# Arbeitsblatt #5

Einführung in die Programmierung mit *Go*

12. März 2015

## 1 Kommandozeilenparameter

Schreiben Sie ein Programm, das die Go-Bibliothek für Kommandozeilenparameter verwendet.

a. Neue Kommandozeilenparameter für ein *Go*-Programm können über das Paket `"flag"` definiert werden. Sie schreiben z.B.

```
name := flag.String("name1", "default-Wert", "Beschreibung")
nummer := flag.Int("name2", 42, "Beschreibung")
boolean := flag.Bool("name3", false, "Beschreibung")
flag.Parse() // immer letzter Befehl
```

b. Verwenden Sie diese Bibliothek, um eine Zeichenkette zu übergeben, die entweder `"hallo"` oder `"auf wiedersehen"` ist. Je nachdem, welcher der beiden Werte übergeben wurde, soll Ihr Programm eine andere Ausgabe produzieren; wenn die Zeichenkette nicht gesetzt wurde oder einen anderen Wert hat, soll das Programm abgebrochen werden. Verwenden Sie die **`switch`**-Anweisung zur Fallunterscheidung.

## 2 Benchmarks

Schreiben Sie einen Benchmark für die Fibonacci-Funktion:

$$\begin{aligned} F(0) &= 0 \\ F(1) &= 1 \\ F(n) &= F(n-1) + F(n-2) \end{aligned}$$

Benchmarken Sie Ausführung der Funktion an drei verschiedenen Punkten (für drei verschiedene Parameterwerte).

## 3 Chat

Bauen Sie ein Client-Programm, um sich mit einem vorkonstruierten Chat-Server zu verbinden.

Der Server läuft auf `sepl.cs.uni-frankfurt.de:11713` (dabei ist `sepl.cs.uni-frankfurt.de` der Name des Rechners und `11713` die Port-Nummer); Sie können sich über das `tcp`-Protokoll verbinden.

Das Programm in Abbildung 2 demonstriert das Kommunikationsprotokoll. Verwenden Sie diesen Code für Ihre Lösung. Nachrichten werden im Typ `chat.Message` kodiert:

- `Kind` ist die Art der Nachricht (gesendete bzw. empfangene Nachricht; Sie können dieses Feld ignorieren)
- `Sender` ist ein Sendername. Wählen Sie einen geeigneten Namen für sich selbst zum Chat aus.
- `Body` ist die Textnachricht, die zu senden ist.

```
func getLine() string {
    reader := bufio.NewReader(os.Stdin)
    text, _ := reader.ReadString('\n')
    return text
}
```

Abbildung 1: Lesen von Text

Die Operationen `Read` und `Write` erlauben Ihnen das Lesen und Schreiben von solchen Nachrichten durch `bufio.Writer` bzw. `bufio.Reader`. Die tatsächliche Übertragung findet durch unverschlüsselte json-Nachrichten statt; dieses Detail wird Ihnen von der `chat`-Bibliothek abgenommen.

a. Schreiben Sie einen Chat-Client, der den 'Sender'-Namen als Kommandozeilenparameter akzeptiert.

b. Stellen Sie eine TCP-Netzwerkverbindung her. Verwenden Sie dazu die Funktion `net.Dial`[1].

c. Wenn der Verbindungsaufbau erfolgreich war, können Sie mit dem `bufio`[2]-Paket aus der erzeugten Netzwerkverbindung einen `*bufio.Reader` und einen `*bufio.Writer` erzeugen (z.B. `NewReadWriter`).

d. Lassen Sie zwei Goroutinen gleichzeitig laufen. Eine soll von der Netzwerkverbindung Pakete lesen und diese ausdrucken. Die zweite Goroutine soll Benutzereingaben lesen (z.B. mit dem Code aus Abbildung 1)

e. Testen Sie Ihr Programm und chatten Sie mit den anderen Kursteilnehmern.

f. Untersuchen Sie den Code und stellen Sie sicher, daß Sie die importierten Code-Stücke verstehen.

[1] `http://golang.org/pkg/net`
[2] `http://golang.org/pkg/bufio`

```go
package chat

import "encoding/json"
import "bufio"

const SendMessage    = "SND"
const ReceiveMessage = "RCV"

type Message struct {
    Kind         string
    Sender, Body string
}

const NUL = 0

func Write(writer *bufio.Writer, m Message) {
    data, err := json.Marshal(m)
    if err != nil {
        panic(err)
    }
    n, err := writer.Write(data)
    if err != nil {
        panic(err)
    }
    if n < len(data) {
        panic("Didn't write all bytes")
    }
    err2 := writer.WriteByte(NUL)
    if err2 != nil {
        panic(err2)
    }
    err3 := writer.Flush()
    if err3 != nil {
        panic(err3)
    }
}

func Read(reader *bufio.Reader) Message {
    var m Message
    data, err := reader.ReadBytes(NUL)
    err = json.Unmarshal(data[:len(data)-1], &m)
    if err != nil {
        panic(err)
    }
    return m
}
```

Abbildung 2: Die Datei `chat.go`