

Memory Management in iOS

1DevDay, 2011

Priya Rajagopal

Jeff Kelley

Overview

- Reference Counting System
- You manage memory in iOS4 and below
- iOS5 and above – Automatic Reference Counting (ARC) available
 - Still useful to understand life cycle of an object

Stack vs. Heap

- When you're programming, most of what you do is on the stack:
 - {
 int a = 42;
 a = a * 10;
}
 - Here, a does not exist after the final } because it is *out of scope*
 - a is created on the **stack**

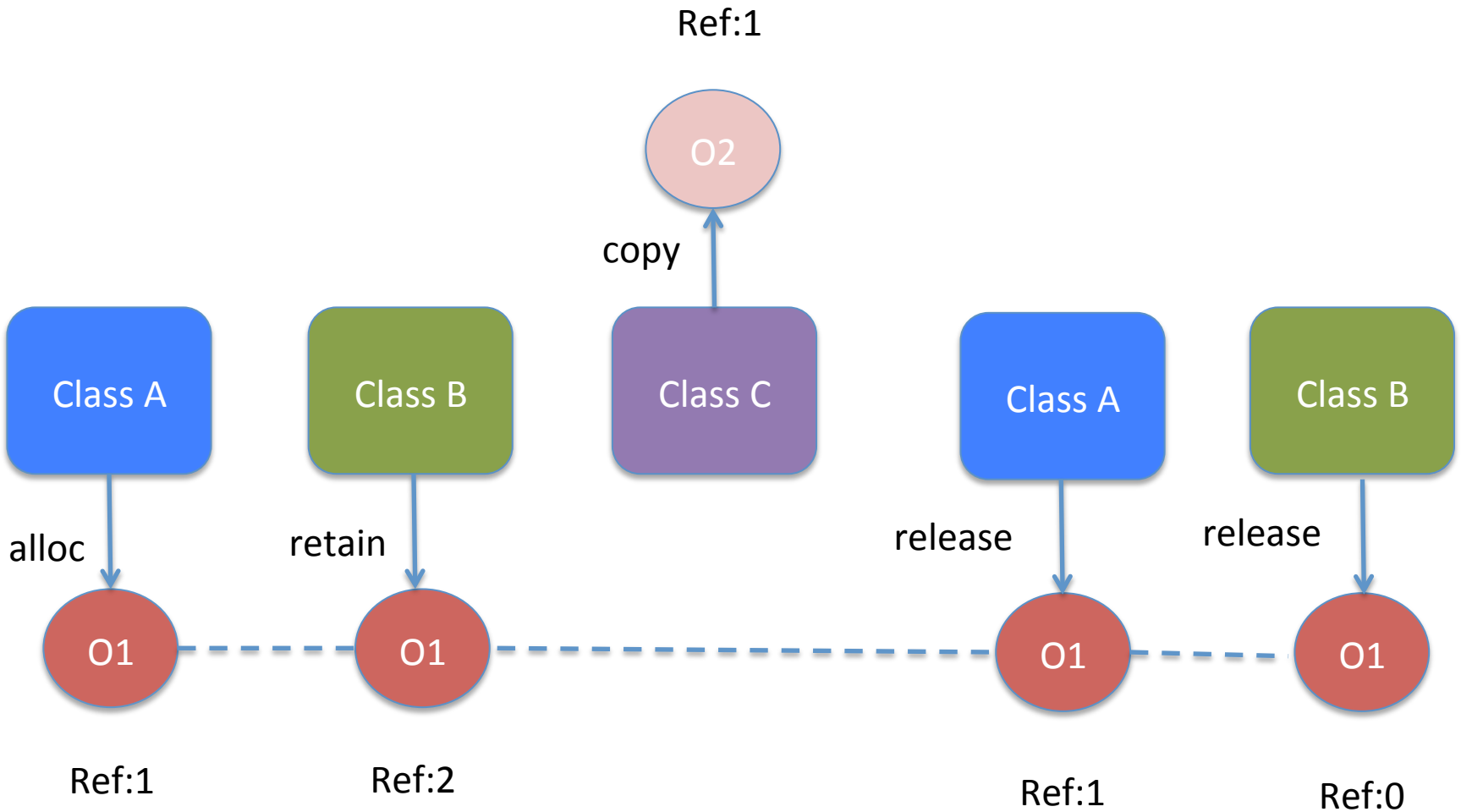
Stack vs. Heap

- Sometimes we want things to live outside of the stack.
- In C, you use `malloc()` to create space on the **heap**, then use `free()` to reclaim the space when you're done with it.
- In Objective-C, objects are (*usually*) created on the heap.
 - Possible to manually create a stack-based object, but this isn't done.
- But when will these objects be freed?

Reference Counting in Objective-C

- When an object is created, it has a reference count of 1. The creator is the owner.
- Every time, a new owner is added, the reference count increases by 1
- Every time, a owner relinquishes control , the reference count decreases by 1
- When reference count is 0, runtime destroys the object by calling `dealloc()` on it.
 - You never call `dealloc()` directly. `release` does it automatically.

A Graphical Explanation...



Taking ownership of an object

- Create an object
 - alloc, new, copy
- Using “retain”
 - Retain the object to ensure it is not deallocated elsewhere
- Strong References
 - Object not freed as long as there is a strong reference to it. Use “retain”

```
@property (retain) id myProperty; // iOS <= 4.3
```

```
@property (strong) id myProperty; // iOS 5+
```

Relinquishing ownership of an Object

- Explicitly using release
 - Object deallocated as soon as reference count is 0
- Put it in an autorelease pool.
 - System takes care of deallocating unreferenced objects periodically from autorelease pool at some later time

Automatic Reference Counting

- Introduced in iOS5
- You don't need to specify retain/release
- Compiler inserts the retain/release code for you
- ARC frees objects as soon as all strong references to it go away
- It's **not** Garbage Collection

Memory Management Tips

- Use Accessors to get/set values of objects
 - Except in dealloc and init
- Release objects (& Reclaim memory) as soon as you are done with them
 - Especially true for scarce resources
- iOS5 and above
 - Leverage ARC