Objektorientierte Programmierung WS 2013/2014 - Aufgabenblatt 3

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

Ausgabe: 12.11.2013

Aufgabe 1 (Fragen zur Generik in Java)

Das Java-Tutorial `http://docs.oracle.com/javase/tutorial/java/generics` zum Thema Generik erläutert den aktuellen Stand der Sprachdefinition. Nutzen Sie es, um folgende drei Fragen zu beantworten:

a) Ist `Box<MyInteger>` ein Untertyp von `Box<MyNumber>`?

```
class Box<T> { // ... some implementation
}
class MyNumber { ...
}
class MyInteger extends MyNumber { ...
}
```

b) In C++ können neben Klassen auch Funktionen parametrisiert werden.

```
template <class T>
  T max(T a, T b) {
     return a > b ? a : b;
}
cout << max<double>(3.0, 5.0)
```

Ist dies in der aktuellen Java-Version auch möglich (im Fall von Java mit Methoden statt Funktionen)? Darf der Typ-Parameter wie in C++ auch ein Grundtyp sein?

c) Warum lässt sich folgende Java-Klasse nicht übersetzen?

```
class Try<T> {
   private T val;
   public Try() { // constructor
      val = new T();
   }
}
```

Aufgabe 2 (Generik in Java)

Schreiben Sie eine generische Klasse `NumberedObject<T>`, mit der Objekte von beliebigem Typ `T` zusammen mit einer automatisch generierten Nummerierung (ab 1 aufsteigend) gespeichert werden.

Für folgende Anwendung

```
public class NumberedThings {
   public static void main(String[] args) {
      NumberedObject<String> no1 = new  NumberedObject<String>("Java");
      NumberedObject<String> no2 = new  NumberedObject<String>("C++");
      NumberedObject<Integer> no3 = new  NumberedObject<Integer>(42);
      System.out.println(no1 + "\n" + no2 + "\n" + no3);
   }
}
```

soll die Ausgabe

```
1: Java
2: C++
3: 42
```

erzeugt werden.

Aufgabe 3 (Generik in Java mit beschränkten Typparametern)

Die generische Klasse `Adder` verwendet einen beschränkten Typparameter:

```
class Adder<T extends Adding<T>> {
   private T val;

   public void init(T v) {
     val = v;
   }
   public void add(T v) {
     val = val.add(v);
   }
   public T getTotal() {
      return val;
   }
}
```

Ein Typparameter kann auch mehrere Beschränkungen haben. Sie werden mit & verknüpft. Dies zeigt das folgende Beispiel:

```
class A { /* ... */ }
interface B { /* ... */ }
interface C { /* ... */ }

class D <T extends A & B & C> { /* ... */ }
```

a) Wozu dient die Beschränkung des Typparameters und warum ist sie hier in unserer Anwendung notwendig?
b) Geben Sie `Adding<T>` an.
c) Entwickeln Sie die Klasse `Preis`, so dass die Anwendung

```
public class AdderProg {
   public static void main(String[] args) {
      Adder<Preis> ap = new Adder<Preis>();
      ap.init(new Preis(0,0));
      ap.add(new Preis(7,99));
      ap.add(new Preis(2,99));
      System.out.println(ap.getTotal());
   }
}
```

die Ausgabe `10,98` erzeugt.

Aufgabe 4 (Generik in C++)

Vervollständigen Sie das C++ Programm in der Datei `stack.cc`, das einen generischen Stack implementiert. Überprüfen Sie Unter- und Überlauf des Kellers.

Übersetzen Sie Ihr C++-Programm mit

```
g++ stack.cc
```

Testen Sie Ihr Programm durch den Aufruf von `./a.out`.