Dr. Andreas Abel
Dr. Ulrich Schöpp
Dr. Steffen Jost

Ludwig-Maximilians-Universität München
Institut für Informatik
24. Januar 2011

13. Musterlösung zur Vorlesung

Einführung in die Informatik: Programmierung und Software-Entwicklung

Aufgabe 13-1 ***Generisches Mapping*** **(4 Punkte)** (Geben Sie alle `.java`-Dateien Ihrer Lösung ab)
Gegeben sind die folgenden zwei Interfaces:

```
public interface Function<A,B> {
  public B apply(A x);
}

public interface List<A> {
  /**
    * @param x   Element, welches zum neuen Kopf der Liste wird.
    */
  public void addFirst(A x);

  /**
    * Entfernt den Kopf der Liste.
    * @return Ehemaliges Kopf-Element
    */
  public A removeFirst();

  /**
    * Generiert einen String, welcher die Liste darstellt. Dabei werden die
    * Elemente der Liste werden durch die von Object geerbten Methode
    * toString dargestellt.
    *
    * @return String, welcher die Liste repräsentiert.
    */
  public String toString();

  public <B> List<B> map(Function<A, B> fun);
}
```

a) Implementieren Sie eine (nicht generische) Klasse `Vorlesen`, welche das angegebene Interface `Function<Integer,String>` implementiert. Dabei soll die Methode `apply` in diesem Falle Zahlen vorlesen, d.h. die Zahl 0 wird auf den String "Null" abgebildet, die Zahl 10 auf den String "Zehn".

Hinweis: Es reicht wenn Ihre Implementation alle Zahlen zwischen 0 und 19 richtig "vorlesen" kann, um die volle Punktzahl zu dieser Teilaufgabe zu erreichen. Anstatt

"Sechzehn" dürfen Sie auch "Sechszehn" schreiben, usw.
Weiterhin dürfen Sie auch das in der Vorlesung nicht behandelte `switch`-statement auf eigene Gefahr verwenden, wenn Sie möchten.

b) Implementieren Sie eine generische Klasse `MyList<A>` welche das Interface `List<A>` implementiert. Sie müssen also eine generische Liste implementieren. Dabei dürfen Sie natürlich *nicht* auf Listen der Java Bibliothek zurückgreifen! Sie können jedoch eigene Implementation zu vorangegangenen Aufgaben, aus den bereitgestellten Musterlösungen oder der Zentralübung wiederverwenden und entsprechend anpassen, um die Aufgabe zu erfüllen.

Die Methode `map` müssen Sie jedoch in jedem Fall komplett neu schreiben. Das Ergebnis des Methodenaufrufs `l.map(f)` soll eine neue Liste sein, welche das Bild der Liste `l` unter der Funktion `f` ist. D.h. die Funktion `f` wird auf jedes Element der Liste `l` angewendet, und die Ergebnisse werden wieder in einer Liste zusammengefasst. Dabei soll die Reihenfolge der Elemente beibehalten werden und die ursprüngliche Liste erhalten bleiben!

Beispiel: Wenn `vorlesen` ein Objekt des Typs `Function<Integer,String>` ist (also Ihre Lösung aus Teil a), und `nummern` auf eine Liste mit den Zahlen $11, 1, 14, 19$ zeigt, dann soll das Ergebnis des Methodenaufrufs `nummern.map(vorlesen)` eine neue Liste sein, welche die Strings "Elf", "Eins", "Vierzehn" und "Neunzehn" in dieser Reihenfolge enthält.

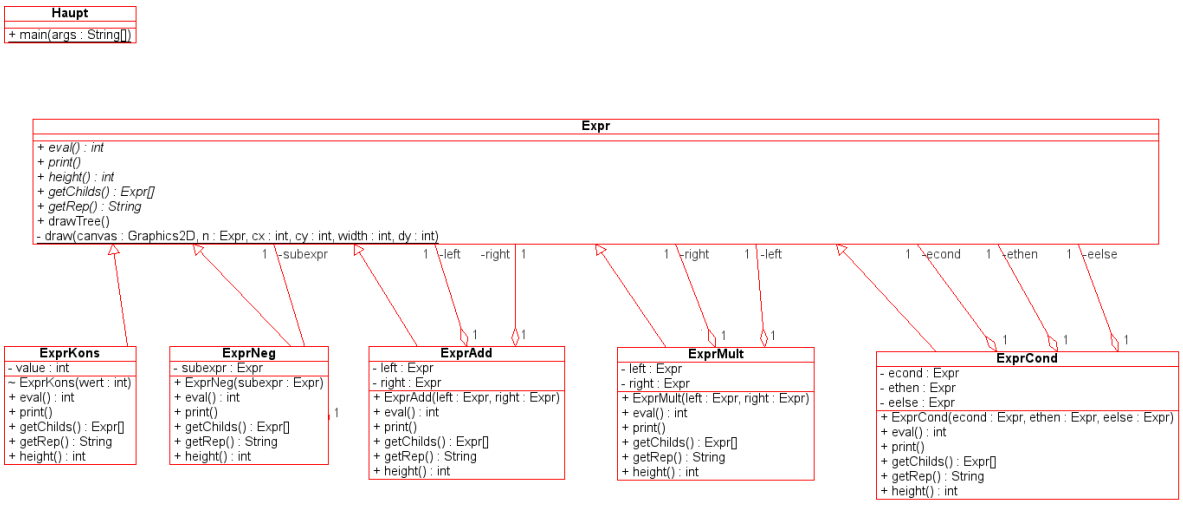
Auf der Homepage der Vorlesung können Sie die beiden Interfaces sowie eine Klasse `Haupt` zum Testen ihrer Lösung herunterladen.

LÖSUNG: Die Lösung ist separat zum download auf Vorlesungshomepage erhältlich.

Aufgabe 13-2 *UML* (4 Punkte) (Abgabeformat: `.txt` oder `.pdf`)
In der Zentralübung am 12.01.11 wurden Komposit-Listen vorgestellt, deren Code auf der Vorlesungshomepage verfügbar ist.

Zeichnen Sie das vollständige UML Diagramm zu den Klassen `Expr`, `ExprAdd`, `ExprCond`, `ExprKons`, `ExprMult`, `ExprNeg` und `Haupt`.

LÖSUNG:



Abgabe: Sie können ihre Lösungen bis Montag, den 31.01.2011, 14 Uhr über UniWorX abgeben. Java Dateien, welche nicht kompilieren, werden nicht beachtet!