What Is Object-Orientation?

Based on Chapter 4 of Bennett, McRobb and Farmer:

Object Oriented Systems Analysis and Design Using UML, (4th Edition), McGraw Hill, 2010.

In This Lecture You Will Learn:

- The fundamental concepts of object-orientation, including:
  - Objects and classes
  - Generalization, specialization and inheritance
  - Information hiding and message passing
- The justifications for an object-oriented approach

Objects

An object is:

"an abstraction of something in a problem domain, reflecting the capabilities of the system to

- keep information about it,
- interact with it,
- or both."

Coad and Yourdon (1990)

Objects

"Objects have state, behaviour and identity."

Booch (1994)

- *State*: the condition of an object at any moment, affecting how it can behave
- *Behaviour*: what an object can do, how it can respond to events and stimuli
- *Identity*: each object is unique

Examples of Objects

Object	Identity	Behaviour	State
A person.	'Hussain Pervez.'	Speak, walk, read.	Studying, resting, qualified.
A shirt.	My favourite button white denim shirt.	Shrink, stain, rip.	Pressed, dirty, worn.
A sale.	Sale no #0015, 18/05/05.	Earn loyalty points.	Invoiced, cancelled.
A bottle of ketchup.	*This* bottle of ketchup.	Spill in transit.	Unsold, opened, empty.

Class and Instance

- All objects are *instances* of some *class*
- A Class is a description of a set of objects with similar:
  - features (attributes, operations, links);
  - semantics;
  - constraints (e.g. when and whether an object can be instantiated).

OMG (2009)

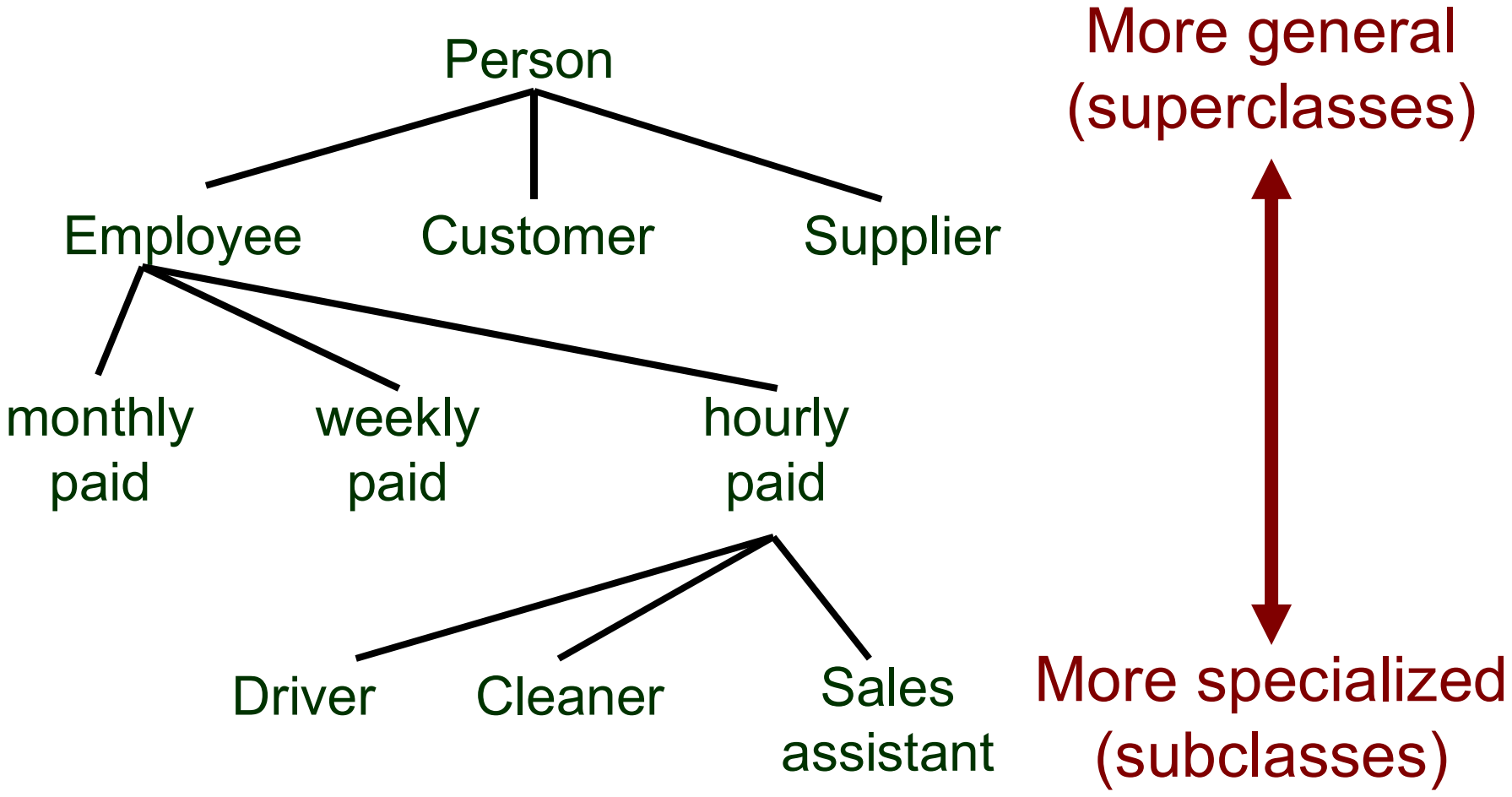
Class and Instance

- An object is an instance of some class
- So, instance = object
  - but also carries connotations of the class to which the object belongs
- Instances of a class are similar in their:
  - *Structure*: what they *know*, what information they hold, what links they have to other objects
  - *Behaviour*: what they *can do*

Generalization and Specialization

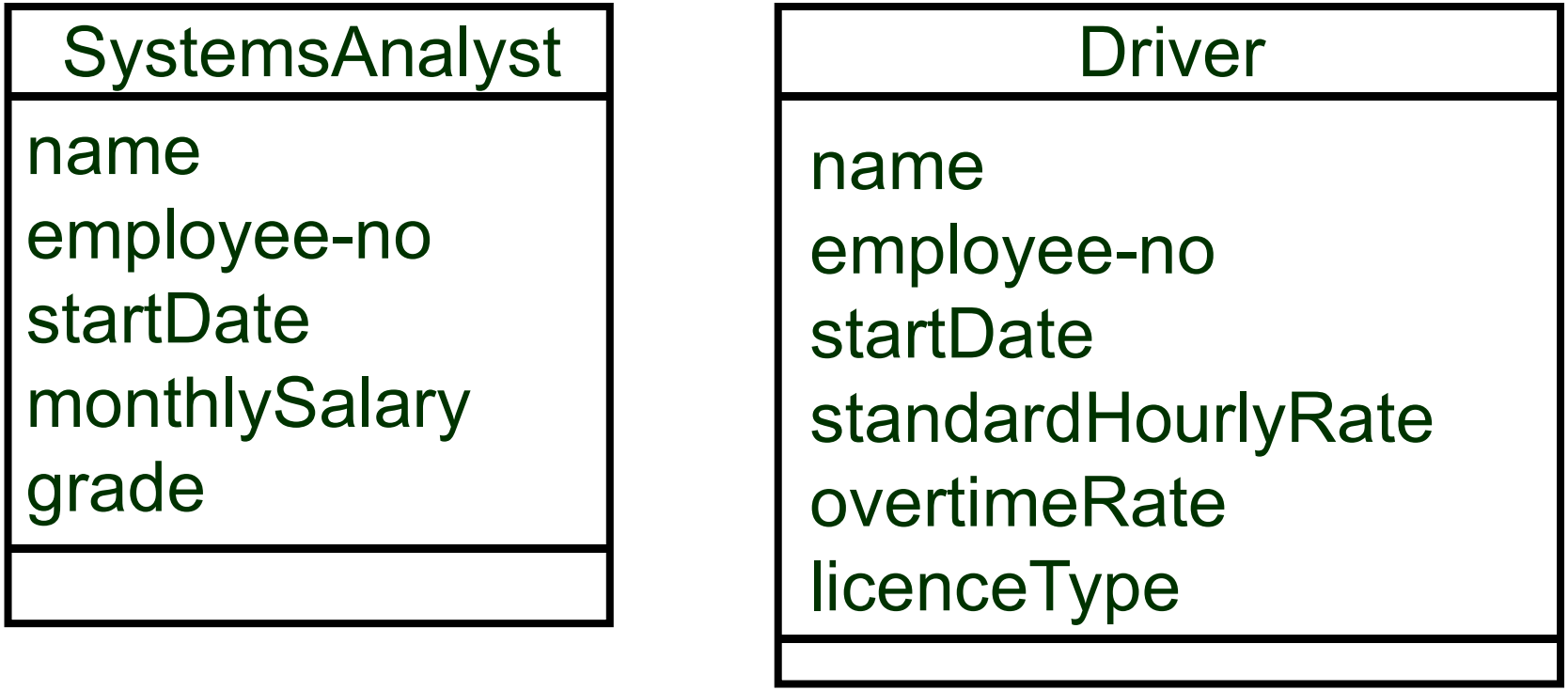
- Classification is hierarchic in nature
- For example, a person may be an employee, a customer, a supplier of a service
- An employee may be paid monthly, weekly or hourly
- An hourly paid employee may be a driver, a cleaner, a sales assistant

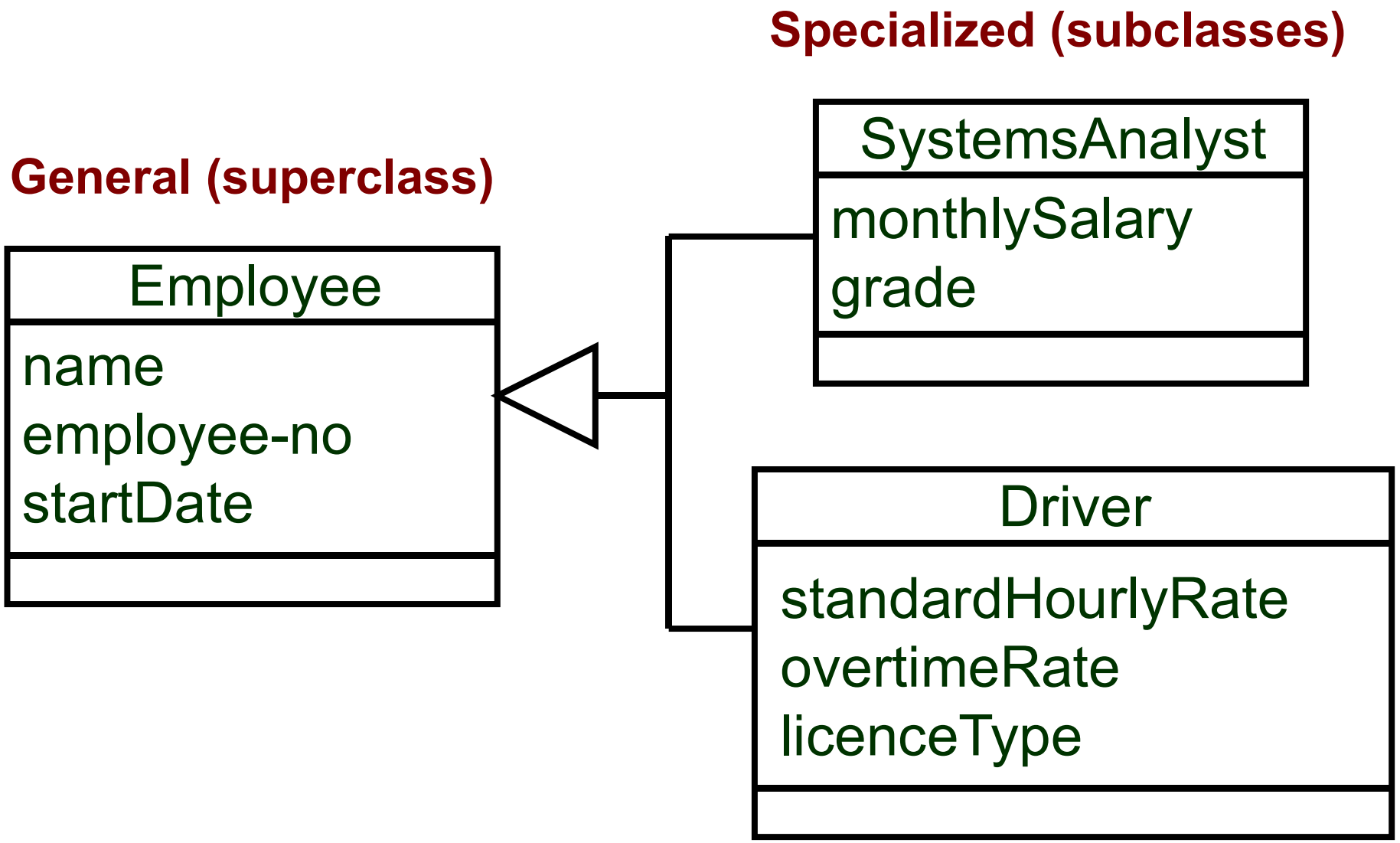
Specialization Hierarchy



Generalization and Specialization

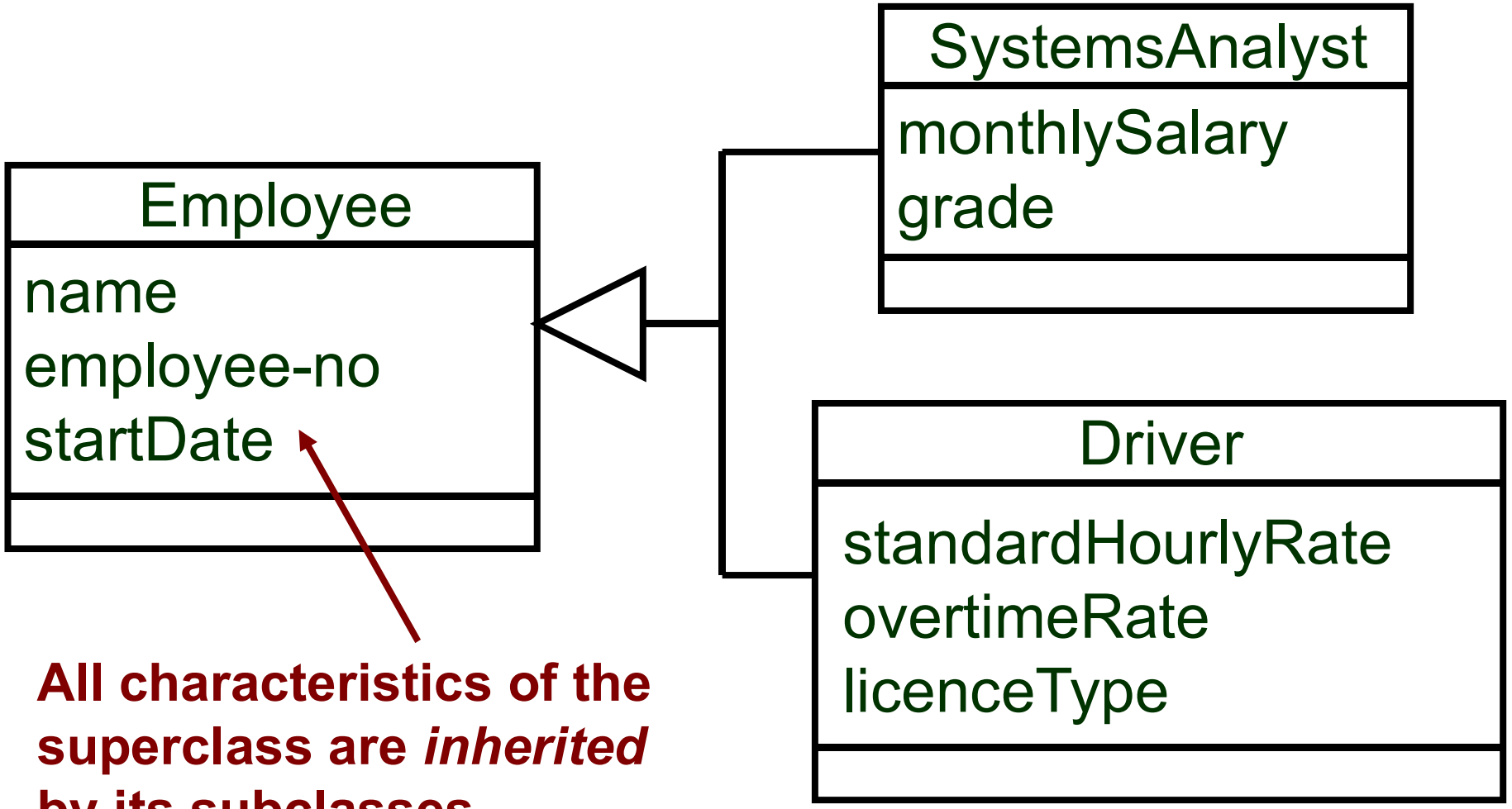
- More general bits of description are *abstracted out* from specialized classes:



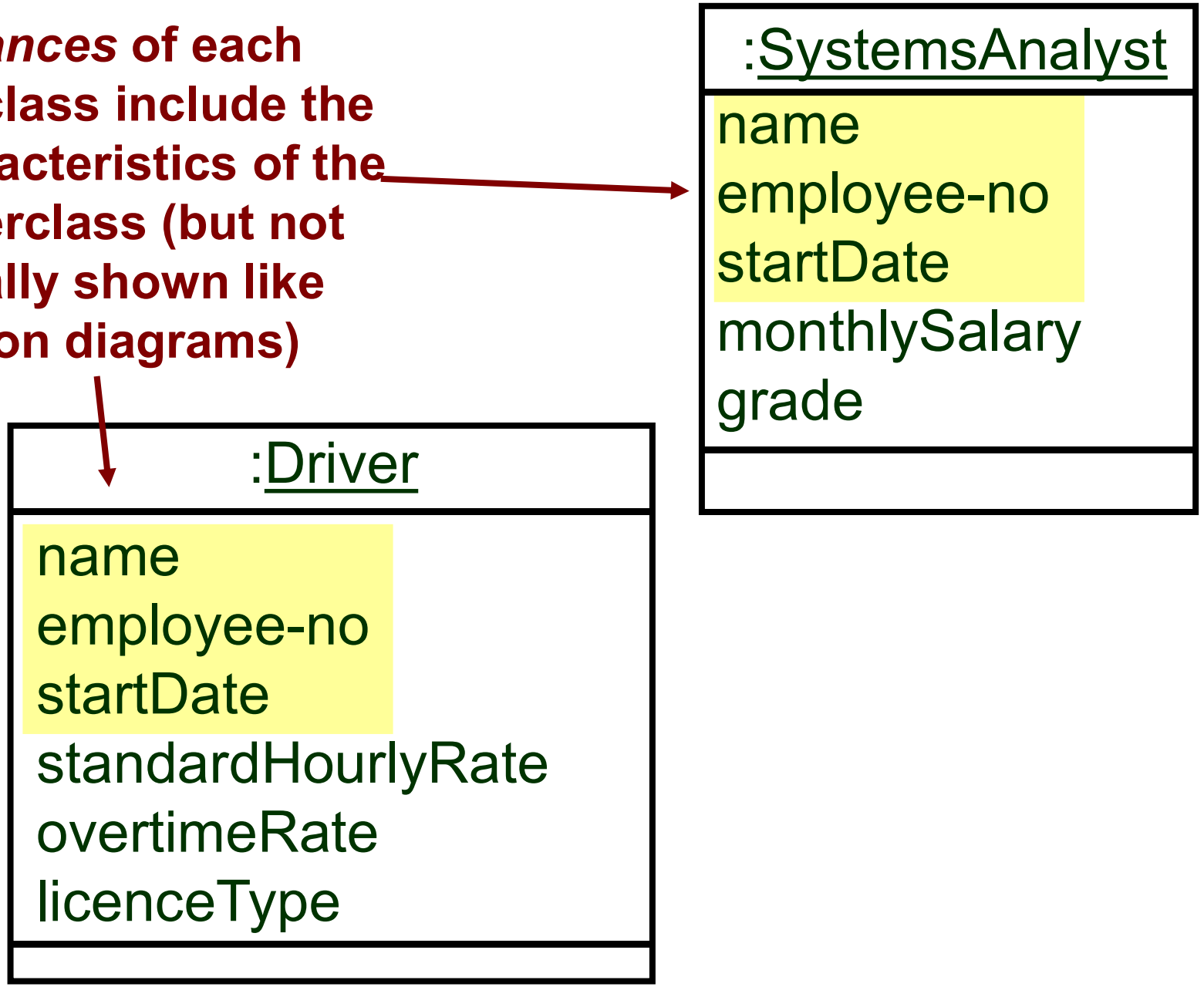


Inheritance

- The *whole* description of a superclass applies to *all* its subclasses, including:
  - Information structure (including associations)
  - Behaviour
- Often known loosely as *inheritance*
- (But actually inheritance is how an O-O programming language *implements* generalization / specialization)



***Instances* of each subclass include the characteristics of the superclass (but not usually shown like this on diagrams)**



Message-passing

- Several objects may collaborate to fulfil each system action
- “Record CD sale” could involve:
  - A CD stock item object
  - A sales transaction object
  - A sales assistant object
- These objects communicate by sending each other messages

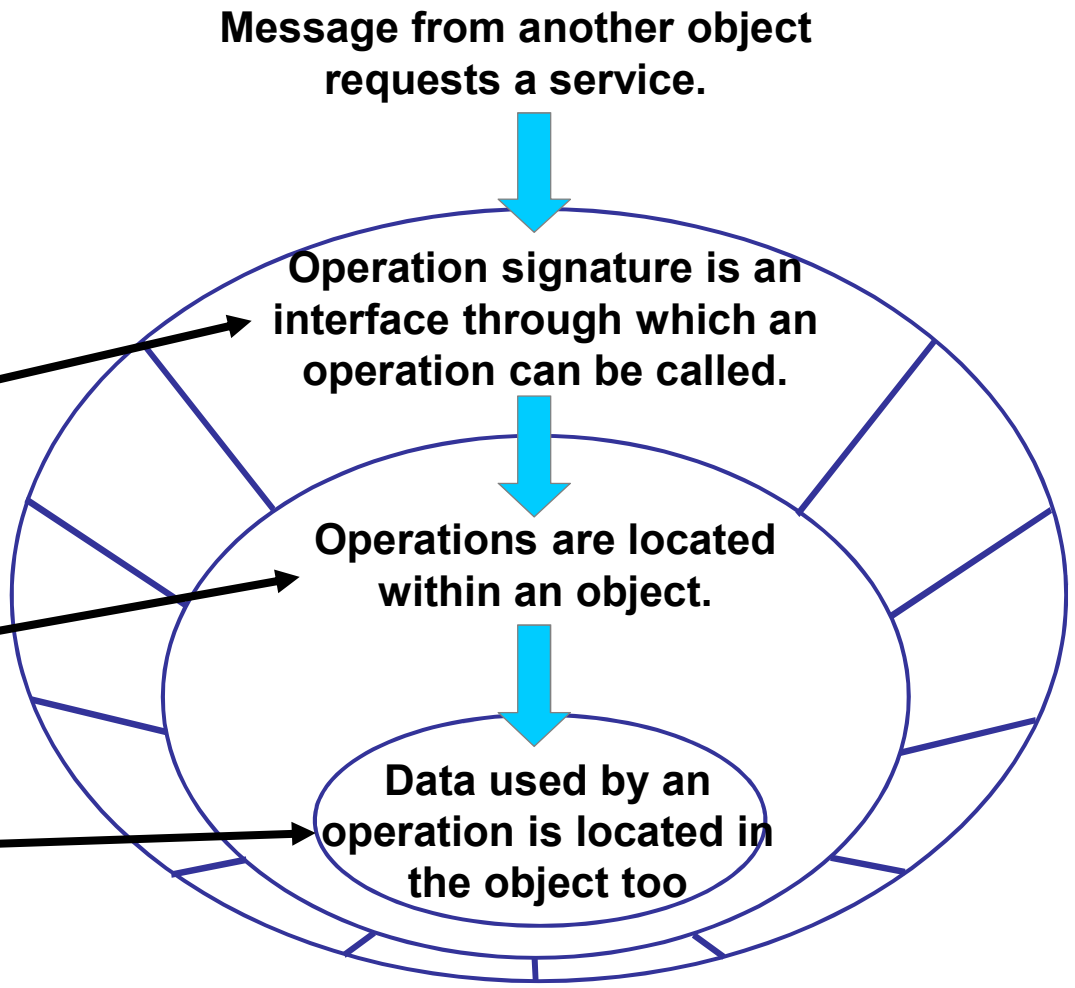
Message-passing and Encapsulation

‘Layers of an onion’ model of an object:

An outer layer of operation signatures…

…gives access to middle layer of operations…

…which access an inner core of data



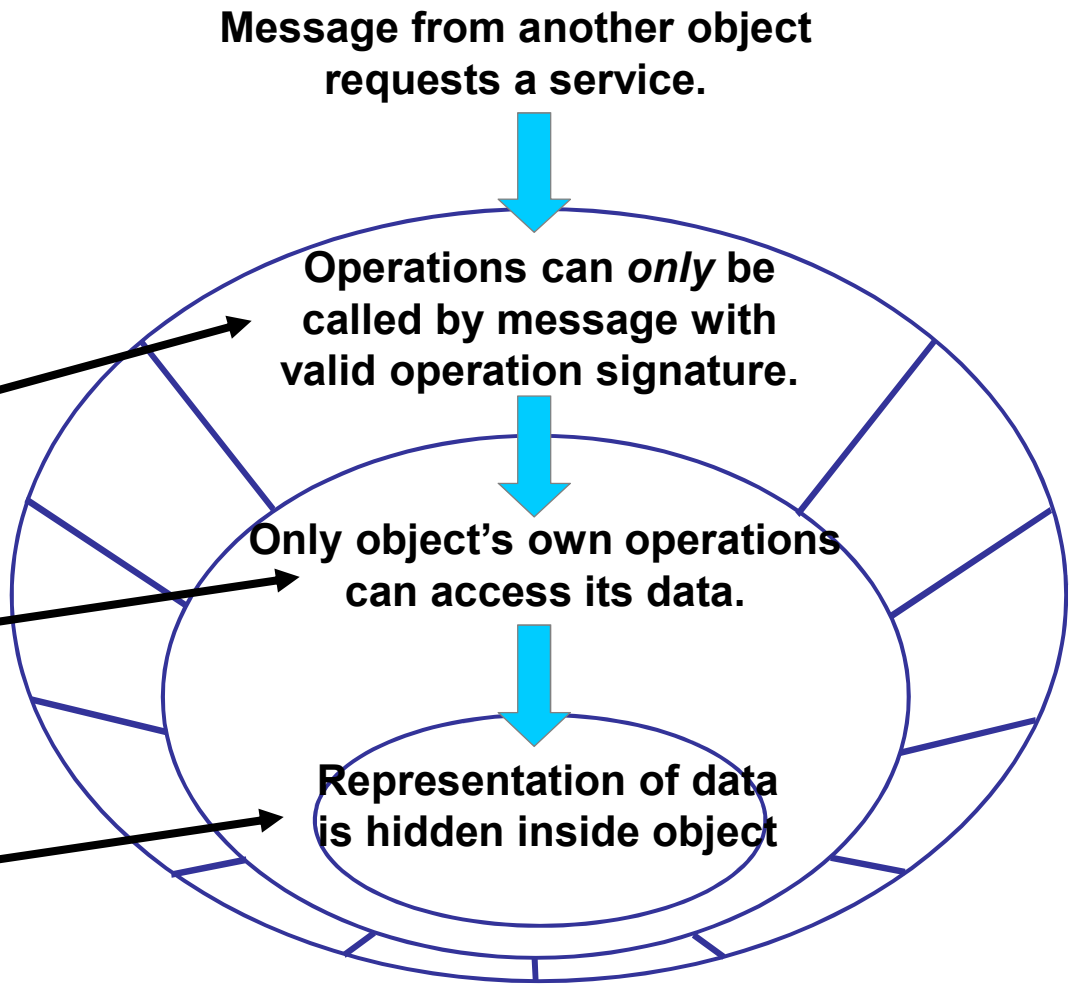
Information Hiding: a strong design principle

'Layers of an onion' model of an object:

Only the outer layer is visible to other objects…

…and it is the only way to access operations…

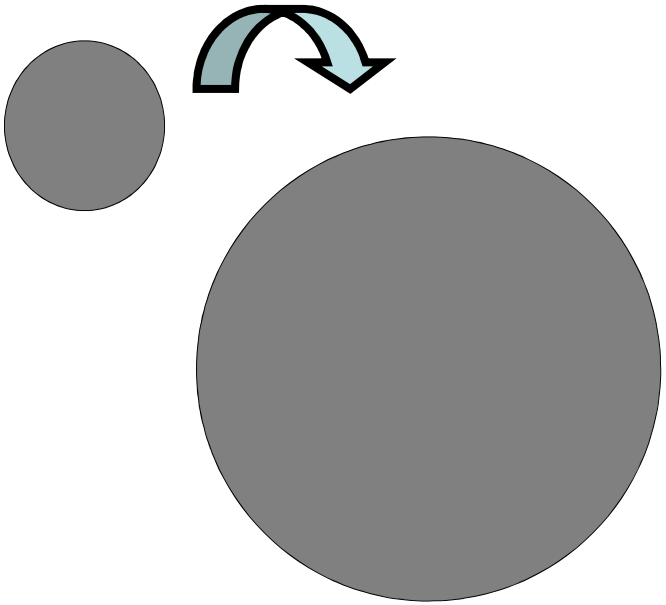
…which are the only way to access the hidden data

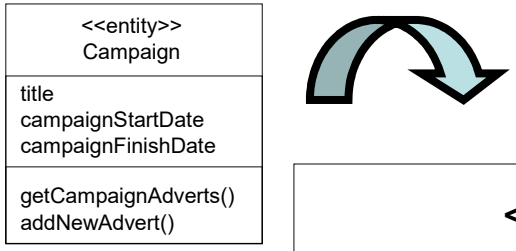


Polymorphism

- Polymorphism allows one message to be sent to objects of different classes
- Sending object need not know what kind of object will receive the message
- Each receiving object knows how to respond appropriately
- For example, a ‘resize’ operation in a graphics package

Polymorphism in Resize Operations





Advantages of O-O

- Can save effort
  - Reuse of generalized components cuts work, cost and time
- Can improve software quality
  - Encapsulation increases modularity
  - Sub-systems less coupled to each other
  - Better translations between analysis and design models and working code

Summary

In this lecture you have learned about:

- The fundamental concepts of O-O
  - Object, class, instance
  - Generalization and specialization
  - Message-passing and polymorphism
- Some of the advantages and justifications of O-O

References

- Coad and Yourdon (1990)
- Booch (1994)
- OMG (2009)

(For full bibliographic details, see Bennett, McRobb and Farmer)