

Extracted from:

Go Brain Teasers

Exercise Your Mind

This PDF file contains pages extracted from *Go Brain Teasers*, published by the Pragmatic Bookshelf. For more information or to purchase a paperback or PDF copy, please visit <http://www.pragprog.com>.

Note: This extract contains some colored text (particularly in code listing). This is available only in online versions of the books. The printed versions are black and white. Pagination might vary between the online and printed versions; the content is otherwise identical.

Copyright © 2021 The Pragmatic Programmers, LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

The Pragmatic Bookshelf

Raleigh, North Carolina

The
Pragmatic
Programmers

Go Brain Teasers

Exercise Your Mind



Miki Tebeka
edited by Margaret Eldridge

Go Brain Teasers

Exercise Your Mind

Miki Tebeka

The Pragmatic Bookshelf

Raleigh, North Carolina



Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and The Pragmatic Programmers, LLC was aware of a trademark claim, the designations have been printed in initial capital letters or in all capitals. The Pragmatic Starter Kit, The Pragmatic Programmer, Pragmatic Programming, Pragmatic Bookshelf, PragProg and the linking *g* device are trademarks of The Pragmatic Programmers, LLC.

Every precaution was taken in the preparation of this book. However, the publisher assumes no responsibility for errors or omissions, or for damages that may result from the use of information (including program listings) contained herein.

For our complete catalog of hands-on, practical, and Pragmatic content for software developers, please visit <https://pragprog.com>.

The team that produced this book includes:

CEO: Dave Rankin

COO: Janet Furlow

Managing Editor: Tammy Coron

Development Editor: Margaret Eldridge

Copy Editor: Jennifer Whipple

Indexing: Potomac Indexing, LLC

Layout: Gilson Graphics

Founders: Andy Hunt and Dave Thomas

For sales, volume licensing, and support, please contact support@pragprog.com.

For international rights, please contact rights@pragprog.com.

Copyright © 2021 The Pragmatic Programmers, LLC.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

ISBN-13: 978-1-68050-899-4

Encoded using the finest acid-free high-entropy binary digits.

Book version: P1.0—September 2021

*To Sharon, who suffered me in quarantine,
and the twenty years before that.*

A Job to Do

```
job.go
package main

import (
    "fmt"
)

type Job struct {
    State string
    done  chan struct{}
}

func (j *Job) Wait() {
    <-j.done
}

func (j *Job) Done() {
    j.State = "done"
    close(j.done)
}

func main() {
    ch := make(chan Job)
    go func() {
        j := <-ch
        j.Done()
    }()

    job := Job{"ready", make(chan struct{})}
    ch <- job
    job.Wait()
    fmt.Println(job.State)
}
```

Guess the Output



Try to guess what the output is before moving to the next page.

This code will print: ready

At first glance, it looks like the code is OK. You're using a pointer receiver in the `Job` struct methods. The fact that the call to `Wait` terminated tells you that the channel was closed.

The problem is with the definition of `ch`. It is a channel of `Job`, not `*Job`, which means that when you send the variable `job` over the channel, you actually send a copy of it. A channel in Go is a *pointer-like* type, so even though there is a copy of `job` inside the goroutine, `j.done` points to the same channel `job.done` is pointing to.

Strings in Go are not pointer-like. When you call `j.Done()`, the string inside the goroutine, you change the value of the `State` field in the goroutine copy of `job`. This change is not reflected in the `job` variable declared in line 28.

The solution is to make `ch` type `*Job`.

```
job_ptr.go
package main

import (
    "fmt"
)

type Job struct {
    State string
    done  chan struct{}
}

func (j *Job) Wait() {
    <-j.done
}

func (j *Job) Done() {
    j.State = "done"
    close(j.done)
}

func main() {
    ch := make(chan *Job)
    go func() {
        j := <-ch
        j.Done()
    }()

    job := Job{"ready", make(chan struct{})}
    ch <- &job
    job.Wait()
    fmt.Println(job.State)
}
```

Further Reading

There Is No Pass-by-Reference in Go

dave.cheney.net/2017/04/29/there-is-no-pass-by-reference-in-go

Channel Types Specification

golang.org/ref/spec#Channel_types

Go Concurrency Patterns: Pipelines and Cancellation

blog.golang.org/pipelines

Channels in the Go Tour

tour.golang.org/concurrency/2