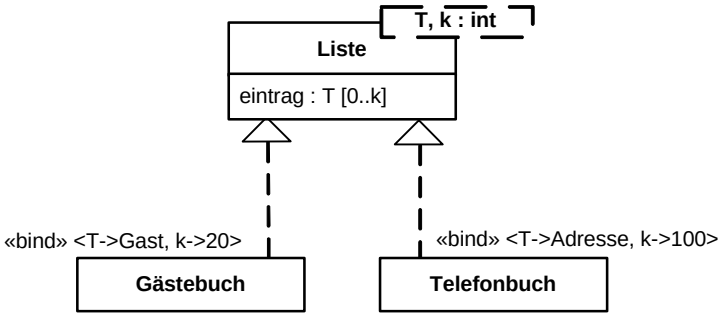
SOPHIST

Codebeispiele für Parametrisierte Klassen

Kapitel 4: Klassendiagramm



Umsetzung in C++

Zur Realisierung des Modells aus dem entsprechenden Beispiel für C# im Buch wird in C++ der parametrisierten Klasse das Schlüsselwort `template` gefolgt von der Liste der Parameter in spitzen Klammern vorangestellt. Parametern, die die Angabe einer Klasse zulassen, wird formal das Wort `class` vorangestellt. Innerhalb der parametrisierten Klasse können die Parameter wie Variablen verwendet werden, die für die entsprechenden Typen oder Werte stehen.

Die Bindung und damit Erzeugung einer konkreten Liste erfolgt durch Definition von Inhalten für definierte Parameter. Auch diese werden in spitzen Klammern dem Namen der parametrisierten Klasse nachgestellt.

```
template <class T, int k>
class Liste {
     T elements[k];
};
...
Liste<Gast,20> Gaestebuch;
Liste<Adresse,100> Telefonbuch;
```

Umsetzung in Java

Java erlaubt ab der Sprachversion 1.5 die Nutzung parametrisierter Klassen. Allerdings sind als Parameter ausschließlich Typen (d.h. Klassen), jedoch keine konkreten Werte zugelassen. Daher kann bei Umsetzung der Abbildung 3.20 der Parameter `k` nicht als Parameter berücksichtigt werden.

Stattdessen wird er als Übergabeparameter des Konstruktors der generischen Klasse Liste definiert, um so die gewünschte Dimensionierung des Feldes zur Aufnahme der `Adress`-Ausprägungen zu erreichen. Die gewählte Umsetzung entspricht daher nicht vollständig der im Klassendiagramm der Abbildung 3.20 formulierten Semantik, bietet aber einen erträglichen Kompromiss.

Zusätzlich verbietet Java die Nutzung der generischen Sprachmechanismen für Arrays. Daher wird nachfolgend die Standardklasse `Vector` verwendet, die einen Array mit dynamischen Grenzen anbietet. Die im Konstruktor übergebene Zahl für `k` definiert daher lediglich Speicherplatz, der

mindestens `k` Elemente aufnehmen kann, verhindert jedoch nicht das Überschreiten dieser Grenze.

Die Parameter einer Klasse werden dieser in spitzen Klammern nachgestellt und müssen nicht zusätzlich typisiert werden. Innerhalb der Klasse kann auf die Parameter zugegriffen werden. Da es sich bei `Vector` selbst um eine parametrisierte Klasse handelt, erfolgt auch dieser Zugriff durch Parameterangabe in spitzen Klammern.

Zur Verwendung ist jeder Parameter an eine existierende Klasse zu binden. Die Bindung erfolgt durch Definition einer Variable und Erzeugung eines neuen Objekts vom Typ der parametrisierten Klasse unter Angabe des gebundenen Typs in spitzen Klammen. Die Angabe wird dem Namen der parametrisierten Klasse nachgestellt.

```
class Liste<T> {
     Vector<T> elements;
     public Liste(int k) {
           elements = new Vector<T>(k);
     }
}
...
Liste<Gast> Gästebuch=new Liste<Gast>(20);
Liste<Adresse> Telefonbuch=new Liste<Adresse>(100);
```