Class Evolution

Nik Shaylor
Doug Simon

Overview

- Mechanisms for provisioning
- Class evolution defined
- Summary of issues
- HotSwap

Provisioning issues

- How are updates delivered to VM?
- Possibilities include (but not limited to):
  1. Use of some delta format (c.f. unix “diff/patch”)
  2. Replace single class (HotSwap)
  3. Replace original unit of installation (i.e. replace whole suite/jar file)
  4. Others?
- RAM requirements for updating?

Class evolution defined

- Class evolution includes:
  1. Method body replacement
  2. Addition/deletion of methods
  3. Addition/deletion of fields
  4. Other changes

- Need to consider instance conversion for each

Method patching

- Most common/desirable evolution (i.e. bug fix)
- Must verify new code → retain some symbolic info
- Does not affect existing instances or vtables

Method addition/deletion

- Method addition is binary compatible, deletion is not → must verify existing code does not called deleted method(s)
- Addition of instance methods requires vtable modification of all affected classes in hierarchy
- Does not affect existing instances

Field addition/deletion

- Field addition is binary compatible, deletion is not → must verify existing code does not access deleted field(s)
- Existing instances must be converted. What should VM guarantee about conversion:
  1. Nothing?
  2. Structural integrity (i.e. type safety)?
  3. Accessibility compatibility (e.g. public → private)?
  4. Semantic compatibility (i.e. field slots have same name+type)?
- Similiar issues for other update categories

Other changes

- Changes in class hierarchy (i.e. modify **extends** or **implements** list for a class)
- Class, field and method attribute changes (e.g. **final** → non-**final**, **private** → **public** etc.)
- Class attribute changes etc.

Updating instances

- Updating instances requires support for Smalltalk-like ‘becomes’ functionality
- Growing instances may cause memory overflow
- May require retention of some symbolic info for classes that will potentially be updated (depending on desired VM guarantees)
- Updated classes must affect only instances in EEPROM?

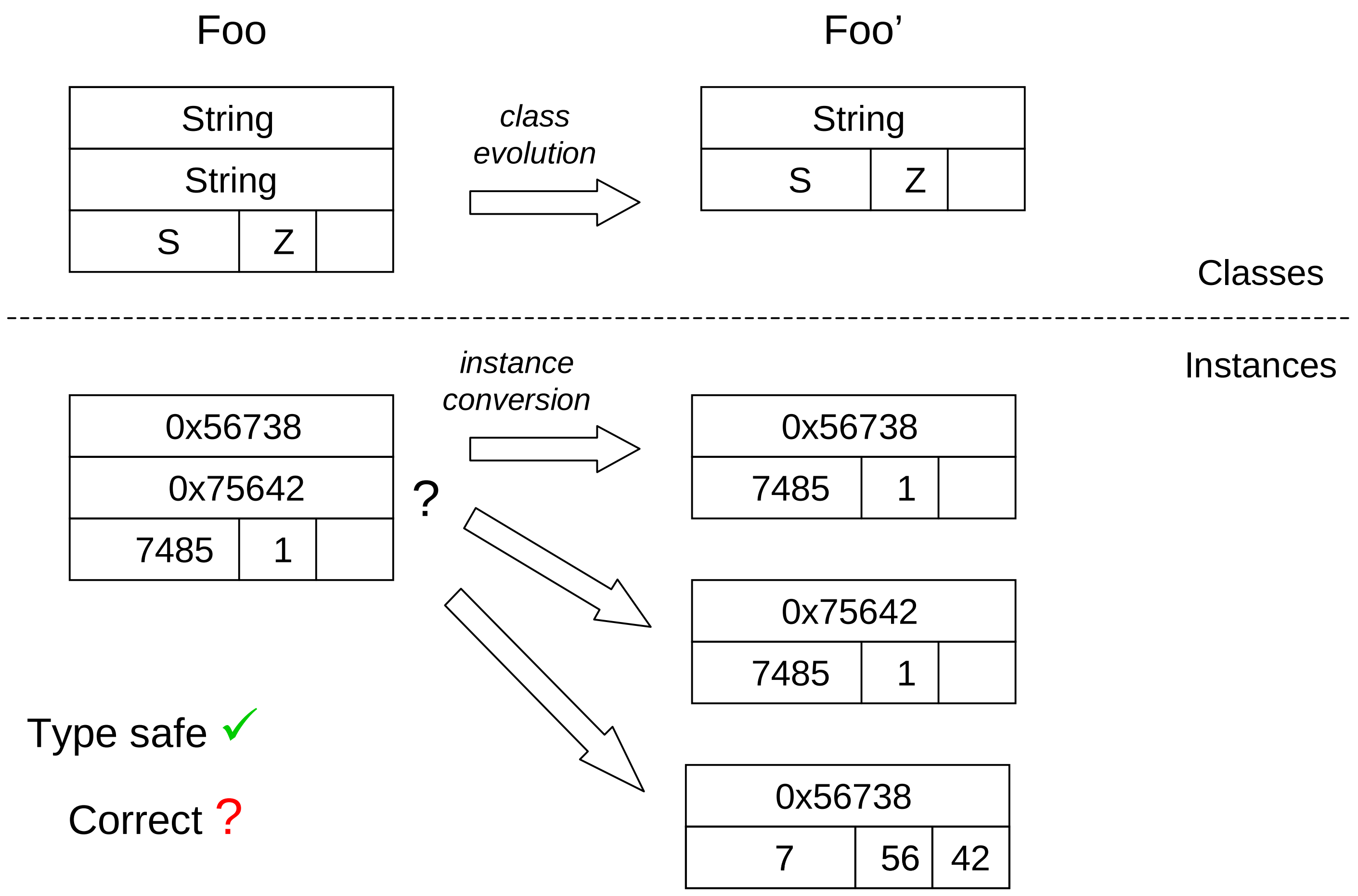
Overall issues

- What level of evolutionary capability is desirable in Java Card?
- VM must verify safety of any changes. Can also be required to verify semantic compatibility.
- Verification of changes is as essential as verification in general.
- Trade off between space and functionality. E.g. must retain symbols to verify semantic compatibility.
- What is the state of the system if evolution does not verify? Need some kind of rollback support.

HotSwap

- HotSwap project at Sunlabs addresses class evolution in J2SE JVM
- Limited functionality (method patching) available as of JDK 1.4.1. Extra functionality in future releases
- http://www.experimentalstuff.com/Technologies/HotSwapTool/publications.html

Field deletion - example



Type safe ✓

Correct ?

Field deletion

- Need field symbolic info to:
  - Verify there are no external references
  - Establish field → offset mapping for instance conversion

- Need field symbolic info for all classes in all hierarchies affected