Objektzerstörung / Speicherfreigabe

Die Speicherorganisation für Objekte ist ein großes Problem bei Sprachen wie C++. Heap-Objekte können dort nur zerstört werden (*delete*), solange ihre Referenzen noch gültig sind. Daher belegen C++-Objekte, deren Referenzen versehentlich überschrieben oder anderweitig unbrauchbar wurden, weiterhin Speicherplatz – und vor allem: sie können dann nicht mehr gelöscht werden. Diese Situation ist gefährlich für lang laufende Programme, weil dadurch Speicherlecks entstehen können. Der freie Arbeitsspeicher des Prozesses wird immer knapper, weil er von toten Objekten (*Zombies*) belegt ist, die weder benutzt noch gelöscht werden können.

Garbage-Collection

Die Speicherverwaltung in Java ist anders geregelt als in C++. Nicht mehr benötigte Objekte werden im Rahmen einer in regelmäßig wiederholten automatischen Speicherbereinigung (Garbage-Collection = Müllsammlung) beseitigt. Der freigewordene Speicher wird reorganisiert und steht für neue Objekte zur Verfügung.

Die Garbage-Collection funktioniert nach einem ausgeklügelten Mechanismus, dessen Hauptbestandteil eine „Referenzzählung“ ist. Während des Programmablaufs werden die Speicheradressen aller Objekte mit den vorhandenen Referenzen verglichen. Dabei wird die Anzahl gültiger Referenzen notiert. Wenn ein Objekt nicht mehr referenziert wird, dann kann dieses Objekt bedenkenlos aus dem Speicher entfernt werden. Dieser Art der Referenzzählung leidet allerdings unter der Gefahr, dass Zirkelreferenzen die Löschung von Objekten verhindern können. Andere Algorithmen zur Bereinigung des Speichers sind die Generationszählung und die „*mark-sweep-collectors*“:

http://www-106.ibm.com/developerworks/java/library/j-jtp10283/#3.0

Expliziter Aufruf

Die Garbage-Collection ist insgesamt ein aufwändiger Vorgang, der viel Rechenzeit benötigt. Daher bestimmt die Java-Laufzeitumgebung am besten selbst, wann der günstigste Zeitpunkt dafür gekommen ist.

Durch Aufruf der Methode `System.gc()` kann die Laufzeitumgebung angewiesen werden, eine „außerordentliche“ Garbage-Collection durchzuführen. In den meisten Fällen wird sie baldmöglichst durchgeführt, es wird aber nicht garantiert, dass dies genau zum Zeitpunkt des Aufrufs geschieht.

finalize ( )

Vor der Zerstörung eines Objekts können Aufräumarbeiten nötig sein. Wenn beispielsweise ein Objekt eine Netzwerk- oder Datenbankverbindung geöffnet hat, dann muss vor der Zerstörung des Objektes diese Ressource wieder freigegeben werden. Für solche Fälle ist die `finalize( )`-Methode vorgesehen. Sie wird von der Garbage-Collection automatisch aufgerufen und ausgeführt, bevor das Objekt aus dem Speicher entfernt wird.

Hinweis: finalize( ) ist formal eine ganz "normale" Methode und es wäre daher durchaus möglich, sie direkt aufzurufen. Finalize( ) ist **kein Destruktor** und bei einem expliziten Aufruf wird nur der Methodenkörper ausgeführt, das betreffende Objekt jedoch **nicht** zerstört!

BEISPIEL

```
class FinalizeTest {

     public FinalizeTest() {
        System.out.println(this+" erzeugt");
     }

     public void finalize() {
        System.out.println(this+" zerstoert\n"+
                                "-----------------------------");
     }
     ////////////////////////////////////////
     public static void main(String[] args) {

      for (int i=0; i < 10; i++) {

        System.gc();
        FinalizeTest eins = new FinalizeTest();
        System.out.println(eins+" lebt");
      }
}
```