**ALP4 SoSe 2013, Di. 16-18**

# Lösung Übungsblatt 8

**Christoph van Heteren-Frese (Matr.-Nr.: 4465677)**
**Sven Wildermann (Matr.-Nr.: 4567553)**
Tutor: Alexander Steen, eingereicht am 13. Juni 2013

---

## Aufgabe 1

### a)

Das Paket `rwmutex.go` implementiert ein spezielles Schloss, das zwar viele Prozesse lesen, aber nur einen schreiben lässt. Es werden dafür vier Zugriffsfunktionen definiert: `RLock()`, `RUnlock()`, `Lock()` und `Unlock()`. Will ein Prozess den Mutex für den Schreibzugriff nutzen obwohl gerade andere Prozesse lesen, wird dieser blockiert. Grundlage der Erläuterung ist folgendes kleines Beispiel:

```
1  package main
2
3  import (
4    "fmt"
5    "sync"
6    "time"
7  )
8
9  var (
10   rwm     sync.RWMutex
11   balance int
12 )
13
14 // "reads" the current balance and prints it
15 // to the sdtout
16 func get(balance *int) {
17   rwm.RLock()
18   fmt.Println(*balance)
19   rwm.RUnlock()
20 }
21
22 // "writes" the current balance: increases the
23 // balance by the given amount
24 func put(balance *int, amount int) {
25   rwm.Lock()
26   *balance += amount
27   rwm.Unlock()
28 }
29
30 func main() {
31   balance = 0
32   go get(&balance)
33   go put(&balance, 100)
34   go get(&balance)
35   go put(&balance, 100)
36   go get(&balance)
37   time.Sleep(1000)
38 }
```

**b)**

**c)**

## Aufgabe 2

## Aufgabe 3

## Aufgabe 4