



**Learning Object-Oriented  
Programming and Design with TDD**

# Inheritance and lookup

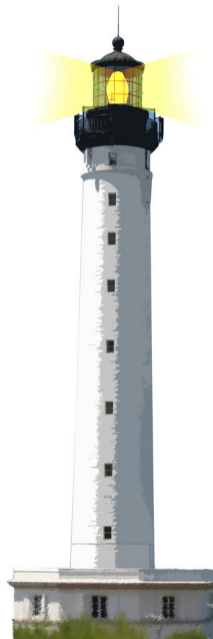
S. Ducasse

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



<http://www.pharo.org>

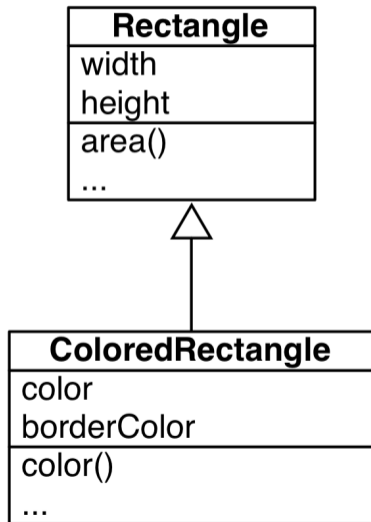
**WXSYY**



# Remember Inheritance

a subclass

- can add state and behavior:
  - color, borderColor, ...
- can use superclass behavior and state
- can specialize and redefine superclass behavior



## Example: state

```
public class Rectangle {  
    protected int length;  
    protected int width;  
  
    public int getArea() {  
        return length * width;  
    }  
}
```

```
class Box extends Rectangle{  
    protected int height;  
}
```



# Box constructors

```
class Box extends Rectangle{  
    public Box(){  
        super();  
        height = 0;  
    }  
}
```

```
public Box(int length, int width, int height) {  
    super(length, width);  
    this.height = height;  
}
```



# Constructors and inheritance

- Each subclass should call its superclass constructor explicitly
  - as a first line of the constructor
- The call to the default constructor can be omitted (but the compiler does it for you)
- `super` in that case indicates the superclass.

A constructor is a static function not a method

- There is NO LOOKUP/Inheritance of constructors



# Simple Lookup

Methods of the superclass can be invoked on subclass instances

```
Box b = new Box (10,20,20);  
System.out.println(b.getArea());
```



# Subclass can enrich its API

```
public class Box extends Rectangle {  
    ...  
    public double getVolume() {  
        return getArea() * height;  
    }  
}
```



# Lookup

```
class A {  
    public int bar() { return foo();}  
    public int foo() { System.out.println("A"); return 1;}  
}  
class B extends A {  
    public int foo() { System.out.println("B"); return 2;}  
}  
  
class Runner {  
    public static void main(String[] args) {  
        A a = new A();  
        B b = new B();  
        a.bar();  
        b.bar();  
    }  
}
```

We will revisit it in the self (this)/super lecture...



# Subclass may want to access hidden superclass

```
public class Box extends Rectangle {  
    ...  
    public double getArea() {  
        return (super.getArea() + height * length + width * height) * 2;  
    }  
}
```

- `super.getArea()` executes the method `rectangle.getArea` on the box instance



# Lookup with Super

```
class A {  
    public void bar() { foo(); }  
    public void foo() { print("A"); }  
}  
class B extends A {  
    public void bar() { super.bar(); }  
    public void foo() { print("B"); }  
}
```

```
(new A()).bar();  
(new B()).bar();
```



# super in super.myMethod();

- super refers to the object the receiver of the message
  - super == this
- super changes the method lookup to start in the superclass of the class using super.
- super does not start lookup in the superclass of the class of the receiver



# Static variables

- Modifier 'static'
- Linked to the class
- Variable unique and shared between all instances of the class
- Kind of global since the class is a global too

```
class Circle {  
    static int PI = 3.14159265;  
    int rayon; //instance level  
    int circonference() {  
        return 2*PI*rayon; // or Circle.PI  
    }  
}
```



# Static methods

- They are not methods! Just functions
- No dynamic binding!
- No possibility to use classes as registration mechanisms
  - for the same you need a singleton and a factory
- check Class Method at Work from <http://mooc.pharo.org> (W4S6)



# About main and command line

- one single parameter: a table of strings
- argv.length

```
public class Echo {  
    public static void main(String[] argv) {  
        for (int i=0; i<argv.length; i++)  
            System.out.print(argv[i]+" ");  
            System.out.println();  
    }  
}
```

```
>>>java Echo OOP is COOL, Java less
```

```
>>>OOP is COOL, Java less
```



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/>