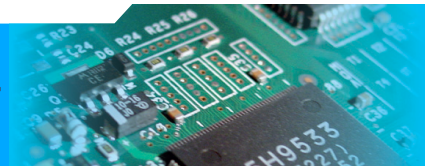
Terminologie

Noël PLOUZEAU

IRISA/ISTIC



A glossary

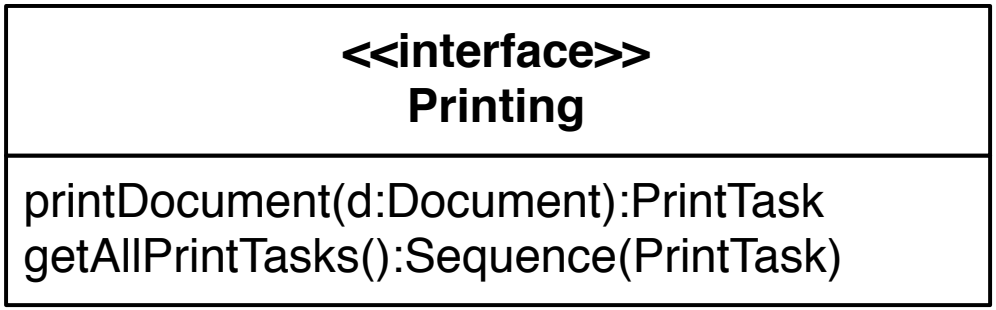
- The Internet and the SE domain provides several meanings for a given word
- For clarity we need one word for one concept
- ACO uses mainly definitions from UML, including for coding

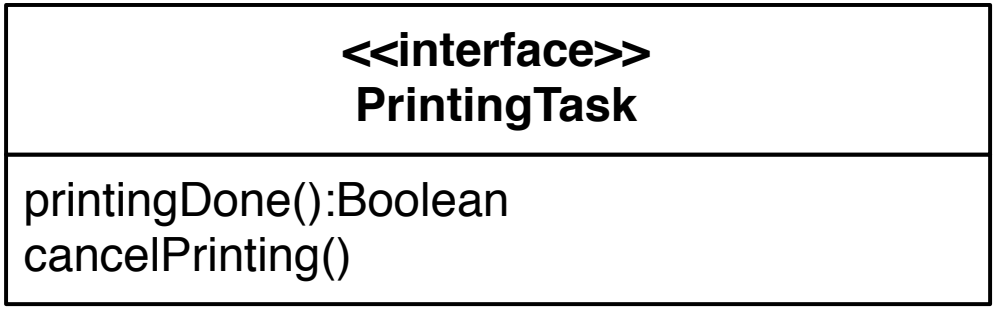
Interface vs implementation

- This separation is a critical key concept in software engineering
  - We will see that many good design principles rely on it
- Object-oriented software has a strong support for it (and this is the main reason to use OO design and code)

Interface concept

- An interface
  - includes all what is needed to ask for a service
  - and nothing more


<<interface>>
Printing
printDocument(d:Document):PrintTask
getAllPrintTasks():Sequence(PrintTask)


<<interface>>
PrintingTask
printingDone():Boolean
cancelPrinting()

Equivalent Java interface

- interface Printing {
  - public PrintTask printDocument(Document d);
  - public List<PrintTask> getAllPrintTasks();
- }

Operations

- An interface (in UML, Java, C#, C++, etc) contains only operations
- To request a service from an object one needs nothing more: no notion of implementation in an operation

Class

- In most cases it is an implementation of an interface
- When accessing an object you don’t need to know its class, only its interface
  - This is an important point

Attributes

- Internal state of an object
- A good design must preserve encapsulation (we will see how to ensure this later)

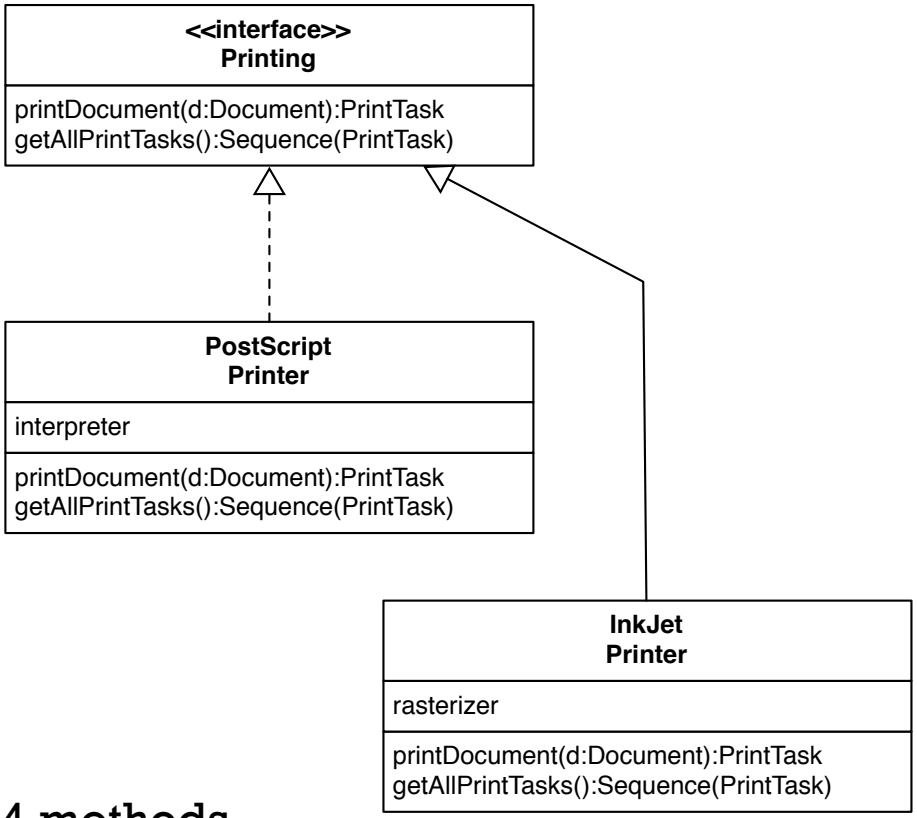
Message

- An object *o1* requests a service from another object *o2*
  - by sending a message
  - and waiting (or not) for a reply
- Messages are native in the UML, not in Java
- In this lecture we will not use the full concept
  - We consider that the message concept is equivalent to the operation concept

Method

- An implementation of an operation (usually code)
- Object-oriented languages allow for 0, 1 or more implementations for an operation
- Beware: many people use the word method where I say operation
  - The *method* word means an algorithm, ie *how* not *what*

Operation != Method



2 operations, 4 methods

Execution of an operation

- Printing printer = …; Document doc = …;
- printer.printDocument(doc)
- The runtime looks for a **method** for this operation
  - if one is defined in printer's object class it is run
  - otherwise the runtime looks in the ancestor class(es)

Abstract versus concrete class

- A concrete class has at least one method for each operation
- An abstract class has at least one operation without a method

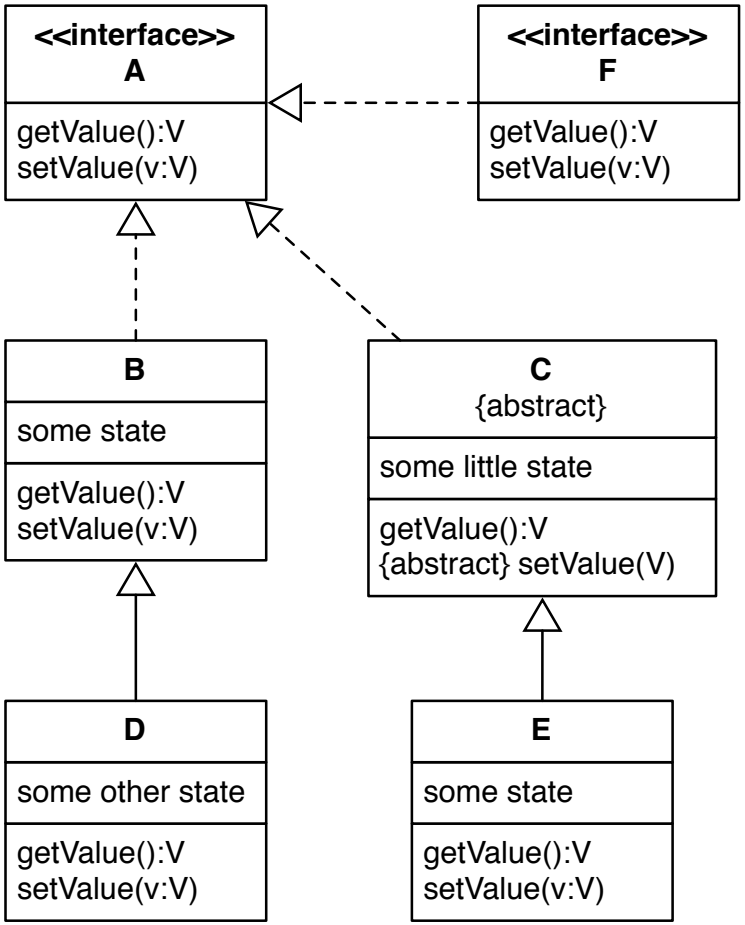
In Java, C#, C++

- A method search is always successful
  - because you cannot instantiate an object of an abstract class
  - this means that each operation has at least one implementation (*ie* method)

Inheritance

- Two forms:
  - operation inheritance
  - method inheritance
- Really different
  - this is why it is wise to use two different terms (operation and method)

Example of inheritance (UML)



References for the languages

- Java spec
  - http://docs.oracle.com/javase/specs/
- UML spec
  - http://www.uml.org