Java Generics - Wiederholung

Dozent: Prof. Dr. Michael Eichberg
Kontakt: michael.eichberg@dhbw.de, Raum 149B
Version: 1.1

1. Datenstrukturen

Kontrollfragen

1.1. LiFo oder FiFo?

1

1. Stack
2. Queue
3. List

1.2. Welches sind die wesentlichen Methoden eines Stacks?

2

3

2. Generics

Kontrollfragen

2.1. Welchem Zweck dienen Generics?

2.2. Was ist der Diamond Operator?

2.3. Wie sieht der Code aus, wenn Sie eine Liste von Ganzzahlen erstellen wollen? D. h. welche Initialisierungen sind korrekt?

```
ArrayList<Integer>   list = new ArrayList<int>();
ArrayList<Integer>   list = new ArrayList<>();
var                  list = new ArrayList<int>();
var                  list = new ArrayList<Integer>();
ArrayList<int>       list = new ArrayList<>();
List<int>            list = new ArrayList<>();
List<Integer>        list = new ArrayList<>();
```

2.4. Welche Zeilen sind korrekt bzw. falsch?

```
List<Integer> list = new ArrayList<>();
list.add(1);
list.add("2");
```

2.5. Was sind Raw-Types?

2.6. Was ist der Unterschied zwischen `List<?>` und `List<Object>`?

2.7. Was ist der Unterschied zwischen `List<? extends Number>` und `List<Number>`?

2.8. Welche Zuweisungen sind gültig?

```
List<? extends Number>  list1 = new ArrayList<Integer>();
List<? extends Number>  list2 = new ArrayList<Double>();
List<? extends Number>  list3 = new ArrayList<String>();
ArrayList<Object>       list4 = new ArrayList<Integer>();
List<Object>            list5 = new ArrayList<Object>();
ArrayList<Integer>      list6 = new ArrayList<Object>();
List<Integer>           list7 = new ArrayList<Object>();
```

2.9. Was ist damit gemeint, dass Generics *invariant* sind?

2.10. Was ist Auto-Boxing?

2.11. Sie haben ein `Set<Integer>`, warum ist es unproblematisch, dass die Methode `contains` die Signatur `boolean contains(Object o)` hat (und nicht `boolean contains(T t)` wenn `T` der generische Typparameter von `Set` sei)?

2.12. Erklären Sie die wesentlichen Elemente des Iterator Patterns.

2.13. Wie ist der Zusammenhang zwischen den Interfaces `Iterable` und `Iterator`?

13

2.14. Erklären Sie die folgenden Methodensignaturen

14

1. `public static <T> void swap(List<T> list, int i, int j)`
2. `public static <T extends Comparable<T>> void sort(List<T> list)`
3. `public static void sort(List<? extends Comparable> list)`

   Ist diese Methode sinnvoll bzw. ist dies Methode vergleichbar mit der vorherigen Methode?
4. `public static <T> void copy(List<? super T> destination, List<? extends T> source)`