
Introduction to Object-Oriented Modelling

Philipp Reinecke
reinecke@cardiff.ac.uk

In this lecture:

- Object Oriented (OO) models and modelling
- Objects and Classes

What is
an
object?

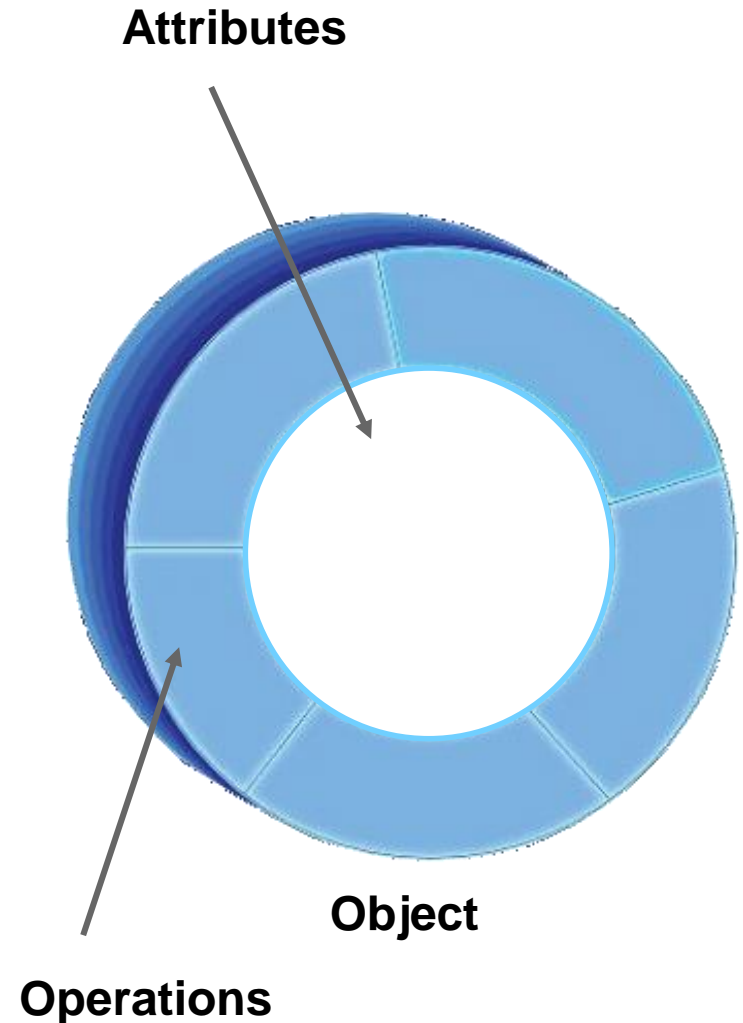
What is a
class?

How do
we model
with
classes?

What is
an
Object?

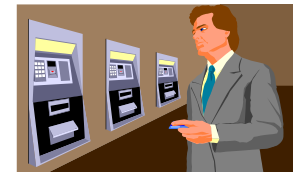
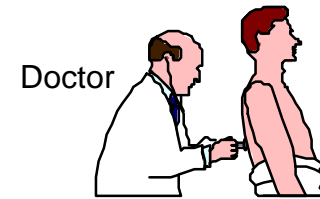
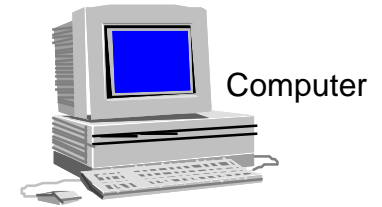
Definition: Object

- An object is an entity with a well-defined boundary and identity that encapsulates state and behaviour.
 - ▶ State is represented by attributes and relationships.
 - ▶ Behaviour is represented by operations.
- Access to attributes is through operations.



An Object may represent anything in the real world

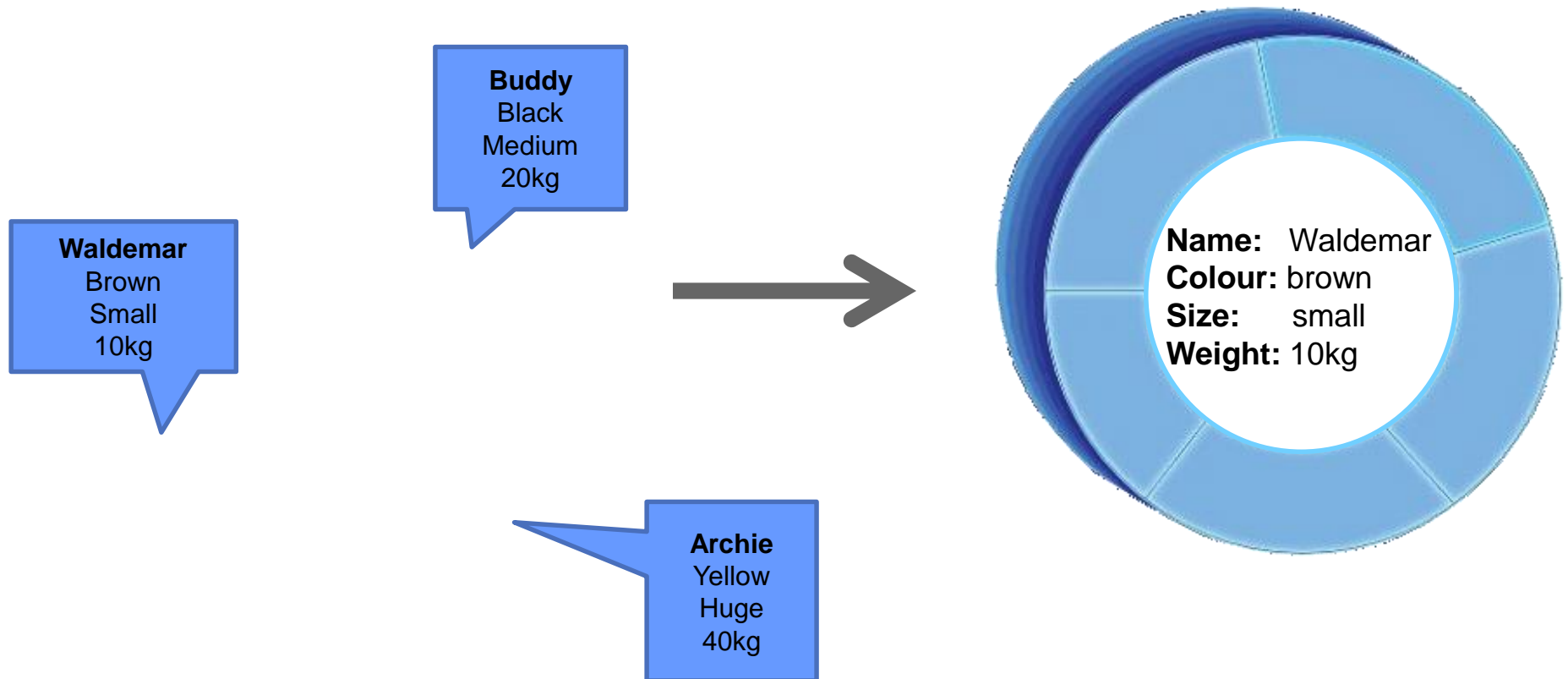
- It can be tangible, like a computer
- It can be a process, like an engine test
- It can be a relationship, like a contract
- It can be a role, like a doctor
- It can be an incident, like a withdrawal
- It can be theoretical, like a matrix
- It can be an interaction like an agreement



Example

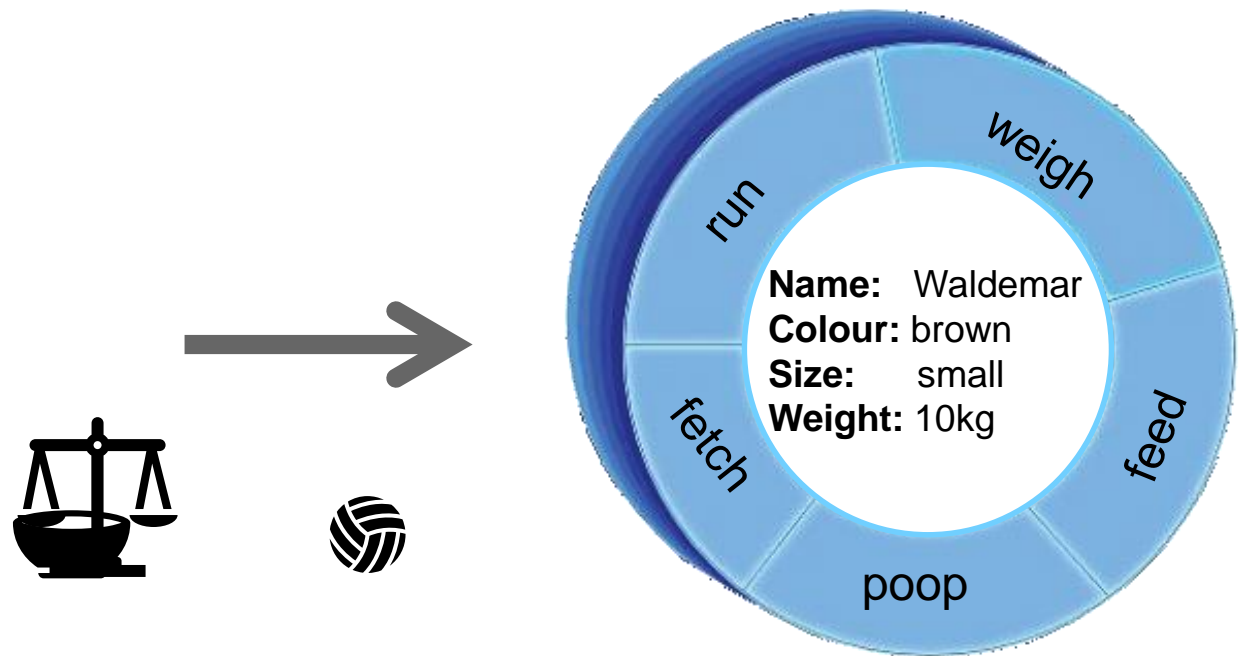
An Object Has State

- State: One of the possible conditions in which the object may exist
- State can change over time



An Object Has Behaviour

- Behaviour determines how an object acts and reacts.
- The visible behaviour of an object is modelled by the set of operations the object can perform.
- Behaviour can change the object state.



An Object Has Identity

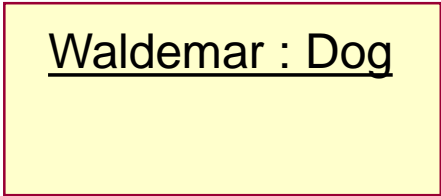
- Each object has a unique identity, even if the state is identical to that of another object.

Waldemar
Brown
Small
10kg

Waldemar
Brown
Small
10kg

Representing Objects in the UML

- An object is represented as a rectangle with an underlined name.



Waldemar : Dog

A yellow rectangular box with a red border. Inside the box, the text "Waldemar : Dog" is underlined.

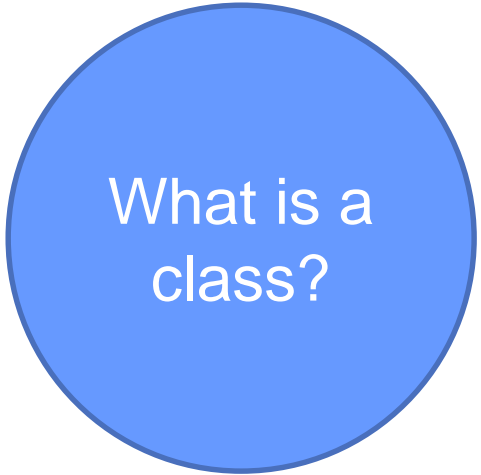
Named Object



_____ : Dog

A yellow rectangular box with a red border. Inside the box, a horizontal line is followed by the text ": Dog".

Unnamed Object



What is a
class?

Definition: Class

- A class is a description of a set of objects that share the same attributes, operations and relationships.
 - ▶ An object is an instance of a class.
- A class is an abstraction in that it
 - ▶ Emphasizes relevant characteristics.
 - ▶ Suppresses other characteristics.
- A class encapsulates
 - ▶ State
 - ▶ Behaviour

Classes and Objects provide Abstraction

- **Abstraction:**

Model most important, essential or distinguishing aspects of something while suppressing or ignoring less important, immaterial or diversionary details.

(Dictionary of Object Technology, Firesmith, Eykholt).

- **Abstraction removes irrelevant detail**

- ▶ Important to manage complexity.
- ▶ Defines a boundary relative to the perspective of the viewer.

Class Design provides Encapsulation

- Encapsulation:

The physical localization of properties and behaviors into a single blackbox abstraction that hides their implementation behind a public interface.

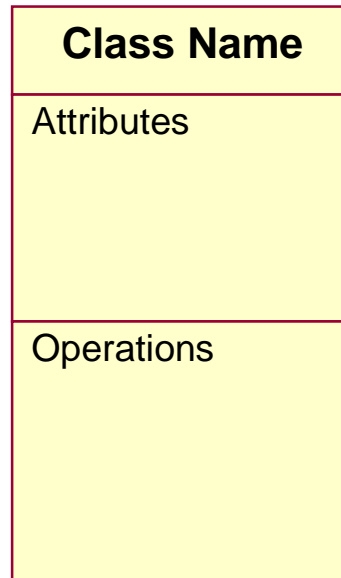
(Dictionary of Object Technology, Firesmith, Eykholt, 1995)

- Encapsulation (information hiding) hides implementation from clients.

- Clients depend on interface
- Clients are independent of implementation
- Class can control access

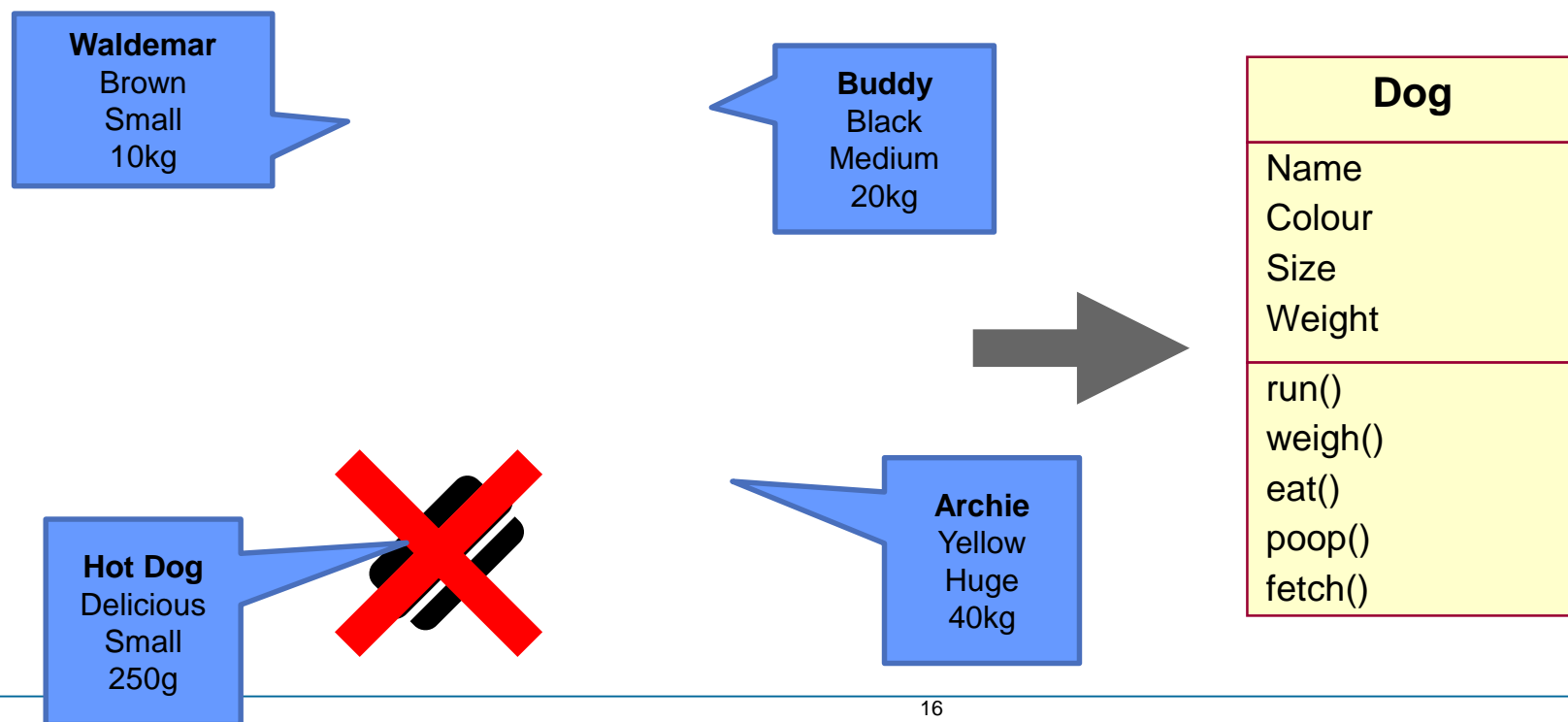
Representing classes in the UML

- A class is represented using a rectangle with compartments.



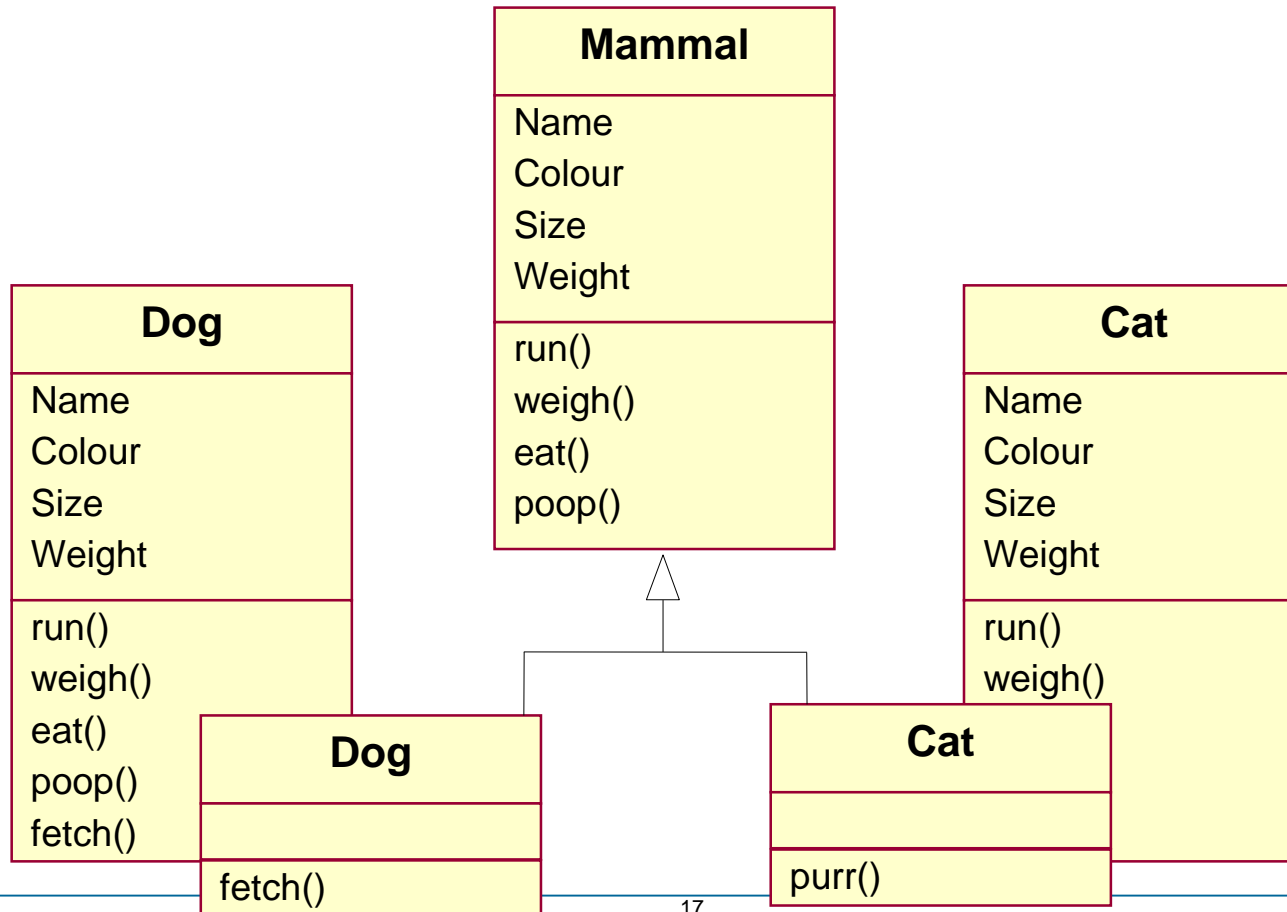
The Relationship Between Classes and Objects

- A class is an abstract definition of an object.
 - ▶ Defines the structure and behavior of objects in the class.
 - ▶ Serves as a template for creating objects.
- Classes group objects with same attributes and behaviour
- Classes are not collections of objects



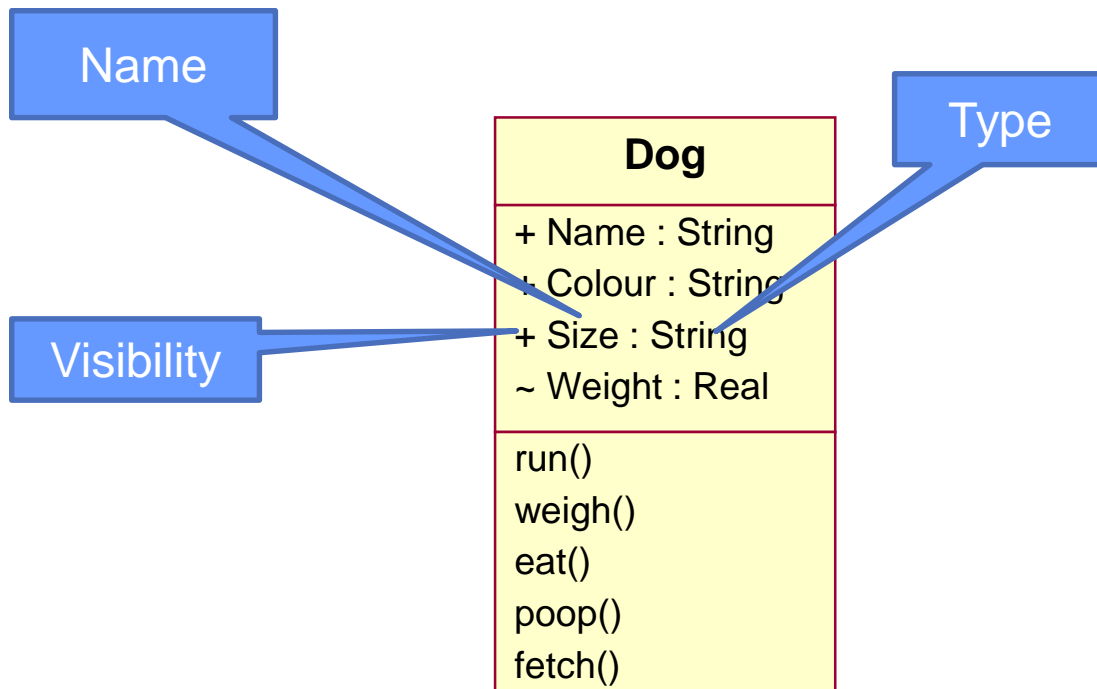
Classes: Inheritance

- Subclasses can inherit attributes and operations from parent class
- Parent class generalises
- Subclasses can specialise



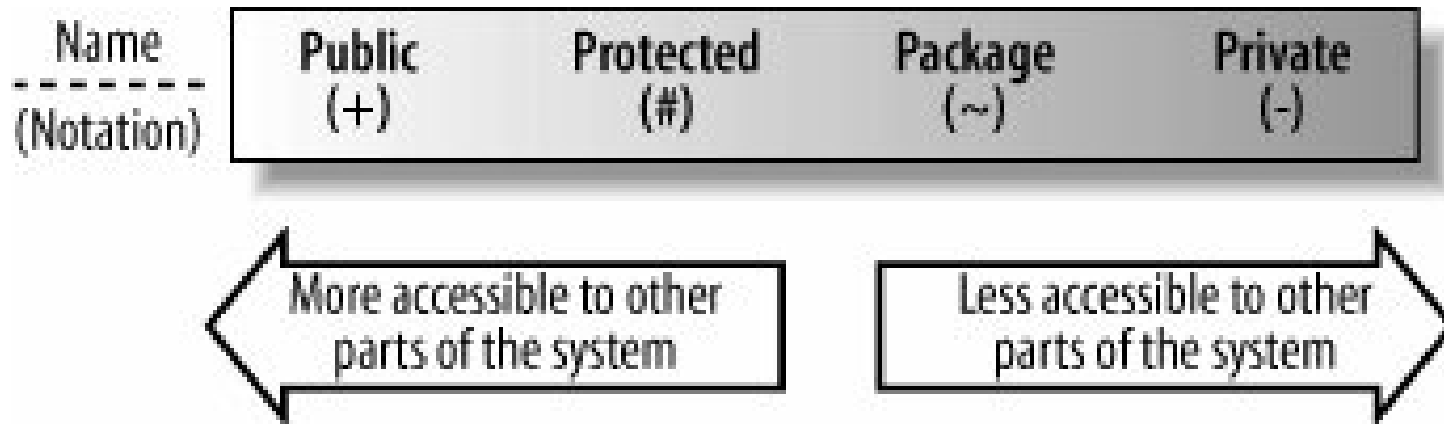
What Is an Attribute?

- An attribute is a named property of a class that describes a range of values that instances of the property may hold.
 - ▶ A class may have any number of attributes
 - ▶ Attribute signature defines the attributes



Attribute: Visibility

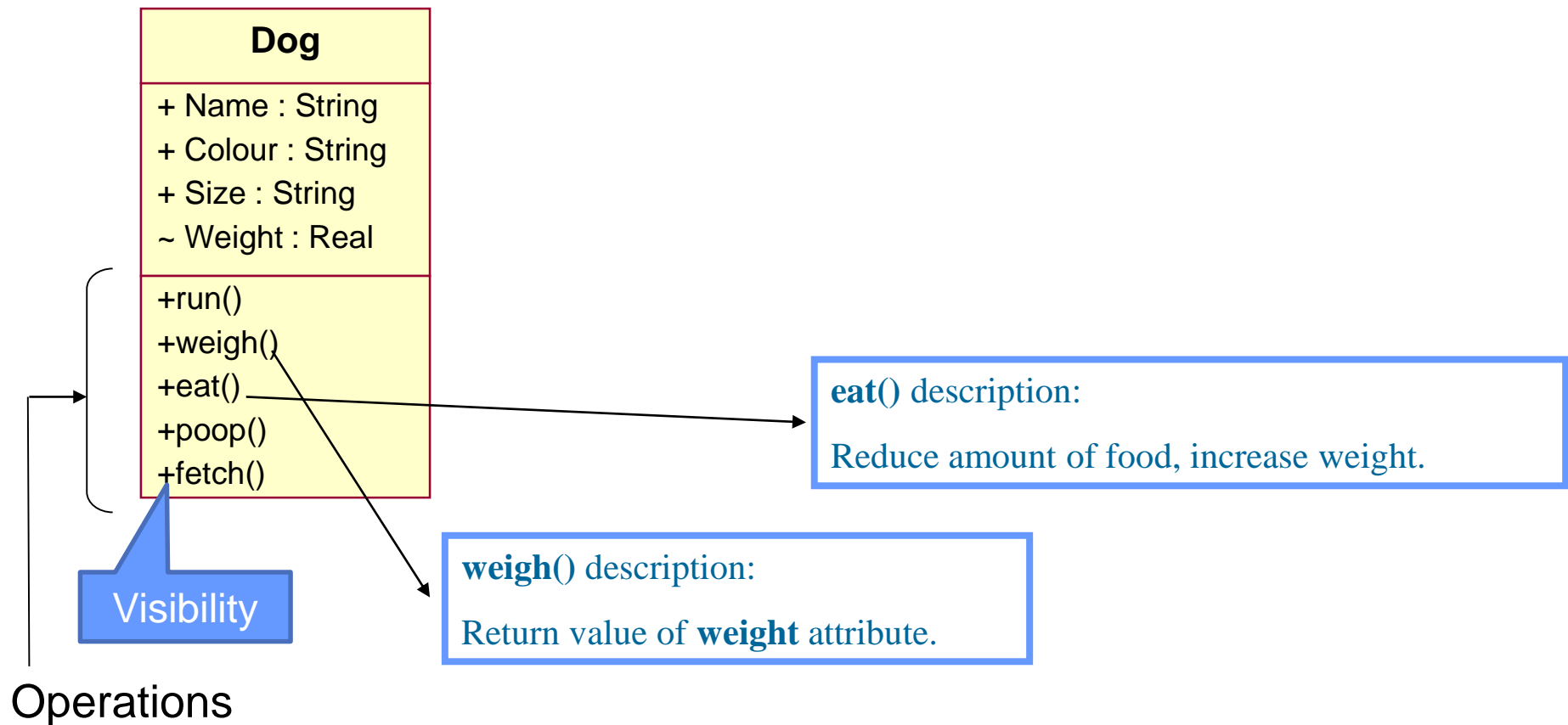
- Visibility indicates how the class reveals its data to other classes.

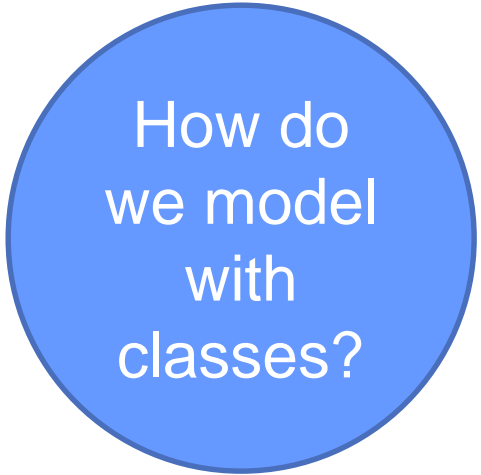


- +Public: accessed from any other class
- #Protected: accessed from subclasses only
- ~Package: accessed from within the same package
- -Private: accessed only within this class

Class Responsibilities and Operations

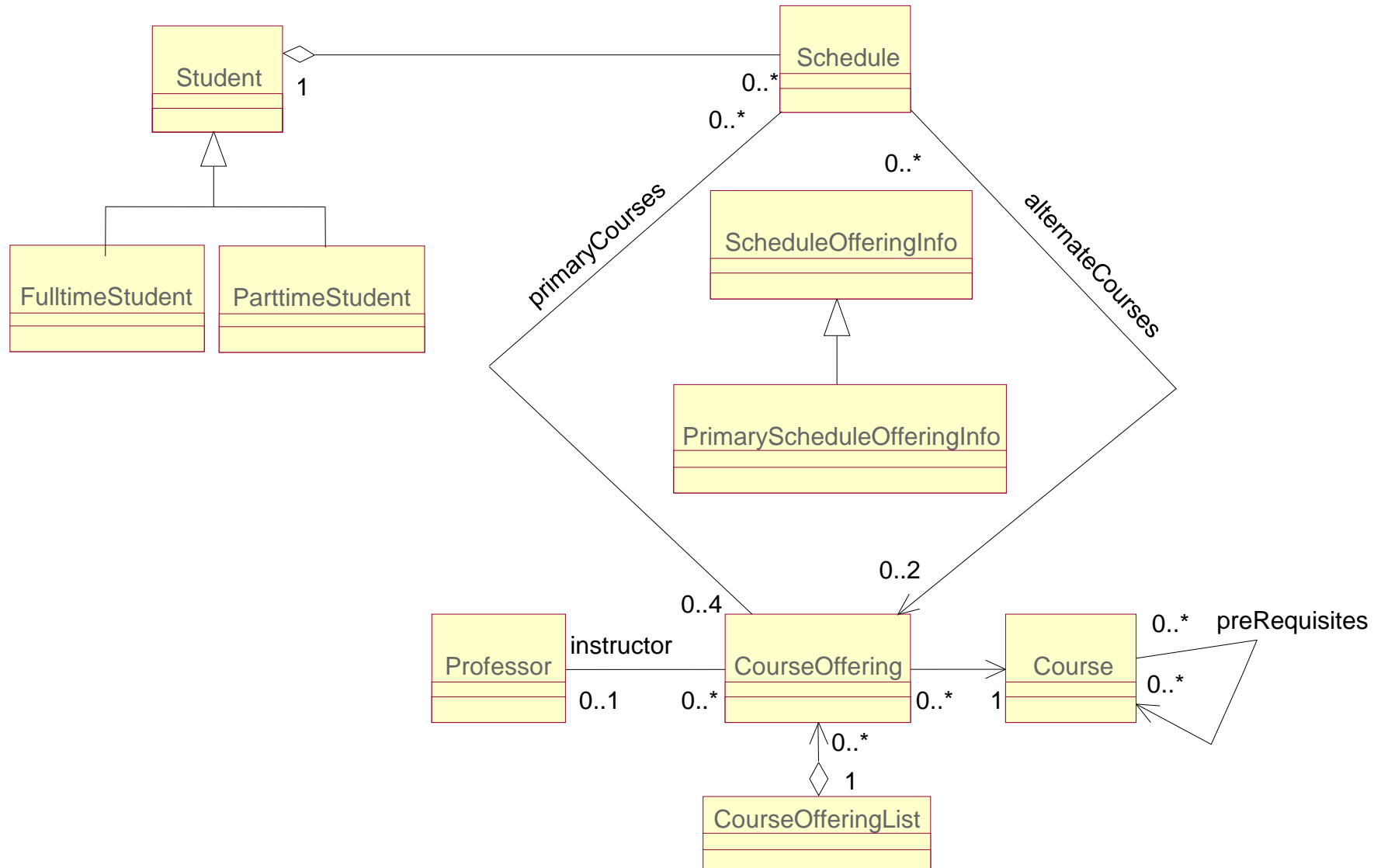
- A **responsibility** is a contract or obligation of a class. It is a statement of something the class can be expected to provide.
- An **operation** is the implementation of a service that can be requested from any object of the class to affect behavior.





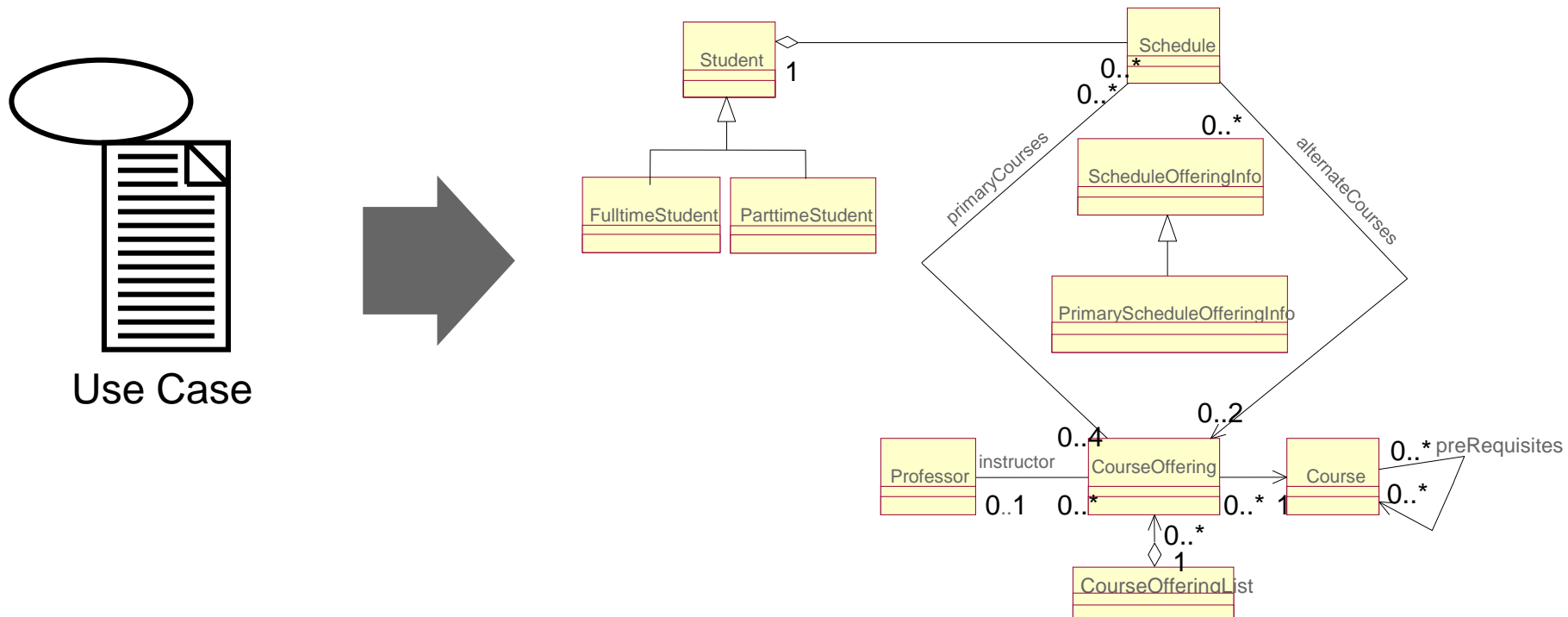
How do
we model
with
classes?

What we want: Class Diagram



Finding Classes

- Key abstractions of the system showing its logical structure.
- Model information to be stored and manipulated to satisfy the use cases.



Example: Finding Classes

- Use **use-case** flow of events as input
- Find key abstractions of the use case
- Traditional, filtering nouns approach
 - ▶ Underline noun clauses in the use-case flow of events
 - ▶ Remove redundant candidates
 - ▶ Remove vague candidates
 - ▶ Remove implementation constructs
 - ▶ Remove attributes (save for later)
 - ▶ Remove operations

Example: Identifying Classes

Register for Courses

1.1 Basic Flow

1. **Registration open.**

This use case starts when someone chooses to register for courses. The system checks that registration is possible at this time.

2. **Select “Create a Schedule”.**

The system displays the functions available to the student. The student selects “Create a Schedule”.

3. **Obtain Course Information.**

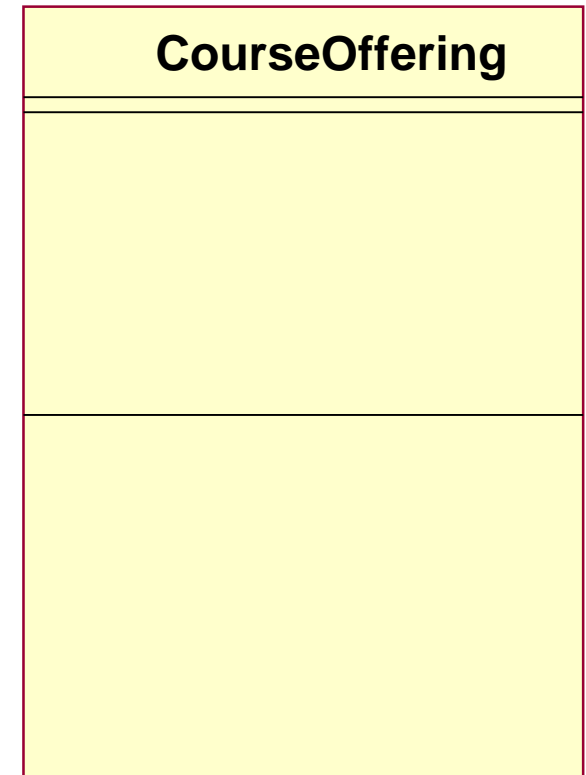
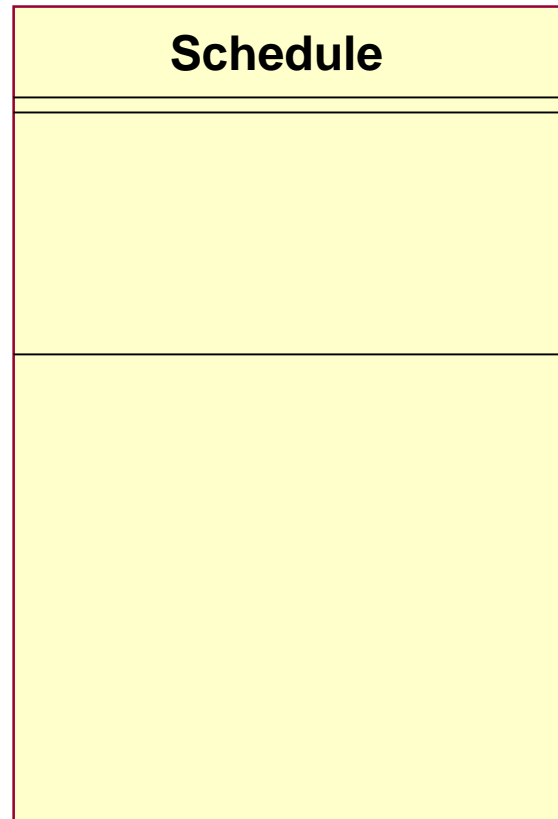
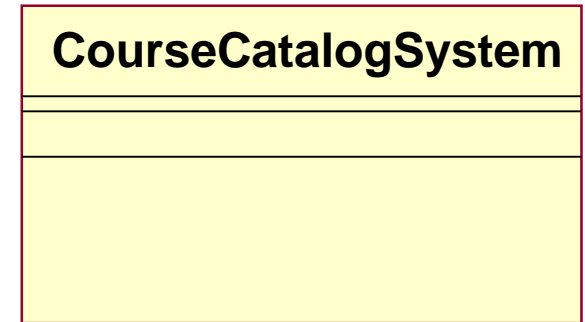
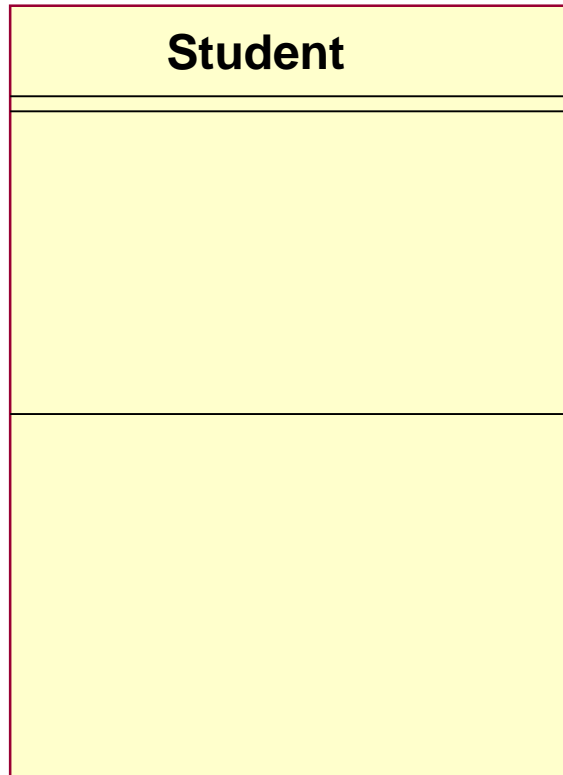
The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the student. The student can search the list by department, professor, or topic to obtain the desired course information.

4. **Select Courses.**

The student selects four primary course offerings and two alternate course offerings from the list of available offerings course offerings.

...

Candidate Classes for Register for Course Use Case



Finding Attributes

- Properties/characteristics of identified classes
- Information retained by identified classes
- “Nouns” that did not become classes
 - ▶ Information whose value is the important thing
 - ▶ Information that is uniquely “owned” by an object
 - ▶ Information that has no behaviour

Candidate Classes for Register for Course Use Case

Student
-name -studentID -address -dateofBirth

CourseCatalogSystem

Schedule
-studentID -primaryOfferings -altOfferings

CourseOffering
-number -startTime -endTime -numStudents

Finding Operations

- Operations describe what the class can do.
- An operation can be either a command or a question. Only a command can change the state of the object; a question should never change it.
- Use the Class-Responsibility-Card (CRC) to find operations (to be discussed later)

Candidate Classes for Register for Course Use Case

Student

-name
-studentID
-address
-dateofBirth

+ get tuition()
+ add schedule()
+ get schedule()
+ delete schedule()

CourseCatalogSystem

+get courseOffering()

Schedule

-studentID
-primaryOfferings
-altOfferings

+ commit()
+ select alternate()
+ remove offering()
+ delete()
+ check conflicts()
+ create()

CourseOffering

-number
-startTime
-endTime
-numStudents

+ setstudentnumbers()
+ getstarttime()
+ getendtime()
+ closeOffering()

Summary

- Objects and Classes
 - Abstraction and Encapsulation
 - State and Behaviour
 - Finding Classes and identifying Attributes
-
- Next: Relationships and Responsibilities