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

# About Constructors
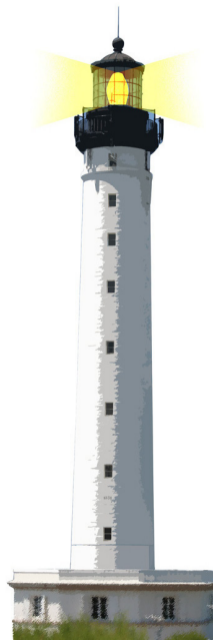
S. Ducasse

http://stephane.ducasse.free.fr

**Pharo**

Extra

# Constructor II

```java
public JButton() {
    this(null, null);
}
```

```java
public JButton(String text) {
    this(text, null);
```

```java
public JButton(String text, Icon icon) {
    setModel(new DefaultButtonModel());
    init(text, icon);
  }
```

- Constructors may invoke each other.
- The keyword this is used for that purpose.

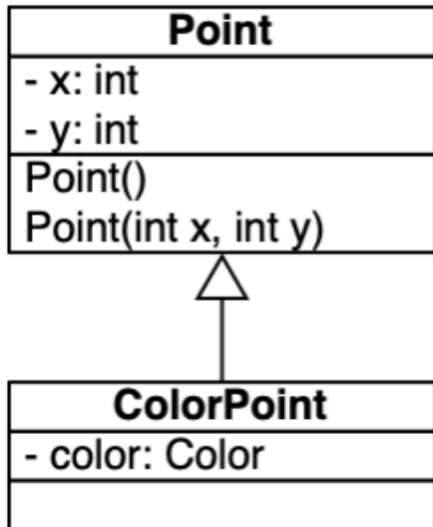Note that this this, used in to invoke constructor, has nothing to do with the this pseudo variable used in method.

# Constructors are not inherited

new Point()
new Point(2, 3)
=> Okay

new ColorPoint()
=> Okay (because of the
default constructor)

new ColorPoint(2, 3)
=> Does not compile

```
Point
- x: int
- y: int
Point()
Point(int x, int y)
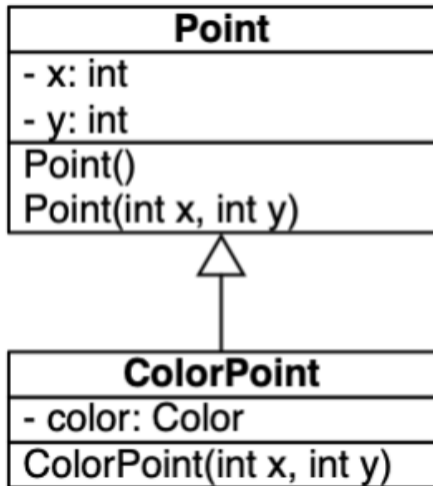```

```
ColorPoint
- color: Color
```

# Missing default constructor

```
new Point()
new Point(2, 3)
=> Okay
```

```
new ColorPoint()
=> Does not compile
(because there is no
default constructor)

new ColorPoint(2, 3)
=> Okay
```

| Point |
| --- |
| - x: int |
| - y: int |
| Point() <br> Point(int x, int y) |

△

| ColorPoint |
| --- |
| - color: Color |
| ColorPoint(int x, int y) |

# Constructors

```
class Rectangle {
    protected int length; protected int width;
    public Rectangle(){
      length = 0;
      width = 0; }
    public Rectangle(int length, int width) {
        this.length = length;
        this.width = width; }
```

```
class Box extends Rectangle {
    protected int height;
    public Box() {
      super();
      height = 0;
    }
    public Box(int length, int width, int height) {
        super(length, width);
        this.height = height;
```

}

# About implicit constructor invocation

```java
class Super {
  String s;
  public Super(){ System.out.println("Super"); }
}

public class Sub extends Super {
  public Sub(){
   super (); // implicitly added if not present
   System.out.println("Sub"); }
  public static void main(String[] args){
   Sub s = new Sub(); }
}

>>> Super
>>> Sub
```

- If a constructor does not explicitly invoke a superclass constructor, the Java compiler automatically inserts a call to the no-argument constructor of the

# Example of missing super constructor

"Implicit super constructor is undefined for default constructor. Must define an explicit constructor"

```java
class Super {
    String s;
    public Super(String s) { this.s = s; }
}

public class Sub extends Super {
    public Sub(){
     System.out.println("Sub"); }

    public static void main(String[] args){
     Sub s = new Sub(); }
}
```

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

Reusing some parts of the Pharo Mooc by

Damien Cassou, Stéphane Ducasse, Luc Fabresse
`http://mooc.pharo.org`