

# Eine Einführung in go

Hauke Stieler

16. Januar 2018

Fachbereich Informatik der Universität Hamburg

# **Agenda**

- some history
- basic features
- cool web stuff
- concurrency212121
- interfaces

# Why go?

In 2007, three guys at Google were frustrated with the existing languages for writing server software:

- Compiling C++ was too slow
- Writing Java felt too verbose
- Aversion against inheritance and design patterns
- Getting concurrency right was hard

### C++

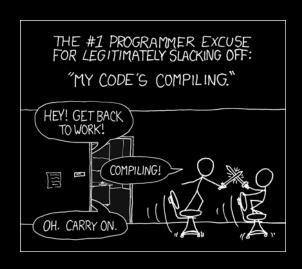
```
// Within large projects, popular header files
// get included thousands of times and hence
// have to be recompiled over and over again
// winclude <iostream>
// include <string>
// include <vector>
```

### C++

```
// Within large projects, popular header files
// get included thousands of times and hence
// have to be recompiled over and over again
#include <iostream>
#include <string>
#include <vector>
```

gcc copies specified file by #include recursively into source file. The same header file gets recompiled over and over again.

→ Rob Pike: Public Static Void at OSCON 2010



#### Java I

Let's do some Java.

Write a public class Person that does the following:

- store a string name
- store an int age

Simple, right?

#### Java I

NO :(

```
Let's do some Java.

Write a public class Person that does the following:

• store a string name

• store an int age

Simple, right?
```

5

#### Java I

```
public class Person {
        private String name;
 2
        private int age;
3
4
        public PersonBean(String name, int age) {
5
            this name = name;
 6
            this age = age;
8
9
        public String getName() {
10
            return name;
11
12
13
        public void setName(String name) {
14
            this name = name;
15
16
17
```

#### Java II

```
18
            return age;
19
20
21
        public void setAge(int age) {
22
            this age = age;
23
24
25
        @Override
26
        public String toString() {
27
            return "PersonBean [" + "name=" + name + ", "
28
           "age=" + age + "]";
29
30
        @Override
32
            final int prime = 31;
33
```

### Java III

```
int result = 1;
34
            result = prime * result + age;
35
            result = prime * result + ((name == null) ? 0 :
36
        name.hashCode());
            return result;
37
38
39
        @Override
40
        public boolean equals(Object obj) {
41
            if (this == obj)
42
43
            if (obj == null)
44
45
46
            if (getClass() != obj.getClass())
47
            PersonBean other = (PersonBean) obj;
48
            if (age != other.age)
49
```

### Java IV

```
return false;
if (name == null) {
    if (other.name != null)
        return false;
} else if (!name.equals(other.name))
return false;
return true;
}
```

# Design

Initial design by 3 people with different backgrounds:

- Rob Pike (Concurrency)
- Robert Griesemer (Modules)
- Ken Thompson (Operating Systems)

All design decisions had to be agreed upon unanimously. Design team later joined by more people at Google.

# Design

- simplicity
- simplicity
- simplicity
- clean package model for fast compilation
- built-in concurrency based on CSP
- interfaces instead of inheritance
- no radical changes after Go 1.0

