Objektorientierte Programmierung WS 2013/2014 - Lösung 3

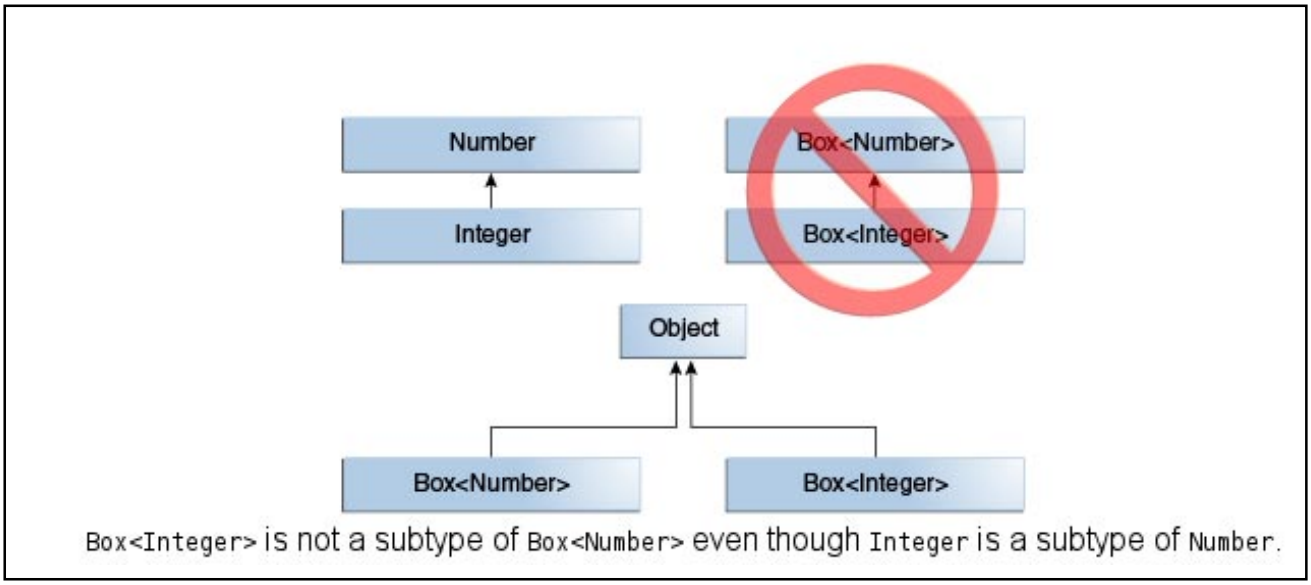
Prof. Dr. U. Kastens
Institut für Informatik, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn

Lösung zu Aufgabe 1

Antworten aus dem Java-Tutorial `http://docs.oracle.com/javase/tutorial/java/generics` zum Thema Generik:

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

Nein.



b)

Können auch Funktionen generische Typparameter haben (ohne dass die umgebende Klasse solche hat)?

Ja, allerdings dürfen sie -wie bei generischen Klassen auch- nicht mit Grundtypen instanziert werden.

The `Util` class includes a generic method, `compare`, which compares two `Pair` objects:

```
public class Util {
    // Generic static method
    public static <K, V> boolean compare(Pair<K, V> p1, Pair<K, V> p2) {
        return p1.getKey().equals(p2.getKey()) &&
               p1.getValue().equals(p2.getValue());
    }
}
```

c)

Die Java-Klasse

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

lässt sich nicht übersetzen, da es nicht erlaubt ist, Instanzen des generischen Typparameters zu erzeugen:

Cannot Create Instances of Type Parameters

You cannot create an instance of a type parameter. For example, the following code causes a compile-time error:

```
public static <E> void append(List<E> list) {
    E elem = new E();  // compile-time error
    list.add(elem);
}
```

Lösung zu Aufgabe 2

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

```
class NumberedObject<T> {
   static int count=1;
   private int numb;
   private T obj;

   public NumberedObject(T v) {
      obj = v;
      numb=count++;
   }

   public String toString() {
      return numb + ": " + obj;
   }
}
```

Lösung zu Aufgabe 3

a) Die Beschränkung des Typparameters stellt sicher, dass der Typ, der als Parameter verwendet wird, bestimmte Anforderungen erfüllt. Im vorliegenden Fall muss sichergestellt werden, dass der Typparameter `T` mit einer Klasse instanziert wird, die eine Methode der Form `T add(T t)` bereitstellt.

b) Wir wählen ein Interface, ebenso wäre eine abstrakte Klasse möglich:

```
interface Adding<T> {
   T add(T v);
}
```

c) Die Klasse `Preis`:

```
class Preis implements Adding<Preis>{
   private int e,c;
   public Preis(int e, int c) {
      this.e = e;
      this.c = c;
   }
   public Preis add(Preis p) {
     Preis res = new Preis(e,c);
     res.c += p.c;
     if (res.c >= 100) {
        res.c -= 100;
        res.e += p.e + 1;
     }
     else res.e += p.e;
     return res;
   }
   public String toString() {
      return e + "," + c;
   }
}
```

Lösung zu Aufgabe 4

Zur Vervollständigung mussten nur die Methoden `push()` und `pop()` ergänzt werden:

```
void pop(){
   assert(!empty());
   --numberofelems;
}

void push(T *x) {
   assert(!full());
   array[numberofelems++] = x;
}
```

Das vollständige C++ Programm finden Sie in der Datei `stackloes.cc`.