



Learning Object-Oriented Programming and Design with TDD

About private methods

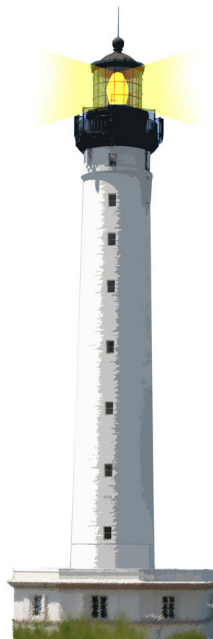
S. Ducasse

<http://stephane.ducasse.free.fr>



<http://www.pharo.org>

WXSYY



Private... which one?

- Pay attention languages can have different interpretations...
- Private Ruby methods are different from private Java ones
- You can have private virtual in CPP but they can be tricky



Are private methods inherited? In Java

```
class A {  
    public void m() { this.p(); }  
    private void p() { println("A.p()"); }  
}  
class B extends A {  
    private void p() { println("B.p()"); }  
}
```

Which is called? A.p() or B.p()?

```
A b = new B();  
b.m();
```



Are private methods inherited? In Java

```
class A {  
    public void m() { this.p(); }  
    private void p() { println("A.p()"); }  
}  
class B extends A {  
    private void p() { println("B.p()"); }  
}
```

Which is called? A.p() or B.p()?

```
A b = new B();  
b.m();  
>>> A.p()
```

Because private methods in Java are statically bound

- No method lookup
- You cannot call them from subclass



Private in Ruby

- Ruby is one of the few dynamic languages that offers access qualifiers: methods can be qualified as public, protected and private.
- a private method may be made public in subclasses. Private methods can only be invoked by sending a message to an implicit receiver (i.e., no use of self).
- However a call to a private method is not statically bound to the method defined in the class but can be overridden in subclasses.



Private in Ruby

```
class C
  def zork(arg) ; return arg.x ; end
  def foo ; self.x end
  def foo2 ; x; end
  private def x;return1; end
end

class D < C
  public
  def x; return2; end
end
```

Results:

```
C.new.foo ==> failed
C.new.foo2 ==> 1
D.new.foo ==> 2
```



Private in Ruby

- class C defines a private method x.
 - The method foo does not invoke this method since it does not use an implicit receiver but sends the message x to self. This is why C.new.foo raises an error.
- The method foo2 invokes x with an implicit receiver (i.e., no self).
 - C.new.foo2 executes the private method x and returns 1.
- Now this is different if a subclass defines a public method x returning 2.
 - D.new.foo2 returns 2 and not 1 even though the method x is private and the method foo2 calls x with an implicit receiver.
- This shows that Ruby's private methods are dynamically resolved.



Virtual Private in CPP

```
class HTMLDocument : public Document, public CachedResourceClient {  
private:  
    virtual bool childAllowed(Node*);  
    virtual PassRefPtr<Element> createElement(const AtomicString& tagName,  
        ExceptionCode&);  
};
```

- Derived classes may override the function to customize the behavior as needed.
- But without exposing the virtual functions directly by making them callable by derived classes
- But but if you declare the method private then you cannot invoke it from its overridden versions.
- A virtual protected would let the derived class be callable in derived classes

<https://isocpp.org/wiki/faq/strange-inheritance#private-virtuals>

Better use protected :)



Virtual Private Limit in C++

```
class Base {  
private:  
    int m_data;  
    virtual void cleanup() { /*do something*/ }  
protected:  
    Base(int idata): m_data (idata) {}  
public:  
    int data() const { return m_data; }  
    void set_data (int ndata) { m_data = ndata; cleanup(); }  
};
```

```
class Derived: public Base {  
private:  
    void cleanup() override {  
        // do other stuff  
        Base::cleanup(); // nope, can't do it }  
public:  
    Derived (int idata): base(idata) {}  
};
```



What you should know

- Each language has its own definition
- Private methods in Java are statically bound



A course by Stéphane Ducasse
<http://stephane.ducasse.free.fr>

Reusing some parts of the Pharo Mocc by

Damien Cassou, Stéphane Ducasse, Luc Fabresse
<http://mocc.pharo.org>



Except where otherwise noted, this work is licensed under CC BY-NC-ND 3.0 France
<https://creativecommons.org/licenses/by-nc-nd/3.0/fr/>