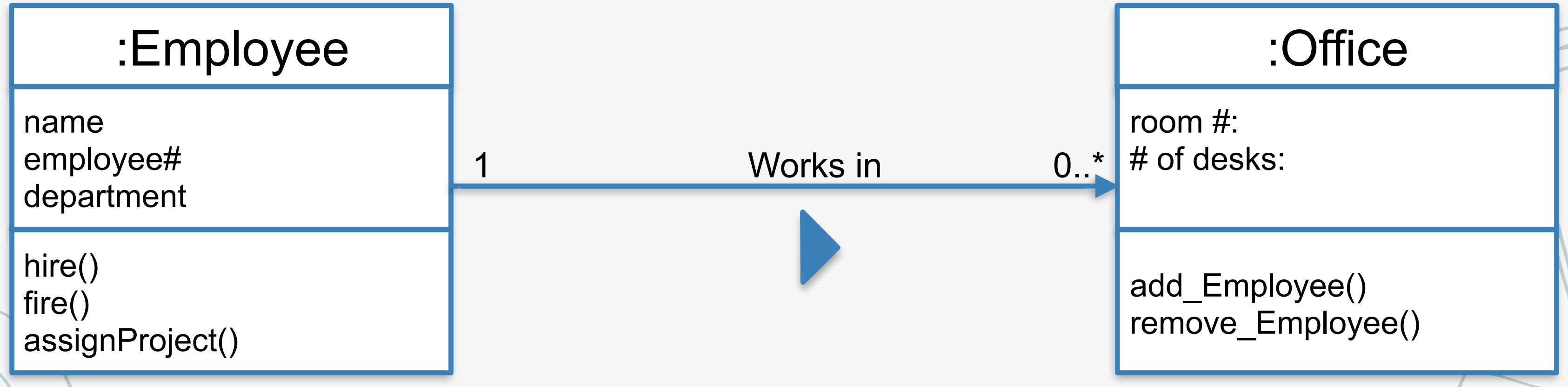
Associations

Objects do not exist in isolation

UML supports:

- Association
- Aggregation and Composition
- Generalization
- Dependency
- Realization

Class associations



:Employee

name
employee#
department

hire()
fire()
assignProject()

1

Works in

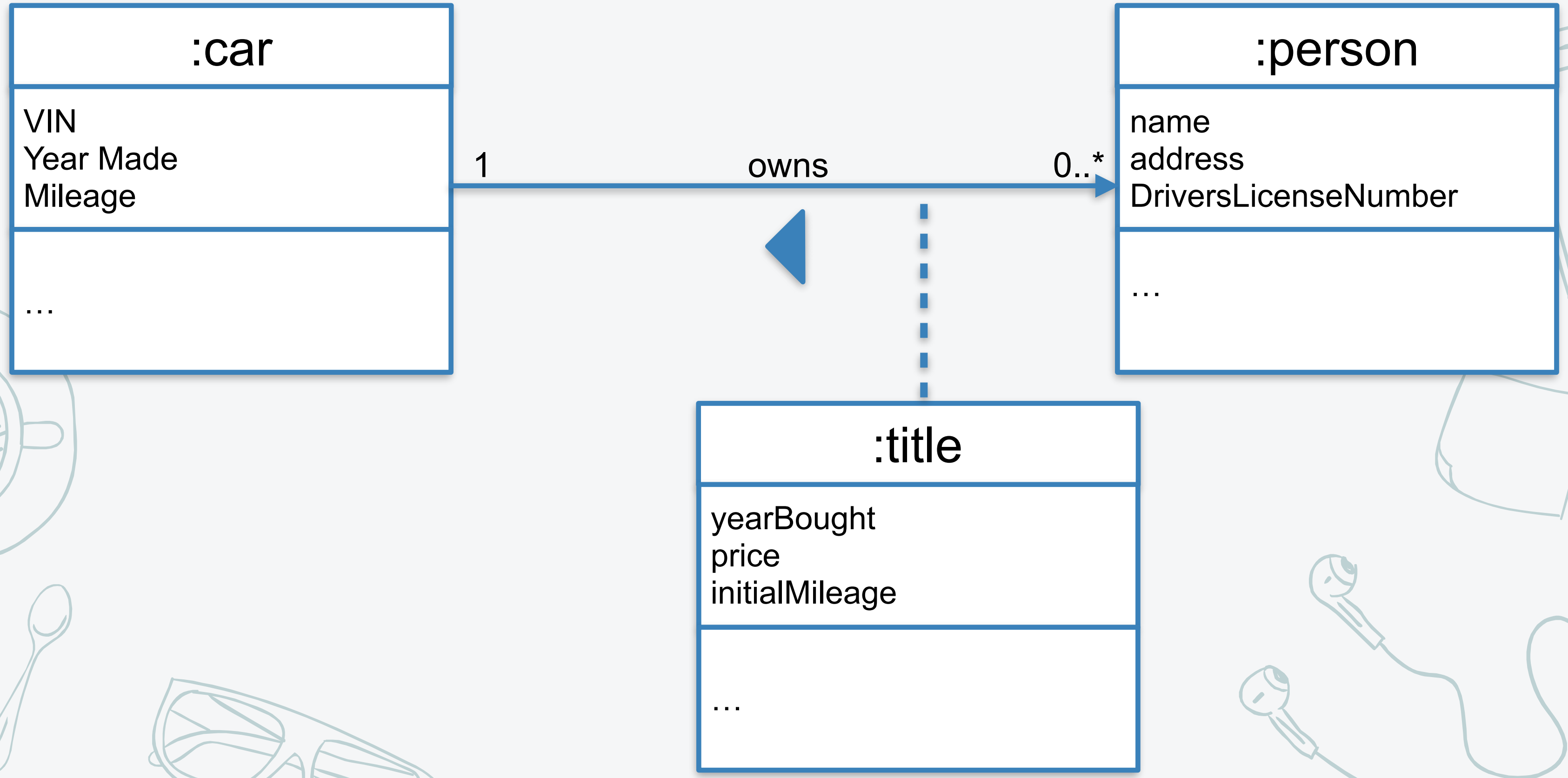
0..*

:Office

room #:
# of desks:

add_Employee()
remove_Employee()

Association Classes



:car

VIN
Year Made
Mileage

…

1

owns

0..*

:person

name
address
DriversLicenseNumber

…

:title

yearBought
price
initialMileage

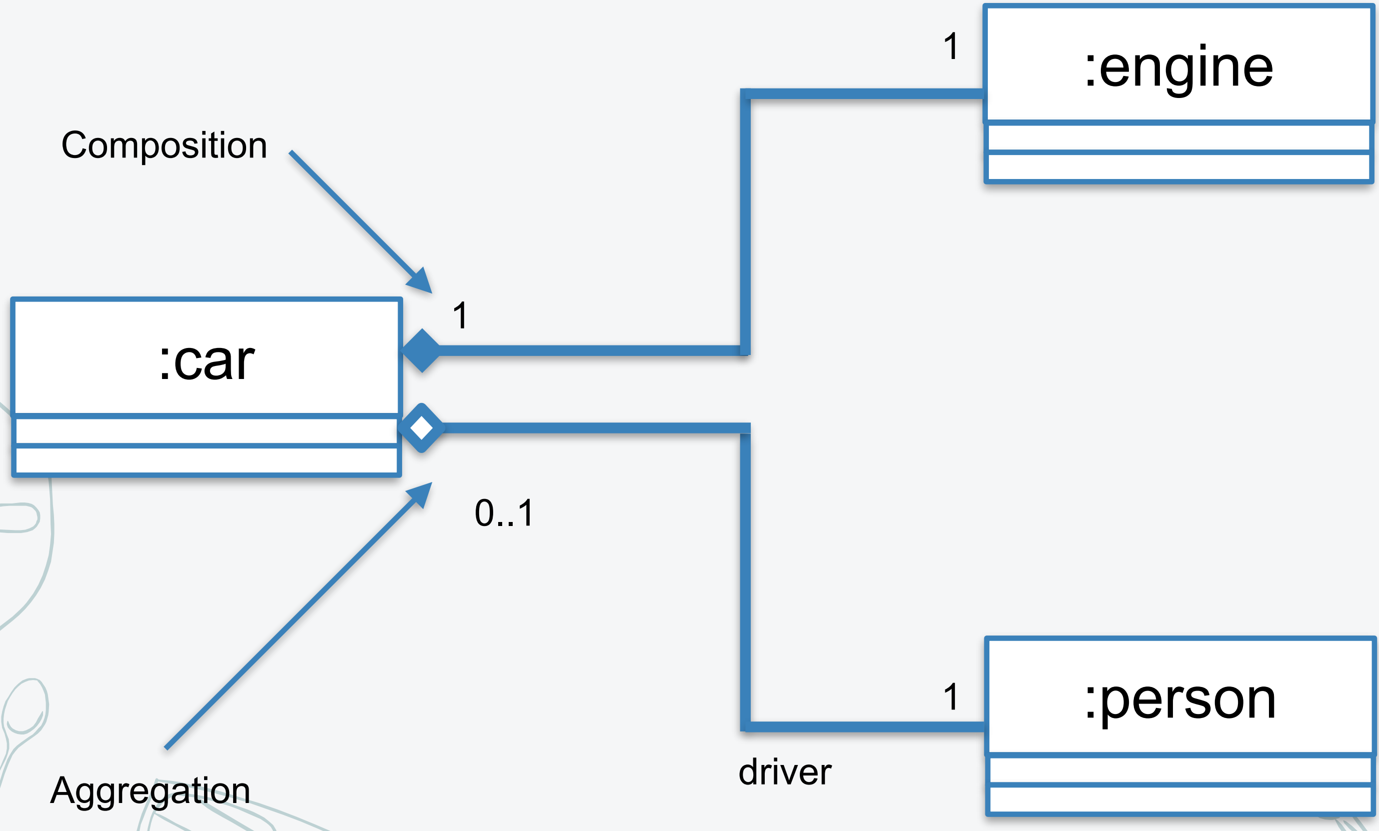
…

Aggregation and Composition

- ✖Aggregation:
  This is the “Has-a” or “Whole/part” relationship

- ✖Composition
  implies ownership:

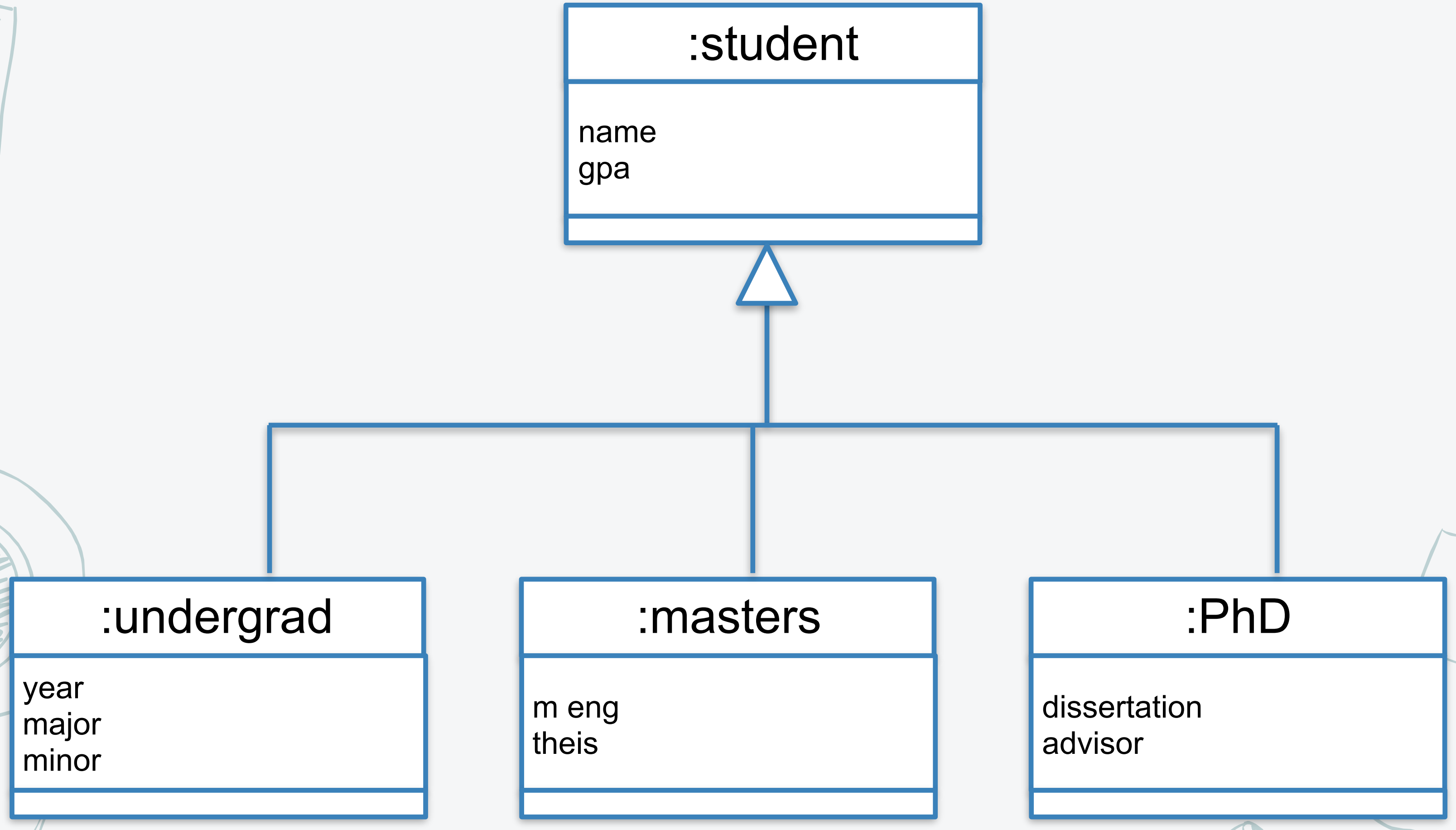
Aggregation and Composition Example



Generalization

- Subclasses are more specific versions of superclasses
- Subclasses inherit attributes, associations, & operations from the superclass
- Subclasses can override an inherited aspect
- Superclass are abstract if they have no instances

Generalization



:student

name
gpa

:undergrad

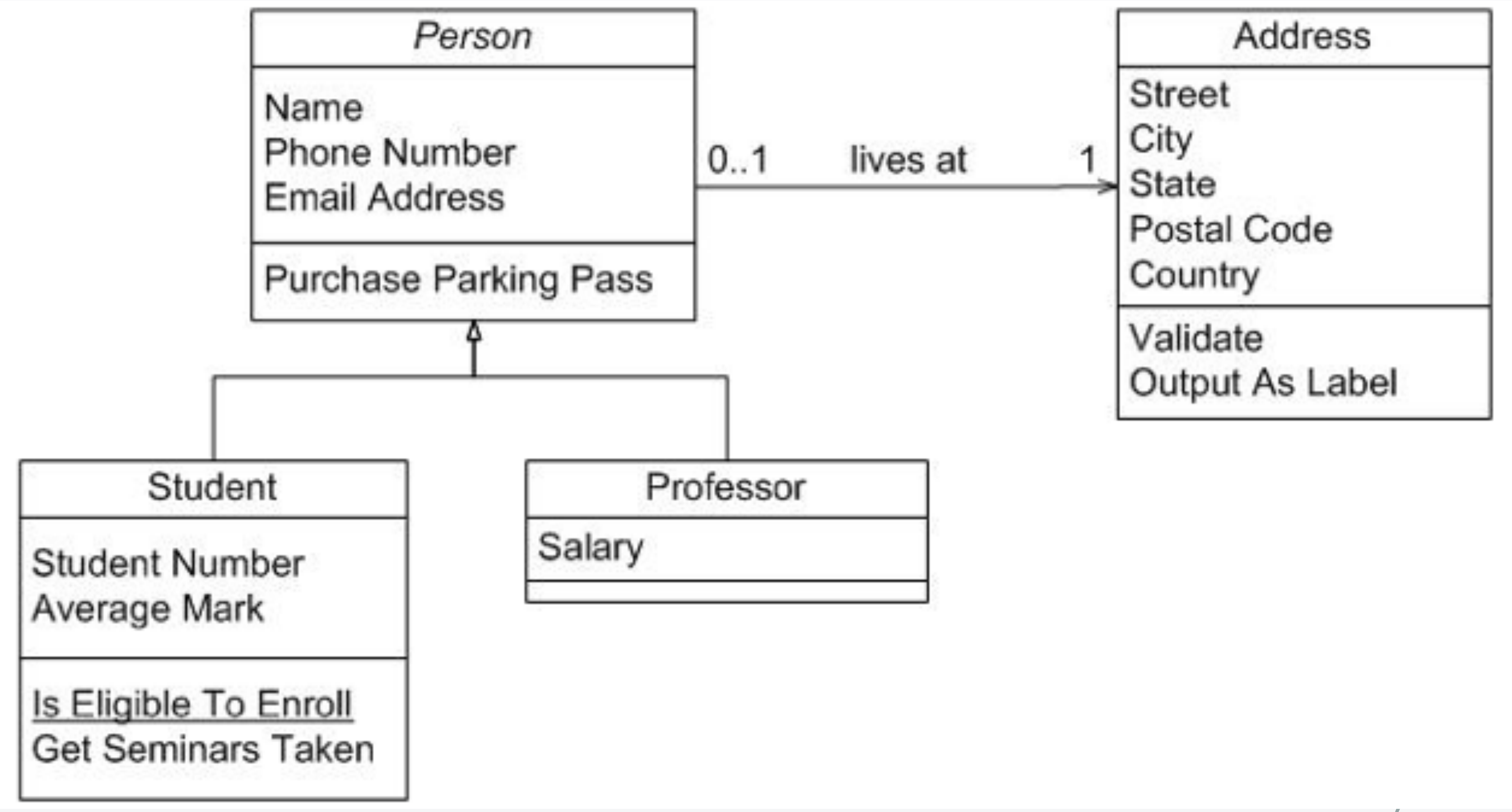
year
major
minor

:masters

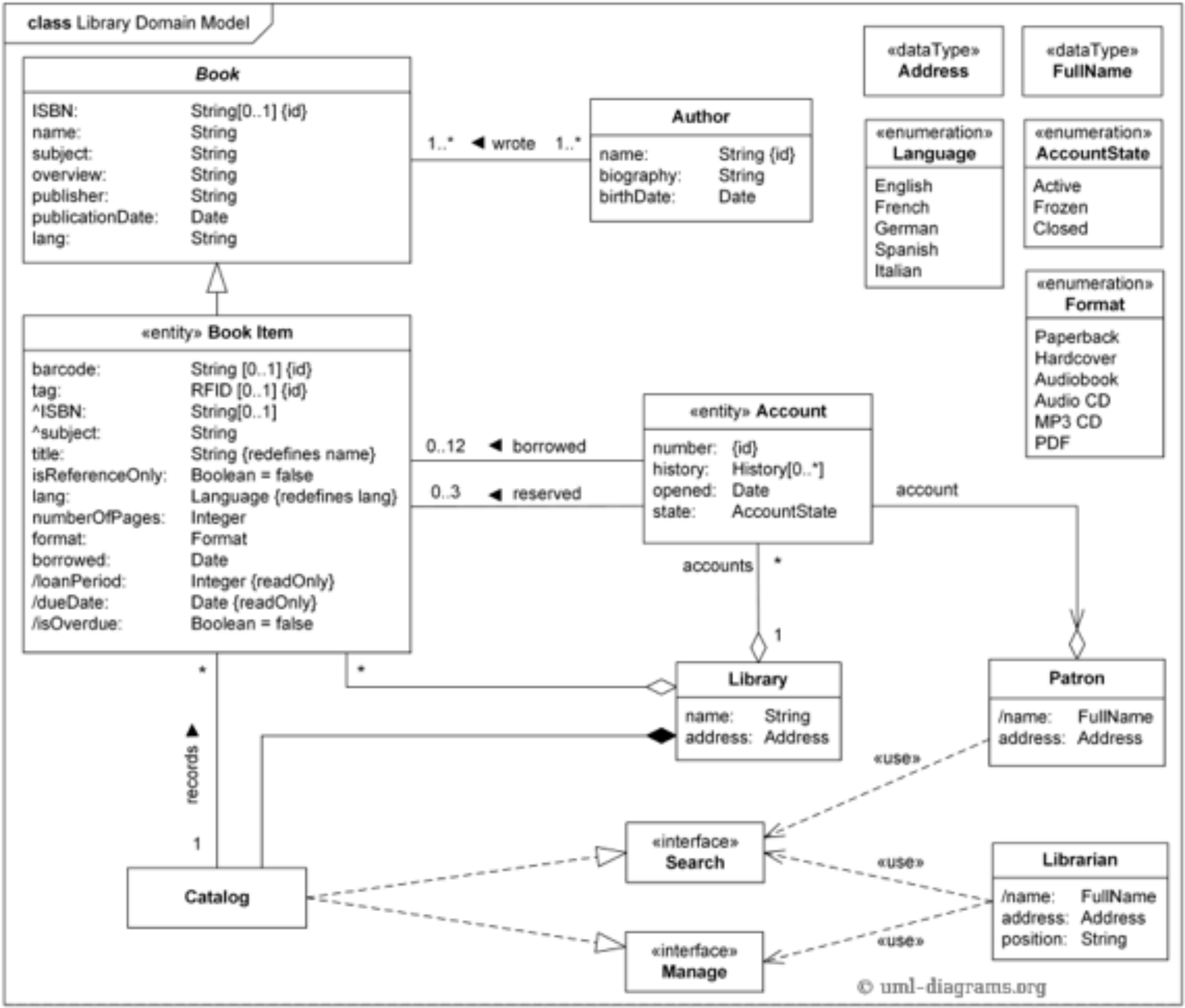
m eng
theis

:PhD

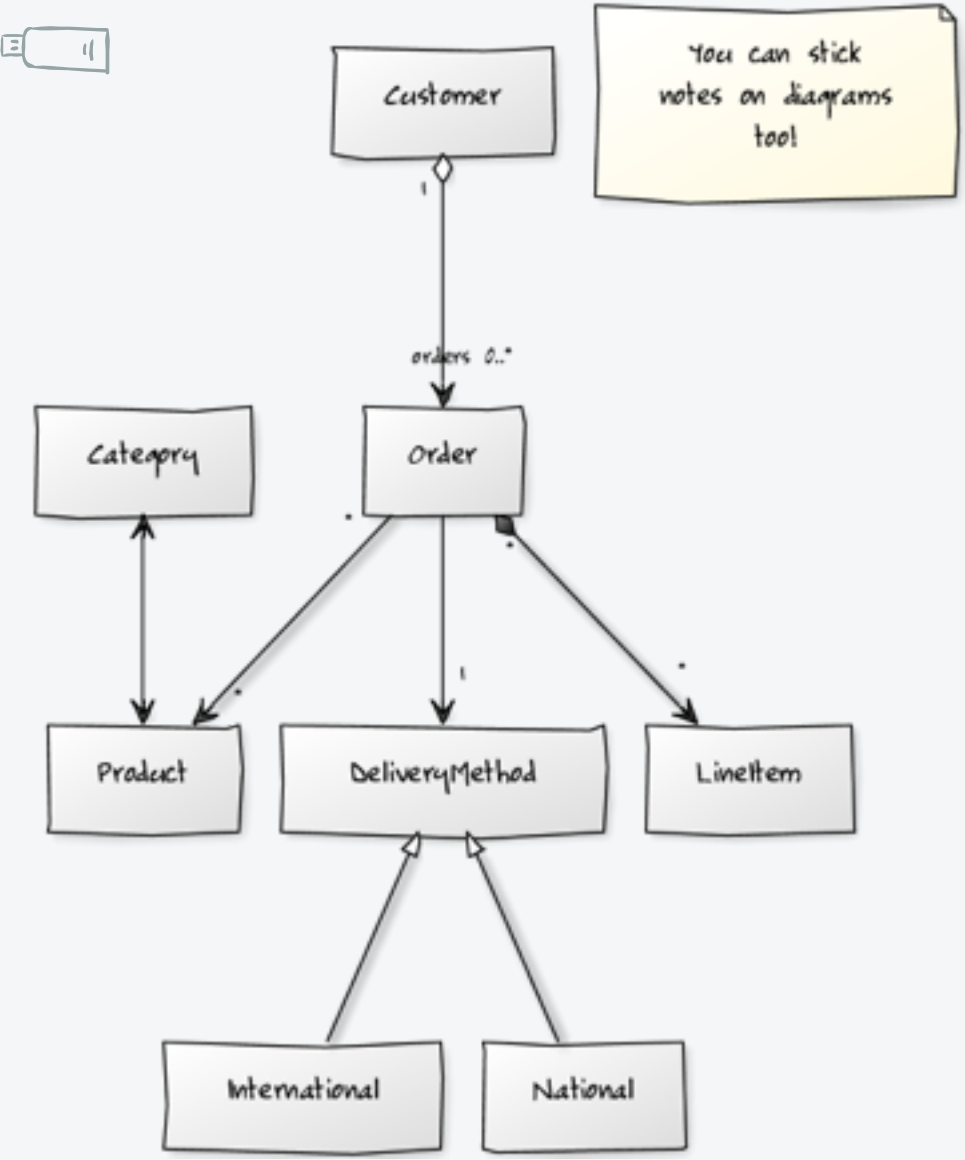
dissertation
advisor



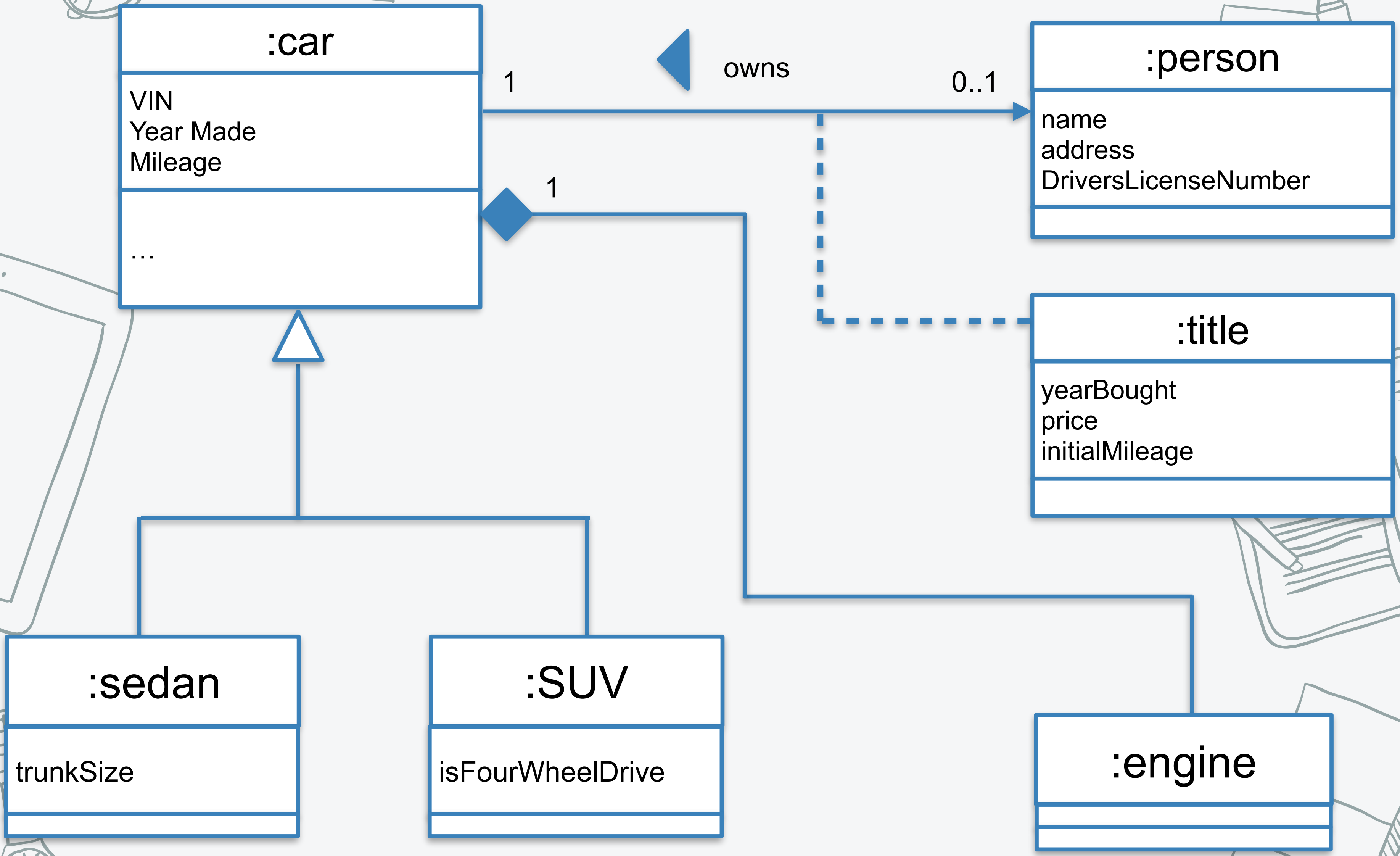
http://agilemodeling.com/artifacts/classDiagram.htm



http://www.uml-diagrams.org/class-diagrams-overview.html



http://yuml.me/diagram/scruffy/class/samples



:car

VIN
Year Made
Mileage

…

owns

1

0..1

:person

name
address
DriversLicenseNumber

1

:title

yearBought
price
initialMileage

:sedan

trunkSize

:SUV

isFourWheelDrive

:engine

Feedback

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash