Memories of Android

(based on a talk and articles by Dianne Hackborn)

Romain Guy	google.com/+RomainGuy	@romainguy
Chet Haase	google.com/+ChetHaase	@chethaase

Why?

- Android is different
- Mobile is different
- Many small decisions can create large problems
- We all want more. Always.

Mobile Dynamics

“The hardware will be faster next year”

Mobile Dynamics

“The hardware will be faster next year”

vs.

“This year’s hardware will be ***cheaper*** *next year”*

Mobile Dynamics

Mobile Dynamics

We want an experience better than desktop

Mobile Dynamics

- We want an experience better than desktop
  - On much slower hardware

Mobile Dynamics

We want an experience better than desktop

On much slower hardware

With higher resolution displays

Mobile Dynamics

We want an experience better than desktop

On much slower hardware

With higher resolution displays

On battery

Mobile Dynamics

We want an experience better than desktop

On much slower hardware

With higher resolution displays

On battery

For as long as possible

Android 2.3

- Still ships with many new [low-end] devices
  - Because of RAM

Agenda

- Android and RAM
- Guidelines
- Measurement

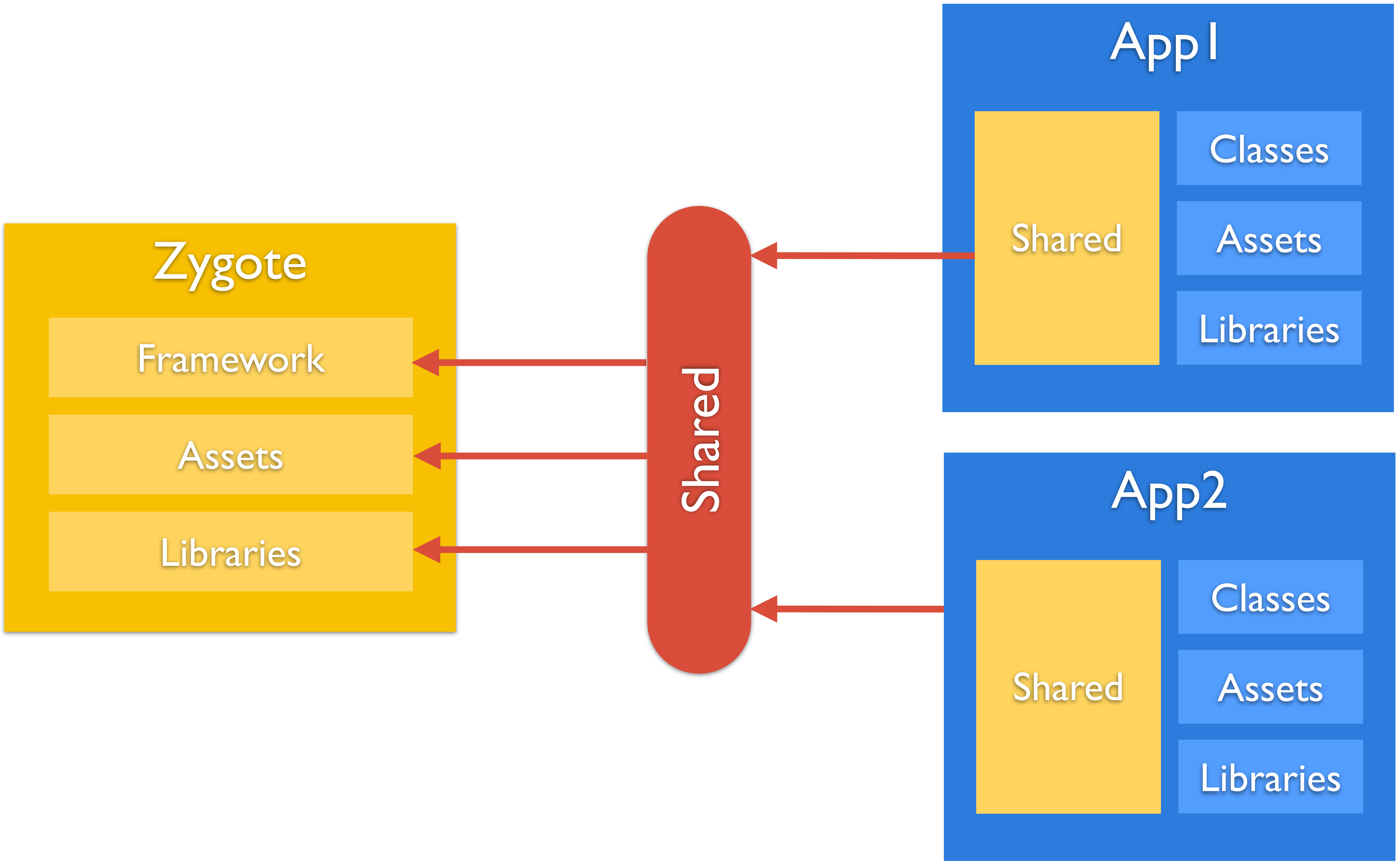
Android and RAM

No Swap

- Clean RAM
  - Paging of mmap’d files as needed
- Dirty RAM
  - Can't swap out
  - Relatively expensive
    - Especially in background processes

Zygote

- Process from which your app is spawned
- Preloads framework classes
- Preloads common assets
- Preloads native libraries



Zygote
Framework
Assets
Libraries
Shared
App1
Shared
Classes
Assets
Libraries
App2
Shared
Classes
Assets
Libraries

Types of memory

	Dirty	Clean
Private	Bad	Okay
Shared	Good	Best

Overcommit

- Reserve address space for an allocation
- Only mapped when needed
- Allocations generally don’t fail
- What happens when no RAM is available?

Processes

- Instead of swap, Android uses processes*
  - Running vs. cached
  - Cached processes can be killed
  - Order of killing is LRU
    - with other modifications
- Cached processes help Android user experience

*See “Processes and Threads” in API Guides documentation

zRAM

- New in Android 4.4
- Enabled on low-memory devices
- Type of swap
  - Swap to compressed RAM

Shared Memory

- Extensively used
    - Requires care in determining RAM use
- Helps minimize memory footprint

Shared Memory

- mmap
  - dalvik code, apk resources, ...
  - Share across processes, allows paging
- zygote
  - First Dalvik process, from which all others fork
  - Preloads commonly used classes, resources
  - Common RAM shared across forked processes
- ashmem
  - Large allocations shared instead of copied
  - Cursors, some graphics resources

Kernel SamePage Merging

- New in Android 4.4
- Share identical pages between processes
  - Copy-on-write
- Really useful with bitmaps
  - Bitmaps allocated on the Dalvik heap
  - byte[] allocations are zeroed out by the VM

How big is an Object?

overhead of Object + overhead of dlmalloc + data

How big is an Object?

overhead of Object + overhead of dlmalloc + data

8 bytes

How big is an Object?

overhead of Object + overhead of dlmalloc + data

8 bytes 4-8 bytes

How big is an Object?

overhead of Object + overhead of dlmalloc + data

overhead of Object	overhead of dlmalloc	data
8 bytes	4-8 bytes	n bytes

How big is an Object?

overhead of Object + overhead of dlmalloc + data

8 bytes 4-8 bytes n bytes

The result must be 8-byte aligned

Size of data

Type	Size as field/variable	Size in array
Object reference	4	4
boolean	4	1
byte	4	1
char	4	2
short	4	2
int	4	4
float	4	4
long	8	8
double	8	8

All sizes in bytes

Object size examples

Object size examples

```
class Empty {
}
```

Object size examples

```
class Empty {
}
```

Allocation	Size in bytes
dlmalloc	4
Object overhead	8

Object size examples

```
class Empty {
}
```

Allocation	Size in bytes
dlmalloc	4
Object overhead	8

Total = 4 + 8 = 12 bytes

Object size examples

```
class Empty {
}
```

Allocation	Size in bytes
dlmalloc	4
Object overhead	8

Total = 4 + 8 = 12 bytes

8-byte aligned total = **16 bytes**

Object size examples

```
class Integer {
    private int value;
}
```

Object size examples

```
class Integer {
    private int value;
}
```

Allocation	Size in bytes
dlmalloc	4
Object overhead	8

Object size examples

```
class Integer {
    private int value;
}
```

Allocation	Size in bytes
dlmalloc	4
Object overhead	8
int	4

Object size examples

```
class Integer {
    private int value;
}
```

Allocation	Size in bytes
dlmalloc	4
Object overhead	8
int	4

Total = 4 + 8 + 4 = **16 bytes**

Object size examples

```
class HashMap$HashMapEntry<K, V> {
    final K key;
    V value;
    final int hash;
    HashMapEntry<K, V> next;
}
```

Object size examples

```
class HashMap$HashMapEntry<K, V> {
    final K key;
    V value;
    final int hash;
    HashMapEntry<K, V> next;
}
```

Allocation	Size in bytes
dlmalloc	4
Object overhead	8

Object size examples

```
class HashMap$HashMapEntry<K, V> {
    final K key;
    V value;
    final int hash;
    HashMapEntry<K, V> next;
}
```

Allocation	Size in bytes
dlmalloc	4
Object overhead	8
Reference	4
Reference	4
int	4
Reference	4

Object size examples

```
class HashMap$HashMapEntry<K, V> {
    final K key;
    V value;
    final int hash;
    HashMapEntry<K, V> next;
}
```

Total = 4 + 8 + 4 * 4 = 28 bytes

Aligned total = 32 bytes

Allocation	Size in bytes
dlmalloc	4
Object overhead	8
Reference	4
Reference	4
int	4
Reference	4

How big is an array?

overhead of Object + overhead of dlmalloc + width + padding + data

How big is an array?

overhead of Object + overhead of dlmalloc + width + padding + data

8 bytes

How big is an array?

overhead of Object + overhead of dlmalloc + width + padding + data

8 bytes 4-8 bytes

How big is an array?

overhead of Object + overhead of dlmalloc + width + padding + data

8 bytes 4-8 bytes 4 bytes

How big is an array?

overhead of Object + overhead of dlmalloc + width + padding + data

overhead of Object	overhead of dlmalloc	width	padding	data
8 bytes	4-8 bytes	4 bytes	4 bytes	

How big is an array?

overhead of Object + overhead of dlmalloc + width + padding + data

8 bytes 4-8 bytes 4 bytes 4 bytes n bytes

How big is an array?

overhead of Object + overhead of dlmalloc + width + padding + data

8 bytes 4-8 bytes 4 bytes 4 bytes n bytes

The result must be 8-byte aligned

Array size examples

Array size examples

```
new byte[1]
```

Object size examples

```
new byte[1]
```

Allocation	Size in bytes
dlmalloc	4
Object overhead	8
width & padding	8

Object size examples

```
new byte[1]
```

Allocation	Size in bytes
dlmalloc	4
Object overhead	8
width & padding	8
data	1

Object size examples

```
new byte[1]
```

Total = 4 + 8 + 8 + 1 = 21 bytes

8-byte aligned total = **24 bytes**

Allocation	Size in bytes
dlmalloc	4
Object overhead	8
width & padding	8
data	1

DEVOXX™

Objects vs primitive types

Objects vs primitive types

Integer	16 bytes

Objects vs primitive types

Integer	16 bytes	vs.	int	4 bytes

Objects vs primitive types

Integer	16 bytes	vs.	int	4 bytes
Boolean	16 bytes			

Objects vs primitive types

Integer	16 bytes	vs.	int	4 bytes
Boolean	16 bytes	vs.	boolean	4 bytes

Objects vs primitive types

Integer	16 bytes	vs.	int	4 bytes
Boolean	16 bytes	vs.	boolean	4 bytes
		vs.	bit-field	1 bit

Primitive types vs primitive types

```
private boolean mProperty1;
private boolean mProperty1;
// …
private boolean mProperty32;
```

Total = 32 * 4= 128 bytes

Primitive types vs primitive types

```
private boolean mProperties = new boolean[32];
```

Total = 4 + 8 + 8 + 32 * 1= 52 bytes

Aligned total = 56 bytes

Primitive types vs primitive types

```
// This is what we use in android.view.View
private int mProperties;
```

Total = 4 bytes

Classes

- Inner class: ~500 bytes of code overhead

```
button.setOnClickListener(new Runnable() {
    public void run() {
        // do stuff
    }
});
```

Enums

Enums

dex file size

```
public static enum Things {
    THING_1,
    THING_2,
};
```

+1,112 bytes

Enums

dex file size

```
public static enum Things {
    THING_1,
    THING_2,
};
```

+1,112 bytes

vs.

```
public static int THING_1 = 1;
public static int THING_2 = 2;
```

+128 bytes

DEVOXX™

Enums vs. ints

Enums vs. ints

```
public static enum Things {
    THING_1,
    THING_2,
};
```

Enums vs. ints

```
.class public final enum LThings;
.super Ljava/lang/Enum;
.source "Things.java"

.annotation system Ldalvik/annotation/Signature;
    value = {
        "Ljava/lang/Enum",
        "<",
        "LThings;",
        ">;"
    }
.end annotation

.field private static final synthetic $VALUES:[LThings;
.field public static final enum THING_1:LThings;
.field public static final enum THING_2:LThings;

.method static constructor <clinit>()V
    .registers 4
    const/4 v3, 0x1
    const/4 v2, 0x0
    new-instance v0, LThings;
    const-string v1, "THING_1"
    invoke-direct {v0, v1, v2}, LThings;-><init>(Ljava/lang/String;I)V
    sput-object v0, LThings;->THING_1:LThings;
    new-instance v0, LThings;
    const-string v1, "THING_2"
    invoke-direct {v0, v1, v3}, LThings;-><init>(Ljava/lang/String;I)V
    sput-object v0, LThings;->THING_2:LThings;
    const/4 v0, 0x2
    new-array v0, v0, [LThings;
    sget-object v1, LThings;->THING_1:LThings;
    aput-object v1, v0, v2
    sget-object v1, LThings;->THING_2:LThings;
    aput-object v1, v0, v3
    sput-object v0, LThings;->$VALUES:[LThings;
    return-void
.end method

.method private constructor <init>(Ljava/lang/String;I)V
    .registers 3
    .annotation system Ldalvik/annotation/Signature;
        value = {
            "()V"
        }
    .end annotation
    invoke-direct {p0, p1, p2}, Ljava/lang/Enum;-><init>(Ljava/lang/String;I)V
    return-void
.end method

.method public static valueOf(Ljava/lang/String;)LThings;
    .registers 2
    .param p0    # Ljava/lang/String;
    const-class v0, LThings;
    invoke-static {v0, p0}, Ljava/lang/Enum;->valueOf(Ljava/lang/Class;Ljava/lang/String;)Ljava/lang/Enum;
    move-result-object v0
    check-cast v0, LThings;
    return-object v0
.end method

.method public static values()[LThings;
    .registers 1
    sget-object v0, LThings;->$VALUES:[LThings;
    invoke-virtual {v0}, [LThings;->clone()Ljava/lang/Object;
    move-result-object v0
    check-cast v0, [LThings;
    return-object v0
.end method
```

Enums vs. ints

```
.class public final enum LThings;
.super Ljava/lang/Enum;
.source "Things.java"

.annotation system Ldalvik/annotation/Signature;
    value = {
        "Ljava/lang/Enum",
        "<",
        "LThings;",
        ">;"
    }
.end annotation

.field private static final synthetic $VALUES:[LThings;
.field public static final enum THING_1:LThings;
.field public static final enum THING_2:LThings;

.method static constructor <clinit>()V
    .registers 4
    const/4 v3, 0x1
    const/4 v2, 0x0
    new-instance v0, LThings;
    const-string v1, "THING_1"
    invoke-direct {v0, v1, v2}, LThings;-><init>(Ljava/lang/String;I)V
    sput-object v0, LThings;->THING_1:LThings;
    new-instance v0, LThings;
    const-string v1, "THING_2"
    invoke-direct {v0, v1, v3}, LThings;-><init>(Ljava/lang/String;I)V
    sput-object v0, LThings;->THING_2:LThings;
    const/4 v0, 0x2
    new-array v0, v0, [LThings;
    sget-object v1, LThings;->THING_1:LThings;
    aput-object v1, v0, v2
    sget-object v1, LThings;->THING_2:LThings;
    aput-object v1, v0, v3
    sput-object v0, LThings;->$VALUES:[LThings;
    return-void
.end method

.method private constructor <init>(Ljava/lang/String;I)V
    .registers 3
    .annotation system Ldalvik/annotation/Signature;
        value = {
            "()V"
        }
    .end annotation
    invoke-direct {p0, p1, p2}, Ljava/lang/Enum;-><init>(Ljava/lang/String;I)V
    return-void
.end method

.method public static valueOf(Ljava/lang/String;)LThings;
    .registers 2
    .param p0    # Ljava/lang/String;
    const-class v0, LThings;
    invoke-static {v0, p0}, Ljava/lang/Enum;->valueOf(Ljava/lang/Class;Ljava/lang/String;)Ljava/lang/Enum;
    move-result-object v0
    check-cast v0, LThings;
    return-object v0
.end method

.method public static values()[LThings;
    .registers 1
    sget-object v0, LThings;->$VALUES:[LThings;
    invoke-virtual {v0}, [LThings;->clone()Ljava/lang/Object;
    move-result-object v0
    check-cast v0, [LThings;
    return-object v0
.end method
```

```
public static int THING_1 = 1;
public static int THING_2 = 2;
```

Enums vs. ints

```
.class public final enum LThings;
.super Ljava/lang/Enum;
.source "Things.java"

.annotation system Ldalvik/annotation/Signature;
    value = {
        "Ljava/lang/Enum",
        "<",
        "LThings;",
        ">;"
    }
.end annotation

.field private static final synthetic $VALUES:[LThings;
.field public static final enum THING_1:LThings;
.field public static final enum THING_2:LThings;

.method static constructor <clinit>()V
    .registers 4
    const/4 v3, 0x1
    const/4 v2, 0x0
    new-instance v0, LThings;
    const-string v1, "THING_1"
    invoke-direct {v0, v1, v2}, LThings;-><init>(Ljava/lang/String;I)V
    sput-object v0, LThings;->THING_1:LThings;
    new-instance v0, LThings;
    const-string v1, "THING_2"
    invoke-direct {v0, v1, v3}, LThings;-><init>(Ljava/lang/String;I)V
    sput-object v0, LThings;->THING_2:LThings;
    const/4 v0, 0x2
    new-array v0, v0, [LThings;
    sget-object v1, LThings;->THING_1:LThings;
    aput-object v1, v0, v2
    sget-object v1, LThings;->THING_2:LThings;
    aput-object v1, v0, v3
    sput-object v0, LThings;->$VALUES:[LThings;
    return-void
.end method

.method private constructor <init>(Ljava/lang/String;I)V
    .registers 3
    .annotation system Ldalvik/annotation/Signature;
        value = {
            "()V"
        }
    .end annotation
    invoke-direct {p0, p1, p2}, Ljava/lang/Enum;-><init>(Ljava/lang/String;I)V
    return-void
.end method

.method public static valueOf(Ljava/lang/String;)LThings;
    .registers 2
    .param p0    # Ljava/lang/String;
    const-class v0, LThings;
    invoke-static {v0, p0}, Ljava/lang/Enum;->valueOf(Ljava/lang/Class;Ljava/lang/String;)Ljava/lang/Enum;
    move-result-object v0
    check-cast v0, LThings;
    return-object v0
.end method

.method public static values()[LThings;
    .registers 1
    sget-object v0, LThings;->$VALUES:[LThings;
    invoke-virtual {v0}, [LThings;->clone()Ljava/lang/Object;
    move-result-object v0
    check-cast v0, [LThings;
    return-object v0
.end method
```

```
const/4 v0, 0x0
sput v0, LThings;->THING_1:I
sput v0, LThings;->THING_2:I
```

Enums

- Allocate more memory
  - Each value is an instance of the enum class
- Execute more code
  - Class initializer runs when enum is loaded
  - Instantiates each value

Garbage isn’t Free

- Temporary objects can also hurt

Garbage isn’t Free

- Temporary objects can also hurt

```
Integer width = view.getWidth();
```

Garbage isn't Free

- Temporary objects can also hurt

```
Integer width = view.getWidth();
```

Autoboxing

Garbage isn't Free

- Temporary objects can also hurt

```
Integer width = view.getWidth();
```

Autoboxing

```
for (MyListener listener : mListeners) {
    // ...
}
```

Garbage isn't Free

- Temporary objects can also hurt

```
Integer width = view.getWidth();
```

Autoboxing

```
for (MyListener listener : mListeners) {
    // ...
}
```

Iterator created

DEVOXX™

Guidelines

Beware Services

- Very expensive
- Need to stay running
- Directly reduce available cached processes
- Remember, no swap
- Services should have well-defined durations
- Services left running is a common application problem

Release your RAM

```
public void onTrimMemory(int level) {
    // cached activity
    if (level >= TRIM_MEMORY_COMPLETE) {
        // ...
    } else if (level >= TRIM_MEMORY_MODERATE) {
        // ...
    } else if (level >= TRIM_MEMORY_BACKGROUND) {
        // ...
    } else if (level >= TRIM_MEMORY_UI_HIDDEN) {
        // ...
    } else if (level >= TRIM_MEMORY_RUNNING_CRITICAL) {
        // ...
    } else if (level >= TRIM_MEMORY_RUNNING_LOW) {
        // ...
    } else if (level >= TRIM_MEMORY_RUNNING_MODERATE) {
        // ...
    }
}
```

Cached

Running

Memory Class

```
ActivityManager.getMemoryClass();
```

```
ActivityManager.getLargeMemoryClass();
```

Bitmaps

- Often largest RAM user
- RAM size = width * height * depth
  - Optimize for size
- Take care with caches of bitmaps
- Android 3.0: bitmaps in Dalvik heap
  - Reuse when possible
- See: http://developer.android.com/training/displaying-bitmaps/manage-memory.html

ProGuard and Zipalign

- Part of standard build tools
- Use them

Design Guidelines

- App design affects RAM usage
- Harder to fix later
- Common programming practices can be less memory efficient

Know your (Java) programming language

- Java has many challenges for memory use
- Have a general sense of the overhead of language features
- Easier to write efficiently the first time

Abstractions

- Hidden costs

External Libraries

- Not necessarily written for Android
- Potentially large expensive for small benefit

Android Libraries

- Still significant overhead, duplication

Use Optimized Containers

- Sparse arrays
  - Replace hash maps when the key is a primitive type
  - Variants for different key/value types
- Benefits
  - Allocation-free
  - No boxing

Sparse arrays

HashMap	Array class
<Integer, Object>	SparseArray
<Integer, Boolean>	SparseBooleanArray
<Integer, Integer>	SparseIntArray
<Integer, Long>	SparseLongArray
<Long, Object>	LongSparseArray
<Long, Long>	LongSparseLongArray*

* Not a public class, copy from Android's source code

Sparse arrays vs HashMap

- SparseIntArray vs HashMap<Integer, Integer> for 1,000 elements

Sparse arrays vs HashMap

- SparseIntArray vs HashMap<Integer, Integer> for 1,000 elements

```
class SparseIntArray {
    int[] keys;
    int[] values;
    int size;
}
```

Sparse arrays vs HashMap

- SparseIntArray vs HashMap<Integer, Integer> for 1,000 elements

```
class SparseIntArray {
    int[] keys;
    int[] values;
    int size;
}
```

Class = 12 + 3 * 4 = 24 bytes

Array = 20 + 1000 * 4 = 4024 bytes

Total = **8,072 bytes**

DEVOXX™

Sparse arrays vs HashMap

- SparseIntArray vs HashMap<Integer, Integer> for 1,000 elements

Sparse arrays vs HashMap

- SparseIntArray vs HashMap<Integer, Integer> for 1,000 elements

```
class HashMap<K, V> {
    Entry<K, V>[] table;
    Entry<K, V> forNull;
    int size;
    int modCount;
    int threshold;
    Set<K> keys;
    Set<Entry<K, V>> entries;
    Collection<V> values;
}
```

Sparse arrays vs HashMap

- SparseIntArray vs HashMap<Integer, Integer> for 1,000 elements

```
class HashMap<K, V> {
    Entry<K, V>[] table;
    Entry<K, V> forNull;
    int size;
    int modCount;
    int threshold;
    Set<K> keys;
    Set<Entry<K, V>> entries;
    Collection<V> values;
}
```

Class = 12 + 8 * 4 = 48 bytes

Entry = 32 + 16 + 16 = 64 bytes

Array = 20 + 1000 * 64 = 64024 bytes

Total = **64,136 bytes**

Use Optimized Containers

- ArrayMap
  - Replaces HashMap
- Benefit
  - Allocation-free
  - Same API as HashMap (implements Map interface)
  - Available in support library
- Drawback
  - Slower than HashMap, don’t use for large collections

Use Optimized Containers

- Raw arrays
  - When it makes sense
- android.view.ViewGroup
  - Children stored in a View[]

Measurement

Process Memory

- USS (Unique Set Size)
  - Private Clean + Private Dirty
  - RAM committed to only that process
- PSS (Proportional Set Size)
  - USS + memory shared with other processes

Meminfo

```
Applications Memory Usage (kB):
Uptime: 27233364 Realtime: 252885787

** MEMINFO in pid 15976 [com.android.systemui] **
                   Pss  Private  Private  Swapped     Heap     Heap     Heap
                 Total    Dirty    Clean    Dirty     Size    Alloc     Free
                ------   ------   ------   ------   ------   ------   ------
  Native Heap     5308     5280        0        0    16172     7658      741
  Dalvik Heap     7015     6684        0        0    19288    13124     6164
 Dalvik Other     3328     3184        0        0
        Stack      188      188        0        0
       Ashmem        2        0        0        0
    Other dev     4648     4356        4        0
     .so mmap     1296      404       20        0
    .apk mmap     1014        0      564        0
    .ttf mmap      299        0      200        0
    .dex mmap     1904       60     1132        0
   Other mmap      100        4       44        0
     Graphics     7904     7904        0        0
           GL    15916    15916        0        0
      Unknown      120      120        0        0
        TOTAL    49042    44100     1964        0    35460    20782     6905
```

Meminfo

```
Applications Memory Usage (kB):
Uptime: 27233364 Realtime: 252885787

** MEMINFO in pid 15976 [com.android.systemui] **
                 Pss  Private  Private  Swapped     Heap     Heap     Heap
               Total    Dirty    Clean    Dirty     Size    Alloc     Free
              ------   ------   ------   ------   ------   ------   ------
  Native Heap   5308     5280        0        0    16172     7658      741
  Dalvik Heap   7015     6684        0        0    19288    13124     6164
 Dalvik Other   3328     3184        0        0
        Stack    188      188        0        0
       Ashmem      2        0        0        0
    Other dev   4648     4356        4        0
     .so mmap   1296      404       20        0
    .apk mmap   1014        0      564        0
    .ttf mmap    299        0      200        0
    .dex mmap   1904       60     1132        0
   Other mmap    100        4       44        0
     Graphics   7904     7904        0        0
           GL  15916    15916        0        0
      Unknown    120      120        0        0
        TOTAL  49042    44100     1964        0    35460    20782     6905
```

Meminfo

```
Applications Memory Usage (kB):
Uptime: 27233364 Realtime: 252885787

** MEMINFO in pid 15976 [com.android.systemui] **
                 Pss  Private  Private  Swapped     Heap     Heap     Heap
               Total    Dirty    Clean    Dirty     Size    Alloc     Free
              ------   ------   ------   ------   ------   ------   ------
  Native Heap   5308     5280        0        0    16172     7658      741
  Dalvik Heap   7015     6684        0        0    19288    13124     6164
 Dalvik Other   3328     3184        0        0
        Stack    188      188        0        0
       Ashmem      2        0        0        0
    Other dev   4648     4356        4        0
     .so mmap   1296      404       20        0
    .apk mmap   1014        0      564        0
    .ttf mmap    299        0      200        0
    .dex mmap   1904       60     1132        0
   Other mmap    100        4       44        0
     Graphics   7904     7904        0        0
           GL  15916    15916        0        0
      Unknown    120      120        0        0
        TOTAL  49042    44100     1964        0    35460    20782     6905
```

DEVOXX™

Meminfo

```
Applications Memory Usage (kB):
Uptime: 27233364 Realtime: 252885787

** MEMINFO in pid 15976 [com.android.systemui] **
                   Pss  Private  Private  Swapped     Heap     Heap     Heap
                 Total    Dirty    Clean    Dirty     Size    Alloc     Free
                ------   ------   ------   ------   ------   ------   ------
  Native Heap     5308     5280        0        0    16172     7658      741
  Dalvik Heap     7015     6684        0        0    19288    13124     6164
 Dalvik Other     3328     3184        0        0
        Stack      188      188        0        0
       Ashmem        2        0        0        0
    Other dev     4648     4356        4        0
     .so mmap     1296      404       20        0
    .apk mmap     1014        0      564        0
    .ttf mmap      299        0      200        0
    .dex mmap     1904       60     1132        0
   Other mmap      100        4       44        0
     Graphics     7904     7904        0        0
           GL    15916    15916        0        0
      Unknown      120      120        0        0
        TOTAL    49042    44100     1964        0    35460    20782     6905
```

Meminfo

```
Applications Memory Usage (kB):
Uptime: 27233364 Realtime: 252885787

** MEMINFO in pid 15976 [com.android.systemui] **
                 Pss  Private  Private  Swapped     Heap     Heap     Heap
               Total    Dirty    Clean    Dirty     Size    Alloc     Free
              ------   ------   ------   ------   ------   ------   ------
  Native Heap   5308     5280        0        0    16172     7658      741
  Dalvik Heap   7015     6684        0        0    19288    13124     6164
 Dalvik Other   3328     3184        0        0
        Stack    188      188        0        0
       Ashmem      2        0        0        0
    Other dev   4648     4356        4        0
     .so mmap   1296      404       20        0
    .apk mmap   1014        0      564        0
    .ttf mmap    299        0      200        0
    .dex mmap   1904       60     1132        0
   Other mmap    100        4       44        0
     Graphics   7904     7904        0        0
           GL  15916    15916        0        0
      Unknown    120      120        0        0
        TOTAL  49042    44100     1964        0    35460    20782     6905
```

Meminfo

```
Applications Memory Usage (kB):
Uptime: 27233364 Realtime: 252885787

** MEMINFO in pid 15976 [com.android.systemui] **
                   Pss  Private  Private  Swapped     Heap     Heap     Heap
                 Total    Dirty    Clean    Dirty     Size    Alloc     Free
                ------   ------   ------   ------   ------   ------   ------
  Native Heap     5308     5280        0        0    16172     7658      741
  Dalvik Heap     7015     6684        0        0    19288    13124     6164
 Dalvik Other     3328     3184        0        0
        Stack      188      188        0        0
       Ashmem        2        0        0        0
    Other dev     4648     4356        4        0
     .so mmap     1296      404       20        0
    .apk mmap     1014        0      564        0
    .ttf mmap      299        0      200        0
    .dex mmap     1904       60     1132        0
   Other mmap      100        4       44        0
     Graphics     7904     7904        0        0
           GL    15916    15916        0        0
      Unknown      120      120        0        0
        TOTAL    49042    44100     1964        0    35460    20782     6905
```

Meminfo (2)

```
 Objects
               Views:      288         ViewRootImpl:        5
         AppContexts:       12           Activities:        1
              Assets:        6        AssetManagers:        6
       Local Binders:       66        Proxy Binders:       47
    Death Recipients:        2
     OpenSSL Sockets:        0
```

Meminfo (2)

```
 Objects
               Views:      288         ViewRootImpl:        5
         AppContexts:       12           Activities:        1
              Assets:        6        AssetManagers:        6
       Local Binders:       66        Proxy Binders:       47
    Death Recipients:        2
     OpenSSL Sockets:        0
```

Meminfo (2)

```
 Objects
               Views:      288         ViewRootImpl:       5
         AppContexts:       12           Activities:       1
              Assets:        6        AssetManagers:       6
       Local Binders:       66        Proxy Binders:      47
    Death Recipients:        2
     OpenSSL Sockets:        0
```

Meminfo (2)

```
Objects
               Views:      288          ViewRootImpl:        5
         AppContexts:       12            Activities:        1
              Assets:        6         AssetManagers:        6
       Local Binders:       66         Proxy Binders:       47
    Death Recipients:        2
     OpenSSL Sockets:        0
```

Meminfo (3) -a

```
** MEMINFO in pid 15976 [com.android.systemui] **
                 Pss      Pss   Shared  Private   Shared  Private  Swapped
               Total    Clean    Dirty    Dirty    Clean    Clean    Dirty
              ------   ------   ------   ------   ------   ------   ------
  Native Heap   5308        0      780     5280        0        0        0
  Dalvik Heap   7031        0     7484     6700        0        0        0
 Dalvik Other   3332        0     3204     3188        0        0        0
        Stack    188        0        8      188        0        0        0
       Ashmem      2        0        4        0        0        0        0
    Other dev   4648        0      604     4356        0        4        0
     .so mmap   1296       20     2692      404     6620       20        0
    .apk mmap   1014      564        0        0     1536      564        0
    .ttf mmap    299      200        0        0      388      200        0
    .dex mmap   1904     1132      416       60     8304     1132        0
   Other mmap    100        0       12        4      332       44        0
     Graphics   7904        0        0     7904        0        0        0
           GL  15916        0        0    15916        0        0        0
      Unknown    120        0        4      120        0        0        0
        TOTAL  49062     1916    15208    44120    17180     1964        0
```

Meminfo (3) -a

```
** MEMINFO in pid 15976 [com.android.systemui] **
                   Pss      Pss   Shared  Private   Shared  Private  Swapped
                 Total    Clean    Dirty    Dirty    Clean    Clean    Dirty
                ------   ------   ------   ------   ------   ------   ------
  Native Heap     5308        0      780     5280        0        0        0
  Dalvik Heap     7031        0     7484     6700        0        0        0
 Dalvik Other     3332        0     3204     3188        0        0        0
        Stack      188        0        8      188        0        0        0
       Ashmem        2        0        4        0        0        0        0
    Other dev     4648        0      604     4356        0        4        0
     .so mmap     1296       20     2692      404     6620       20        0
    .apk mmap     1014      564        0        0     1536      564        0
    .ttf mmap      299      200        0        0      388      200        0
    .dex mmap     1904     1132      416       60     8304     1132        0
   Other mmap      100        0       12        4      332       44        0
     Graphics     7904        0        0     7904        0        0        0
           GL    15916        0        0    15916        0        0        0
      Unknown      120        0        4      120        0        0        0
        TOTAL    49062     1916    15208    44120    17180     1964        0
```

Exercise

```java
// 1MB + some overhead
class Chunk {
    byte[] padding = new byte[1024 * 1024];
}
```

Exercise

```
List<Chunk> mRetainedChunks = new ArrayList<Chunk>();
List<Chunk> mTempChunks = new ArrayList<Chunk>();

int i = 0;
Runtime runtime = Runtime.getRuntime();
long max = runtime.maxMemory() - 1024 * 1024;

while (runtime.totalMemory() < max) {
    ((i++ % 2 == 0) ? mRetainedChunks : mTempChunks)
        .add(new Chunk());
}
```

Exercise

```
Log.d("Heap", "max=" + toMB(runtime.maxMemory()));
Log.d("Heap", String.format("heap: %.2f/%.2f",
    toMB(runtime.freeMemory()),
    toMB(runtime.totalMemory())));
```

Exercise

```
Log.d("Heap", "max=" + toMB(runtime.maxMemory()));
Log.d("Heap", String.format("heap: %.2f/%.2f",
    toMB(runtime.freeMemory()),
    toMB(runtime.totalMemory())));
```

```
# < 1 MB of free memory
D/Heap(13055): max=192.0
D/Heap(13055): heap: 0.84/191.99
```

Exercise

```
// Remove half the chunks
mTransientChunks.clear();

// Force a GC to free up memory
System.gc();
```

Exercise

```
// Remove half the chunks
mTransientChunks.clear();

// Force a GC to free up memory
System.gc();
```

```
# < 87 MB of free memory
D/Heap(13055): heap: 87.86/191.99
```

Exercise

```java
// Allocate ~2MB
Bitmap b = Bitmap.createBitmap(1024, 512,
    Bitmap.Config.ARGB_8888);
```

Exercise

```
// Allocate ~2MB
Bitmap b = Bitmap.createBitmap(1024, 512,
    Bitmap.Config.ARGB_8888);
```

```
D/dalvikvm: GC_BEFORE_OOM freed 0K, 46% free 106633K/196600K, ...
E/dalvikvm-heap: Out of memory on a 2097168-byte allocation.
```

Dalvik Heap Management

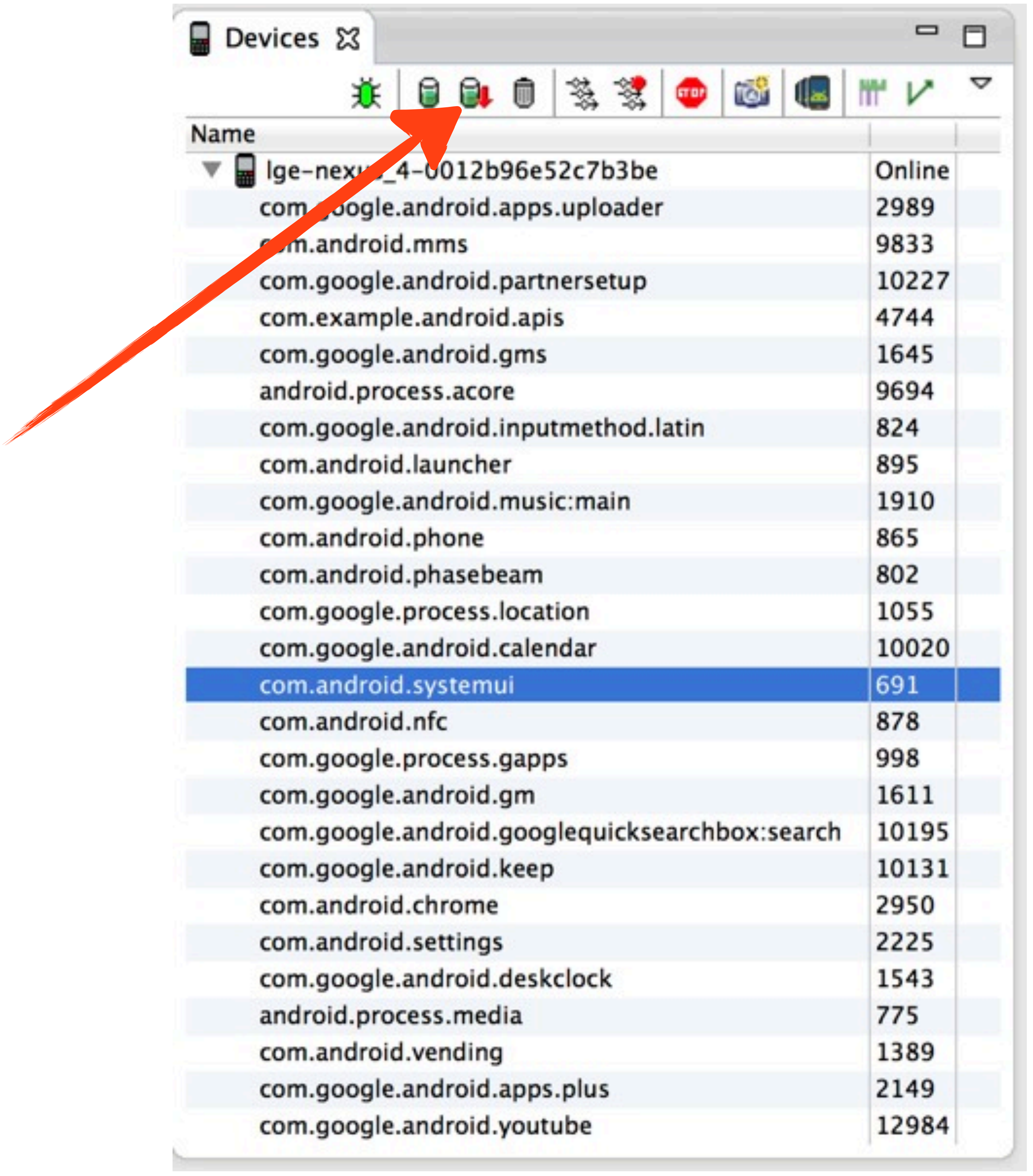
- Single virtual memory range
- Non-compacting
  - The heap will fragment!
- Can shrink if unused space at end of range
- madvise used to free individual pages inside of the range
  - returns memory to kernel
    - even if the heap size does not shrink

Dalvik Heap Analysis

- Zygote allocations are generally not of concern for an app
- Convert data prior to heap analysis: hprof-conv

Collect Heap Data

- Run your app
- Select your app in DDMS
- Press "Dump HPROF File" button
- Save file

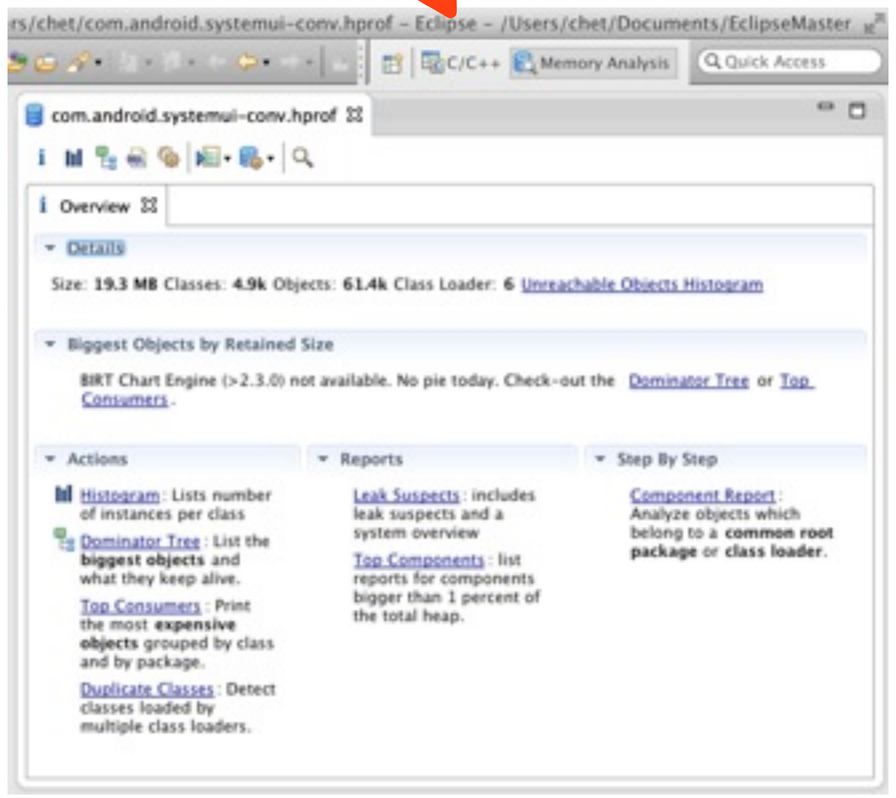


Name	
lge-nexus_4-0012b96e52c7b3be	Online
com.google.android.apps.uploader	2989
com.android.mms	9833
com.google.android.partnersetup	10227
com.example.android.apis	4744
com.google.android.gms	1645
android.process.acore	9694
com.google.android.inputmethod.latin	824
com.android.launcher	895
com.google.android.music:main	1910
com.android.phone	865
com.android.phasebeam	802
com.google.process.location	1055
com.google.android.calendar	10020
com.android.systemui	691
com.android.nfc	878
com.google.process.gapps	998
com.google.android.gm	1611
com.google.android.googlequicksearchbox:search	10195
com.google.android.keep	10131
com.android.chrome	2950
com.android.settings	2225
com.google.android.deskclock	1543
android.process.media	775
com.android.vending	1389
com.google.android.apps.plus	2149
com.google.android.youtube	12984

Analyze with [jh|m]at

```
$ hprof-conv ~/systemui.hprof ~/systemui-conv.hprof
$ jhat ~/systemui-conv.hprof
```

- localhost:7000 in browser
- or load into mat


rs/chet/com.android.systemui-conv.hprof – Eclipse – /Users/chet/Documents/EclipseMaster

Details

Size: **19.3 MB** Classes: **4.9k** Objects: **61.4k** Class Loader: **6** Unreachable Objects Histogram

Biggest Objects by Retained Size

BIRT Chart Engine (>2.3.0) not available. No pie today. Check-out the Dominator Tree or Top Consumers.

Actions

Histogram : Lists number of instances per class

Dominator Tree : List the **biggest objects** and what they keep alive.

Top Consumers : Print the most **expensive objects** grouped by class and by package.

Duplicate Classes : Detect classes loaded by multiple class loaders.

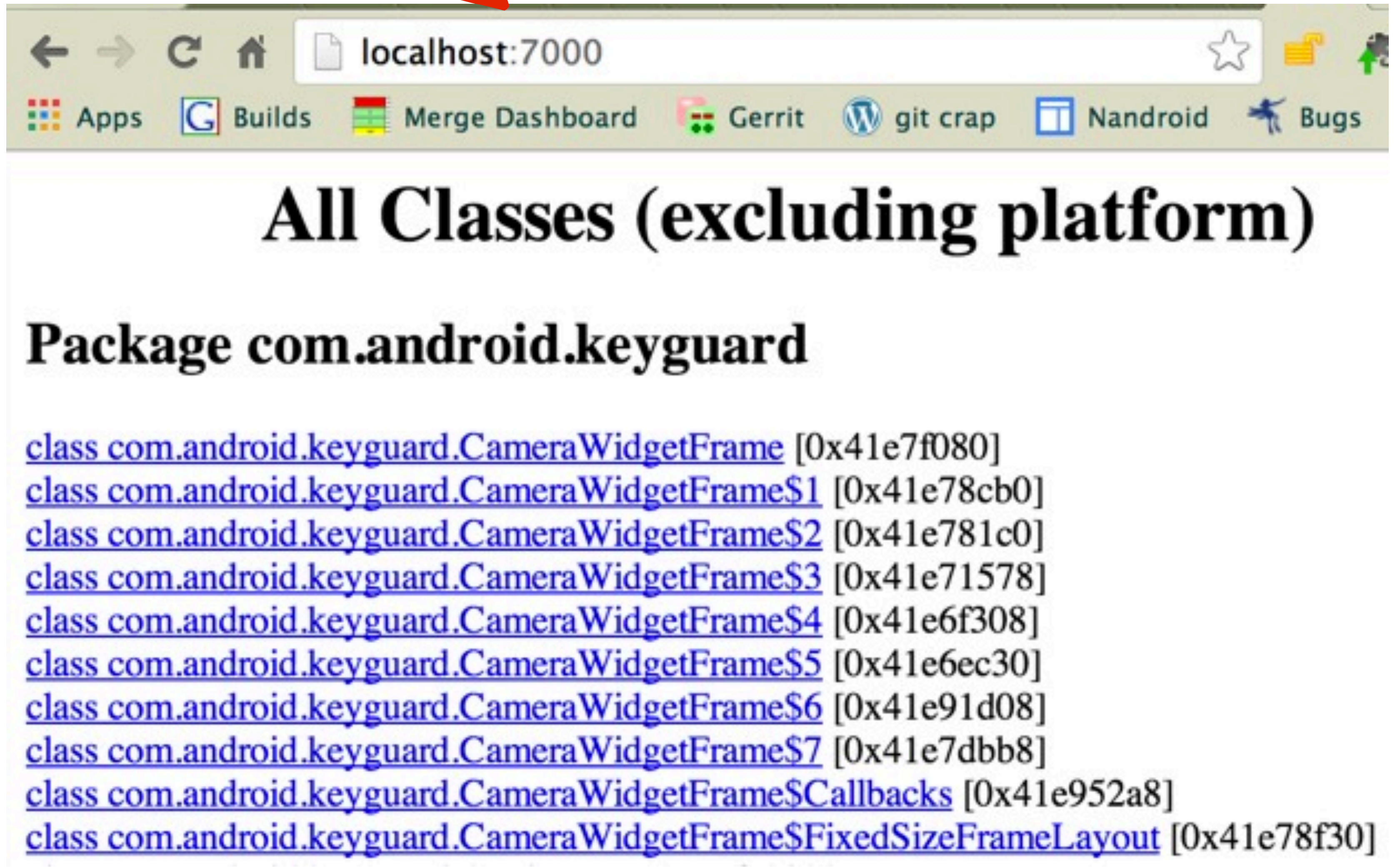
Reports

Leak Suspects : includes leak suspects and a system overview

Top Components : list reports for components bigger than 1 percent of the total heap.

Step By Step

Component Report : Analyze objects which belong to a **common root package** or **class loader**.



All Classes (excluding platform)

Package com.android.keyguard

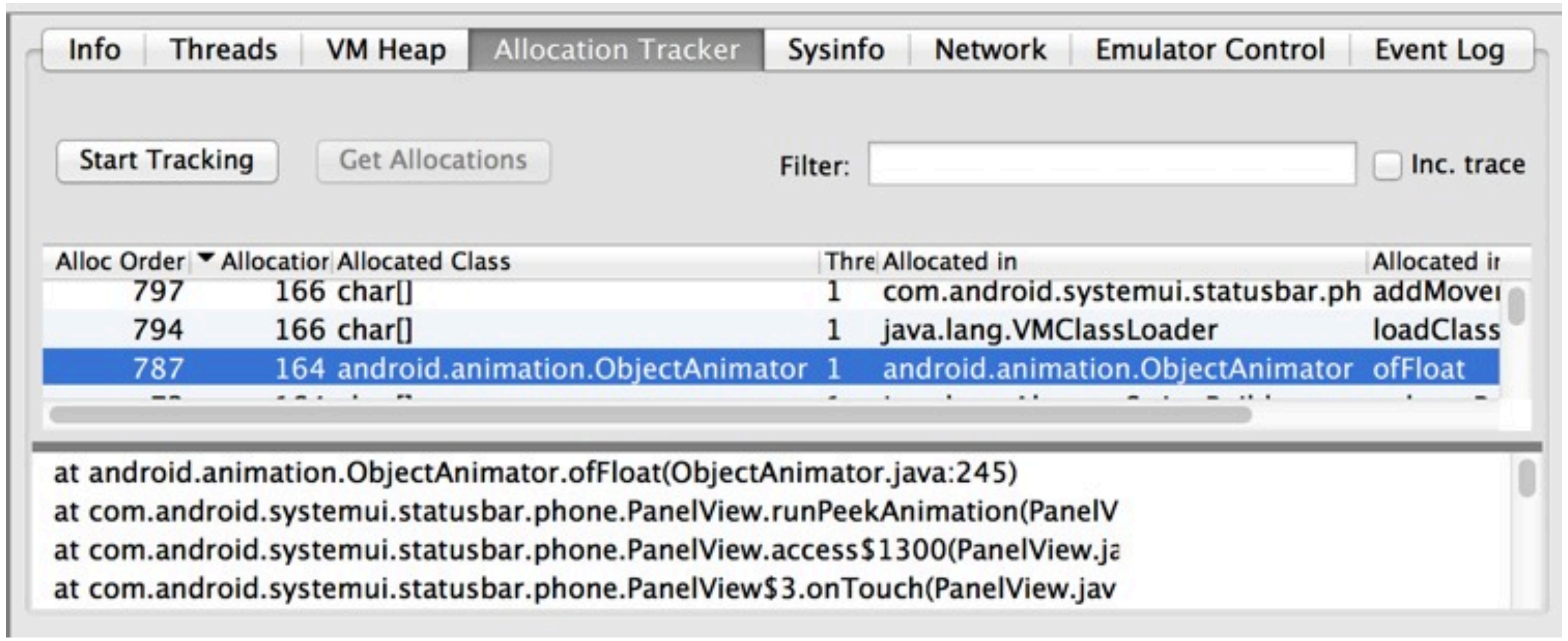
class com.android.keyguard.CameraWidgetFrame [0x41e7f080]
class com.android.keyguard.CameraWidgetFrame$1 [0x41e78cb0]
class com.android.keyguard.CameraWidgetFrame$2 [0x41e781c0]
class com.android.keyguard.CameraWidgetFrame$3 [0x41e71578]
class com.android.keyguard.CameraWidgetFrame$4 [0x41e6f308]
class com.android.keyguard.CameraWidgetFrame$5 [0x41e6ec30]
class com.android.keyguard.CameraWidgetFrame$6 [0x41e91d08]
class com.android.keyguard.CameraWidgetFrame$7 [0x41e7dbb8]
class com.android.keyguard.CameraWidgetFrame$Callbacks [0x41e952a8]
class com.android.keyguard.CameraWidgetFrame$FixedSizeFrameLayout [0x41e78f30]

Finding Leaks

- Simple way:
  - 1. Run app for a while
  - 2. Look at heap
  - 3. Profit!
- Caveats:
  - Use “adb shell dumpsys meminfo <app>” for initial overview
  - Finding large leaks is easy (sort by size)
  - finding systemic memory problems is often hard

Dalvik Allocation Tracker

- Allocations over a set period of time
- DDMS allocation tracker:
  - Select app
  - "Start tracking"
  - Interact with app
  - "Get allocations"
  - Click to see stack
- Good tool for jank, too!

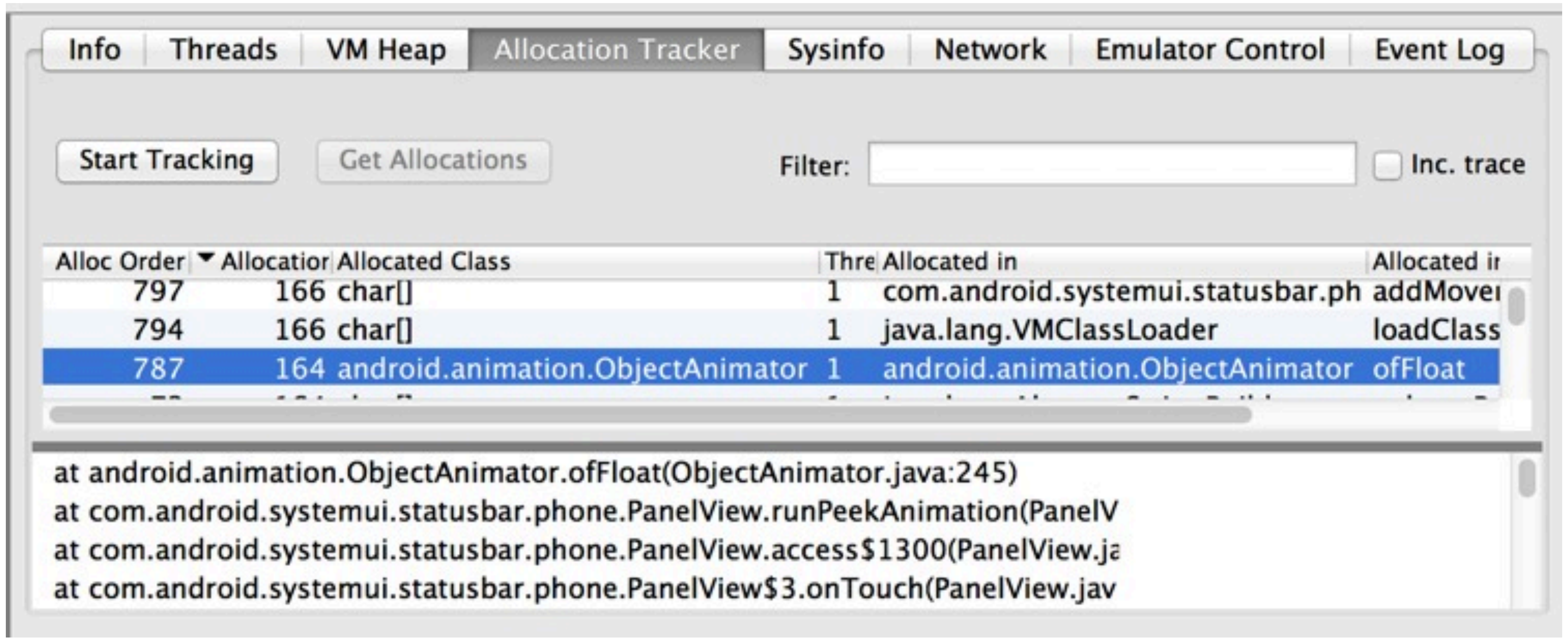


Alloc Order	Allocatior	Allocated Class	Thre	Allocated in	Allocated ir
797	166	char[]	1	com.android.systemui.statusbar.ph	addMover
794	166	char[]	1	java.lang.VMClassLoader	loadClass
787	164	android.animation.ObjectAnimator	1	android.animation.ObjectAnimator	ofFloat

```
at android.animation.ObjectAnimator.ofFloat(ObjectAnimator.java:245)
at com.android.systemui.statusbar.phone.PanelView.runPeekAnimation(PanelV
at com.android.systemui.statusbar.phone.PanelView.access$1300(PanelView.ja
at com.android.systemui.statusbar.phone.PanelView$3.onTouch(PanelView.jav
```

Dalvik Allocation Tracker

- Allocations over a set period of time
- DDMS allocation tracker:
  - Select app
  - “Start tracking”
  - Interact with app
  - “Get allocations”
  - Click to see stack
- ~~Good~~ Great tool for jank, too!



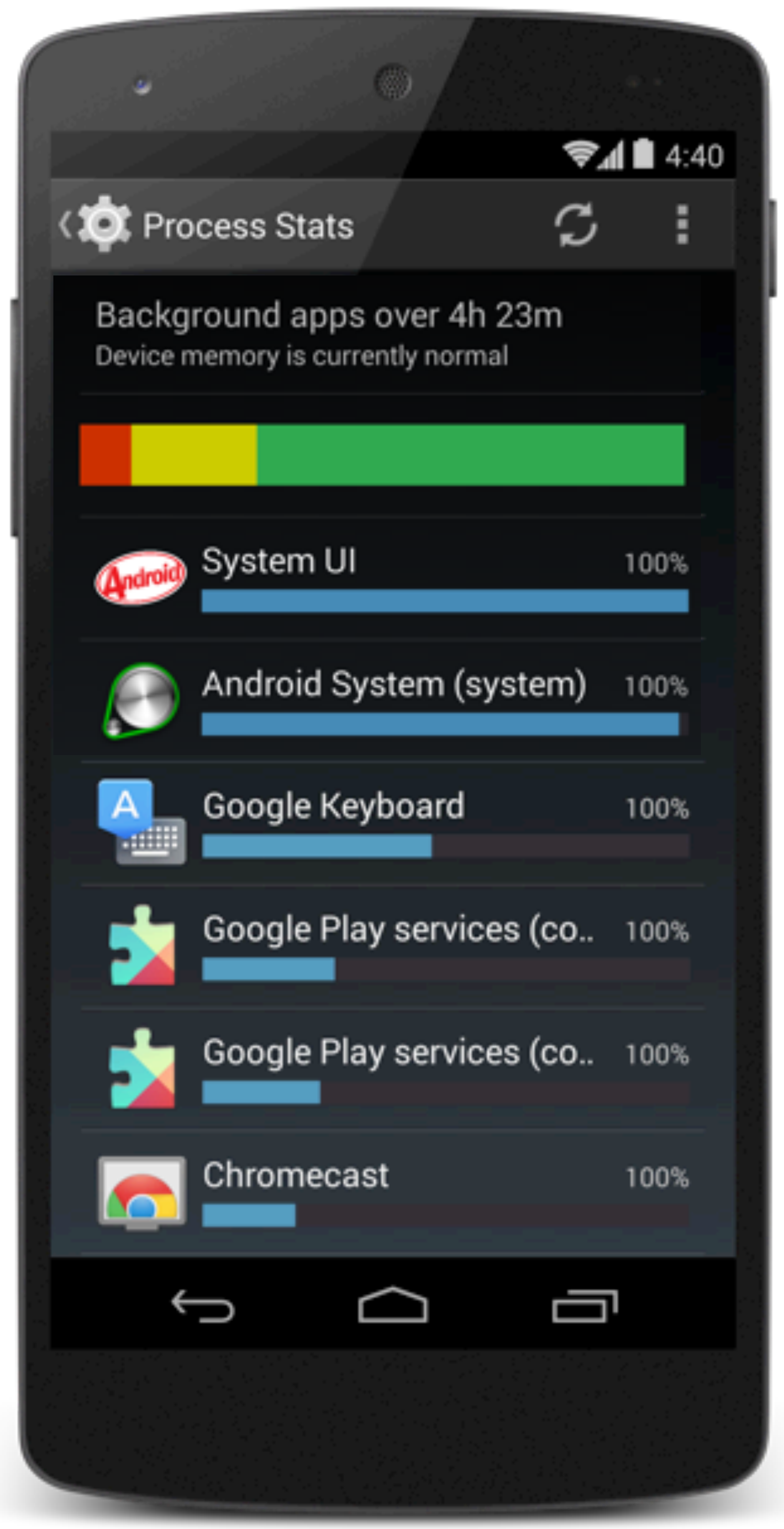
Alloc Order	Allocatior	Allocated Class	Thre	Allocated in	Allocated ir
797	166	char[]	1	com.android.systemui.statusbar.ph	addMovei
794	166	char[]	1	java.lang.VMClassLoader	loadClass
787	164	android.animation.ObjectAnimator	1	android.animation.ObjectAnimator	ofFloat

```
at android.animation.ObjectAnimator.ofFloat(ObjectAnimator.java:245)
at com.android.systemui.statusbar.phone.PanelView.runPeekAnimation(PanelV
at com.android.systemui.statusbar.phone.PanelView.access$1300(PanelView.ja
at com.android.systemui.statusbar.phone.PanelView$3.onTouch(PanelView.jav
```

Processes

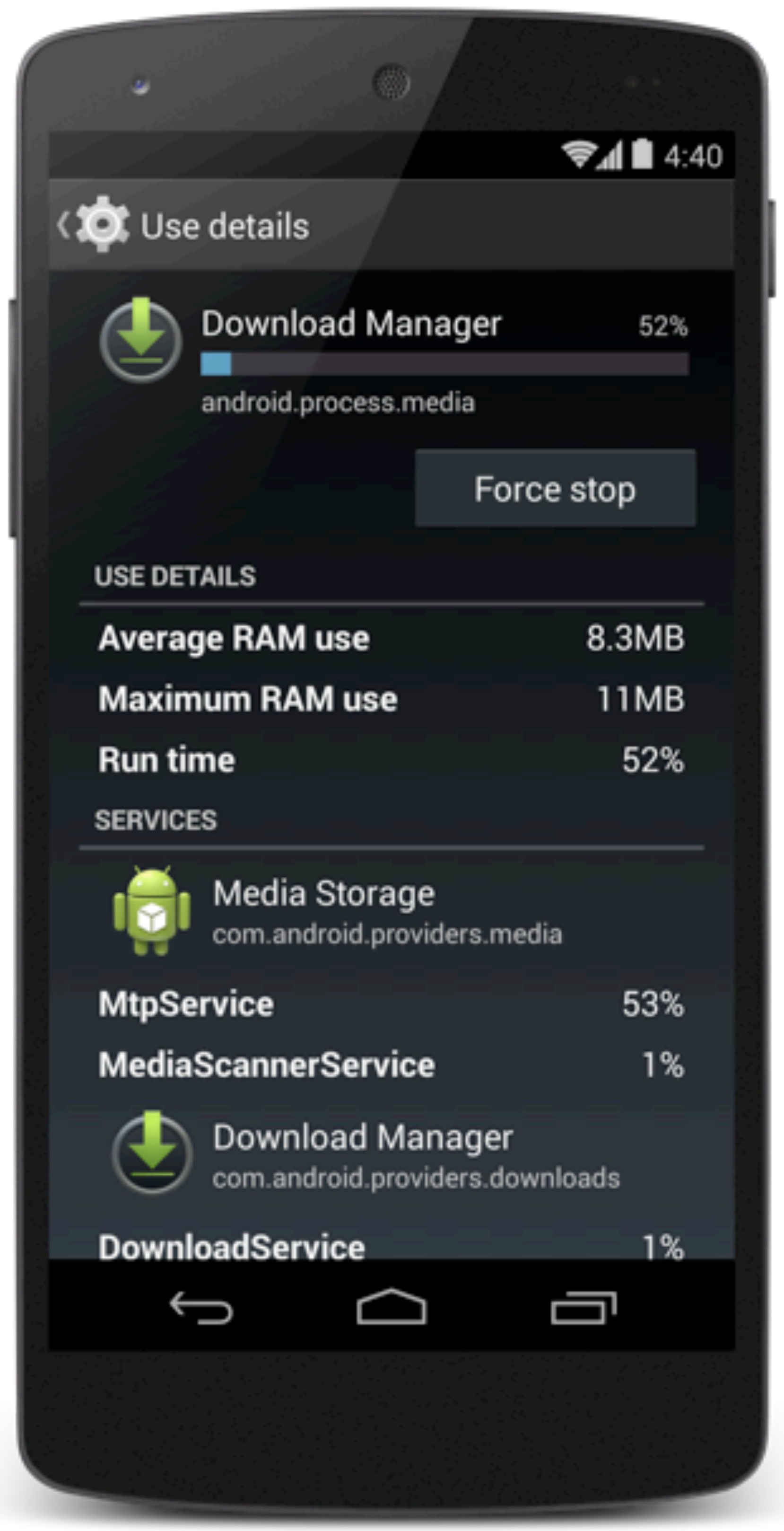
- Every process has overhead
  - Empty, do-nothing process: 1.5 MB USS
  - Ready to show UI: 4 MB USS
  - Showing UI: much more
- Avoid multiple processes in general
- Possible to run multiple apps in one process
  - Activity’s `android:process` attribute

procstats: UI


Process Stats

Background apps over 4h 23m
Device memory is currently normal

System UI	100%
Android System (system)	100%
Google Keyboard	100%
Google Play services (co..	100%
Google Play services (co..	100%
Chromecast	100%


Use details

Download Manager 52%
android.process.media

Force stop

USE DETAILS

Average RAM use	8.3MB
Maximum RAM use	11MB
Run time	52%

SERVICES

Media Storage
com.android.providers.media

MtpService	53%
MediaScannerService	1%

Download Manager
com.android.providers.downloads

DownloadService	1%

DEVOXX™

procstats: command line

```
$ adb shell dumpsys procstats com.google.android.apps.maps

COMMITTED STATS FROM 2013-11-05-18-04-58:
  * com.google.android.apps.maps / u0a60:
           TOTAL: 1.1%
         Service: 1.1%
        (Cached): 99% (98MB-98MB-99MB/96MB-97MB-97MB over 7)

Run time Stats:
  Screen Off / Norm / +1h19m25s22ms
  Screen On  / Norm / +10m43s963ms
               TOTAL: +1h30m8s985ms

          Start time: 2013-11-05 18:04:58
  Total elapsed time: +5h9m53s44ms (complete) libdvm.so chromeview
```

For More Information

- Managing Your App’s Memory
  - http://developer.android.com/training/articles/memory.html