Programmieren C++: Copy-Konstruktor, `clone`-Trick

Klaus Kusche

Wir erweitern unser Beispiel aus der vorigen Übung:

Zum schnelleren Eingeben mehrerer neuer, gleicher Gegenstände wollen wir in unserem Hauptprogramm eine zusätzliche Auswahlmöglichkeit anbieten, die einfach den *zuletzt eingegebenen Artikel nochmals in die Liste einträgt* (natürlich mit *neuer Inventarnummer*, aber gleichem Namen, gleichem Wert, gleicher RAM- und Disk-Ausstattung usw.).

Als Zusatz könntest du noch einen weiteren Auswahlpunkt machen, der einen beliebigen (mittels Inventarnummer auszuwählenden) Gegenstand statt dem letzten kopiert.

Die Herausforderung dabei ist, dass wir nur *einen einzigen Menüpunkt für alle Arten von Gegenständen* wollen, keine getrennten Punkte für "Kopiere den letzten Computer", "Kopiere das letzte Zubehör" usw.. Dieser Menüpunkt soll auch *ohne Fallunterscheidung* zwischen den einzelnen Klassen auskommen.

Das funktioniert in **zwei Schritten**:

1.) Copy-Konstruktor

Gib jeder Klasse einen Copy-Konstruktor:

- Der Copy-Konstruktor in der **Basisklasse** *kopiert den Namen* aus dem Original, aber *vergibt eine neue Inventarnummer*.
- Die Copy-Konstruktoren in den **abgeleiteten Klassen** *kopieren zuerst das Basis-Objekt* (wie?) und setzen dann die in der abgeleiteten Klasse *dazugekommenen Member auf dieselben Werte wie im Original*.
- Der Copy-Konstruktor von **`Comp`** muss so wie der normale **`Comp`**-Konstruktor auch die *statischen Variablen* (Anzahl der Computer, Summe von RAM und Disk) *aktualisieren*.

2.) `clone`-Methode

Das Problem ist, dass **Konstruktoren nicht virtuell** sind: In unserem Hauptprogramm haben wir einen Pointer auf **`Inventar`**-Objekte, der in Wirklichkeit auf ein Objekt einer davon abgeleiteter Klassen (**`Comp`** oder **`Zubeh`**) zeigt.

Wir können nicht einfach einen **`Inventar`**-Copy-Konstruktor für dieses Objekt aufrufen: Das würde nur ein **`Inventar`**-Objekt erzeugen (was gar nicht geht, weil **`Inventar`** ja abstrakt ist) und die zusätzliche Information von **`Comp`** oder **`Zubeh`** verlieren.

Wir müssten in einer Fallunterscheidung ausdrücklich den Copy-Konstruktor von **`Comp`** oder **`Zubeh`** aufrufen, aber das ist umständlich: Erstens haben wir noch nicht gelernt, wie man herausfindet, zu welcher Klasse das Objekt gehört, auf das der Pointer zeigt, und zweitens müsste man den Code jedesmal ändern, wenn eine neue abgeleitete Klasse dazukommt.

Für dieses Problem wurde der "**clone**-Trick" erfunden:

- Wir definieren in **Inventar** eine neue, *rein virtuelle Methode*, die keine Parameter hat und einen *Pointer auf eine neue Kopie des eigenen Objektes zurückliefert*.

  Eine solche Methode wird meist **clone** genannt.

- In jeder *abgeleiteten* Klasse wird **clone** implementiert, indem man einfach den *Copy-Konstruktor der eigenen Klasse* mit **new** aufruft (mit dem *eigenen Objekt* als Argument) und den von **new** gelieferten Pointer returniert.

Unser Hauptprogramm ruft dann keinen Copy-Konstruktor auf, sondern verwendet **clone**, um einen beliebigen Gegenstand zu kopieren. Zeigt der Pointer, für den **clone** aufgerufen wird, beispielsweise auf ein **Comp**-Objekt, so ruft der **virtual**-Mechanismus das **clone** aus der Klasse **Comp** auf, und dieses kopiert das Objekt durch Aufruf des Copy-Konstruktors von **Comp**.