Prof. Dr. Peter Thiemann
Sergio Feo-Arenis
Sergiy Bogomolov

Summer Term 2014

Softwaretechnik

http://proglang.informatik.uni-freiburg.de/teaching/swt/2014/

Exercise Sheet 4

Exercise 1: Design by Contract (5 Points)

The homepage of the lecture provides the code of a class `Stack` implementing stacks. Unfortunately, all pre- and postconditions as well as the invariants are missing. Please add them to the code.

Exercise 2: Contract Monitoring (5 Points)

The following code fragments are parts of Java classes and Java interfaces used for a spread sheet application annotated with contracts. Analyze the code and identify contract violations that may occur during run-time.

```
interface IIntegerInterval {
  int getLowerBound();
    @post { 0 <= getLowerBound() < getUpperBound() }
  int getUpperBound();
    @post { 0 <= getLowerBound() < getUpperBound() }
  void changeContent (int i);
    @pre { this.getLowerBound() <= i < this.getUpperBound() }
}

class IntegerInterval implements IIntegerInterval {
  int getLowerBound() { ... }
    @post { 0 <= getLowerBound() < getUpperBound() }
  int getUpperBound() { ... }
    @post { 0 <= getLowerBound() < getUpperBound() }
  void changeContent (int i) { ... }
    @pre { this.getLowerBound() <= i < this.getUpperBound() }
}

class NegativeIntegerInterval extends IntegerInterval {
   void changeContent (int i) {
     super.changeContent (-i);
   }
    @pre { this.getLowerBound() <= -i < this.getUpperBound() }
}

class Run {

  public static void main (String[] a) {
    ...
    int i = ...
```

```
    IntegerIntervall c = ...
    NegativeIntegerInterval n = ...

    if (i >= 0 && i <= 10) {
     c.changeContent(c.getLowerBound()+(c.getUpperBound()-c.getLowerBound)*i/10);
    }

    ...
    n.changeContent(-42);
    ...

  }
    @pre { true }
}
```

Exercise 3: Hoare Calculus (10 Points)

Prove the following Hoare triples.

(i)
```
{ x >= 10, y >= 0 }
y = y + x;
{ x >= 0, y >= 5 }
```

(ii)
```
{ true }
if (a > b) {
  m = a;
} else {
  m = b;
}
{ m == max (a, b) }
```

(iii)
```
{ A, i < n }
i = i + 1;
sum = sum + i;
{ A }
```
where we define the assertion `A` by $\texttt{A} \equiv (sum + \Sigma_{j=i+1}^{n} j == n(n+1)/2) \wedge i \leq n$.
Hint: express `A` first in a simpler, equivalent form.

(iv)
```
{ n >= 0, sum=0, i=0 }
while (i<n)
  {
    i = i + 1;
    sum = sum + i;
  }
{ sum == n*(n + 1)/2 }
```

(v)
```
{ n >= 0 }
sum = 0;
i = 0;
while (i<n)
```

```
  {
    i = i + 1;
    sum = sum + i;
  }
{ sum == n*(n + 1)/2 }
```

Submission

- Submit this sheet *before* the lecture of Thursdays.
- Late submissions will not be accepted.
- Deadline: Thursday 11:59 a.m..