

Algorithmen und Programmierung IV

SS 2006

11. Mai 2006

Musterlösung zum Aufgabenblatt 1

Aufgabe 1

- `sqrt(a*a+b*b)`

Unteilbar, weil der Ausdruck keine Nebenwirkung hat und keine gemeinsamen Variablen beteiligt sind.

- `x == max(a,x)`

Teilbar, weil in dem Ausdruck die gemeinsame Variable `x` zweimal gelesen wird.

```
a = 3;  
x = 5;  
x == ...
```

```
-----  
x = 7;  
-----
```

```
... max(a,x)
```

Erklärung: Sowohl für `x == 5` als auch für `x == 7` sollte der Ausdruck `x == max(a,x)` den Wert `true` liefern, jedoch findet bei diesem Zeitschnitt der Vergleich `5 == 7` statt, so dass der Ausdruck den Wert `false` liefert.

- `x = max(a,b);`

Unteilbar, weil bei der Zuweisung die gemeinsame Variable `x` einfach ist und der Ausdruck `max(a,b)` keine Nebenwirkung und keine gemeinsamen Variablen hat.

- `z ? a+b : a-b`

Unteilbar, weil der Ausdruck keine Nebenwirkung hat, die gemeinsame Variable `z` einfach ist, genau einmal gelesen wird und sonst nur private Variablen beteiligt sind.

- `sqrt(a*x+b*y)`

Teilbar, weil zwei gemeinsame Variablen `x` und `y` gelesen werden.

```
a = b = 1;  
x = y = 0;  
sqrt(a*x+...)
```

```
-----  
x = 9;  
y = 16;  
-----
```

```
sqrt(...+b*y)
```

Erklärung: Der Ausdruck `sqrt(a*x+b*y)` sollte den Wert 0 (für `x == 0` und `y == 0`), den Wert 3 (für `x == 9` und `y == 0`) oder den Wert 5 (für `x == 9` und `y == 16`) liefern, jedoch liefert er bei diesem Zeitschnitt den Wert 4, weil der Wert von `x` vor und der Wert von `y` nach der Änderung von `x` und `y` gelesen wird.

- `z = !z;`

Teilbar, weil bei der Zuweisung die gemeinsame Variable `z` sowohl gelesen als auch geschrieben wird.

```
z = true;  
... !z;
```

```
-----  
z = !z  
-----
```

```
z = ...
```

Erklärung: Nach der Ausführung der beiden Zuweisungen `z = !z` sollte der Wert von `z` wieder `true` sein, jedoch ist der Wert von `z` bei diesem Zeitschnitt `false`, weil beide Anweisungen den Wert von `z` lesen, bevor die jeweils andere die Zuweisung durchgeführt hat.

Aufgabe 2

- a) Zur besseren Übersicht bei den Zeitschnitten wurde die `for`-Schleife „entrollt“ und der Inhalt der Methode `prettyPrint` eingesetzt, wobei zu beachten ist, dass die Parameter `c` und `norm` Kopien sind und daher mit `c0` bis `c3` bzw. `norm0` bis `norm3` bezeichnet werden.

- Fall 1: Das vierte `prettyPrint` überholt das dritte.

```
...  
  
c.real = reals[2];  
c.img = imgs[2];  
norm = norm(c);  
-----  
// Parameterübergabe  
String res2 = "| "+c2.real+" + "+c2.img+" *i|="+norm2;  
-----  
  
c.real = reals[3];  
c.img = imgs[3];  
norm = norm(c);  
-----  
// Parameterübergabe  
String res3 = "| "+c3.real+" + "+c3.img+" *i|="+norm3;  
System.out.println(res3);  
// Ausgabe: | 1.0 + 1.0 *i| = 1.41...  
-----  
  
System.out.println(res2);  
// Ausgabe: | 0.0 + 1.0 *i| = 1.0
```

- Fall 2: Die Attribute von `c` erhalten die Werte der vierten komplexen Zahl, bevor das dritte `prettyPrint` seine Parameter übernommen hat.

```
...  
  
c.real = reals[2];  
c.img = imgs[2];  
norm = norm(c);  
-----  
// FORK, aber noch keine Ausführung von prettyPrint  
-----  
  
c.real = reals[3];  
c.img = imgs[3];  
-----  
// Parameterübergabe  
String res2 = "| "+c2.real+" + "+c2.img+" *i|="+norm2;  
System.out.println(res2);  
// Ausgabe: | 1.0 + 1.0 *i| = 1.0  
-----  
  
norm = norm(c);  
-----  
// Parameterübergabe  
String res3 = "| "+c3.real+" + "+c3.img+" *i|="+norm3;  
System.out.println(res3);  
// Ausgabe: | 1.0 + 1.0 *i| = 1.41...
```

Hier ist zu beachten, dass die Variable `c` nur einen Verweis zu einem Exemplar vom Typ `Complex` enthält. Somit wird bei der Parameterübergabe auch nur der Verweis kopiert, aber das Exemplar der komplexen Zahl bleibt das gleiche.

- Fall 3:

```
...  
  
c.real = reals[3];  
c.img = imgs[3];  
norm = norm(c)  
// Methode test wird verlassen  
-----  
// andere Operationen  
-----  
// Parameterübergabe (!?)  
String res3 = "| "+c3.real+" + "+c3.img+" *i|="+norm3;  
System.out.println(res3);  
// Ausgabe: | 1.0 + 1.0 *i| = -1.98...
```

Da lokale Variablen nur solange existieren, solange die jeweilige Methode ausgeführt wird, aber aufgrund der Gabelungsanweisung die Parameterübergabe unter Umständen erst dann stattfindet, sobald die Methode `test` verlassen worden ist, könnte der Speicher der Variablen `c` und `norm` zwischenzeitig anderweitig verwendet worden sein. Im obigen Zeitschnitt wurde offenbar der Speicherbereich von `norm` bereits überschrieben, aber der Verweis in `c` zeigte noch auf ein Exemplar von `Complex`, welches noch nicht von der Speicherbereinigung entfernt worden ist.

- b) Auf diese Weise kann nur das im Fall 3 beschriebene Problem mit ungültig gewordenen lokalen Variablen gelöst werden, da die Methode `test` erst dann verlassen wird, wenn die Parameter `c` und `norm` übergeben worden sind.

Ferner wird zumindest der Wert von `norm` immer korrekt ausgegeben, da er nun von der Methode `test` nicht vor der Parameterübergabe mit einem neuen Wert überschrieben werden kann.

- c) Im Gegensatz zu Teil b) wird das Problem durch diese Modifikation der Sprache verschärft: Während bisher nur die Änderung der Werte `c.real` und `c.img` in der Methode `test` eine direkte Auswirkung auf die Methode `prettyPrint` hatte, wirkt sich mit Variablenparametern auch eine Änderung der Variable `norm` in der Methode `test` unter Umständen nachträglich auf `prettyPrint` aus.

Aufgabe 3

Zur besseren Übersicht bei dem Zeitschnitt wurde der Inhalt der Methode `add` eingesetzt.

```
Cell cell1 = new Cell();
cell1.head = x;
cell1.tail = head;
-----
                        Cell cell2 = new Cell();
                        cell2.head = y;
                        cell2.tail = head;
                        head = cell2;
-----
head = cell1;
```

Anders als erwartet enthält die Liste nach Abschluss des `CO/OC`-Block weder `x` gefolgt von `y` noch `y` gefolgt von `x`, sondern nur `x`, weil unabhängig von einander zwei Zellen vom Typ `Cell` erzeugt wurden, von denen keine auf die jeweils andere, sondern beide auf den alten Anker `head` (in diesem Fall `null`) verweist. Wird der Anker `head` nun auf eine der beiden Zellen gesetzt, so geht die andere Zelle verloren.