II.4.4 Generische Datentypen

Dienstag, 22. Dezember 2020 16:01

Nachteile der allgemeinen Liste (mit Werten vom Typ Object):

1. Keine Anforderungen an Werte ausdrückbar.
   Abhilfe: Verwende abstr. Klasse oder Interface "Vergleichbar" anstelle von Object.
2. Verschiedene Objekt-Typen in der gleichen Liste.
3. Wenn man Werte v. Typ Bruch aus der Liste ausliest sind es zunächst Werte der Oberklasse (Object oder Vergleichbar). Sie müssen erst explizit zur U-Klasse "gecasted" werden, bevor man mit ihnen weiterarbeiten kann.

Lösung für 2+3: generische Typen

Eine generische Klasse wie Element<T> definiert viele Typen:

Element<Bruch>
Element<Wort>
⋮

Typvariable T kann durch bel. Datentypen instantiiert werden.
=> stellt sicher, dass in einer Liste vom Typ Liste<Bruch> nur Werte vom Typ Bruch sind.

Polymorphismus

- ad hoc Polymorphismus:
  Verschiede Implementierungen einer Methode.
  Es hängt vom Typ der Objekte ab, welche Implementierung ausgeführt wird.
  (typisch für OO-Sprachen)
- parametrischer Polymorphismus:

(typisch für OO-Sprachen)

- parametrischer Polymorphismus:
  Dieselbe Implem. einer Methode kann für Objekte verschiedener Typen ausgeführt werden.
  (typisch für funktionale Sprachen, seit Java 5)

Kurzschreibweise:

```
xs = new Liste<>()
```

↑ möglich, wenn aufgrund des Typs von xs klar ist, wie der Typparameter instantiiert werden muss.

Durch statisches (d.h. zur Compile-Zeit) Typ-Checking ist nun sichergestellt, dass nur Werte des gleichen Typs in der Liste sind.

Auch möglich:

- Klassen mit mehreren Typparametern
- Interfaces mit Typparametern

Generische Typen existieren nur zur Compile-Zeit.
Im Java Bytecode gibt es nur noch raw types, bei denen die Typparameter durch Object ersetzt werden.

Raw Type zu Liste<T> ist Liste. Hier kann man Werte v. Typ Object speichern.

"instanceof" ist nur mit raw types verwendbar.

~~xs instanceof Liste<Bruch>~~ gener. Typ nicht verfügbar zur Laufzeit

xs instanceof Liste

Statische Methoden in gener. Klassen

- existieren nur einmal pro Klasse
  ⇒ können nicht auf den Typparameter einer generischen Klasse zugreifen
- Lösung: statische Methoden können eigene Typparameter haben
  Diese haben nichts mit dem Typparameter der Klasse zu tun.

Kombiniere param. Polymorphismus mit Typhierarchie

Typvar. soll nur mit U-Klassen von best. Oberklasse instantiiert werden.

Bsp: T soll nur mit Vergleichbar oder seinen Unterklassen instantiiert werden.

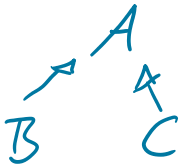
⇒ Lösung: Type Bounds

```
<T extends VergleichBar>
```

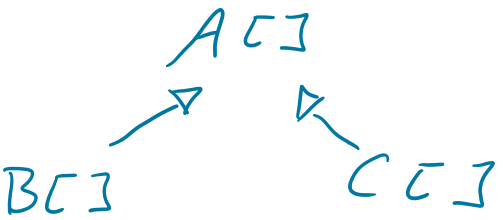
Es ex. noch weitere Type Bounds:

- wild cards: ?
- Einschränkung v. Wild Cards nach oben oder unten in der Typhierarchie.

Typhierarchie bei generischen Klassen



bei Arrays:



Schlechte Idee: So kann man ein C-Objekt

Schlechte Idee: So kann man ein C-Objekt in ein Objekt vom Typ B[] schreiben.

bei Listen: