Aufgabe 5: Typprüfungen

Erklären Sie mit eigenen Worten den Unterschied zwischen Namensäquivalenz und Strukturäquivalenz bei Typprüfungen. Warum wird in Java (und auch in MicroJava) zwar in der Regel Namensäquivalenz verwendet, für Arrays jedoch Strukturäquivalenz?

Lösung

Bei Namensäquivalenz werden zwei Typen als gleich betrachtet, wenn sie durch denselben Typnamen beschrieben werden. Im Compiler erfordert das lediglich eine Prüfung, ob sie durch denselben Strukturknoten repräsentiert werden. Die Typprüfung läuft also auf einen Vergleich hinaus, ob zwei Typzeiger auf denselben Knoten zeigen (`a.type == b.type`), was einfach und effizient ist.

Bei Strukturäquivalenz werden zwei Typen als gleich betrachtet, wenn sie die gleiche Struktur haben. Bei Arrays erfordert das eine Prüfung ob sie denselben Elementtyp haben. Bei Klassen muss geprüft werden, ob sie die gleiche Anzahl von Feldern haben und ob die Typen der entsprechenden Felder strukturäquivalent sind, was auf einen rekursiven Vergleich hinausläuft. Obwohl Strukturäquivalenz mächtiger ist als Namensäquivalenz, ist sie aufwändiger und erfordert einen komplizierteren und auch ineffizienteren Vergleich strukturierter Typen.

Auf Grund der Einfachheit (und auch besseren Verständlichkeit) verwenden die meisten modernen Programmiersprachen daher Namensäquivalenz. In Java (und auch in MicroJava) wird allerdings bei Arrays Strukturäquivalenz verwendet, weil Arraytypen in Java nicht durch einen Typnamen (z.B. `PersonArray`) beschrieben werden können, sondern nur durch eine Struktur (z.B. `Person[]`).