MentorMate™

Mobile & Web, Applied.

Васил Нончев

Димитър Николов

Android Team (Plovdiv)

MentorMate™

Memory Leaks in Android

MentorMate™

Agenda

- Why?
- What do we call a leak?
- Memory information in Android Logs
- GC and GC roots
- How to find memory leak(demo)?
- (Bonus) Efficient Bitmap loading

Why should we care

- OutOfMemory crashes
- Speed up both the app and the system
- Faster GC

Your options:

- Know your objects, fix your problems
- Or…

MentorMate™

Leaks in Android

- What do we call a leak in Android
- Overview of the GC process
  - What it does - frees up the no longer needed objects
  - Where does it start - the GC Root objects
  - References types:
    - Strong
    - Soft
    - Weak
    - Phantom
    - (Finalizer)

Bisection of a GC log message

```
GC_CONCURRENT freed 456K, 19% free 2753K/3360K, paused 5ms+9ms, total 378ms
```

`GC_CONCURRENT` type of GC
`freed 456K` amount of space freed from GC
`19%` free current heap's available space
`2753K` the amount of heap currently used
`3360K` the "soft" limit on the heap, that would trigger a GC
`paused 5ms+9ms` Concurrent GC pause times at the start and at the end
`total 378ms` total time of the GC cycle
Types of GC:
`GC_CONCURRENT GC_EXPLICIT GC_EXTERNAL_MALLOC GC_FOR_MALLOC GC_HPROF_DUMP_HEAP`

How GC works

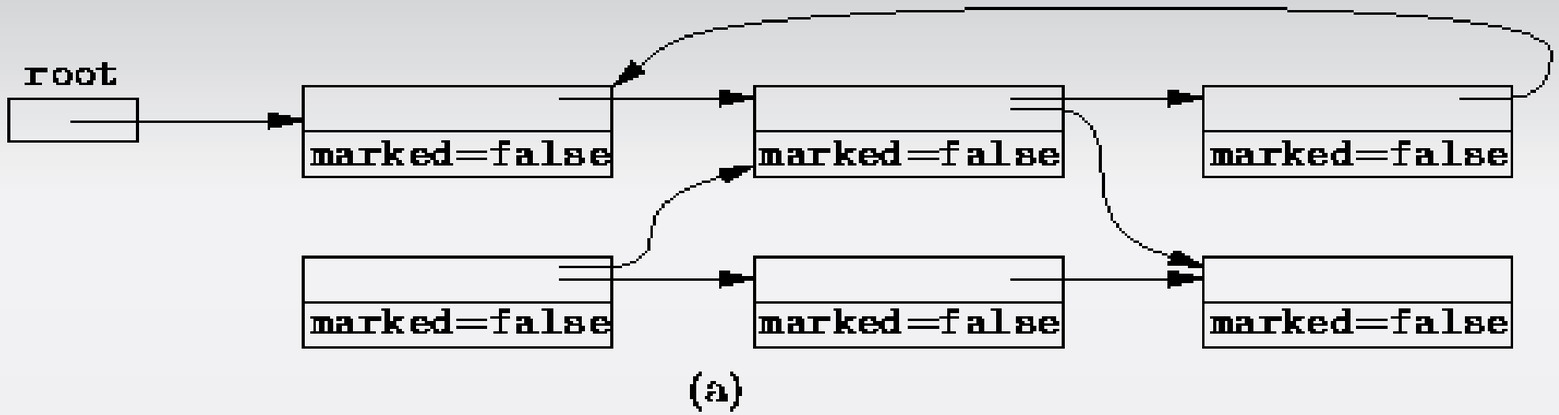
GC Roots Objects that serve as a starting point for the garbage collection. These are not garbage collectable.

- **Class** - class loaded by system class loader. Such classes can never be unloaded. They can hold objects via static fields.
- **Thread** - live thread
- **Stack Local** - local variable or parameter of Java method
- **JNI Local** - local variable or parameter of JNI method
- **JNI Global** - global JNI reference
- **Monitor Used** - objects used as a monitor for synchronization
- **Held by JVM** - objects held from garbage collection by JVM for its purposes.

Unfortunately, JVM provides absolutely no additional detail for such objects.

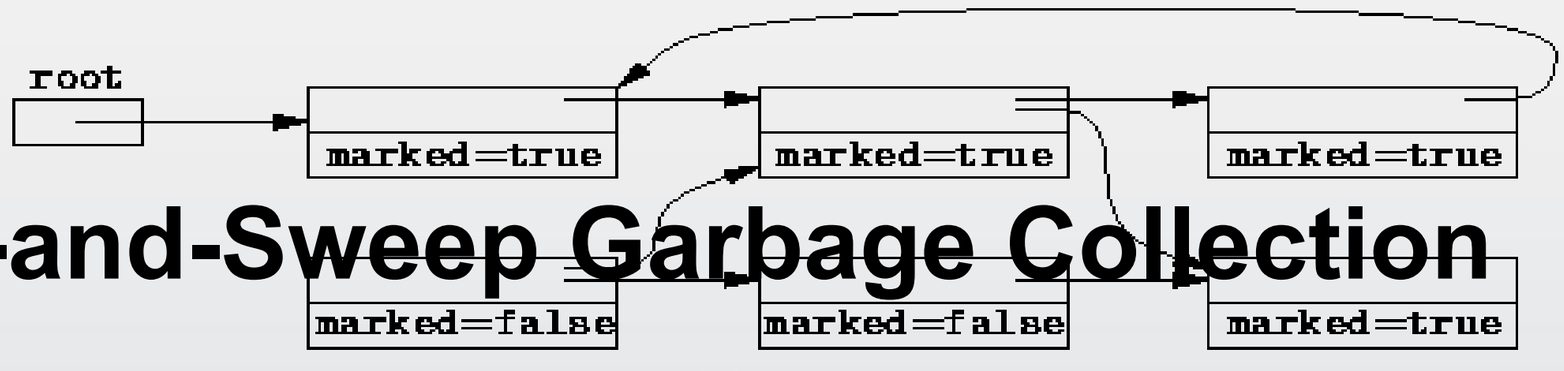
Mark-and-Sweep Garbage Collection

Initial state



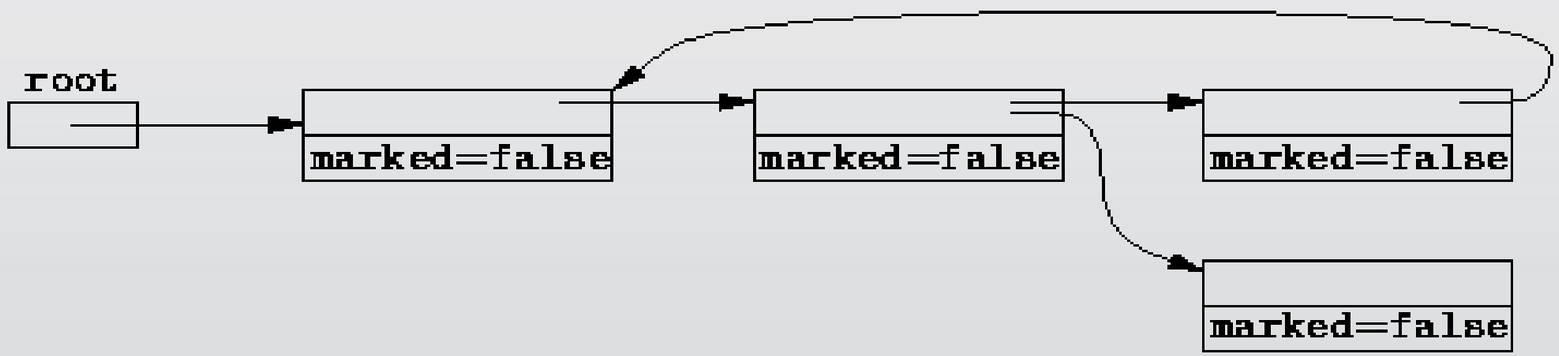
(a)

Mark



(b)

Sweep



(c)

MentorMate™

Almost demo time ...

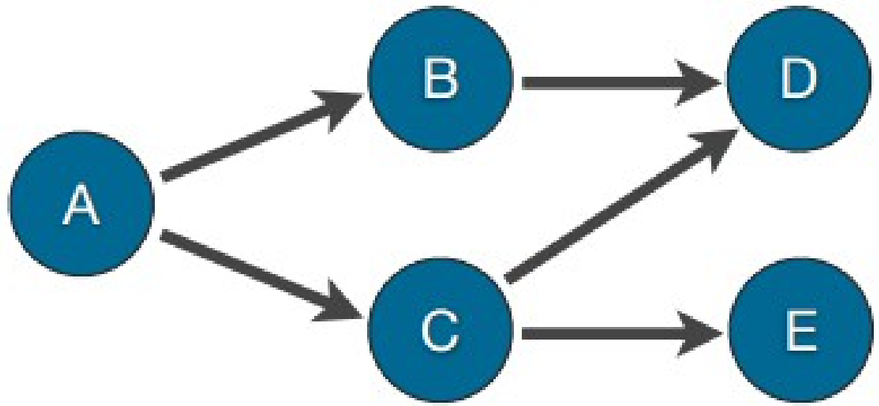
Dominator tree:

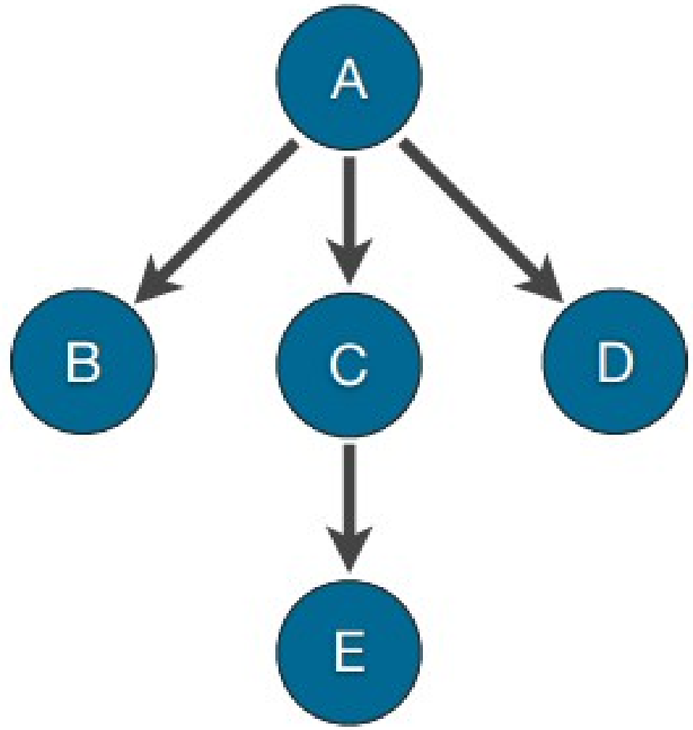
A node **d** dominates a node **n** if every path from the entry node to **n** must go through **d**.

Shallow heap is the memory consumed by one object. An object needs 32 or 64 bits (depending on the OS architecture) per reference, 4 bytes per Integer, 8 bytes per Long, etc.

Retained set of X is the set of objects which would be removed by GC when X is garbage collected.

Retained heap of X is the sum of shallow sizes of all objects in the retained set of X, i.e. memory kept alive by X.





MentorMate™

MentorMate™

Plan for finding the leak

1. Obtain a heap dump
   a. Make sure you browse the application as much as possible
   b. Return to the place where you started
2. Create histogram view
3. Find objects that should not be in there
4. Find the cause of them not being GCd

(Bonus points) Use OQL - SELECT * FROM "java.lang.Thread"

Popular memory leak causes:

1. Callbacks
2. Singletons
3. Static fields
4. Timers
5. Long running threads

(Bonus) Efficient bitmap loading

- When loading images in memory use the most optimal size for you.
- The image loaded should not be bigger than the canvas that you need to show it in.
- Tutorial on how to do it - in the references
- Image resource resolution
  - Android assumes images provided as in density independent pixels unless the image is in the nodpi folder
  - Android find the best matching image and scales it in memory if needed
  - Use ImageView scaling options instead

What to do at home (as a conclusion)

- Using DDMS find out how much memory your app uses
- Click through the app and go back to the main screen.
- Make a heap dump. See what objects live in memory. Should they still be in there?

Thank you.

MentorMate™

Q&A?

References

https://www.youtube.com/watch?v=_CruQY55HOk
http://www.yourkit.com/docs/java/help/gc_roots.jsp
http://developer.android.com/training/displaying-bitmaps/load-bitmap.html
http://stackoverflow.com/questions/12709603/mat-eclipse-memory-analyzer-how-to-view-bitmaps-from-memory-dump
https://developer.android.com/training/articles/memory.html