

## Aufgabe 1 (12 Punkte)

a) (6 P.)  $x, y, \dots$  seien *einfache* gemeinsame Variable;  $a, b, \dots$  seien *einfache* private Variable. Welche der folgenden sechs Ausdrücke bzw. Zuweisungen sind unteilbar? Warum bzw. warum nicht?

```
sqrt(a*a+b*b)
x == max(a,x)
x = max(a,b);
z ? a+b : a-b
sqrt(a*x+b*y)
z = !z;
```

b) (6 P.) Geben Sie für die nicht unteilbaren Fälle jeweils ein einfaches Szenario an, bei dem die Aktivitäten eines zweiten Prozesses unerwünschte Effekte verursachen können! (Zeitschnitte verwenden!)

*Zur Erinnerung:* Dass das Lesen von  $x$  bei nebenläufiger Änderung von  $x$  nicht-deterministisch entweder den alten oder den neuen Wert von  $x$  liefert, liegt in der Natur der Sache und ist nicht weiter problematisch. *Unerwünscht* wäre aber zum Beispiel, wenn die Auswertung von  $x*x$  weder den alten Quadratwert noch den neuen lieferte.

## Aufgabe 2 (14 Punkte)

Wir betrachten die Gabelungsanweisung

*ForkStatement:* **FORK** *Process*

die die angegebene Anweisung *Process* asynchron ausführt. Eine solche Gabelungsanweisung wird in den folgenden Programmfragmenten benutzt:

```
class Complex { double real, img; }
```

```

...
static double norm(Complex c) {
    return sqrt(sqr(c.real)+sqr(c.img));
}
static void prettyPrint(Complex c, double norm) {
    String res = "| "+c.real+" + "+c.img+" *i|="+norm;
    System.out.println(res);
}
static void test() {
    double reals[] = {0.0, 1.0, 0.0, 1.0};
    double imgs[] = {0.0, 0.0, 1.0, 1.0};
    Complex c = new Complex();
    double norm;
    for(int i=0; i<4; i++) {
        c.real = reals[i]; c.img = imgs[i];
        norm = norm(c);
        // langsame Ausgabe nebenläufig im Hintergrund:
        FORK prettyPrint(c,norm); }
}

```

a) (6 P.) Das Ergebnis ist nicht ganz so wie erwartet. Verschiedene Aufrufe von `test` produzieren mitunter verschiedene Ausgaben. Unter anderem werden die folgenden beiden Ausgaben beobachtet:

```

| 0.0 + 0.0 *i | = 0.0
| 1.0 + 0.0 *i | = 1.0
| 1.0 + 1.0 *i | = 1.4142135623730951
| 0.0 + 1.0 *i | = 1.0

```

```

| 0.0 + 0.0 *i | = 0.0
| 1.0 + 0.0 *i | = 1.0
| 1.0 + 1.0 *i | = 1.0
| 1.0 + 1.0 *i | = 1.4142135623730951

```

Beschreiben Sie unter Einsatz von Zeitschnitten, wie beide Ausgaben entstanden sind!

Damit nicht genug, gelegentlich werden auch Ausgaben mit völlig unerwarteten Werten beobachtet, wie zum Beispiel:

```

...
| 1.0 + 1.0 *i | = -1.9823791387912837e+44
...

```

Erklären Sie, wie eine derartige Ausgabe entstehen kann! (Beachten Sie dabei die Lebensdauer lokaler Variablen.)

b) (4 P.) Wir betrachten eine eingeschränkte Variante der Gabelungsanweisung: *Process* darf nicht jede beliebige Anweisung sein; es muss sich um den Aufruf einer **void**-Methode handeln. Zudem gibt es einen subtilen semantischen Unterschied: *erst nach erfolgter Parameterübergabe* wird mit dem Erzeugerprozess fortgefahren. Lassen sich damit die Probleme aus a) lösen und, wenn ja, wie?

c) (4 P.) Lösen Sie b) für den Fall, dass die Programmiersprache nicht Wertparameter, sondern *Variablenparameter* verwendet!

### Aufgabe 3 (8 Punkte)

Die folgende Klasse realisiert *Objektlisten* als einfach verkettete Listen; Die Eintragung eines Objekts erfolgt wie bei einer *push*-Operation auf einem Keller:

```
class ObjectList {
    private Cell head;
    private static class Cell {Object head; Cell tail;}
public void add(Object x) {
    Cell cell = new Cell();
    cell.head = x;
    cell.tail = head;
    head = cell;
}
...
}
```

Zwei nebenläufige Prozesse erzeugen Objekte *x*, *y* und wollen sie in einer Objektliste *objects* ablegen:

```
ObjectList objects = new ObjectList();
CO ... objects.add(x); .....
|| ..... objects.add(y); ...
OC
.....
```

Das geht nicht immer gut! Was kann passieren? (Bitte mit Zeitschnitten verdeutlichen!)