Object-Oriented Programming

Encapsulation Control/Visibility (VRH 7.3.3)

Carlos Varela

RPI

Adapted with permission from:

Seif Haridi

KTH

Peter Van Roy

UCL

April 16, 2015

Controlling visibility

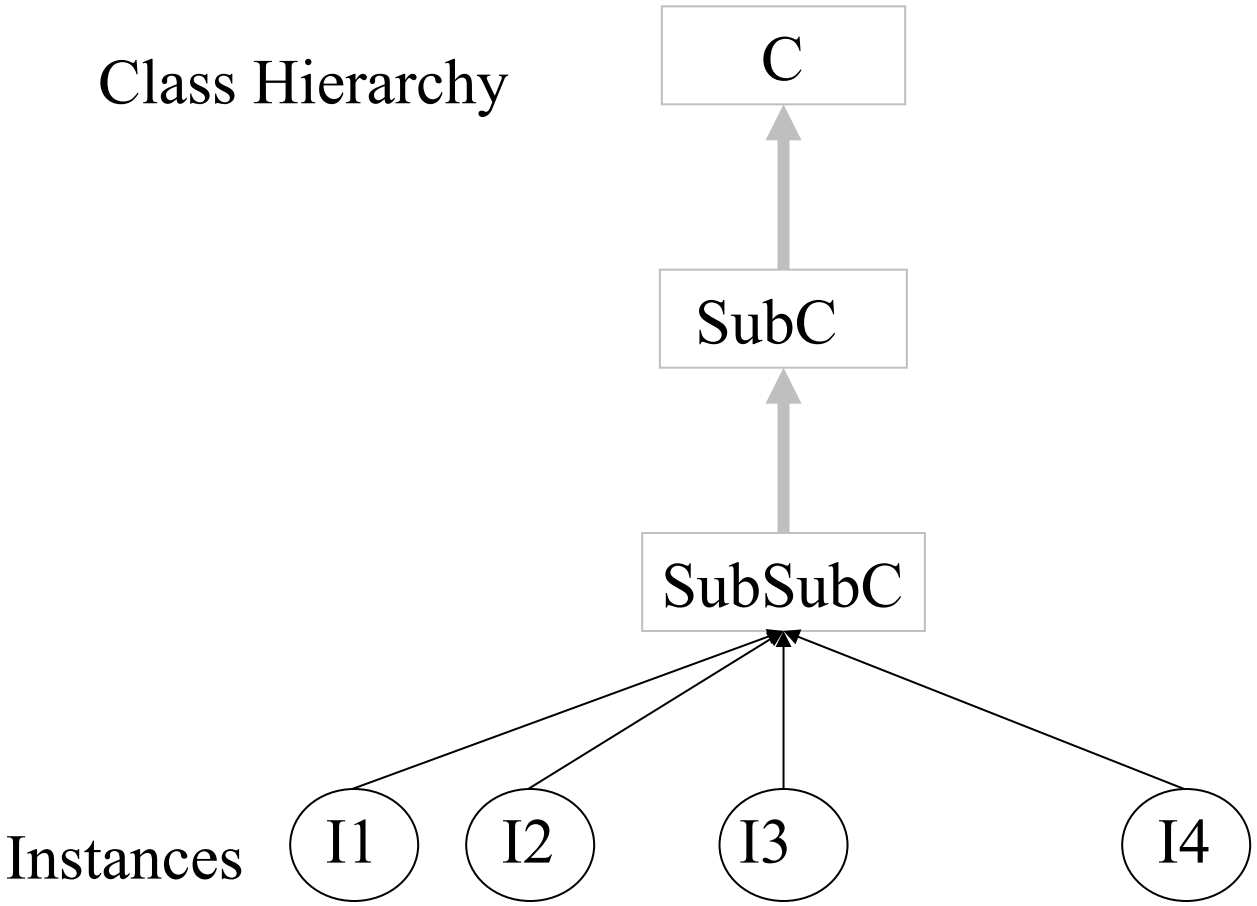
- Visibility is the control given to the user to limit access to members of a class (attributes, methods and properties)
- Each member is defined with a scope (part of program text that the member can be accessed by name)
- Programming languages use words like *public*, *private* and *protected* to define visibility
- Unfortunately different languages use these keywords to define different scopes

Public and private scopes in ADTs

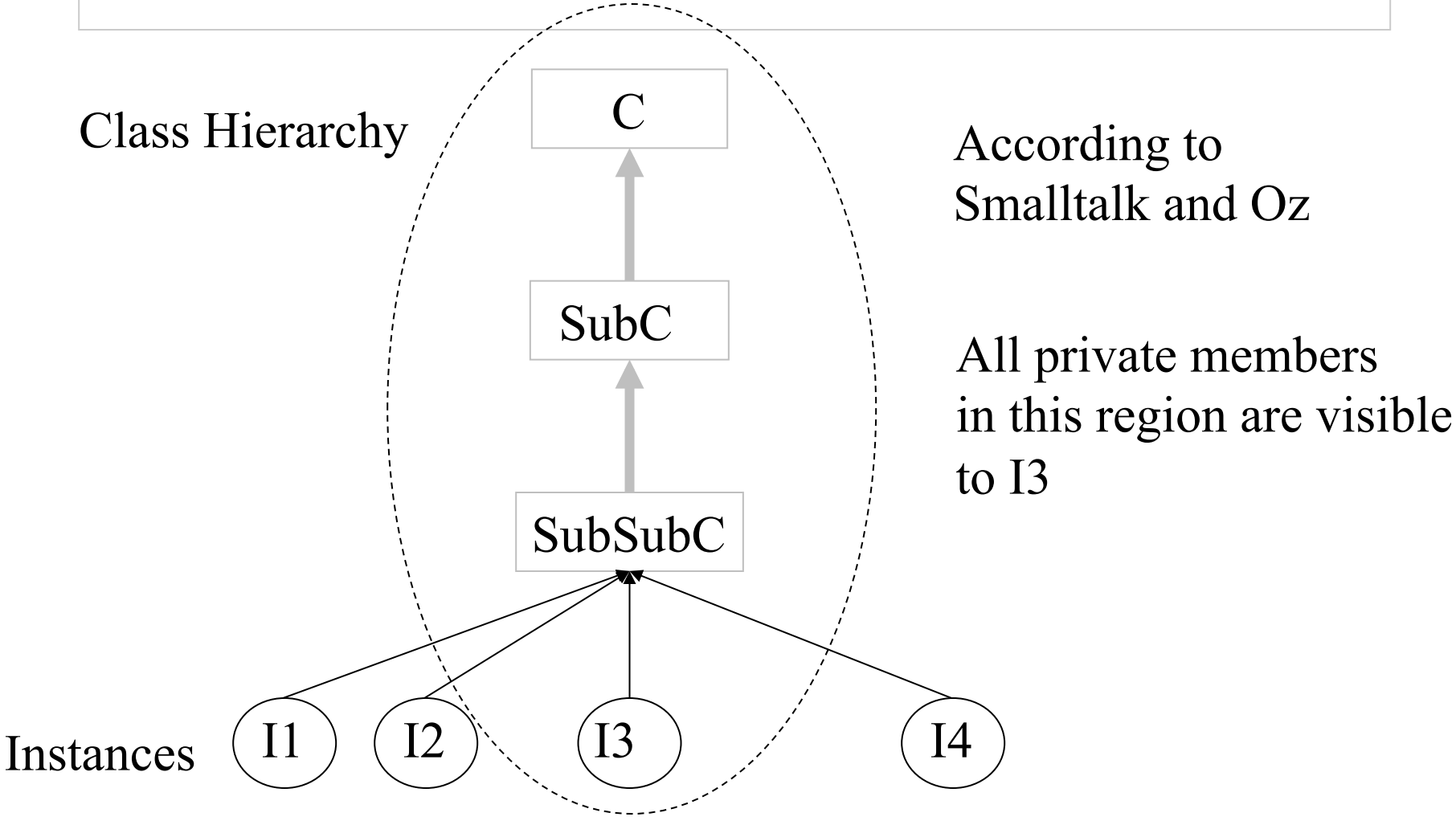
- A private member is one which is only visible in the object instance (it is used for implementing the ADT)
- The object instance can see all the private members in its class and its super classes
- A public member is visible anywhere in the program
- It is part of the interface of the ADT

- In Oz (and Smalltalk) attributes are private and methods are public (the default rule)
- In Java and C++ private has another meaning

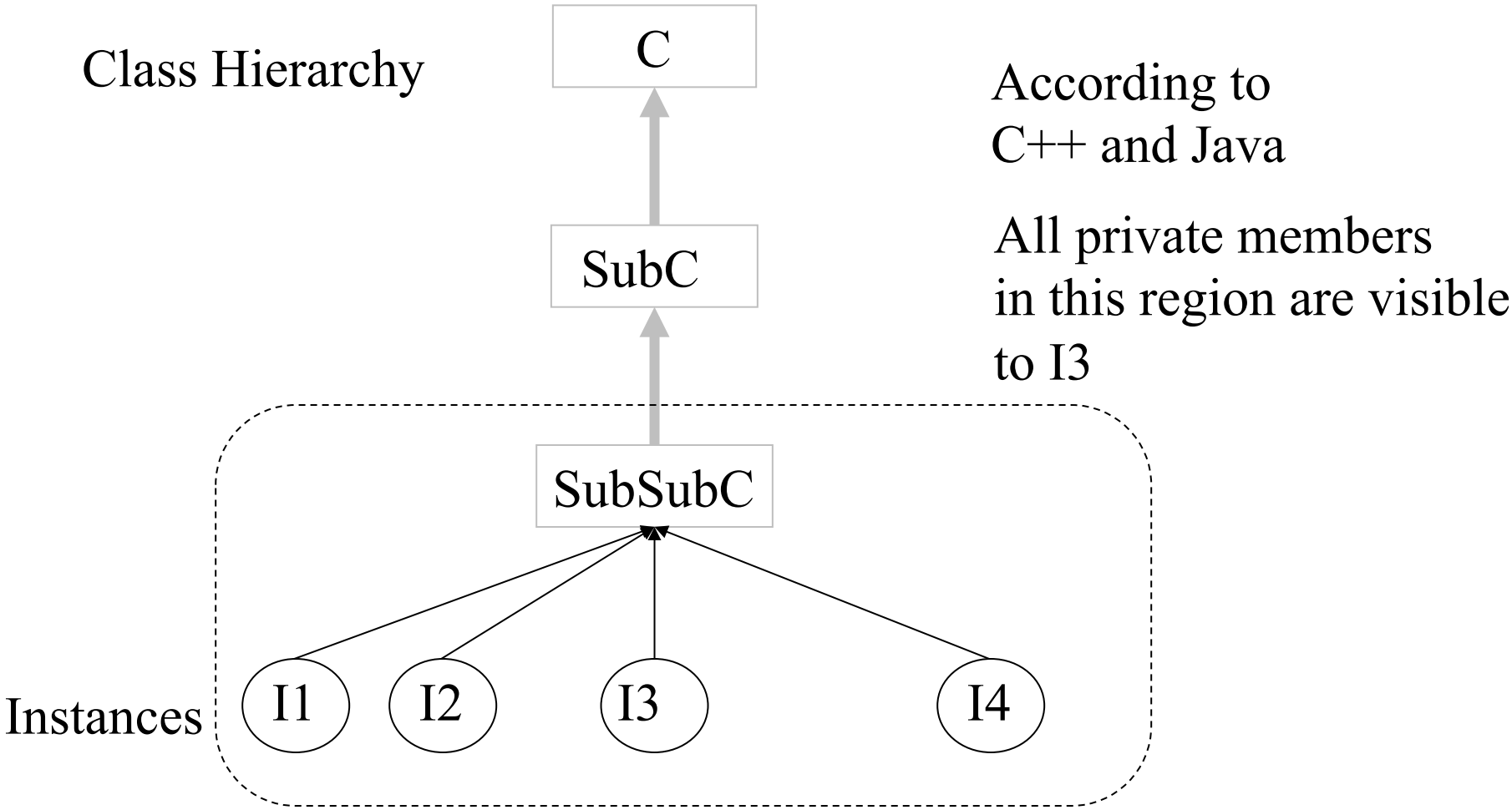
The meaning of Private



The meaning of Private



The meaning of Private



Public and private scopes in ADTs

- In Oz (and Smalltalk) attributes are private and methods are public
- It is possible in Oz to make a method private within a class
- Using a variable identifier as a method head will make the method local to the class
- The variable is automatically bound to a unique name

```
class C
   meth A(...) ... end
....
end
```

Public and private scopes in ADTs

- In Oz (and Smalltalk) attributes are private and methods are public
- It is possible in Oz to make a method private within a class
- Using a variable identifier as a method head will make the method local to the class
- The variable is automatically bound to a unique name
- ! is an escape character, !A means escape the class scope

```
class C
   meth A(...) ... end
....
end
```

```
local A = {NewName} in
   class C
      meth !A(...) ... end
    ....
   end
end
```