Tentamen Formele Methoden voor Software Engineering (213520)

16 april 2009, 9.00-12.30 uur.

BELANGRIJK: geef op je tentamen duidelijk aan:

- je studierichting
- of je beide huiswerkopgaven gemaakt hebt, en bij welke werkcollegeleider.

Bij dit tentamen mag je de de FSP Quick Reference Card en de JML Cheat Sheet gebruiken, alsmede de sheets van de colleges (geen boeken).

---

1. *(25 punten)*

   In een zonnebankcentrum kan een klant een solarium cabine reserveren bij de toonbank. Zodra een cabine vrijkomt wordt ie aangezet, en wordt het nummer van de cabine aan de klant gegeven. De klant gaat naar de aangegeven cabine en geeft daar aan hoeveel tijd de cabine in werking moet zijn. Nadat de cabine stopt signaleert de cabine naar de toonbank het bedrag dat de klant moet betalen. De klant betaalt vervolgens bij de toonbank (ga er niet vanuit dat de klant precies weet wat ie moet betalen). Verder kan de desk een cabine laten schoonmaken (uiteraard als er geen klant in de cabine is).

   (a) Specificeer het zonnebankcentrum voor 1 klant en 1 cabine. Neem het volgende proces voor de toonbank:

   ```
   DESK = ( reserve -> turn_on -> get_number -> DESK
          | amount[m:M] -> pay[m] -> DESK
          | clean -> DESK).
   ```

   (hint: specifeer processen CUSTOMER en CABIN).

   (b) Specificeer nu een zonnebankcentrum met 2 cabines en 3 klanten (hint: denk goed na over hoe je het proces DESK moet veranderen om deadlocks te voorkomen).

   (c) Specificeer een safety property die zegt dat het verschil tussen het aantal `reserve` acties en het aantal `pay` acties nooit groter is dan 2. Geldt deze eigenschap voor jouw zonnebankcentrum?

   (d) Specificeer een progress property die zegt dat klant nummer 3 uiteindelijk een cabine krijgt. Geldt deze eigenschap? Geeft nu prioriteit aan de `reserve` acties van klant 1 en klant 2. Geldt de eigenschap nu nog steeds?

   (e) Voeg een proces toe aan het systeem dat ervoor zorgt dat elke cabine na precies 2 keer gebruikt te zijn wordt schoongemaakt.

2. *(10 punten)* Beschouw de volgende methode:

   ```
   int[] keerOm(int[] a) {
       int[] result = new int[a.length];
       int i = a.length-1;
       while (i >= 0) {
           result[a.length-1-i] = a[i];
           i--;
       }
       return result;
   }
   ```

   (a) Specificeer een zo volledig mogelijk contract voor deze methode in JML.

   (b) Bewijs (met behulp van een lus-invariant) de correctheid van de methode. Geef de bewijsstappen duidelijk aan!

3. *(15 punten)* Terug naar de sauna. Beschouw de volgende declaraties:

```
interface Cabin {
    /** Sets the customer field to a given value. */
    void setCust(Customer cust);
    /** Returns the current value of the customer field. */
    Customer getCust();
    /** Returns true if the customer is currently null. */
    boolean isFree();
    /** Returns the rate of this cabin (a positive amount). */
    double getRate();
}

class Customer {
    /** Returns the current payable amount. */
    double getAmount() {
        return amount;
    }
    /** Decreases the payable amount by a (positive) payment. */
    void payAmount(double payment) {
        amount -= payment;
    }
    /**
     * Reserves a cabin (which should be empty at the time of call)
     * and sets the payable amount to the cabin rate.
     * Should only be called if the customer has no cabin yet.
     */
    void setCabin(Cabin cabin) {
        cabin.setCust(this);
        this.cabin = cabin;
        this.amount = cabin.getRate();
    }
    /**
     * Frees the current cabin of this customer. Should only be called
     * if the customer has a cabin, and the due amount has been payed.
     */
    void resetCabin() {
        this.cabin.setCust(null);
        this.cabin = null;
    }
    /** Returns the currently reserved (possibly null) cabin. */
    Cabin getCabin() {
        return cabin;
    }
    // The current payable amount; should be non-negative.
    private double amount = 0;
    // The currently reserved cabin. If null, the payable amount
    // should be zero.
    private Cabin cabin;
}
```

(a) Stel een contract in JML op voor `Cabin`, waarin het JavaDoc-commentaar zoveel mogelijk formeel gespecificeerd is.

(b) `Cabin` wordt geïmplementeerd in een klasse `CabinImpl`. Geef van `CabinImpl` de constructor (waarin de `rate` wordt gezet) en de instantievariabelen, met bijbehorend JML-commentaar zodat de specificatie correct te bewijzen is. (Het bewijs zelf hoeft niet geleverd te worden!)

(c) Stel een contract in JML op voor `Customer`, waarin het JavaDoc-commentaar zoveel mogelijk formeel gespecificeerd is.

(d) Bewijs de correctheid van de methode `resetCabin()`.