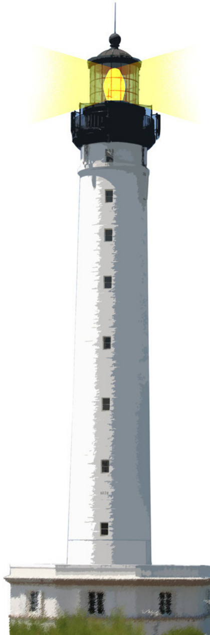
Advanced Object-Oriented Design

Key OO concepts in Java

S. Ducasse



http://www.pharo.org

Objectives

- Illustrate key OO concepts
- In Java but limited as much as possible to the essential points

Thanks Alexandre Bergel for parts of the materials used in this lecture!

Quotes of the day

- "Perfection is attained, not when no more can be added, but when no more can be removed." Antoine de Saint-Exupéry
- "I invented the term ’Object-Oriented’, and I can tell you I did not have C++ in mind." Alan Kay (nor Java :))

Simple Java is an oxymoron

- Java is gigantic and even more... (looks more and more like an old verbose language)
- Full of conceptual glitches (public fields, strange protected semantics, overloading...)

This lecture will not cover: packaging, enums, lambdas, generics, inner classes, modules, private methods, visibility, synchronised, meta data, overloading, primitives vs. boxed....

- But we will provide extras slides on more advanced topics

Java

Not pure object-oriented programming language

- Static methods are not looked up
- Primitive types are not objects: int and Integer....
- Classes are not first class: cannot send messages to classes
- Mixing physical representation (files) with concepts
  - there is not need to have files to have classes
  - class definitions can be saved in databases

Outline

- Instances, instance creation
- Classes / instance variables
- Methods
- Inheritance (single)
- Method lookup
- this / super
- Constructor
- Dynamic type vs. static types
- Interface
- Cast

Instances

- Remember: one state, identity, behavior
- Created using `new` construct

```
new Tomagotchi()
```

Often

```
Tomagotchi t = new Tomagotchi()
```

Class

- Mold/Generators of instances

```
public class Rectangle {
  protected int length;
  protected int width;
  ...
```

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

Class

- Class import packages (group of classes)

O_o

- One public class per file (well)
- File name should have the name of the public class

Instance variables

- Describe instance structure
- Have a visibility: Avoid public, private and final :)
- Better use protected (see companion extra lectures)

```
public class Rectangle {
  protected int length;
  protected int width;
  ...
```

- Accessible by method of the class and subclasses

Methods

- this represents the receiver
- The lookup of methods at runtime starts in the class of the receiver.

```
public class Rectangle{
  protected int length;
  protected int width;

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

Constructor (I)

- A Constructor is a static function, it is not a method!
- Responsible to properly initialize an object
- `<class>()` is a default constructor

```
public Rectangle() {
  length = 0;
  width = 0;
}
```

Multiple constructors in a class

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

What you should know

- Class/Instances
- Methods
- Constructors = functions

A course by

S. Ducasse, G. Polito, and Pablo Tesone