Objektorientierte Programmierung WS 2013/2014 - Lösung 2

Prof. Dr. U. Kastens
Institut für Informatik, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn

Lösung zu Aufgabe 1

Klassen stellen Methoden bereit um Objekte zu instanziieren, Initialwerte und Methodenrümpfe zu setzen. Typen beschreiben eine Menge von Operationen, die auf einem Objekt ausgeführt werden können. Operationen sind durch ihre Signatur bestimmt (Parameter, Rückgabetype u. ggf. Name). Sind die Mengen von Signaturen zweier Typen gleich, so haben sie denselben Typ, auch wenn ihre Namen unterschiedlich sind.

Subclassing bedeutet, dass eine Klasse von einer anderen Klasse erbt. Sie erbt dabei ihre Methoden und Variablen und kann diese überschreiben oder überladen sowie neue Methoden/Variablen hinzufügen. In manchen Sprachen können Methoden/Variablen der Oberklasse modifiziert werden. Subclassing betrachtet das die Ebene der Implementierung.

Subtyping beschreibt eine Relation zwischen Typen (nicht zwischen Klassen, wie beim subclassing). Ein Untertyp kann an Stellen eingesetzt werden, an denen ein Obertyp verlangt ist. Subtyping muss sich nicht zwingend an Typen von Klassen orientieren, auch subtyping bei Verbunden, Variablen oder Funktionen ist möglich.

Subclassing muss nicht notwendigerweise bedeuten, dass auch eine subtyping Relation erfüllt ist.

Lösung zu Aufgabe 2

Wenn nur Teile der Oberklasse geerbt werden, so gilt die subtying Relation nicht mehr. Es kann Aufrufstellen geben, an denen gerade eine nicht geerbte Methode aufgerufen wird.

a) Wir übernehmen das `undefine`-Konstrukt in die Java-Syntax:

```
class Vater { void m(); }
cass Sohn extends Vater  {undefine m();}

Vater v = new Sohn();
v.m();  // Fehler: Methode nicht definiert, Unterklasse aber keine Untertyp
```

b) Die Eiffel-Forderung, dass eine Klasse, die mindestens eine Methode mit `undefine` kennzeichnet, als abstract angesehen wird, verhindert, dass `Sohn`-Objekte erzeugt werden. Es kann daher nicht zum Aufruf undefinierter Methoden kommen.

Lösung zu Aufgabe 3

a) Zwei Typen `A` und `B` stehen in einer Untertyprelation, etwa `A <: B`. Durch Typkonstruktoren werden aus `A` und `B` komplexere Typen konstruiert, z.B. Strukturen mit Komponenten vom Typ `A` und `B`, Arrays mit Elementen vom Typ `A` und `B` oder Funktionen mit Parametern vom Typ `A` und `B`. Stehen die so konstruierten Typen in der gleichen Untertyprelation wie die Ausgangstypen, spricht man von Kovarianz, stehen sie in der entgegengesetzten Relation, spricht man von Kontravarianz.

   Beispiele:

   Ein nicht änderbares Array von Typ `A` ist ein Untertyp eines nicht änderbaren Arrays von Typ `B` (Kovarianz).

   Eine Funktion mit Parametertyp `B` ist ein Untertyp einer Funktion mit Parametertyp `A` (Kontravarianz).

b) Für die überschreibende Funktionen gilt: kovariante Rückgabetypen und kontravariante Parametertypen. Damit `(B -> D) <: (A -> C)` ist, muss also gelten: `A <: B` und `D <: C`

Lösung zu Aufgabe 4

Ballspiel mit `BoundedBall`: `blatt2/boundedball`

Lösung zu Aufgabe 5

Ballspiel mit `MultiballWorld`: `blatt2/multiballworld`