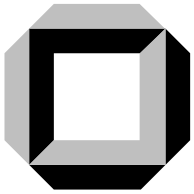

Universität Karlsruhe
Institut für Theoretische Informatik
Prof. Dr. P. H. Schmitt
C. Engel, B. Weiß

07. Januar 2009

Praktikum
Formale Entwicklung objektorientierter Software
Übungsblatt 9

Aufgabe 17

Wichtig: Verwenden Sie bitte ab sofort für alle KeY-Aufgaben die aktuellere KeY-Version 1.3.816-beta anstelle von 1.4 TP. Falls Sie von zuhause aus arbeiten, erhalten Sie diese Version als "nightly build" von der KeY-Homepage. Die Installation auf den Praktikumsaccounts ist bereits aktualisiert.

Passen Sie in der Musterlösung zu Bag.java aus Aufgabe 11 die Methode `removeAll` an, so dass sie folgendermaßen aussieht:

```
/*@ normal_behavior
  @   ensures (\forall int x; 0 <= x && x < n; contents[x] != elt);
  @*/
 void removeAll(int elt) {
     for (int i = 0; i < n; i++) {
         if (contents[i] == elt ) {
             n--;
             contents[i] = contents[n];
             i--;
         }
     }
 }
```

Beweisen Sie mit KeY die "EnsuresPost"-Beweisverplichtung für diese Methode.

Hinweise:

- Bei "EnsuresPost" wird bewiesen, dass die Methode ihre Nachbedingung erfüllt. Sie können den "EnsuresPost"-Beweis starten, indem Sie die Datei mit KeY laden und in dem sich öffnenden "Proof Obligation Browser"-Fenster links die gewünschte Methode (`removeAll`) und rechts die gewünschte Beweisverpflichtung (EnsuresPost) auswählen. Klicken Sie dann auf "Start proof". Es öffnet sich ein weiteres Fenster, in dem Sie aufgefordert werden, einen Methodenvertrag und eine Menge von Klasseninvarianten auszuwählen. Für diese Aufgabe können Sie diesen Dialog einfach unverändert mit "OK" quittieren.
- Ihre Hauptaufgabe ist, für die Schleife eine geeignete Invariante, `assignable`-Klausel und Variante (`decreases`-Klausel) anzugeben. Fügen Sie diese als JML-Annotationen in den Code ein.

- Die Datei enthält Klasseninvarianten, die besagen, dass `contents` nicht `null` ist, und dass sich `n` im Intervall [0, `contents.length`] bewegt. Diese Klasseninvarianten erscheinen auch in der Vorbedingung der Beweisverpflichtung. Sie sind nötig, um den Beweis schließen zu können. Die Aussage über den Wertebereich von `n` müssen Sie zusätzlich auch in die *Schleifen*invariante aufnehmen, da diese Information sonst beim Anwenden der Invariantenregel verlorengeht.

- Mit der richtigen Schleifeninvariante, `assignable`-Klausel und `decreases`-Klausel kann der Beweis vollautomatisch durchgeführt werden. Aktivieren Sie dazu im Reiter "Proof search strategy" die Strategie "Java DL" und nehmen Sie folgende Einstellungen vor:

  - "Loop treatment": "Invariant" (Schleifen sollen mit der Invariantenregel behandelt werden)
  - "Method treatment" - "Expand" (Methoden sollen durch Auspacken des Rumpfes behandelt werden)
  - "Quantifier treatment - No Splits with Progs" (Quantoren sollen heuristisch instantiiert werden, aber der Beweis soll dadurch nicht aufgespaltet werden, solange noch Programme enthalten sind)

  Wenn Sie nun die Strategie starten, sollte der Beweis nach etwa 1000 bis 2000 Regelanwendungen geschlossen werden. Falls nicht, ist höchstwahrscheinlich Ihre Schleifeninvariante, `assignable`-Klausel oder `decreases`-Klausel falsch oder zu ungenau.

- Schauen Sie sich den erfolgreichen Beweis an und verschaffen Sie sich ein grundlegendes Verständnis seiner Struktur. Insbesondere sind die anfängliche Beweisverpflichtung und die von der Invariantenregel erzeugten Beweisknoten interessant.

- Geben Sie sowohl den gespeicherten Beweis als auch die veränderte Datei `Bag.java` ab.

Aufgabe 18

Ersetzen Sie in der Musterlösung zu `Amount.java` aus Aufgabe 11 den Vertrag der Methode `subtract` mit folgendem Vertrag:

```
/*@ public normal_behavior
  @   assignable \nothing;
  @   ensures \result.euros*100+\result.cents
  @             == euros*100+cents - (a.euros*100+a.cents);
  @*/
  public Amount subtract(Amount a){
      ...
```

Ihre Aufgabe ist auch dieses Mal, "EnsuresPost" mit KeY zu verifizieren. Dabei sollen die Aufrufe der Methoden `negate` und `add` *nicht* durch Auspacken der Methodenrümpfe (Regel methodBodyExpand), sondern durch Verwenden von Verträgen für die beiden aufgerufenen Methoden (Regel Use Operation Contract) behandelt werden.

Hinweise:

- Die vorhandenen Verträge für `negate` und `add` sind nicht aussagekräftig genug, um mit ihnen die Korrektheit von `subtract` zu beweisen. Schreiben Sie daher zunächst bessere Verträge für die beiden Methoden. Die beiden Verträge sollen korrekt sein (d.h. von der jeweiligen Methodenimplementierung erfüllt werden), Sie brauchen das aber nicht zu beweisen. Denken Sie daran, geeignete `assignable`-Klauseln zu wählen!

- Um den Beweis durchzuführen, stellen Sie in den Einstellungen der Strategie "Java DL" unter "Method treatment" die Option "Contracts" ein. Der Beweis sollte dann automatisch zugehen, nach insgesamt größenordnungsmäßig 1000 Schritten.
- Geben Sie sowohl den gespeicherten Beweis als auch die veränderte Datei `Amount.java` ab.

Abgabe bis 13.01.

Es braucht pro Gruppe nur *eine* Lösung abgegeben werden.
Die Abgabe der Übungsblätter erfolgt mit dem SVN System. Dazu legen Sie die abzugebenden Dateien im SVN ab und kopieren sie mit SVN in den Unterordner *abgabe/<nr>* wie in Aufgabe 2 auf Blatt 1 beschrieben.
Einige Aufgaben verlangen eine schriftliche Bearbeitung, diese ist dann je nach Komplexität als ASCII, html, ps- oder pdf-Dokument abzugeben. Auf *keinen* Fall im MS Word `doc`-Format.

Praktikums-Webseite: `http://lfm.iti.uni-karlsruhe.de/keyprakt0809.php`

Christian Engel: R. 106, Tel. 608-4338, E-Mail: `engelc@ira.uka.de`
Benjamin Weiß: R. 309, Tel. 608-4324, E-Mail: `bweiss@ira.uka.de`