CS4125
SYSTEMS ANALYSIS
SPRING SEMESTER 2010-2011

J.J. Collins
Dept of CSIS
University of Limerick

Lecture C (W2L1)

(a) Interfaces cont.
(b) The Object-Oriented (OO) paradigm – Part 1.

1. More on Interfaces.

- Look at coding fragment at end of handout
  - Person has an address
  - Will see these coding fragments later in semester

2. More on Interfaces - compliance

	Signature / Call	Syntactically Correct	Semantically Correct
Server	+ debit (amount: integer, accountNo: integer): inetger		
Client 1 call	debit(100,176588932)	YES	YES
Client 2 call	debit(100.00, 176588932)	NO	YES
Client 3 call	debit(17658892,100)	YES	NO

Note:

1. Compilers enforce syntactic checks.
2. Cannot enforce semantic compliance.

1. More on Interfaces.

- Operations in interfaces have pre and post conditions
  - Conceptually similar to the fine print in a contract
  - If pre-condition is not satisfied, no obligation on service provider to support a valid engagement
  - Otherwise, service provider guarantees that the state of the salient parts of the system will be as specified in post condition
  - Example: operation debit(account no, amount) on class Account
    - Pre: Account.balance – amount > Account.overdraftLimit;
    - Post: Account.balance > Account.overdraftLimit;
  - Pre conditions can help eliminate defensive programming

2. Key Features of the OO Paradigm

- Classes and Objects
- Generalisation
  - Programmers refer to this as inheritance
- Polymorphism
- Templates

3. Classes and Objects

- Object. Abstraction of something in a problem domain, reflecting the capabilities of the system to keep information about it, interact with it, or both. (Yourdon and Coad, 1989).
- An object has state, behaviour and identity (Booch, 1994).
- Objects are sometimes deliberately characterised as if each is a person with roles (Wrifs-Brock et al., 1990):
  - Who am I?
  - What can I do?
  - What do I know?
- An object represents a particular instance of a class (Rational, 1997).
- An object is not required to have a physical manifestation in the real world e.g. sales object, campaign object.
- Classes are intended to be loosely coupled, highly cohesive modules.

3. Classes and Objects

- A class is a specification or template from which instances of objects are derived.
- Terms "class" and "object" are synomonous in the profession.
- Logical tests for class membership:
  - Share a common set of descriptive characteristics.
  - Share a common set of valid behaviours.

3(a). Method and Messages

- Procedural decomposition:
  - Focus on specifying procedures, implement using
    - Procedural language such as C, Basic, Fortran, Modula, etc.
  - Data is globally visible.
  - This gives rise to high dependency in the system. For example:
    - Operation *o1()* uses global data *d*
    - Operation *o2()* uses global data *d*
    - Therefore, *o1()* and *o2()* have a derived dependency on each other
  - Data dependency
  - Dependency of process upon data structure can cause problems e.g. if data structures changed.
  - Example of coupling between subsystems.

3(a). Methods and Messages

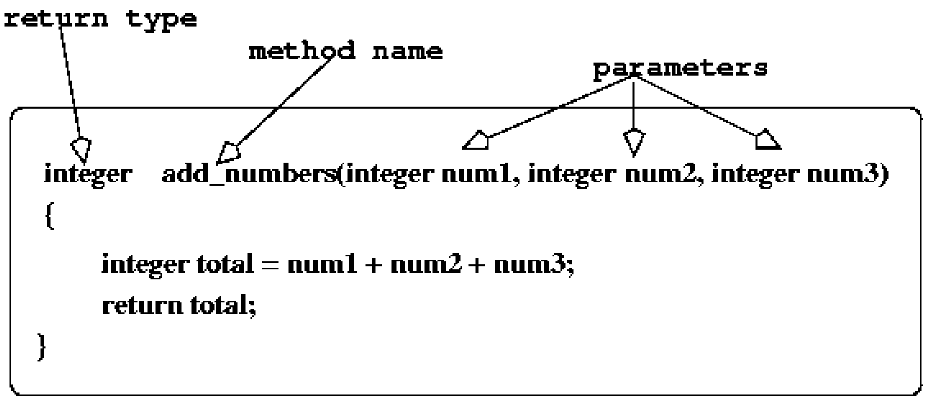
- Object oriented: locate each process with the data it uses.
- Processes are called operations or methods.
- Each has a specific signature - also known as message protocols.
- An operations signature is a definition of its interface.
- In order to invoke an operation, its signature must be given by the caller.
- Caller sends a message to the callee.
- Caller invokes callee.

3(a). Methods and Messages

Methods:

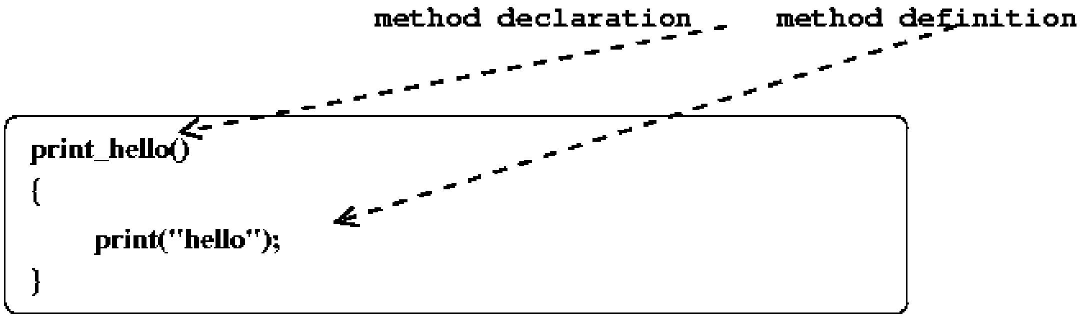
- A method is an operation that can be performed or executed by the object that is responsible for it, the object being an instance of a class.
- A method is made up of:
  - A method heading or declaration.
  - A method body or definition.

3(a). Methods and Messages



```
integer    add_numbers(integer num1, integer num2, integer num3)
{
      integer total = num1 + num2 + num3;
      return total;
}
```

Method that adds three numbers.



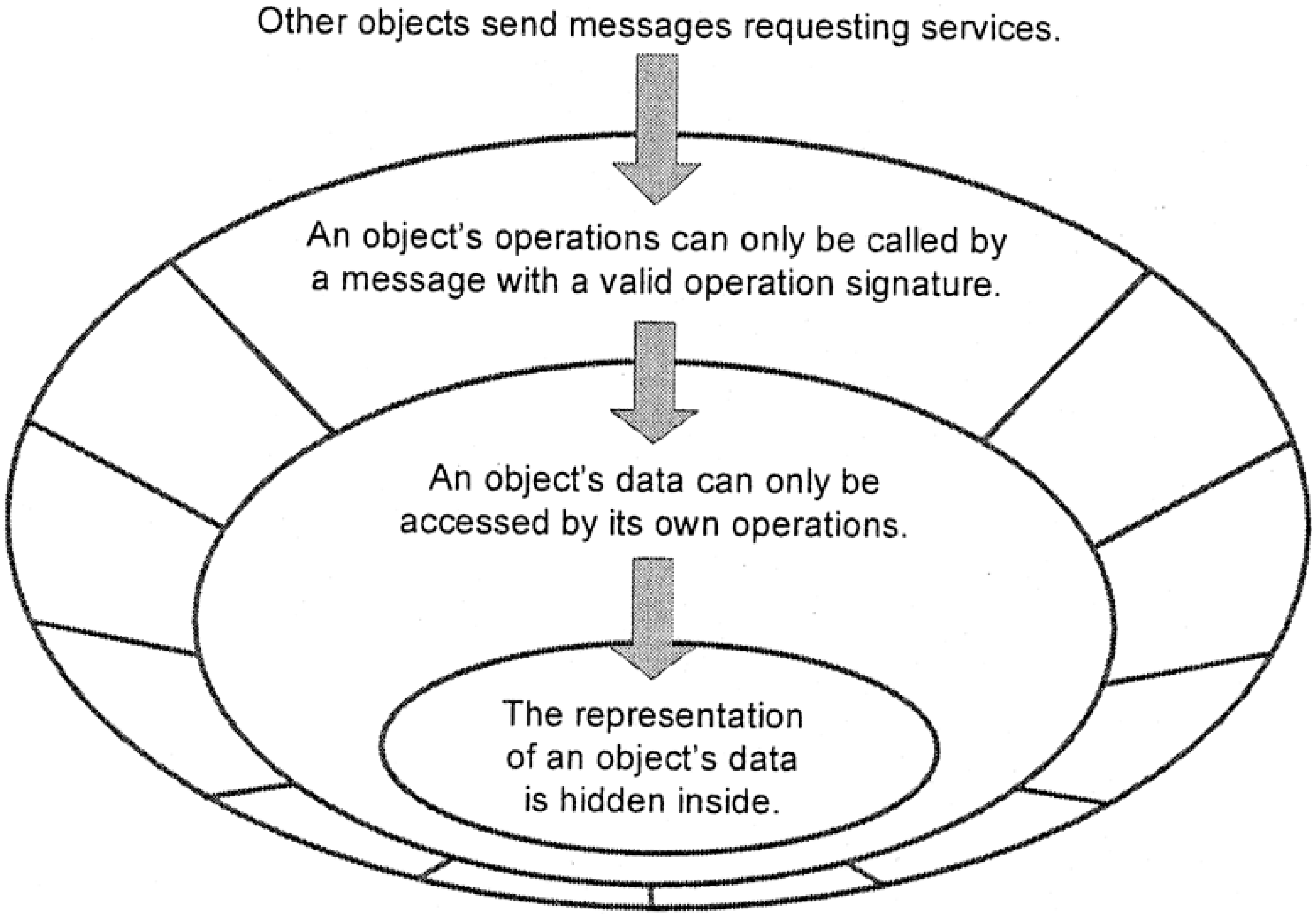
```
print_hello()
{
      print("hello");
}
```

Method that prints "hello" to the screen.
Note that this method does not have a return type, and no parameters.

3(a). Methods and Messages

- The definition supplies an implementation that supports the operation specified in the declaration.
- The method heading (declaration) specifies an interface or signature for that method.
- Programmers code the implementation (definition) for each method in the class diagram.

3(a). Methods and Messages



Other objects send messages requesting services.

An object's operations can only be called by a message with a valid operation signature.

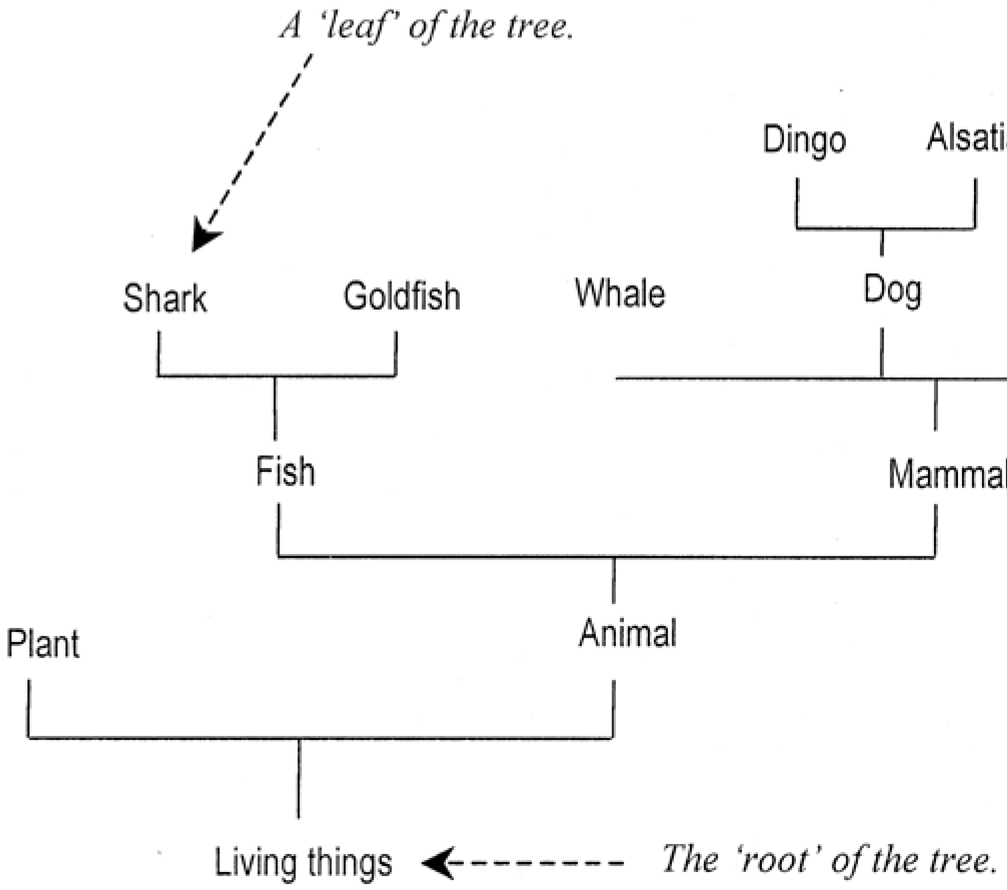
An object's data can only be accessed by its own operations.

The representation of an object's data is hidden inside.

4. Generalisation

- Generalisation: taxonomic relationship between a more general element and a more specific element that is fully consistent with the first element and that adds additional information - UML Semantics Guide (Rational, 1997).
- Taxonomic: a hierarchy of relationships.
- Object classes can be arranged into hierarchies.
- Known as Inheritance in OO languages
- superclass and subclass.

4. Generalisation

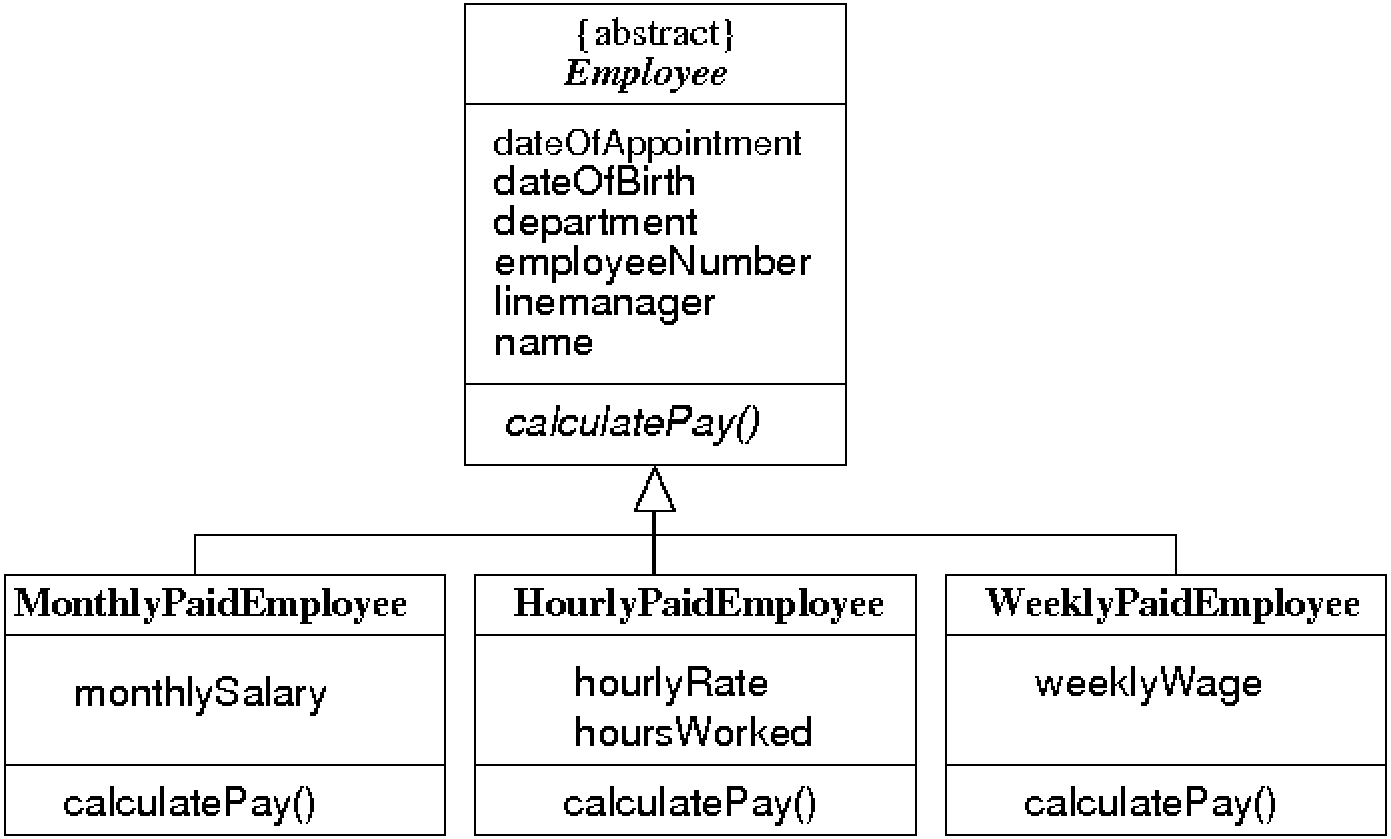


Hierarchies can be drawn with the root at the base, like this. This can seem intuitively right when one level 'grows out of' another.

Hierarchies can also be drawn with the root at the top. This can seem intuitively right for abstract things, like class diagrams.

4. Generalisation

- Class diagram showing generalisation
- Superclass in this example happens to be an abstract class.



4. Generalisation

- Suppose that we have designed the class *Lecturer*, and now must develop the class *DirectorOfStudies*.
- Implement using inheritance.
- Terminology
  - *DirectorOfStudies* inherits from *Lecturer*
  - *DirectorOfStudies* is a subclass (or derived class) of Lecturer
  - *DirectorOfStudies* is a specialisation of *Lecturer*
  - *DirectorOfStudies* is more specialised than *Lecturer*
  - *Lecturer* is a superclass (or base class) of *DirectorOfStudies*.
  - *Lecturer* is a generalisation of *DirectorOfStudies*.

Reading

- Bennett, McRobb, and Farmer: chapter 4