HotSwap

The project website including binaries for download and a bugtracking system can be found at http://ssw.jku.at/dcevm/.

The class data structures of the Java HotSpot virtual machine are immutable during the execution of a program. While new classes can be loaded, it is not possible to change existing classes (e.g. add/remove methods or fields). This project tries to relax this condition to allow arbitrary changes to existing Java classes (including changes to the class hierarchy) while running a Java program.

Prototype

Work in progress.

Latest patch update: December 17, 2010
Patch base: jdk7-b118
Full patch (advanced version): hotswap.patch
Patch with VM modifications only and limited feature set: hotswaplight.patch
Please send any comments or questions to: wuerthinger@ssw.jku.at

Implementation Notes

Initial implementation description (April 1, 2009): Thomas Wuerthinger, Dynamic Code Evolution for the Java HotSpot Virtual Machine
For more recent information, see the list of publications.

General Design Decisions

- Arbitrary changes possible (including changes to subtype relationships)
- No performance penalty before or after the change
- No introduced indirections
- Continued execution of old active methods
- Only simple, comprehensible strategies for field matching or method transitions
- Change possible at any point when the VM is suspended
- Graceful handling of accesses to deleted fields or calls of deleted methods

Status

<table>
<thead>
<tr>
<th>Type of Change</th>
<th>Supported?</th>
<th>Possible problems after resume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swap Method Body</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Add Method</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Remove Method</td>
<td>yes</td>
<td>NoSuchMethodError / none in advanced version</td>
</tr>
<tr>
<td>Add Field</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Remove Field</td>
<td>yes</td>
<td>NoSuchFieldError / none for static fields in advanced version</td>
</tr>
<tr>
<td>Add Supertype</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Remove Supertype</td>
<td>no / yes in advanced version</td>
<td>none, but not always guaranteed to be possible</td>
</tr>
</tbody>
</table>

Transformer Methods

Classes can implement a method "void $transformer()".
This method is called on every object whose class was redefined and can therefore be used to initialize the new instances of a class.
Additionally, it is possible to implement a static method "void $staticTransformer()" that will be called on the class only.
Example code:
### Initial version

```java
class A {
    int x;
}
```

### New version

```java
class A' {
    int x;
    int doubleX;

    static void staticTransformer() {
        System.out.println("Class A has a new version");
    }

    void transformer() {
        doubleX = 2 * x;
    }
}
```

### Future Work

- NetBeans support for advanced features such as transformer methods and method forwarding.