Karlsruher Institut für Technologie
Institut für Theoretische Informatik
Prof. Dr. Peter H. Schmitt
Dr. V. Klebanov, Dr. M. Ulbrich, C. Scheben

Formale Systeme, WS 2013/2014

Übungsblatt 9

Dieses Übungsblatt wird in der Übung am 24.01.2014 besprochen.

Wir haben unter Adresse `http://i12www.ira.uka.de/~mulbrich/teaching/jmlcheck` einen Syntaxchecker für JML installiert. Sie können dort für Ihre Lösungen überprüfen, ob sie syntaktisch korrekt sind.

Aufgabe 1

Zur Implementierung einer Personendatenbank wird folgende Java-Klasse verwendet:

```
class Person {
   String name;
   int age;
   boolean isFemale;
   Person father;
   Person mother;
   Person[] children;

   void celebrateBirthday() {
      age ++;
   }

   void becomeParentTo(Person child) {
      children = addToArray(child);
      if(isFemale) {
         child.mother = this;
      } else {
         child.father = this;
      }
   }

   Person[] addToArray(Person child) {
      Person[] result = new Person[children.length + 1];
      System.arraycopy(children, 0, result, 0, children.length);
      result[children.length] = child;
      return result;
   }
}
```

(a) Formulieren Sie für die folgenden Aussagen Klassen-Invarianten in JML:

(i) *Der Vater einer Person ist männlich und älter als die Person selbst.*

(ii) *Alle Kinder einer Person sind echt jünger als die Person selbst.*

(iii) *Jede Person ist Kind ihrer Mutter.*

(iv) *Wenn eine Person weiblich ist, dann ist sie Mutter all ihrer Kinder.*

(b) Formulieren Sie für die Methode `celebrateBirthday()` einen Methoden-Vertrag in JML, der folgendes besagt: *Die Methode darf in jedem Zustand aufgerufen werden. Nach Ausführung der Methode ist der Wert des Feldes* `age` *um genau 1 erhöht, alle anderen Speicherstellen sind unberührt geblieben.*

(c) Beschreiben Sie das Verhalten der Methode `becomeParentTo(Person)` möglichst präzise mit einem JML-Vertrag.

(d) Vervollständigen Sie folgenden Methodenvertrag für die Methode `addToArray(Person)`:

```
/*@ public normal_behaviour
  @  ensures \result.length == ...
  @  ensures (\forall int i; 0 <= i && ...
  @  assignable \nothing;
  @*/
```

(e) Die JML-Semantik gestattet es nicht, dass eine Person ohne Vater/Mutter angelegt wird. Wie muss die Spezifikation geändert werden, wenn Personen ohne Angabe ihrer Eltern angelegt werden können sollen?

Aufgabe 2

(a) Schreiben Sie eine Java-Methode, die die folgende Spezifikation erfüllt: Gegeben ein Feld[1] a von ganzzahligen Werten $a_1, \ldots, a_n$ soll die Methode Index i_0 des ersten Auftretens der Zahl 0 liefern, also den Index i_o ausgeben, so dass

(i) $a_{i_0} = 0$ und

(ii) $a_k \neq 0$ für alle $0 \leq k < i_0$.

Falls keiner der Werte $a_1, \ldots, a_n$ gleich 0 ist, so soll die Methode -1 zurückliefern.

(b) Schreiben Sie einen JML-Methodenvertrag, der diese Spezifikation wiedergibt.

(c) Die Methodenimplementierung wird eine Schleife benutzen. Geben Sie für die Schleife eine möglichst starke Invariante und eine Variante[2] an.

Aufgabe 3

Für eine Anwendung im Bankenbereich wird die folgende Klasse zur Modellierung von Geldbeträgen verwendet:

[1] engl. array

[2] engl. decreases clause

```
class Betrag {
   int euros;
   int cents;

   Betrag(int euros, int cents) {
      this.euros = euros;
      this.cents = cents;
   }

   Betrag add(Betrag b) { ... }
}
```

Es ist das Ziel, die Benutzung dieser Klasse auf *normalisierter Beträge* einzuschränken. Ein Betrag heißt normalisiert, wenn er nicht-negativ ist und der Cent-Anteil im Betrag weniger als einen Euro ausmachen. Die Einschränkung in der Verwendung der Klasse soll mittels Annotatationen in JML formal umgesetzt werden.

(a) Geben Sie eine JML-Klasseninvariante für die Klasse `Betrag` an, die besagt, dass das Betrag-Objekt normalisiert ist.

(b) Geben Sie einen Vertrag für die Method `add` an, der besagt, dass der Betrag, der als Eregebnis zurückgeliefert wird, der Summe des Arguments und des Aufrufempfängers entspricht.

Die Methode darf ein neues Ergebnisobjekt erstellen, darf aber keine existierenden Speicherstellen abändern.

(c) In einem ersten Versuch wird die Methode `add` nun folgendermaßen implementiert:

```
Betrag add(Betrag b) {
   int e = euros + b.euros;
   int c = cents + b.cents;
   return new Betrag(e, c);
}
```

Warum verstößt diese Implementierung gegen die JML-Spezifikation und wie kann das repariert werden?