What is Object-Orientation?

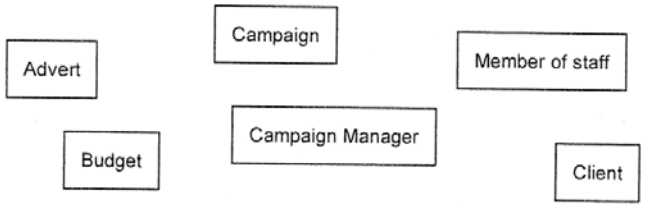
Chapter 4

In this Lecture you will Learn

- The main concepts introduced here are:
  - Objects, Classes and Instances
  - Object State
  - Generalization and Specialization
  - Message-passing and Encapsulation
  - Polymorphism

What is Object?

- An object is an abstraction of something in a problem domain, reflecting the capabilities of the system to
  - Beep information about it
  - Interact with it



More about Object

- Objects have state, behavior and identity
  - **State** – The condition of an object at any moment, affecting how it can behave
  - **Behavior** – What an object can do, how it can respond to events and stimuli
  - **Identity** – Each object is unique

State

- The state of the object is all the data which it currently encapsulates
- An object normally has a number of named attributes (or instance variables or data members) each of which has a value
- Some attributes can be mutable
  - An attribute ADDRESS
- Other attributes may be constant (immutable)
  - Date of birth
  - Identifying number

Behaviour

- The way an object acts and reacts, in terms of its state changes as message passing.
- An object understands certain messages, which is to say that it can receive the message and act on them.
- The set of messages that the object understands, like the set of attributes it has, is normally fixed.

Identity

- The idea is that objects are not defined just by the current values of their attributes
- An object has continues existence
  - For example the values of the object's attributes could change, perhaps in response to a message, but it would still be the same object.
- An object is normally referred to by a name, but the name of the object is not the same thing as the object, because the same object may have several different names

Examples

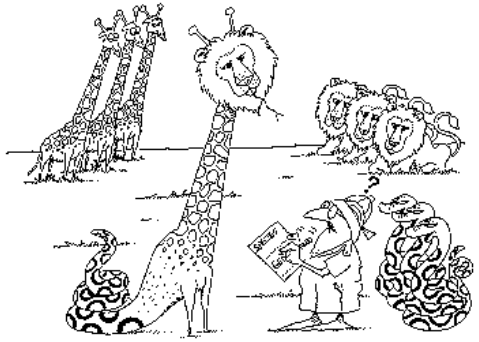
Object	Identity	Behaviour	State
A person.	**'Mary Lee'**	**Speak, walk, read.**	**Studying, resting, qualified.**
A shirt.	**My favourite button white denim shirt.**	**Shrink, stain, rip.**	**Pressed, dirty, worn.**
A sale.	**Sale no #0015, 16/07/05.**	**Earn loyalty points.**	**Invoiced, cancelled.**
A bottle of ketchup.	***This* bottle of ketchup.**	**Spill in transit.**	**Unsold, opened, empty.**

Class Exercise

- What does “Object State” mean?

Classification

- Objects with the same data structure (attributes) and behaviour (operations) are grouped into a class
- Each class defines a possibly infinite set of objects



Classes

- An object belongs to a group or category called a **Class**.
- Each object is an instance of a class
- Each object knows its class
- Each instance has its own value for each attribute (state) but shares the attribute names and operations with other instances of the class
- Class encapsulates data and behaviour, hiding implementation details of an object

Attributes

- Attributes describe the characteristics of an object.
- Objects can have a specific attribute called a state.
- The state of an object describes the object's current status.

Method

- A **Method** defines specific tasks that an object can perform.
  - Example:
    - ***Constructor Method:*** A method that creates a new instance of an object.
    - ***Update Method:*** A method that changes existing data.
    - ***Query Method:*** Method that provides information about an object's attributes

Class and Instance

- All objects are instances of some class
- A Class is a description of a set of objects with similar
  - Attributes
  - Operations and Methods
  - Relationships and Semantics

Relationships among Objects and Classes

- Relationships describe:
  - What objects need to know about each other
  - How objects respond to changes in other objects
  - The effects of membership in classes, super-classes, and sub-classes

Dependency

- Dependency occurs when one object must be informed about another.
  - E.g.: the SCHOOL_BUS object must be kept informed about the BUS_ROUTE object in order to make the correct stops at the correct time. (Any chances in BUS_ROUTE object will affect the SCHOOL_BUS object)

Association

- Association occurs when certain attributes of one object are determined by its interaction with another objects.

Aggregation

- Aggregation (part-of) relationship: An object can be composed of many component objects and may itself be a component object.
  - E.g., Airplane is an object composed of hundreds or thousands of component objects, such as wheels, wings, instruments, etc.

Class Exercise

- Define object, class and instance.

Class Exercise

- What do you think is meant by “semantics”?

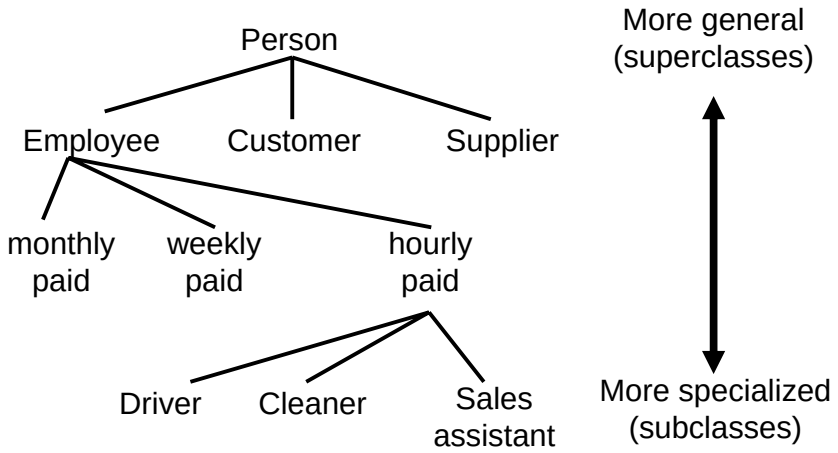
Structure and Behaviour

- An object is an instance that originates from a class, it is structured and behaves according to its class.
- The purpose of a class is to declare a collection of Methods, Operations and Attributes that fully describe the **Structure** and **Behaviour** of objects.
  - ***Structure***: what an object *knows*, information that it holds
  - ***Behaviour***: what an object *can do*

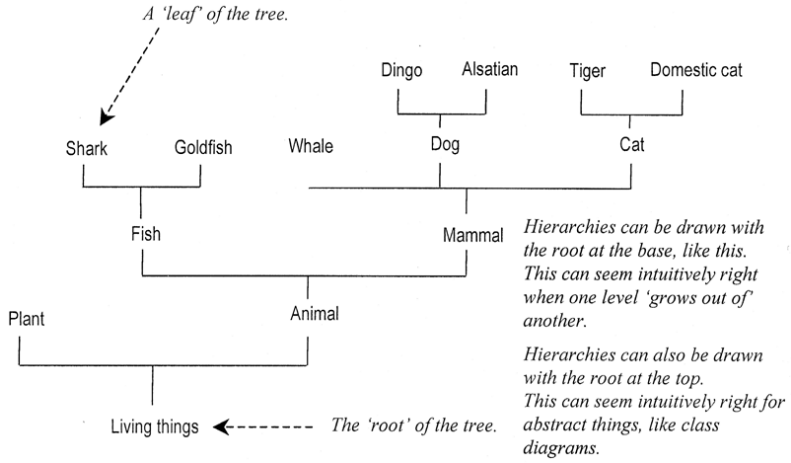
Generalization and Specialization

- 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



Another Hierarchy Example



Generalization and Specialization

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

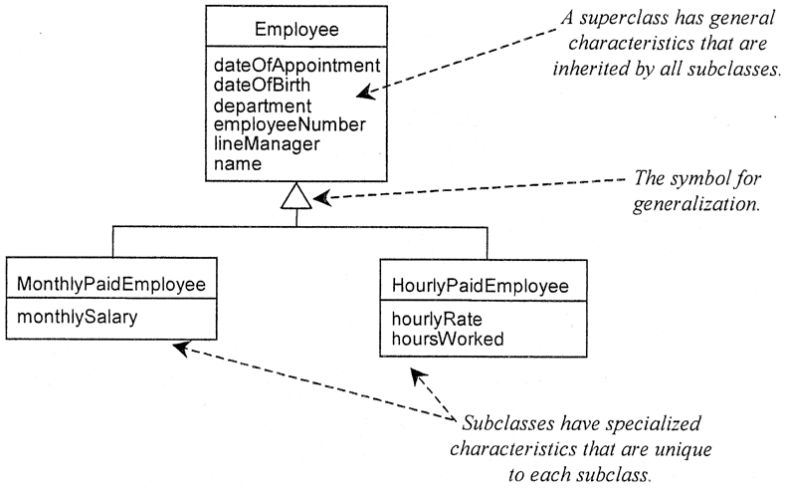
General (superclass)

Person
Name Date of birth Gender Title

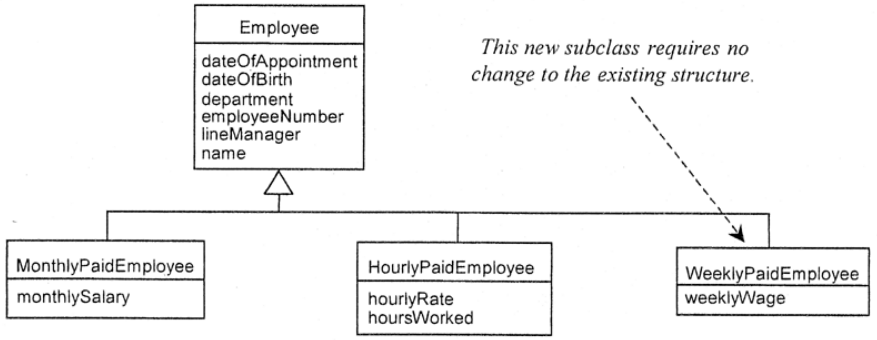
Specialized (subclass)

HourlyPaidDriver
StartDate StandardRate OvertimeRate LicenceType

Generalization Notation



Extend in Hierarchies



Class Exercise

- What is the difference between generalization and specialization?

Inheritance

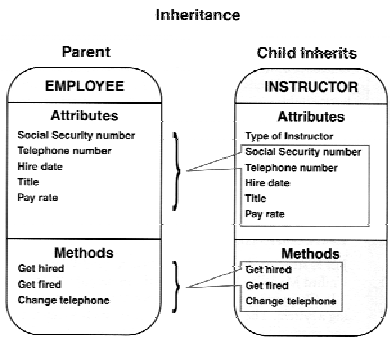
- The *whole* description of a superclass applies to *all* its subclasses, including:
  - Information Structure
  - Behaviour
- Often known loosely as *inheritance*
  - Actually, inheritance is the facility in an O-O language that implements generalization / specialization

Inheritance

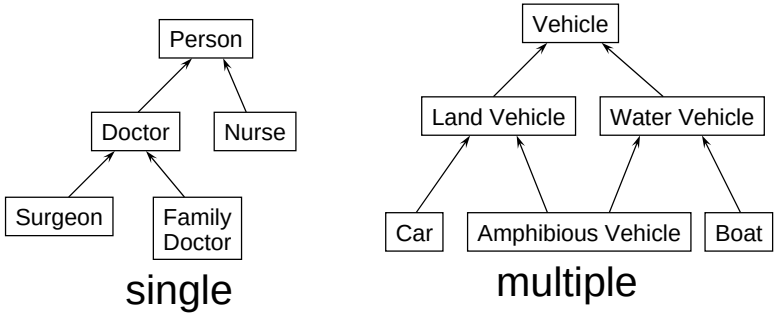
- Inheritance is the mechanism by which object-oriented programming languages implement a relationship of generalization and specialization between classes.
- A subclass automatically acquires features of its superclasses.

Subclass vs. Superclass

- A **Subclass** is an extended, specialized version of its **Superclass**. It includes the operations and attributes of the Superclass, and possibly some more.
- Inheritance enables an object to derive one or more of its attributes from another object.



Type of Inheritance

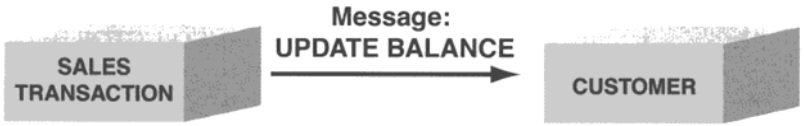


Class Exercise

- What rules describe the relationship between a subclass and its superclass?

Message

- A **Message** is a command that tells an object to perform a certain method.

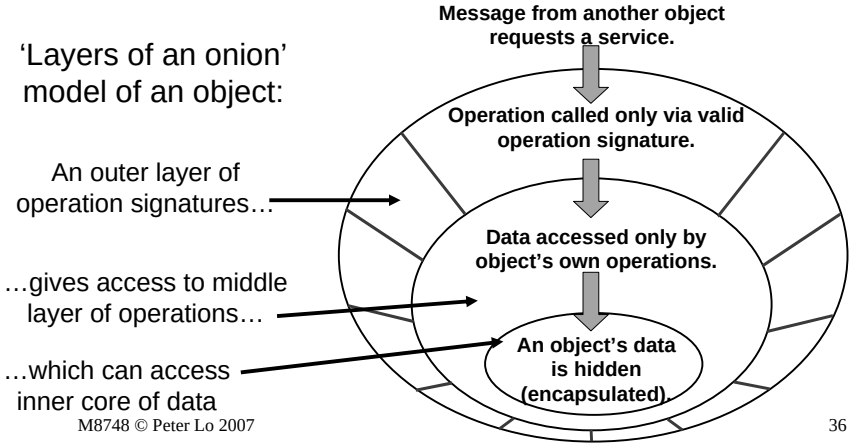


Message-passing

- 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

- The layers of protection that surround an object



Encapsulation

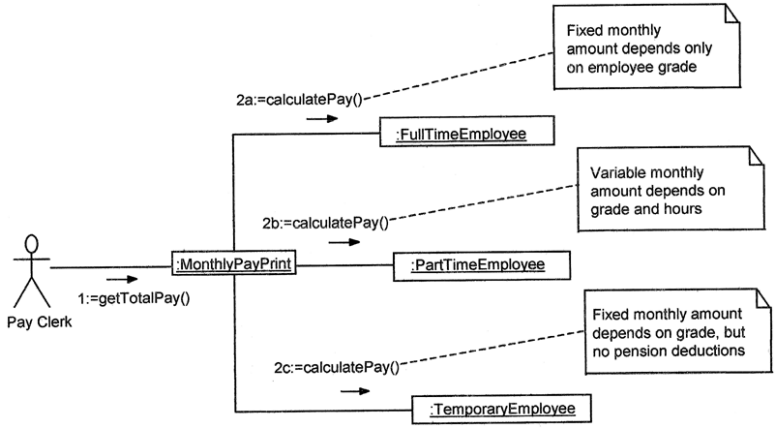
- An object can viewed as a black box because a message to the object triggers changes within the object without specifying how the changes must be carried out.
- The black box concept is an example of **Encapsulation:** all data and methods are self-contained.
- Definition of Encapsulation: *The object encapsulates or hides both data and the logical procedures required to manipulate the data*

Class Exercise

- How does the object-oriented concept of message passing help to encapsulate the implementation of an object, including its data?

Polymorphism

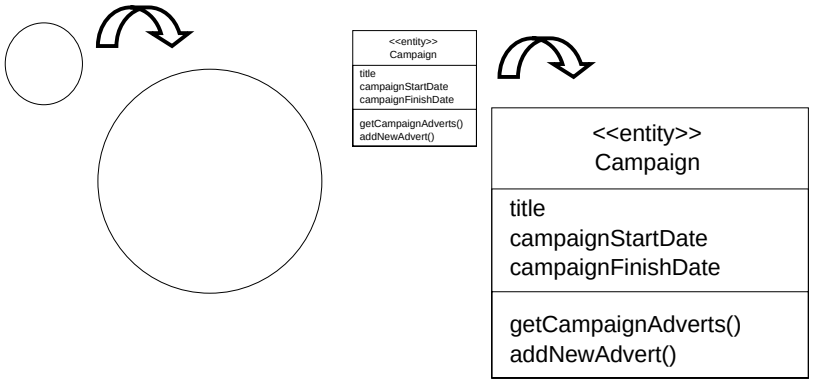
- Polymorphism allows one message to be sent to objects of different classes



Polymorphism

- 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



Class Exercise

- What is polymorphism?

Advantages of OO

- 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

Class Exercise

- Why is it particularly hard for a designer to anticipate a user's sequence of tasks when using a GUI application?