Destruktor

Wechseln zu:Navigation, Suche

Dieser Artikel erfüllt die GlossarWiki-Qualitätsanforderungen **nur teilweise**:

Korrektheit: 5 (vollständig überprüft)	**Umfang**: 5 (wesentliche Fakten vorhanden)	**Quellenangaben**: 2 (wichtige Quellen fehlen)	**Quellenarten**: 3 (gut)	**Konformität**: 5 (ausgezeichnet)

Inhaltsverzeichnis

1 Definition

Ein **Destruktor** ist eine spezielle Prozedur zur „Zerstörung“ existierender Objekte.

Ein Destruktor führt bei Aufruf folgende Aufgaben durch:

1. Optional: Aktivierung weiterer Finalisierungsmethoden; insbesondere können Beziehungen aufgelöst und weitere Objekte zerstört werden
2. Optional: Löschung des Objektzustandes, d.h. Überschreibung des Speicherplatzes mit geeigneten Finalwerten (Datenschutz!)
3. Freigabe des für den Objektzustand reservierten Speicherplatzes
4. Markieren des Objektidentifikators als ungültig (oder Freigabe des Objektidentifikators)

2 Bemerkungen

2.1 Destruktor als Methode

Ein Destruktor kann einer Klasse als Modifikationsmethode zugeordnet sein. Ein derartiger Destruktor verringert bei jedem Aufruf die zugehörige Klassenextension um ein Objekt.

Es gibt aber auch Destruktoren, die keiner Klasse zugeordnet sind. Derartige Konstruktoren zerstören Objekte, die ebenfalls keiner Klasse zugeordnet sind.

2.2 Beziehungen

Beziehungen zu anderen Objekten können nicht nur vom Destruktor, sondern auch früher von anderen Prozeduren und Modifikationsmethoden gelöscht werden.

2.3 Löschung des Objektzustandes

Die Löschung des Objektzustandes durch Überschreiben des Speicherplatzes mit binären Nullen oder Zufallszahlen wird i. Allg. nicht direkt unterstützt. Das heißt, wenn dies aus Datenschutzgründen notwendig sein sollte, muss der Programmierer entsprechenden Code selbst schreiben.

2.4 Freigabe des Objektidentifikators

Aus Sicht der "reinen objektorientierten Lehre" sollte ein Objektidentifikator niemals wiederverwendet werden, insbesondere dann nicht, wenn es möglich ist, Objektidentifikatoren im Klartext abzuspeichern oder auszudrucken. Anderenfalls könnten außerhalb des eigentlichen Systems Verweise auf falsche Objekte existieren.

Allerdings werden vor allem in hauptspeicherbasierten Programmiersystemen (Java, C++ etc.) meist Hauptspeicher-Adressen als Objektidentifikatoren verwendet. In derartigen Systemen müssen Objektidentifikatoren wiederverwendet werden, da (realer oder virtueller) Hauptspeicher nicht unbegrenzt zur Verfügung steht.

In objektorientierten Datenbanksystemen werden dagegen Objektidentifikatoren meist nicht wiederverwendet. Aus Performanzgründen werden die Objektidentifikatoren dann allerdings temporär durch Hauptspeicheradressen ersetzt, wenn ein Objekt in den Hauptspeicher geladen wird. Diese (nicht-triviale) Optimierungstechnik heißt Pointer Swizzling.

2.5 Garbage Collection

Viele objektorientierte Programmiersysteme unterstützen Garbage Collection, d. h. die automatische Zerstörung von Objekten, auf die nicht mehr zugegriffen werden kann. In derartigen System braucht der Programmierer Destruktoren nicht explizit zu programmieren.

Diese (ebenfalls nicht-triviale) Technik hilft, viele Programmierfehler zu vermeiden. Sie kann allerdings temporär zu Perfomanzeinbußen führen.

3 Quellen

1. **Kowarschick (MMProg)**: Wolfgang Kowarschick; Vorlesung „Multimedia-Programmierung“; Hochschule: Hochschule Augsburg; Adresse: Augsburg; Web-Link; 2018; Quellengüte: 3 (Vorlesung)
2. **Kowarschick (2002a)**: Wolfgang Kowarschick; Multimedia-Programmierung – Objektorientierte Grundlagen; Hrsg.: Michael Lutz und Christian Märtin; Reihe: Informatik interaktiv; Verlag: Fachbuchverlag Leipzig im Carl Hanser Verlag; ISBN: 3446217002; 2002; Quellengüte: 5 (Buch)
3. **Kowarschick (2002)**: Wolfgang Kowarschick; Vorlesung „Multimedia Softwareentwicklung II“ – Wintersemester 2001/2002; Hochschule: Fachhochschule Augsburg; Adresse: Augsburg; Web-Link; 2002; Quellengüte: 4 (Skript)

4 Siehe auch

Konstruktor

Kategorien:
Objektorientierte Programmierung
Glossar
Kapitel:Multimedia-Programmierung