Memory Management

CS 274 iPhone App Development
Robert Muller

Today

• Memory Management in Objective C
• With side-orders of:
  – Pointers
  – Stack vs. Heap Storage
  – Dynamic Method Dispatch.
Values with **indefinite extent**. I.e., values whose lifetimes aren’t bounded by the invocation of a procedure or method. After such values are created, they may live until program exit.
In many languages, (e.g., LISP, CAML, Java, Python, ...) memory in the heap is allocated and reclaimed automatically using garbage collection.

The iPhone engineers at Apple decided that garbage collection consumed too much battery power.

So on the iPhone, the programmer is required to manage heap allocation and deallocation using reference counting.
Reference Counting in Obj-C

- When you allocate or copy storage for a heap value, its reference count is set to 1;
- When you retain a heap value, its reference count is incremented.
- When you release a heap value, its reference count is decremented and checked for 0. If it is 0, the storage is returned to the free-space pool.
Reference Counting Rules of the Road

• If you allocate or copy a heap value, you need to **release** it when you’re done with it;

• If you need a heap value to persist, you may it **retain** it and thereby become a co-owner of it;

• Do not release a heap value that you didn’t allocate, copy or retain.

Bad Things may Happen to You

• If you release a heap value that you didn’t allocate, copy or retain, the heap storage space may be reclaimed and reallocated prematurely. Your app will have **dangling pointers** and you are going to be miserable.

• If you fail to release heap values when you are done with them, your app will have **memory leaks**, it will probably run out of memory and your app will be terminated.

Odds and Ends

• Who is “you”?

• The **autorelease** pool.