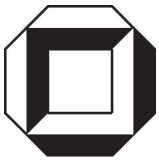

# Universität Karlsruhe (TH)

## Lehrstuhl für Programmierparadigmen


| | | |
|---|---|---|
| Fortgeschr. Objektorientierung SS 2009 | | http://pp.info.uni-karlsruhe.de/ |
| Dozent: | Prof. Dr.-Ing. G. Snelting | snelting@ipd.info.uni-karlsruhe.de |
| Übungsleiter: | Andreas Lochbihler | lochbihl@ipd.info.uni-karlsruhe.de |
| | Dennis Giffhorn | giffhorn@ipd.info.uni-karlsruhe.de |


Blatt 8 | Ausgabe: 10.06.2009 | Besprechung: 17.06.2009

1. **Bounded Polymorphism**

   Geben Sie je ein Beispiel an, das zeigt, dass es sinnvoll sein kann,

   (a) zwei Typschranken für einen Parameter

   (b) zwei verschiedene Typparameter mit verschiedenen Schranken

   haben zu können.

2. **Generische Klassen und Vererbung**

   Verwendet man generische Klassen für Container, möchte man natürlich auch generische Klassen für Iteratoren verwenden:

   (a) Erstellen Sie die Signatur einer generischen Klasse *Node*<*A*>, in der jeder Knoten eine Liste von <*A*> enthält. Weiterhin soll jeder Knoten eine Liste von Vorgänger und Nachfolge-Knoten enthalten, auf die nur über Iteratoren zugegriffen werden darf.

   (b) Erstellen Sie die Signatur einer Klasse *SpecialNode*, die von *Node*<`Integer`> erbt und weiterhin die Methode `void` *foo*() und `void` *bar*() beinhaltet.

   (c) Schreiben Sie den Code für die Methode *bar*. Sie soll für alle Nachfolger des aktuellen Knotens die Methode *foo* aufrufen. Welches Problem entsteht dabei? Sehen Sie eine Möglichkeit, es zu umgehen?

3. **Probleme mit Type Erasure bei Generics in Java**

(a) **Typcasts**: Betrachten Sie folgendes Java-Codefragment:

```
1  public interface Board extends Cloneable {
2      ...
3      Board clone();
4      ...
5  }
6
7  public class SudokuBoardSolver implements SudokuSolver {
8      ...
9      private <B extends Board> LinkedList<B>
10     backtrack(B board) {
11         ...
12         B newBoard = (B) board.clone();
13         ...
14     }
15     ...
16 }
```

Der Java Compiler liefert standardmäßig folgende Bemerkung:

```
Note: sudoku/SudokuBoardSolver.java uses unchecked or unsafe operations.
```

Bei Verwendung des Parameters `-Xlint:unchecked` erhält man folgende Meldung:

```
sudoku/SudokuBoardSolver.java:86: warning:
  [unchecked] unchecked cast
found   : sudoku.Board
required: B
        B newBoard = (B) board.clone();

1 warning
```

i. Wieso tritt diese Meldung auf? Überlegen Sie sich, was intern in der JVM passiert.

ii. Kann man dies verhindern? Wenn ja, wie, wenn nein, warum nicht.

(b) **Implementierung generischer Interfaces:** Worüber beschwert sich der Java-Compiler bei folgender Klasse? Warum?

```
1  class B implements Comparable<B>, Comparable<String> {
2      public int compareTo(B b) { ... }
3      public int compareTo(String s) { ... }
4  }
```

(c) **Generische Arrays:** Der Compiler weigert sich in folgendem Beispiel, Code für Array-Erzeugungen mit Elementen generischen Typs zu erzeugen. Warum? Wie kann man trotzdem generische Arrays erzeugen?

```
1  class C<T> {
2      T[] createArray(int size) {
3          return new T[size];
4      }
5  }
```