Referenz-Objekte in Java

soft, weak, phantom references

Maria Gensel, Ben Kraufmann

Freie Universität, Berlin

17. November 2005

Motivation

Das Szenario

- Ressource mit teurem Zugriff ist gegeben
- häufiger Zugriff auf Teile dieser Ressource
- daher Caching-Mechanismus benötigt

Das Problem dabei

- intuitiver Ansatz (pooling) ist schlecht
  - nicht benötigte Objekte verweilen im Speicher
  - $\Rightarrow$ unerwünschte Effekte in generationaler GC

Java bietet mit verschieden starken Referenz-Arten eine Lösung.

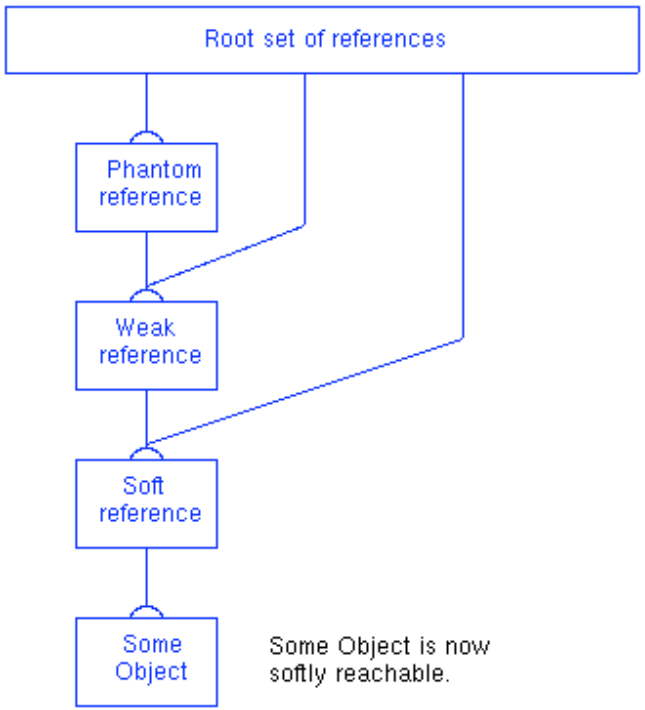
Referenz-Arten ...

...absteigend sortiert nach Stärke.

- strong references
- soft references
- weak references
- (phantom references)

Referenz-Pfade

- Objekte können vom root set ausgehend über mehrere Pfade erreichbar sein.
- Ein Objekt ist nur so stark erreichbar, wie die schwächste Referenz auf dem stärksten Pfad.



soft references

- Objekt ist nicht stark-, nicht schwach- und nicht phantomreferenziert.
- GC kann soft-referenzierte Objekte jederzeit abräumen.
- Strategie: Objekt lange vorhalten bis OOME[a] droht.
- Länger nicht benutzte Objekte werden bevorzugt abgeräumt.
- `java.lang.ref.SoftReference`

[a] out of memory error

weak references

- Schwach-referenzierte Objekte können von GC jederzeit eingesammelt werden.
- Strategie: sofort abräumen.
- `java.lang.ref.WeakReference`

`java.util.WeakHashMap`

- Eignet sich, um kurzlebigen Cache zu implementieren.
- Schlüssel sind schwach-referenziert.
- Ohne starke Referenz werden sie abgeräumt
- und damit effektiv auch die zugehörigen Wert-Objekte.

weak references

- Schwach-referenzierte Objekte können von GC jederzeit eingesammelt werden.
- Strategie: sofort abräumen.
- `java.lang.ref.WeakReference`

`java.util.WeakHashMap`

- Eignet sich, um kurzlebigen Cache zu implementieren.
- Schlüssel sind schwach-referenziert.
- Ohne starke Referenz werden sie abgeräumt
- und damit effektiv auch die zugehörigen Wert-Objekte.

reference queues

- GC fügt (wenn gewünscht) Referenz-Objekt in Schlange ein, wenn zugehöriges Objekt nicht mehr stark-referenziert.
- Auf dieses Ereignis kann gewartet werden.
  - blockierend `remove()`
  - nicht-blockierend `poll()`
- Ermöglicht Finalisieren von Objekten - zb in einem eigenen Thread.

phantom references

- Ist nicht für Caching relevant
- sondern ermöglicht sauberes Abräumen eines Objekts.
  - $\Rightarrow$ Freigabe von Ressourcen
- Deswegen **muss** Phantom-Referenz zusammen mit einer Schlange erzeugt werden.

Quelle

http://java.sun.com/developer/technicalArticles/ALT/RefObj/

FERTIG!