Methodenkonzept: Sichtbarkeit von Variablen

Gate's Gesetz: Die einzig wichtige Information in einer Hierarchie ist: Wer darf was wissen?

Methoden lösen ein spezifisches Problem in einer derartig allgemeinen Weise, so dass diese Problemlösung in allen Programmen ohne weitere Anpassung an die spezielle Situation eingebunden werden kann.

Sie deklarieren daher vor Beginn des Hauptprogramms eine Methode (in Java kann sie auch danach folgen) mit einem geeigneten Namen und rufen sie von verschiedenen Stellen des Hauptprogramm aus auf - mit der gleichzeitigen Übergabe der Liste der aktuellen Werte, die bearbeitet werden sollen.

Das Besondere von Unterprogrammen bzw. Methoden in höheren Programmiersprachen liegt - neben der "vollautomatischen" Übergabe und Rückgabe der Parameter - in der völligen Abschottung des Unterprogramms gegenüber dem Hauptprogramm und weiteren einzelnen Unterprogrammen.

Dem Hauptprogramm ist nur der Name der Methode zusammen mit einer Liste der Parameter, die von der Methode als "Eingabe" erwartet werden und - soweit vorhanden - der Typ des Rückgabewertes, der wieder an das Hauptprogramm zurückgeliefert wird, bekannt. Das Hauptprogramm weiß aber nicht, wie und mit welchen Variablen das betreffende Problem innerhalb der Methode gelöst wird.

Verschiedene Methoden dürfen ohne weiteres intern dieselben Variablennamen benutzen, ohne dass es zu Konflikten kommt. Derartige Variablen, die nur innerhalb einer Methode Gültigkeit besitzen, heißen lokale Variablen.

Tauchen in einem Hauptprogramm und in einer Methode derselbe Variablenname auf, so wird die Hauptprogramm-Variable für die Dauer der Ausführung des Unterprogramms eingefroren, sie ist währenddessen nicht sichtbar.

In einem Unterprogramm, in dessen Variablenliste (bestehend aus Parameterliste und lokalen Variablen) eine für die gesamte Klasse deklarierte Variable nicht vorkommt, lässt sich diese Klassenvariable dagegen abfragen (und auch verändern - in der Regel schlechter Programmierstil!). Eine derartige Klassenvariable bezeichnet man auch als globale Variable.

Dieses Abschottungsprinzip durch lokale Variablen - die nur in dem Block bekannt sind, in dem sie deklariert sind, und sonst nirgends - und globale Variablen - die innerhalb der gesamten Klasse bekannt sind, aber durch gleichlautende Variablen verdeckt werden - ist ein wesentliches Kennzeichen einer höheren Programmiersprache.

Werden mehrere Entwickler gleichzeitig auf verschiedene Teile eines Programms angesetzt, so ist jeder einzelne völlig frei in der Art, wie das Problem gelöst wird, und in der Wahl der verwendeten Variablennamen. Dem Gesamtteam müssen nur die Namen der Methoden mit Eingabeliste und Ausgabewert bekannt sein. Dann kann jeder die Prozeduren des anderen beliebig innerhalb seiner eigenen Problemlösung benutzen.

Häufig benutzte und immer wieder benötigte Routinen können in Form einer Methoden-Bibliothek gespeichert werden.

Gültigkeitsbereich und Lebensdauer einer Variablen

Als Gültigkeitsbereich einer Variablen wird der Teil eines Programms bezeichnet, in dem die Variable genutzt werden kann, d.h. in dem der Name der Variablen verwendet werden darf. Hierbei macht es einen Unterschied, ob die Variablendefinition als Anweisung oder als Definition eingesetzt wird.

Variablendefinition als Anweisung

Der Gültigkeitsbereich einer in einer Anweisung definierten Variablen - auch lokale Variable genannt - erstreckt sich von der der Variablendefinition folgenden Anweisung bis zum Ende desselben Blockes und umschließt alle inneren Blöcke. Im Gültigkeitsbereich einer lokalen Variablen darf keine weitere Variable mit demselben Namen definiert werden.

Da Methodenrümpfe immer einen abgeschlossenen Block bilden, können jederzeit Variablen mit dem gleichen Namen in

verschiedenen Methoden definiert werden. Was zwar zulässig ist, jedoch zu Missverständnissen führen kann und daher vermieden werden sollte, ist die Verwendung ein und desselben Namens sowohl für eine Methode als auch für eine (lokale) Variable.

Variablendefinition als Definition

Wird eine Variable im Definitionsteil eines Programms definiert, also neben den Prozeduren und Funktionen, dann gilt folgende Vereinbarung: Der Gültigkeitsbereich einer im Definitionsteil definierten Variablen - auch globale Variable genannt - umfasst das gesamte Programm mit Ausnahme des Initialisierungsausdrucks der Variablen selbst sowie der Initialisierungsausdrücke von anderen globalen Variablen, die erst später definiert werden. Im gesamten Programm dürfen nicht mehrere globale Variablen mit demselben Namen definiert werden.

Es ist jedoch erlaubt, lokale Variablen mit dem Namen einer globalen Variablen zu versehen. Insbesondere ist es definitionsgemäß auch möglich, globale Variablen zu benutzen, ohne sie vorher definiert zu haben; die Definition erfolgt dann weiter unten im Programmcode. Vermeiden Sie solche Fälle jedoch, da derartige Programme schwer verständlich sind. Gewöhnen Sie sich an, auch globale Variablen vor ihrer ersten Benutzung zu definieren.

Wird eine lokale Variable im Gültigkeitsbereich einer gleichnamigen globalen Variablen definiert, dann "überdeckt" der Gültigkeitsbereich der lokalen Variablen den Gültigkeitsbereich der globalen Variablen, d.h. tritt im Gültigkeitsbereich der lokalen Variablen der zweifach vergebene Name auf, dann ist damit die lokale Variable gemeint. Es ist nicht möglich, im Gültigkeitsbereich der lokalen Variablen auf eine globale Variable mit demselben Namen zuzugreifen.

Die Benutzung globaler Variablen gilt in der imperativen Programmierung als "unsauber", da die Gefahr, dass sich dadurch Fehler in ein Programm einschleichen, steigt. Programme sind übersichtlicher, wenn Prozeduren bzw. Funktionen nur auf solche Variablen zugreifen, die lokal in der Prozedur bzw. Funktion definiert sind ("Lokalitätsprinzip"). Benutzen Sie das Parameterkonzept, mit dessen Hilfe die Nutzung globaler Variablen weitgehend vermieden werden kann.

Lebensdauer einer Variablen

Die Lebensdauer einer globalen Variablen umfasst die gesamte Ausführungszeit eines Programms. Die Lebensdauer einer lokalen Variablen beginnt bei ihrer Definition und endet nach der vollständigen Abarbeitung des Blocks, in dem sie definiert wurde.

Das heißt, wird zwischen der Definition einer lokalen Variablen und dem Ende des Blocks eine Prozedur oder Funktion aufgerufen, so ist die Variable auch während der Ausführung der Funktion "lebendig", aber da die Variable in der Prozedur bzw. Funktion nicht gültig ist, ist sie dort nicht zugreifbar.

Aufgabe:

Eure Gruppe soll den Unterschied zwischen lokalen und globalen Variablen vorstellen. In unserer Programmierumgebung können lokale und globale (Kasten unten links) angelegt werden. Am besten kann man den Unterschied mit Hilfe eines Programms mit einem Unterprogramm (hier: Neues Struktogramm) und dem Debugger vorstellen. Das neue Struktogramm kann man als Unterprogramm (auch: Methode/Prozedur) innerhalb des main-Struktogramms mit Hilfe des Namens aufgerufen werden.

Auf folgende Stichwörter sollt Ihr insbesondere eingehen: lokale und globale Variable, Lebensdauer, Namenskonflikt

Erstellt dazu einzelne Unterprogramme.

Zeigt der Lerngruppe auch wie man die globalen und lokalen Variablen in unserer Programmierumgebung anlegt.