What’s wrong with Object Oriented Programming

a tale about weird habits of software developers

Habit *noun* /ˈhæbɪt/

a thing that you do often and almost without thinking, especially ***something that is hard to stop doing***

Alan Kay

Father of Object Oriented Programming

Concived Laptop

Contributed to the Client-Server Model

Father of User Interface

Father of Smalltalk

Contributed to the definition of Ethernet

Alan Kay

Father of Object Oriented Programming

“ I made up the term object-oriented, and I can tell you I didn't have C++ in mind ”

Alan Kay

Father of Object Oriented Programming

“Java is the most distressing thing to happen to computing since MS-DOS”

Object

O

An object is a data structure that is made up of behavior and data

Inheritance

I

It's a mechanism for code reuse to allow independent extension of the original code. It give rise to hierarchies

Method

M

Methods provide the interface that other classes use to access and modify the data of an object the kingdom of

nouns

take your requirements and circle all nouns, those are your classes. Then underline all the adjectives, those are your properties. Then highlight all your verbs, those are your methods.

Bob, the bald guy

Bob, a bald and nearsighted guy, smokes a cigarette

nouns	◯
verbs	🟨
adjectives	🟩

Bob, the bald guy

Bob, a bald and nearsighted guy, smokes a cigarette

```
class Person {
        public method <set/get> hairStyle(...)
        public method <set/get> bodyLook(...)

        public method smoke() {
                # smoke the cigarette
        }
}
```

nouns	◯
verbs	🟨
adjectives	▬

Bob, the bald guy

Bob, a bald and nearsighted guy, smokes a cigarette

01 retrieve *a cigarette packet*

02 grab *the packet*

03 take *one cigarette*

04 find *a lighter*

05 light up *the cigarette*

06 smoke

Bob, the bald guy

Bob, a bald and nearsighted guy, smokes a cigarette

```
class Person {
        public method <set/get> hairStyle(...)
        public method <set/get> bodyLook(...)

        public method smoke() {
                # retrieve
                # grab
                # take
                # find
                # light-up
                # smoke
        }
}
```

Bob, the bald guy

Bob, a bald and nearsighted guy, smokes a cigarette

```
class Person {
	public method <set/get> hairStyle(...)
	public method <set/get> bodyLook(...)

	public method smoke() {
		# call the method sequence
	}
	private method retrieve() {}
	private method grab() {}
	private method take() {}
	private method find() {}
	private method lightUp() {}
	private method smoke() {}
}
```

“ give the people the opportunity of choice and it will be confused with the opportunity to be mistaken ”

Bob, the bald guy

The pitfall of public and private accessors

```
class Person {
    public variable hairStyle;
    public variable bodyLook;
```

Object Data/Variables should be private by design

```
    ~~public method <set/get> bodyLook(...)~~
```

Object Behaviour/Methods should be public by design

```
    }
    private method retrieve() {}
    private method grab() {}
    private method take() {}
    private method find() {}
    private method lightUp() {}
    private method smoke() {}
}
```

Bob, the bald guy

The pitfall of nouns

Bob, a bald and nearsighted guy, smokes a cigarette

```
class Person {
        method smoke(injected cigaretteWierdClassName) {
                // smoke!
        }
}
```

```
class CigaretteDispatcher{}     class LighterLightningStrategy{}     class CigarettePackageGrabber {}

class LighterFinder {}          class SmokingSessionProvider {}      class CigaretteChooser {}

class CigarettePacketProvider {}     class LighterFinder {}      class CigarettePacketOpener {}

class CigaretteBrandChecker {}     class CigaretteChooser {}       class LighterGasRecharger {}
```

Bob, the bald guy

The pitfall of nouns

Bob a bald and nearsighted guy, smokes a cigarette

```
class Person {
        method smoke(injected cigaretteWierdClassName) {
                // smoke!
        }
}
```

```
class CigaretteDispatcher{}     class LighterLightningStrategy{}     class CigarettePackageGrabber {}

class LighterFinder {}          class SmokingSessionProvider {}      class CigaretteChooser {}

class CigarettePacketProvider {}    class LighterFinder {}    class CigarettePacketOpener {}

class CigaretteBrandChecker {}      class CigaretteChooser {}     class LighterGasRecharger {}
```

ld guy

class Cig

n {}

der {}

class Male {}

class BigCigarette {}

class Cigar {}

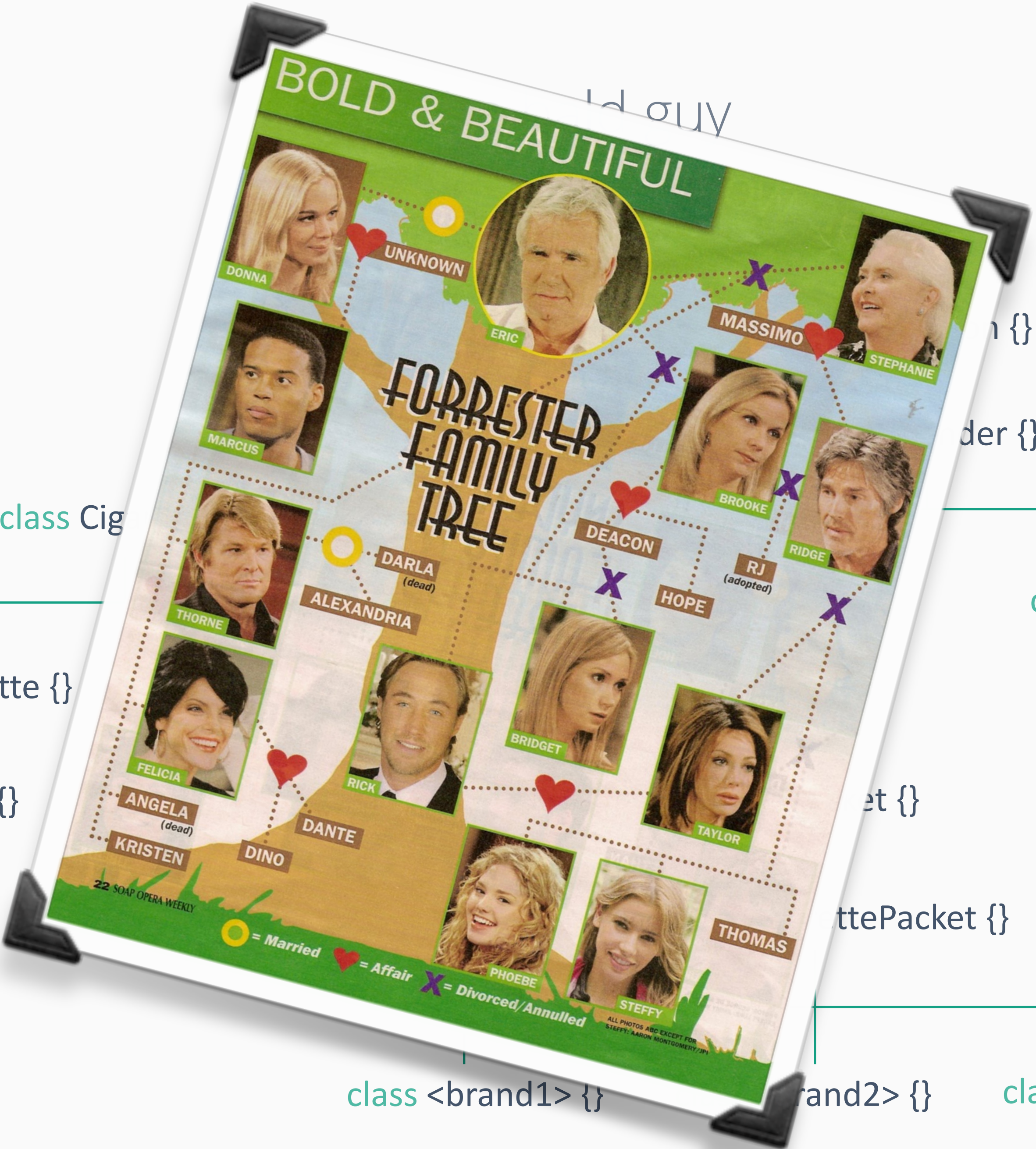
et {}

ttePacket {}

class <brand1> {}

rand2> {}

class <brand3> {}



The Square-Rectangle Problem

The pitfall of polymorphism

```
public class Rectangle {

    public method setWidth (double size) { … }

    public method getWidth () { … }

    public method setHeight (double size) { … }

    public method getHeight () { … }

    public method getPerimeter() { … }

    public method getArea() { … }
}
```

```
public class Square extends Rectangle {

    public method setSide (double size) {
        setWidth(size);
        setHeight(size);
    }
}
```

How can a so weak paradigm be tought by the same person who invented laptops, UI and so forth?

```
Date: Wed, 23 Jul 2003 09:33:31 -0800
To: Stefan Ram [removed for privacy]
From: Alan Kay [removed for privacy]
Subject: Re: Clarification of "object-oriented"
[some header lines removed for privacy]
Content-Type: text/plain; charset="us-ascii" ; format="flowed"
Content-Length: 4965
Lines: 117

At 6:27 PM +0200 7/17/03, Stefan Ram wrote:
>   Dear Dr. Kay,
>
>   I would like to have some authoritative word on the term
>   "object-oriented programming" for my tutorial page on the
>   subject. The only two sources I consider to be "authoritative"
>   are the International Standards Organization, which defines
>   "object-oriented" in "ISO/IEC 2382-15", and you, because,
>   as they say, you have coined that term.
```

[…]

- I thought of objects being like biological cells and/or individual computers on a network, only able to communicate with messages (so messaging came at the very beginning -- it took a while to see how to do messaging in a programming language efficiently enough to be useful).

[…]

- I didn't like the way Simula I or Simula 67 did inheritance (though I thought Nygaard and Dahl were just tremendous thinkers and designers). So I decided to leave out inheritance as a built-in feature until I understood it better.

[…]

The original Smalltalk at Xerox PARC came out of the above. The subsequent Smalltalk's are complained about in the end of the History chapter: they backslid towards Simula and did not replace the extension mechanisms with safer ones that were anywhere near as useful.

```
OOP to me means only messaging, local retention and protection and
hiding of state-process, and extreme late-binding of all things. It
can be done in Smalltalk and in LISP. There are possibly other
systems in which this is possible, but I'm not aware of them.

Cheers,

Alan
```

Thank you

sassoroberto@gmail.com

Roberto Sasso

robertosasso84

roberto.sasso