# IPHONE

*Development Jump Start*

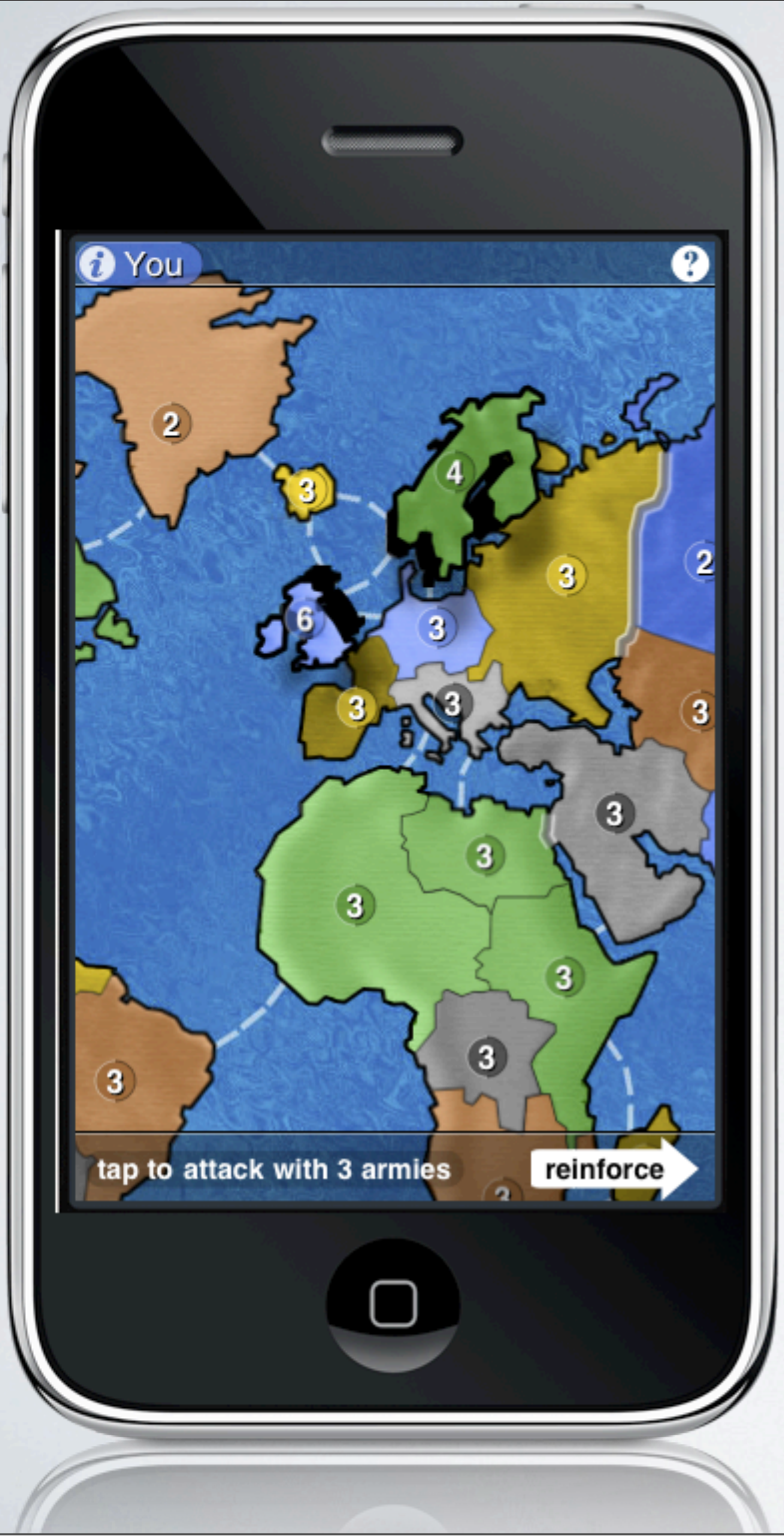
```
(id)initWithFrame:(CGRect)frame {
  if (self = [super initWithFrame:frame]) {

      // Initialization code
      UIImageView* img1 = [[UIImageView alloc]
      UIImageView* img2 = [[UIImageView alloc]

      [self addSubview:img1];
      [self addSubview:img2];

      [img1 release];
      [img2 release];
  }
  return self;

void) touchesBegan:(NSSet *)touches withEvent:

  [UIView beginAnimations:nil context:UIGraph:
  [UIView setAnimationTransition:UIViewAnimat:
```

phil nash

levelofindirection.com

# Who?

been in a professional developer for the last 18 years
- mostly windows
- c++, c#, Java, Python etc
- then, Aug 2008, decided to write and iPhone app -> Obj-C
- three weeks later was finishing my first app...



# vConqr

... in the app store since Sept 2008
- still maintaining - coming up to v2
- no subliminal advertising
- not here to talk about that/
- here to talk about iPhone dev - so what do we need?

## Need?

A Mac

iPhone API stack

XCode

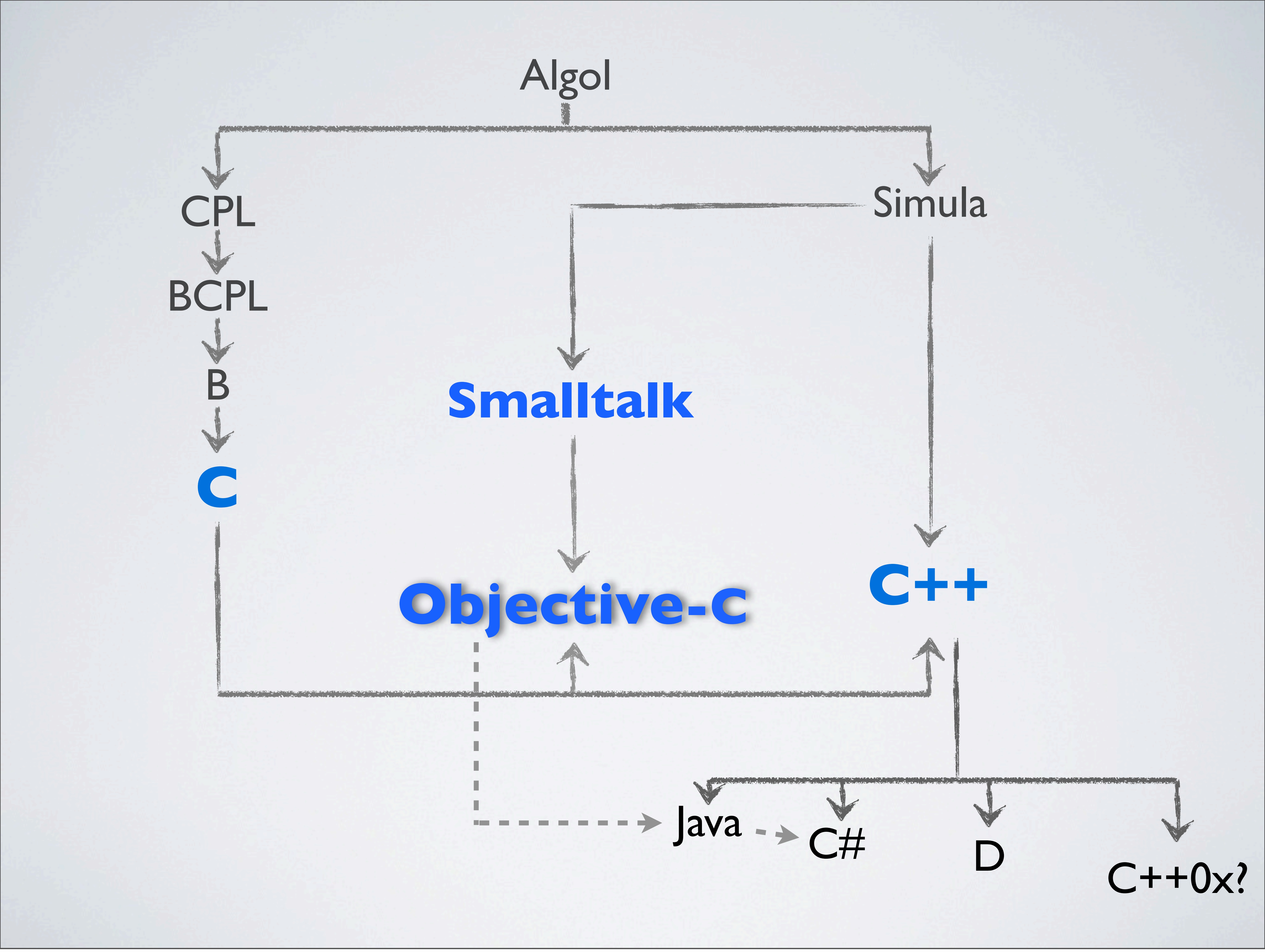
```
[object method1]
```

Objective-C

Mac - great machines. Well built, durable, sleek, well integrated.
XCode
iPhone APIs
Objective-C: Monotouch
- getting very good
- limitations - esp. debugging
- not all bindings
- docs, tutorials, samples
- expensive ($400 - $1000)

Opinion polarised
? - learning hump,
  - modern features early
  - lacking some features we've got used to
 - or is it? ...

| **C#** VS | **Objective-C** |
|---|---|
| Linq | Key-Value Coding/ Path |
| Extension methods | Categories |
| Delegates | Selectors |
| Dynamic keyword | Message-passing |

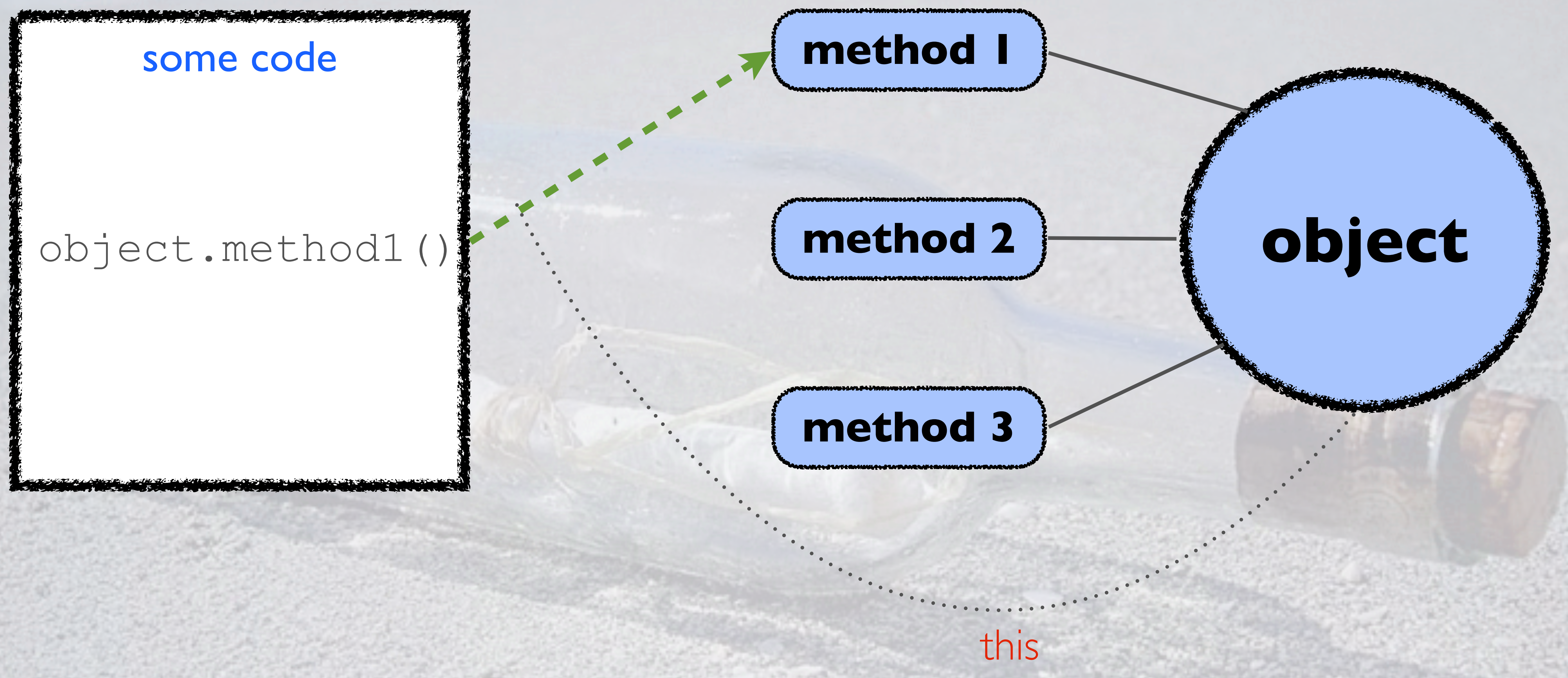

Algol
CPL
BCPL
B
C
Simula
Smalltalk
Objective-C
C++
Java
C#
D
C++0x?

## Message passing

# Objective-C in 20 minutes

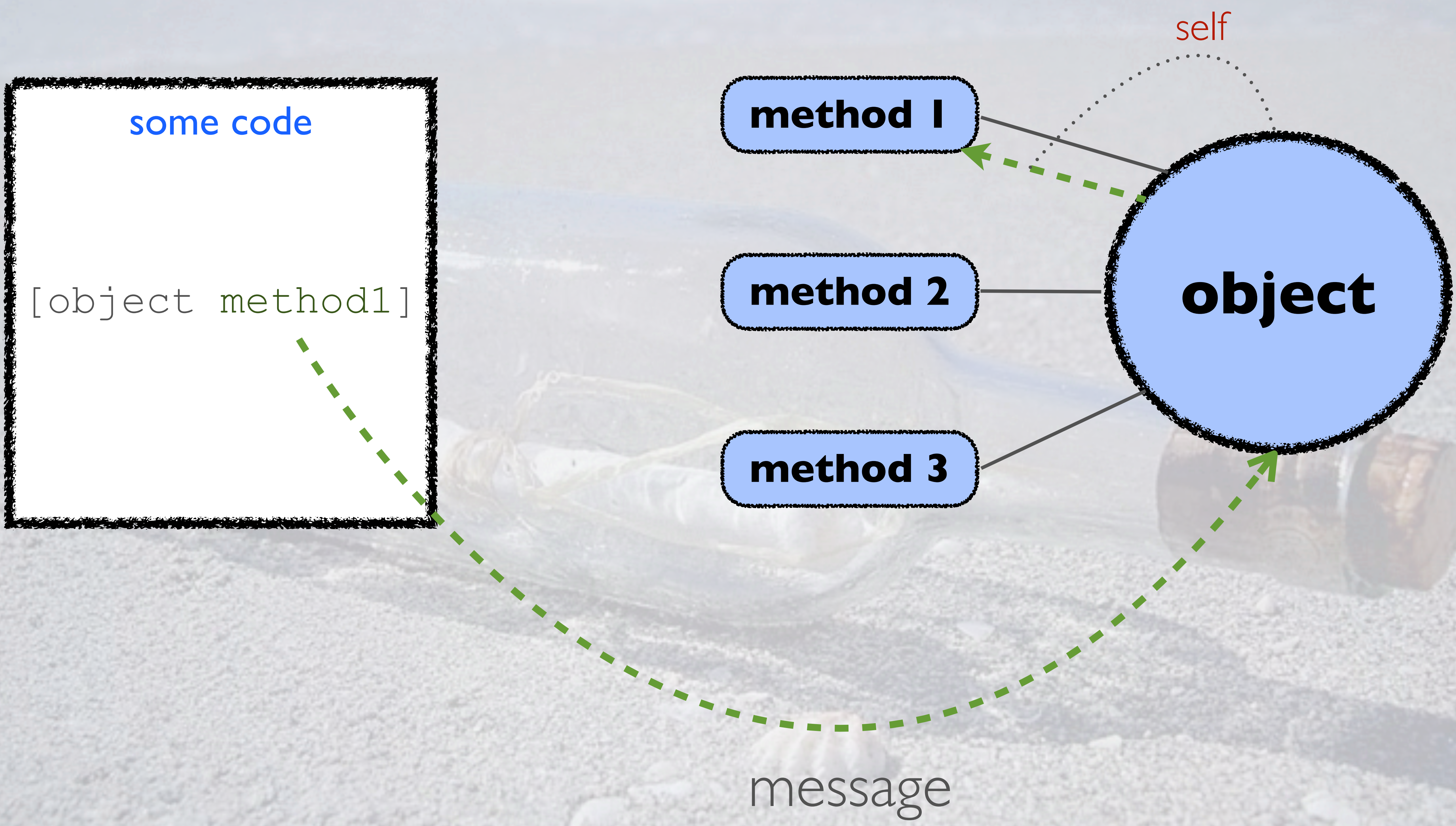
Java

C++ C#...



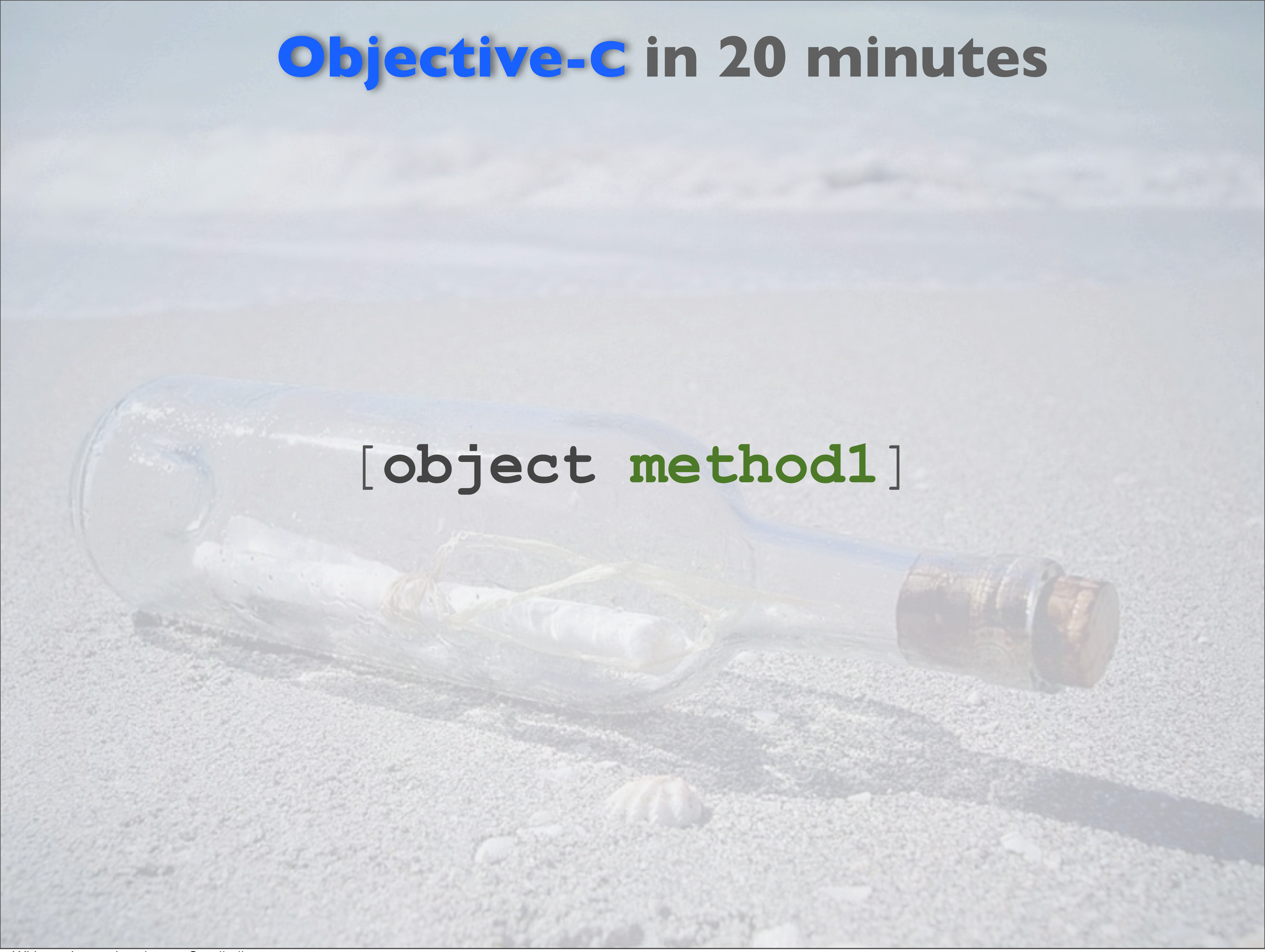
# Objective-C in 20 minutes

## Obj-C



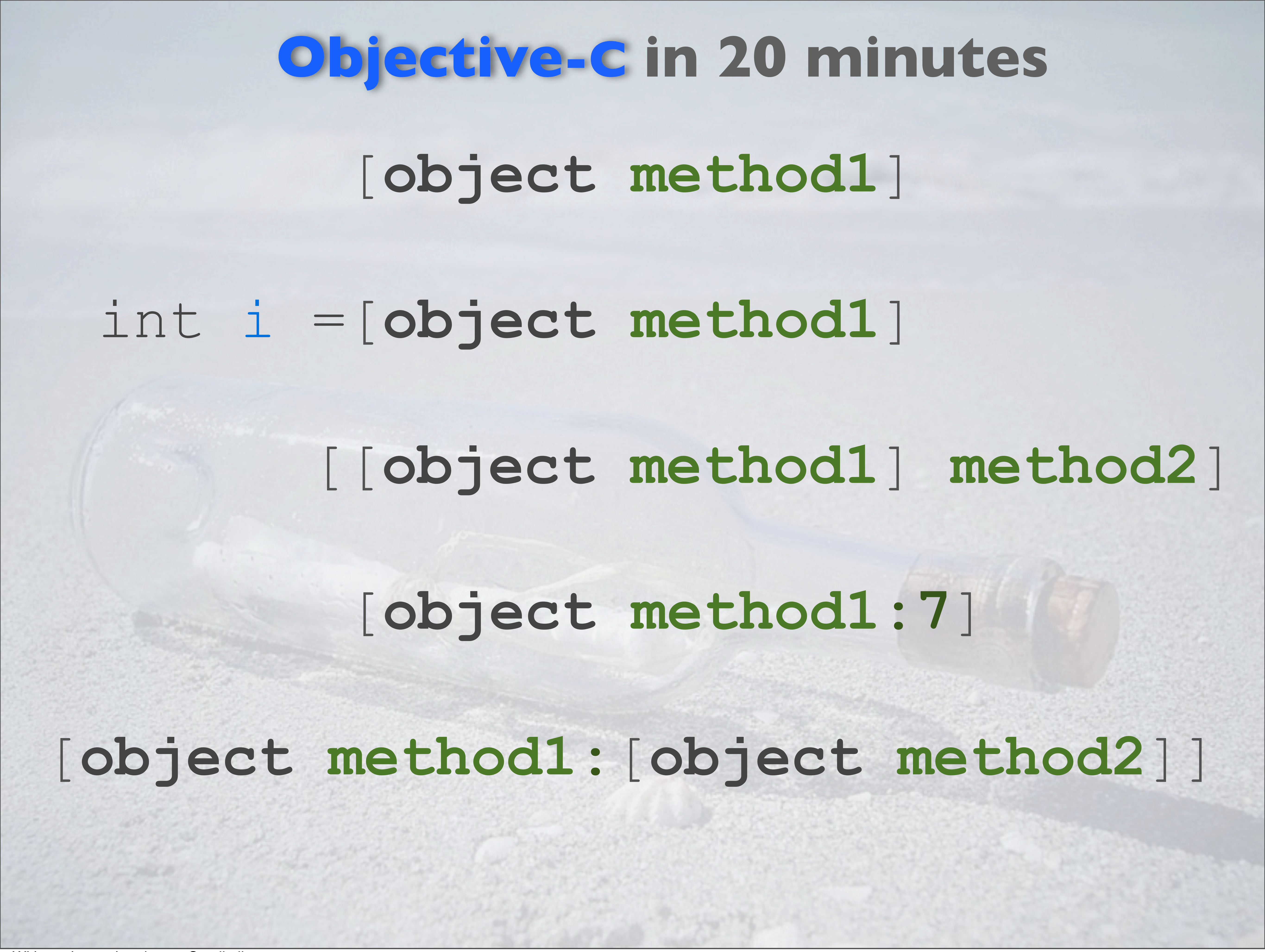
- closer look at the syntax...

# Objective-C in 20 minutes



```
[object method1]
```

Without the sq. brackets = Smalltalk
in Smalltalk: everything an object (even primitives)
 - all operations are messages
Obj-C is fusion of Smalltalk and C
 - type systems must co-exist
------
return a value
pass message to returned object
pass arguments
pass returned object as argument



## Objective-C in 20 minutes

```
          [object method1]

int i =[object method1]

        [[object method1] method2]

          [object method1:7]

[object method1:[object method2]]
```

Without the sq. brackets = Smalltalk
in Smalltalk: everything an object (even primitives)
 - all operations are messages
In Obj-C objects & primitives are different
- C operators are retained
------
return a value
pass message to returned object
pass arguments
pass returned object as argument

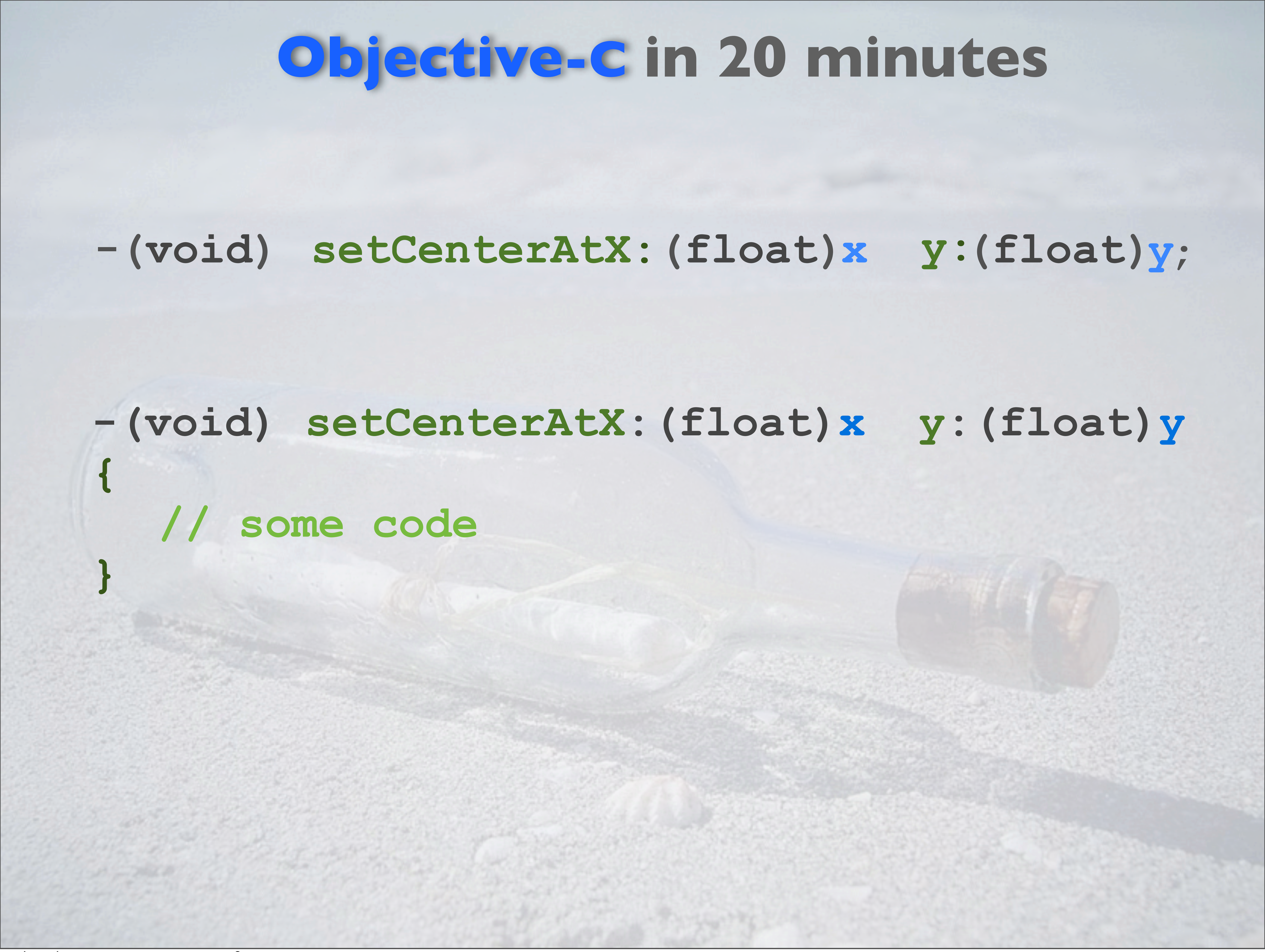
# Objective-C in 20 minutes

```
[object method1:7]

[circle setCenter:100 :100]

[circle setCenterAtX:100 y:100]
```

$$\texttt{circle}\ \underbrace{\texttt{setCenterAtX:y:}}_{\textit{method name}}$$

closer look at arguments
how to pass more than one?
need colon - never see this in real code
label to left of colon
all labels (with colons) = method name
- how do we declare? ...

# Objective-C in 20 minutes

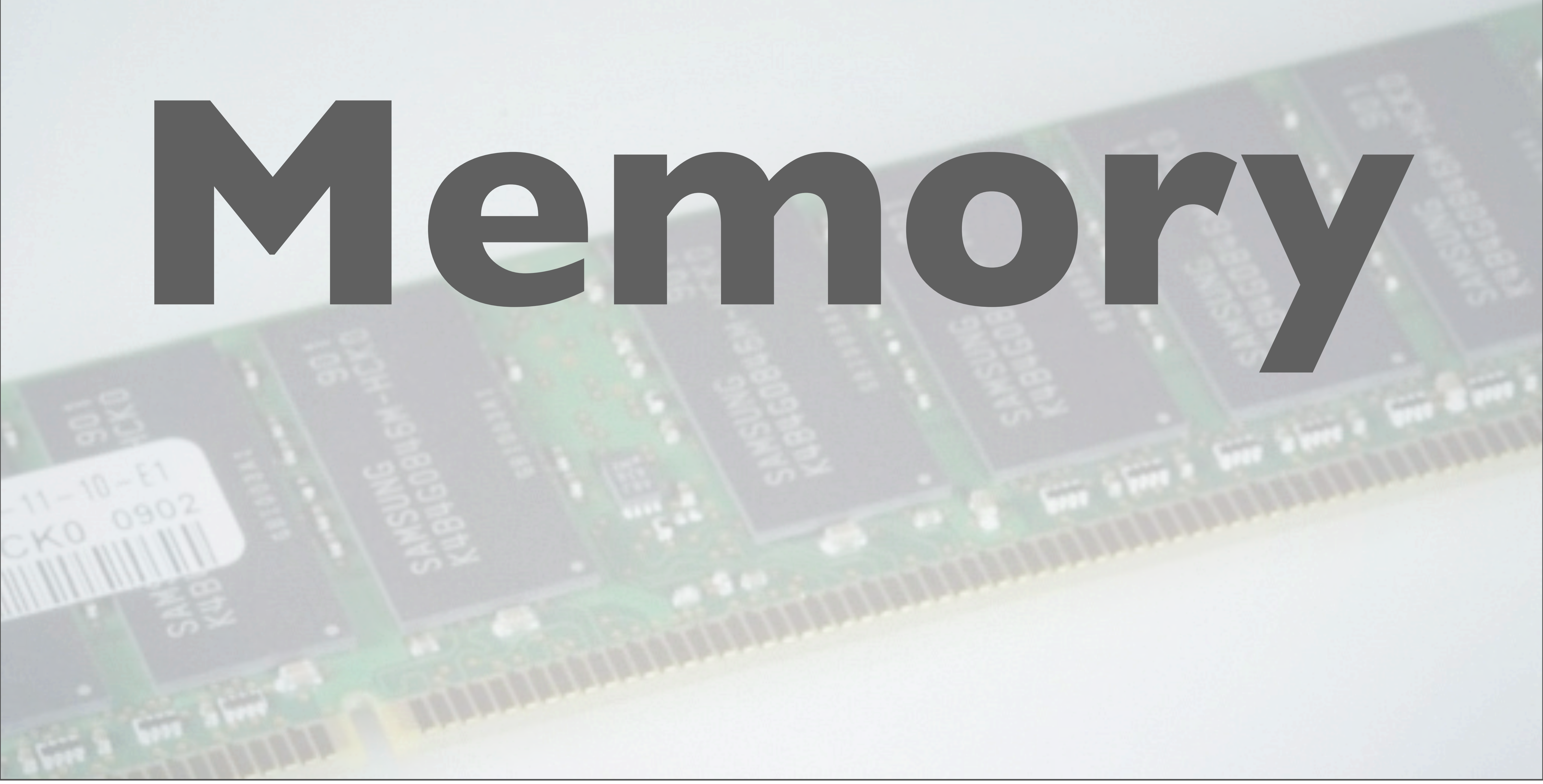


```
-(void) setCenterAtX:(float)x  y:(float)y;
```

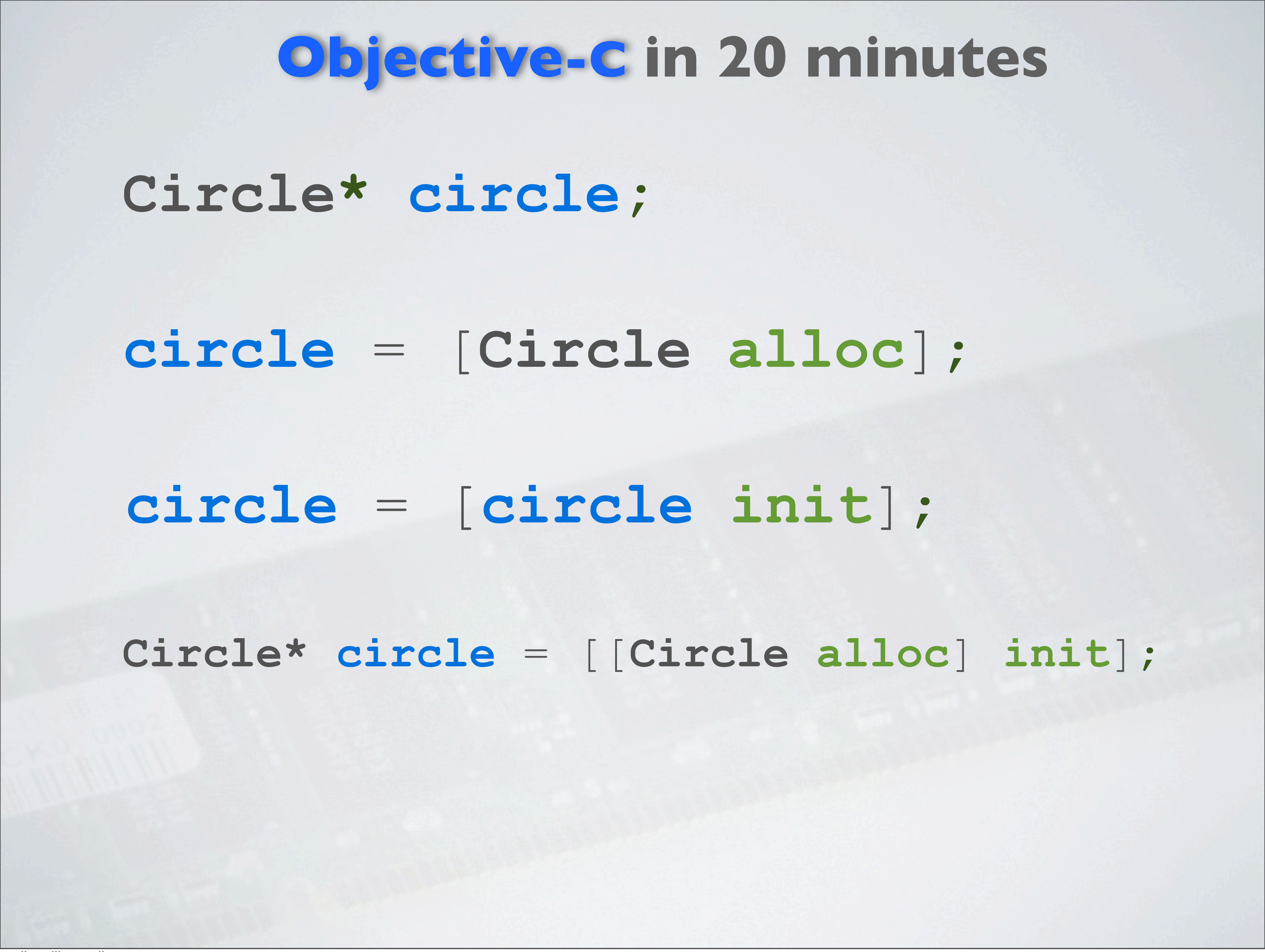
```
-(void) setCenterAtX:(float)x  y:(float)y
{
   // some code
}
```

what about memory management? ...

Objective-C in 20 minutes



# Memory



## Objective-C in 20 minutes

```
Circle* circle;

circle = [Circle alloc];

circle = [circle init];

Circle* circle = [[Circle alloc] init];
```

alloc, like malloc
init, returns object (may differ)
one line
factory method?
... what about parameters?

## Objective-C in 20 minutes

```
    Circle* circle = [[Circle alloc] initAtCenterX: 200 y:200];
```

```
Circle* circle = [[Circle alloc] initAtCenterX:200
                                             y:200
                                    withRadius:100];
```

~~[circle dealloc];~~

```
[circle release];
```

radius
split over lines
dealloc - implement, don’t call
release - ref counting
... retain counts
- go over a little more...

# Objective-C in 20 minutes

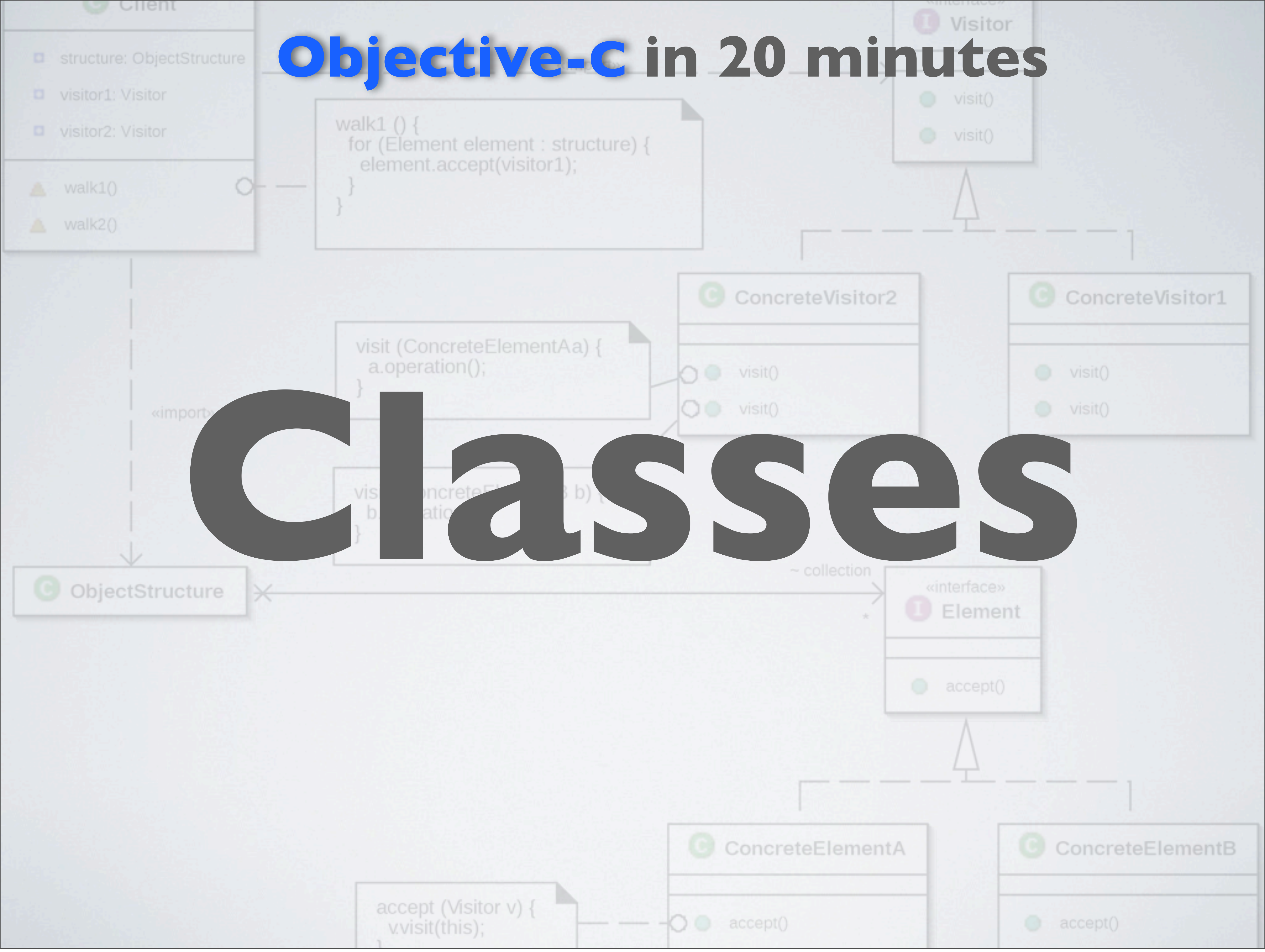
| | retain count |
|---|---|
| `Object* object = [[Object alloc] init];` | 1 |
| `[object retain];` | 2 |
| `[object release];` | 1 |
| `[object release];` | 0 |

*`[object dealloc];`*

- how do we create new classes? ...

## Objective-C in 20 minutes

# Classes



```
walk1 () {
  for (Element element : structure) {
    element.accept(visitor1);
  }
}
```

```
visit (ConcreteElementAa) {
  a.operation();
}
```

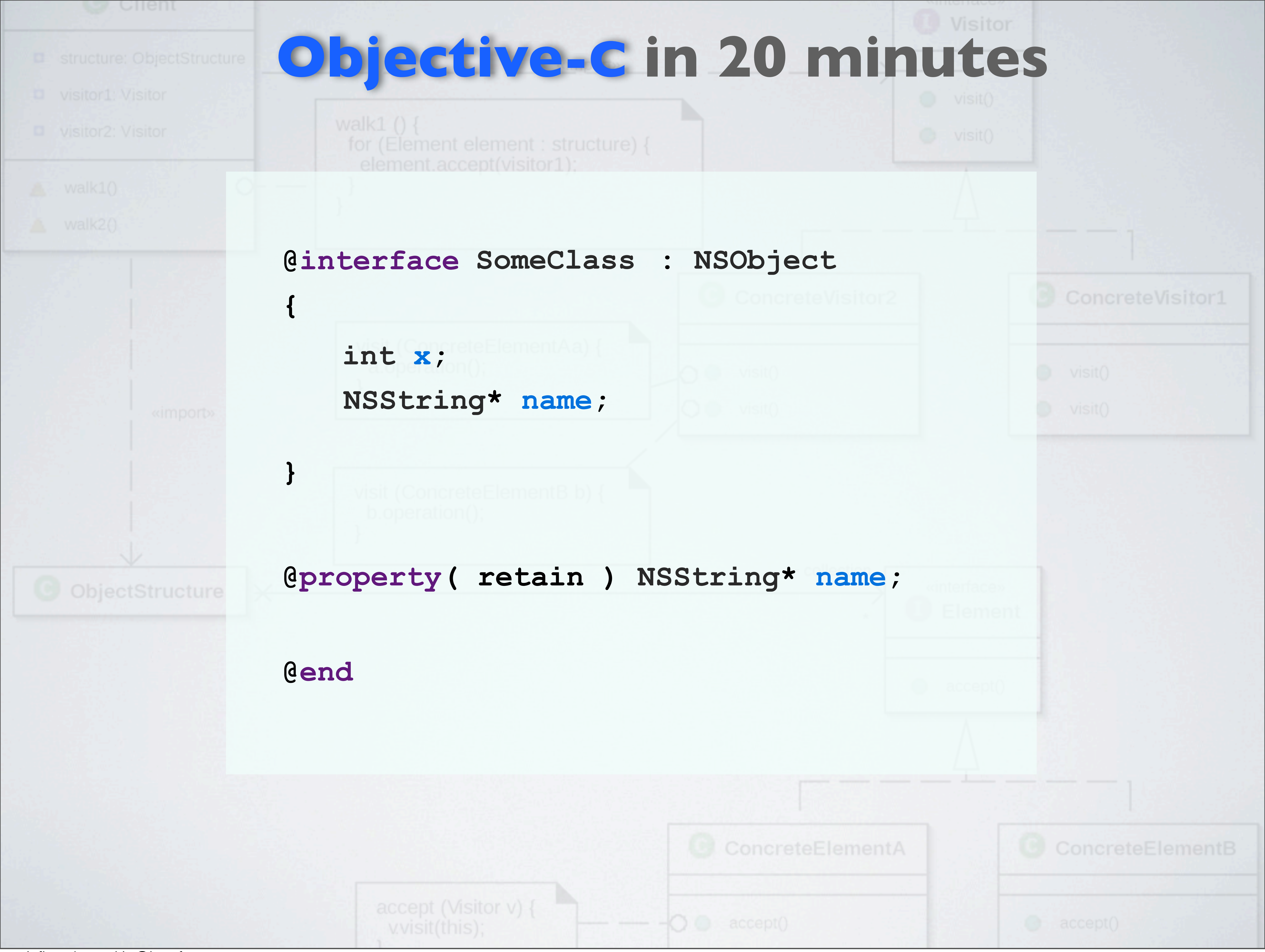
```
accept (Visitor v) {
  v.visit(this);
```

## Objective-C in 20 minutes

```
@interface SomeClass : NSObject
{
    int x;
    NSString* name;

}

-(NSString*) name;
-(void) setName: (NSString*) newName;

@end
```

- define class with @interface
- not the same as interface
- derived from NSObject
- code block
- “blow your mind” - @end

# Objective-C in 20 minutes



```
@interface SomeClass  : NSObject
{
    int x;
    NSString* name;

}

@property( retain ) NSString* name;


@end
```

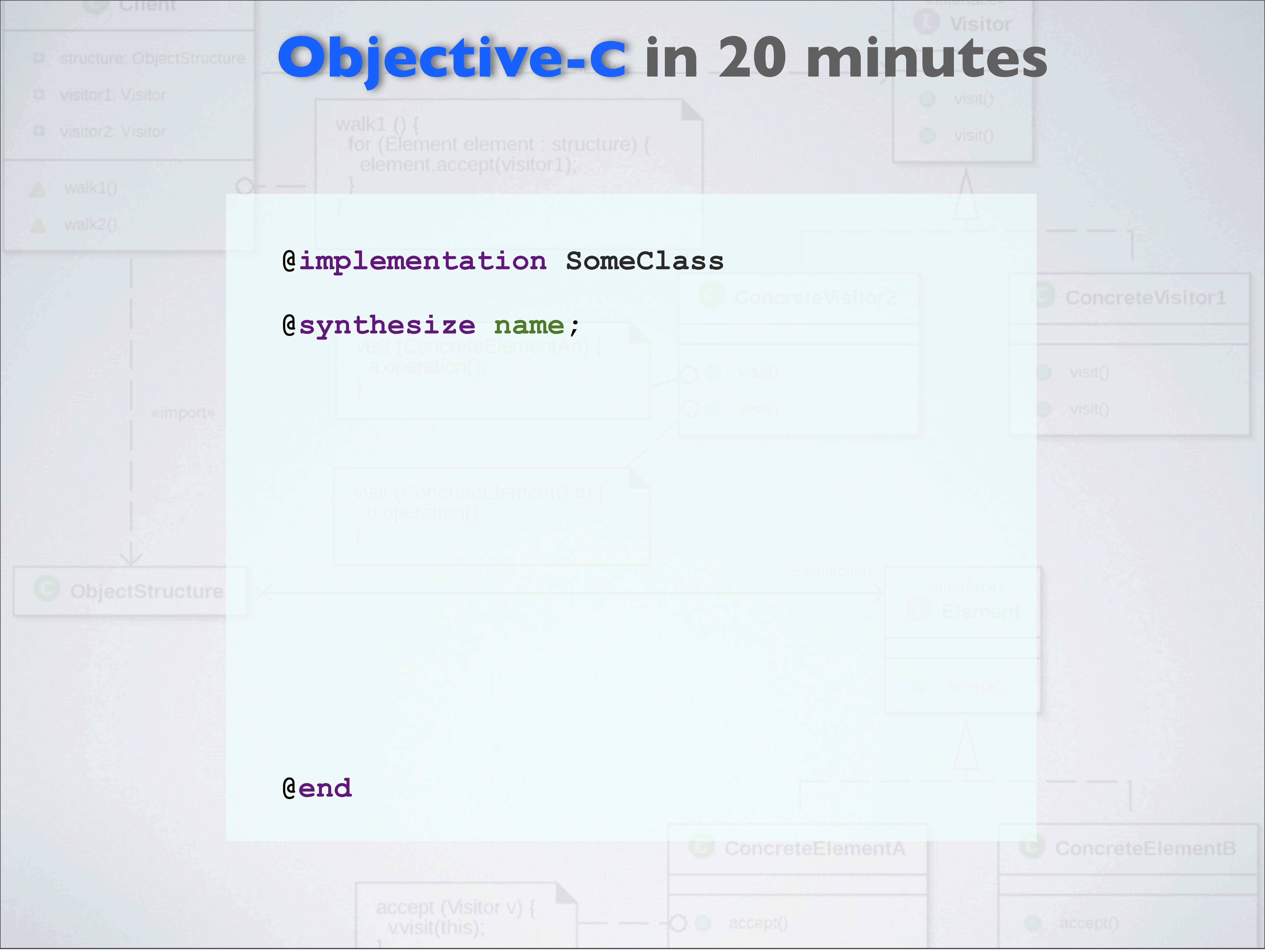
- define class with @interface
- not the same as interface
- derived from NSObject
- code block
- “blow your mind” - @end

# Objective-C in 20 minutes

```
@implementation SomeClass


-(NSString*) name
{
     return name;
}

-(void) setName: (NSString*) newName
{
     [newName retain];
     [name release];
     newName = name;
}

@end
```

# Objective-C in 20 minutes

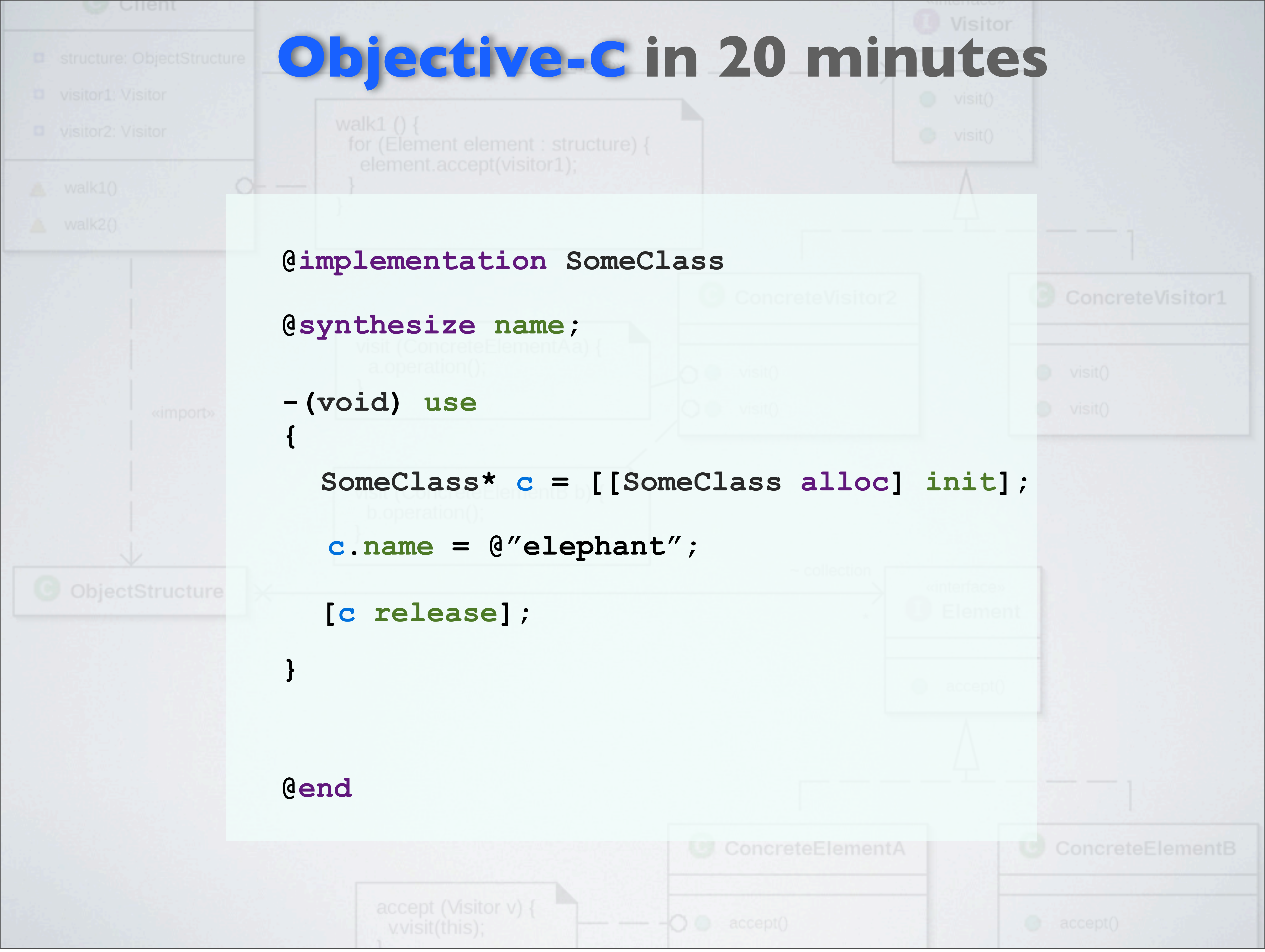
```
@implementation SomeClass

@synthesize name;






@end
```



## Objective-C in 20 minutes

```
@implementation SomeClass

@synthesize name;

-(void) use
{
   SomeClass* c = [[SomeClass alloc] init];

   [c setName: @”elephant”];

   [c release];

}


@end
```

# Objective-C in 20 minutes



```
@implementation SomeClass

@synthesize name;

-(void) use
{
    SomeClass* c = [[SomeClass alloc] init];

    c.name = @”elephant”;

    [c release];

}


@end
```

Demo
A Concentrated Savoury Toast Extract Demo
FOR BEST BEFORE END SEE BACK OF PAGE
250 g ℮

Now onto the meat of the presentation...