

ORACLE®



Java™
ORACLE®

Best Practices for Efficient Java ME Programming

Andrey Petushkov
Architect, Java Embedded implementation
Oracle
October 2, 2014



Note: The speaker notes for this slide include detailed instructions on how to reuse this Title Slide in another presentation.

Tip! Remember to remove this text box.

ME target devices

- Footprint
- CPU power & capabilities
- Battery
- Always on
- Unattended
- Connected

ME target devices

- Footprint: 256k RAM, 1M ROM
- CPU power & capabilities: 120MHz, not caches, 16-bit mem bus
- Battery: a few month on battery
- Always on: always
- Unattended: no leaks, auto-restart, remote monitoring
- Connected: all kinds, wired/wireless, from NFC up to satellite

Java gives you ***power***

- Concept
- Language
- APIs
- Modularity
- Skill set
- Security

Java gives you ***power***

- Concept
- Language
- APIs
- Modularity
- Skill set
- Security

- GC
- Threads
- Multi-VM

Java gives you ***power***

- Concept
- Language
- APIs
- Modularity
- Skill set
- Security

- Classes
- Interfaces
- Objects
- Lambda
- Varargs
- Autoboxing

Java gives you ***power***

- Concept
- Language
- APIs
- Modularity
- Skill set
- Security

- A lot of APIs!
- Libraries
- JSRs
- Buffering
- Iterating
- Paralleling

Java gives you ***power***

- Concept
- Language
- APIs
- Modularity
- Skill set
- Security

- Runtime profiles
- Shared libraries
- On demand downloading

Java gives you ***power***

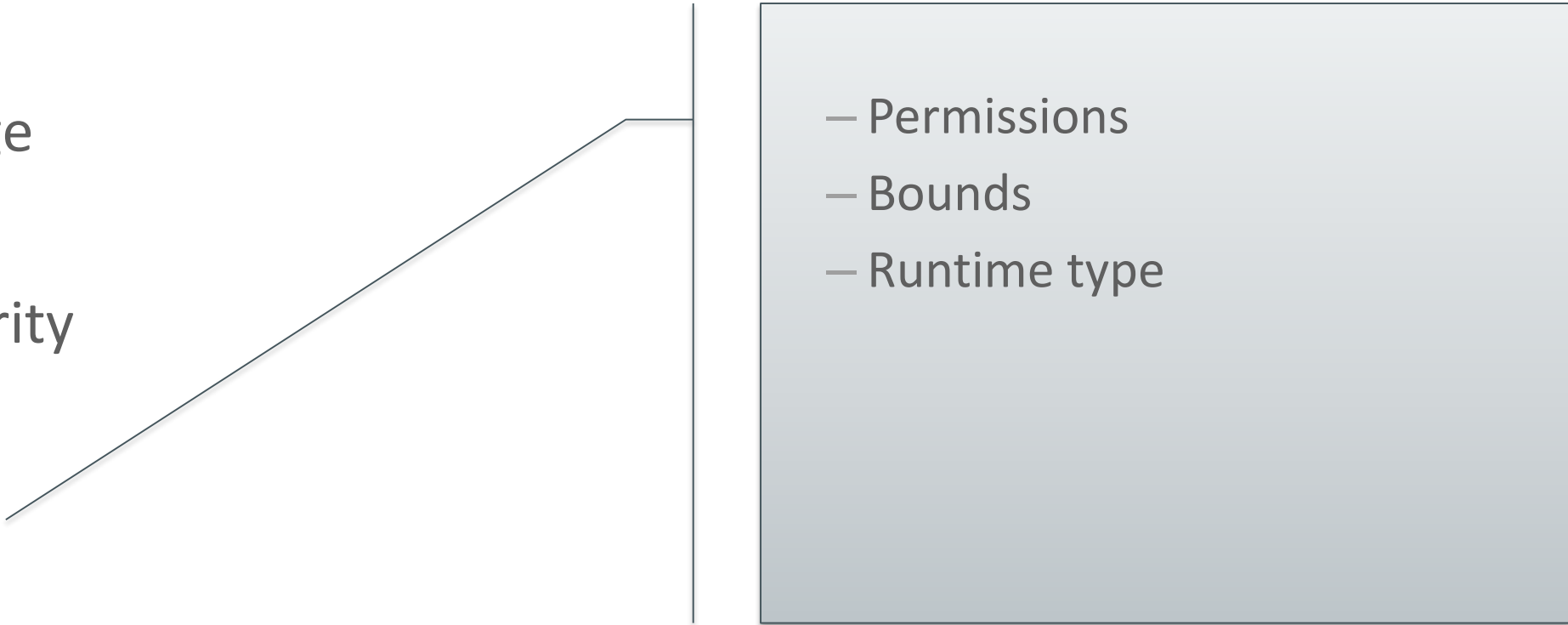
- Concept
- Language
- APIs
- Modularity
- Skill set
- Security



– 9M Java developers worldwide

Java gives you ***power***

- Concept
- Language
- APIs
- Modularity
- Skill set
- Security

- 
- Permissions
 - Bounds
 - Runtime type

“Apparently, miracles don’t come cheap”

– Dana Scully, FBI

So what's wrong?

- GC: pauses
 - Watch your allocation rate!
 - Force GC before time-critical operations
 - `Runtime.freeMemory()` – only estimate (likely there is more, force GC to find out)
 - Use tools (ME SDK Memory Profiling)
- Threads and MVM
 - Synchronization: lightweight in OJMEE but still can block
 - Priorities
 - Native code execution == stop the world
 - 1 thread => 1 execution stack

OOP(s)

- Metadata
 - Classes, interfaces. 200
 - Methods. 50
- Lambda
 - A class indeed
- Varargs
- Autoboxing

Essentially an array

```
void a(String... a);  
a("1", "2", "3");
```

```
==
```

```
0:  iconst_3  
1:  anewarray    #4    // class java/lang/String  
4:  dup  
5:  iconst_0  
6:  ldc          #5    // String 1  
8:  astore  
9:  dup  
10: iconst_1  
11: ldc          #6    // String 2  
13: astore  
14: dup  
15: iconst_2  
16: ldc          #7    // String 3  
18: astore  
19: invokestatic #8    // Method a:([Ljava/lang/String;)V
```

Leaving OOP aside

- Literals
 - Mind the constant pool
 - And its initialization
- Blocks and branches
 - Local variables
 - Stackmaps indeed
- switch (a)
 - Integer is easy
 - String is not

```
switch (args[0]) {
  case "1":
    ...
  ==
  7: invokevirtual #4 // Method String.hashCode()
  10: tableswitch  { // 49 to 52
    ...
  }
  40: aload_1
  41: ldc          #5 // String 1
  43: invokevirtual #6 // Method String.equals(Object)
  46: ifeq        93
  49: iconst_0
  50: istore_2
  51: goto         93
  ...
  93: iload_2
  94: tableswitch  { // 0 to 3
    ...
  }
  124: getstatic   #2 // Field System.out
  ...
  165: return
```


And that's not the end

- Foreach
- Enums
 - Essentially a class
 - Stop, it's actually 2 classes!
 - Maybe even more
 - switch (enum): +1 array
- Initialized arrays
- Debug conditions

```
enum Operation {  
    PLUS {  
        double eval(double x, double y) { return x + y; }  
    }  
    ...  
    abstract double eval(double x, double y);  
}
```

Sandbox, yeah

- Application isolation / MVM
 - Resources
 - Communication == copying
 - Java runtime state
- Application security
 - No ClassLoader
 - Permissions all over the place
 - X.509, likely

Built-in intelligence

- Adaptive compiler
 - Long and cold.. methods
- APIs
 - Buffers of all kinds
 - Class loading, initialization and ~~unloading~~

9M developers,
Are you ready for embedded?

So can embedded and Java co-exist together?

- Yes, it can! Two things make it happen:
- 2. ME APIs and the toolchain
 - Optimized
 - Provides hints and warnings
- 1. You!
 - The **embedded** developers

ME specific hints

- ME SDK provides hints on potentially inefficient constructs

```
static void a(Integer a) {  
    System.out.println("a "+a);  
}  
  
...  
HelloHeap.a(1);
```



```
HelloHeap.java:20: warning: Integer is unboxed  
into int  
        System.out.println("a "+a);  
HelloHeap.java:29: warning: int is boxed into  
Integer  
        HelloHeap.a(1);  
  
2 warnings
```

- Need to install ME SDK plugins into IDE

Choose wisely

- Structure
 - Tasks, classes, methods, branches, constants
 - Values are nicely inlined when final. Methods too
 - Less objects
- Post-process
 - Obfuscate
- More code for performance, less code for footprint
- Off-load processing to higher tier
- The goal and the target

Java ME 8 Resources

- Java ME 8 Oracle Technology Network (OTN) downloads
Free for development and evaluation purposes
 - Oracle Java ME Embedded 8.1 Developer Preview
 - Oracle Java ME SDK 8.1 Early Access #3
 - <http://www.oracle.com/technetwork/java/embedded/javame/embed-me/downloads/index.html>
- Java ME 8 Documentation
 - Developer Preview on FRDM-K64F: *Release Notes, Getting Started Guide*
 - *Java ME 8 Developer Guide*, plus new chapter: *Java ME Optimization Techniques*
 - Full Java ME 8 API doc set
 - <http://docs.oracle.com/javame/8.0/>
- Terrence Barr's blog
 - <http://terrencebarr.wordpress.com/>

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

CREATE THE FUTURE

