Deep Dive into Interfaces

MD Kabir
Twitter: @kabir_md5
Github: msk610

Overview

- Introduction
- Some tips that will help
- Things to avoid with interfaces
- Some good examples use cases of interfaces
- Some sample scenarios
- Conclusion
- Useful links

Introduction

- An *interface type* is defined as a set of method signatures
- Provides a way to achieve Polymorphism
- Lets you use duck typing like Python but provides compile time validation

Some tips that will help

- Start with concrete data structures
- Do not export interfaces unless you need to
- When you need users of the package to provide an implementation detail
- Multiple implementations that need to be maintained internally
- Try (Do) not to use empty interfaces

Things to avoid with interfaces

```go
package api

// Service interface
type Service interface {
    // Start
    Start()
    // HandlePutAPI
    HandlePutAPI(key string, value string)
    // HandleGetAPI
    HandleGetAPI(key string) string
}
```

```go
// service structure
type service struct {
    // store to manage key value
    store map[string]string
}

// Start method for service
func (s *service) Start() {
    s.store = make(map[string]string)
}

// HandlePutAPI method for service
func (s *service) HandlePutAPI(key string, value string) {
    s.store[key] = value
}

// HandleGetAPI method for service
func (s *service) HandleGetAPI(key string) string {
    if val, ok := s.store[key]; ok {
        return val
    }
    return ""
}

// MakeAPIService function
func MakeAPIService() Service {
    newService := service{}
    newService.store = make(map[string]string)
    return &newService
}
```

Things to avoid with interfaces (interface pollution)

```
go test -bench=.

goos: darwin

goarch: amd64

pkg: github.com/msk610/talk/api

BenchmarkInterfacePutAPI-4      20000000          60.9 ns/op

BenchmarkStructPutAPI-4         20000000          57.8 ns/op

BenchmarkInterfaceGetAPI-4      30000000          39.1 ns/op

BenchmarkStructGetAPI-4         50000000          24.6 ns/op
```

```go
package main

// Foo implements Speaker
type Foo struct{}

// Talk method for Foo
func (*Foo) Talk() {}

// Answer method for Foo
func (Foo) Answer() {}

// Speaker interface
type Speaker interface {
    Talk()
    Answer()
}
```

```go
// GiveMeetupTalk needs a speaker
func GiveMeetupTalk(speaker Speaker) {
    if speaker != nil {
        speaker.Talk()
        println("Speaker gave talk")
        speaker.Answer()
        println("Speaker answered questions")
    }
}

func main() {
    var badFoo *Foo = nil
    var badSpeaker Speaker = badFoo
    GiveMeetupTalk(badSpeaker)
}
```

Breaks here

(method dispatch table for Foo, nil)

Things to avoid with interfaces (nil values)

```
$ go run main.go

Speaker gave talk
panic: value method main.Foo.Answer called using nil *Foo pointer
```

Some good example use cases of interfaces

```go
package main

import (
    "fmt"
    "sort"
)

// Foo struct
type Foo struct {
    // Bar
    Bar int
    // Baz
    Baz int
}

// ByBarBaz implements sort.Interface
type ByBarBaz []Foo

func (b ByBarBaz) Len() int           { return len(b) }
func (b ByBarBaz) Swap(i, j int)      { b[i], b[j] = b[j], b[i] }
func (b ByBarBaz) Less(i, j int) bool { return b[i].Bar+b[i].Baz < b[j].Bar+b[j].Baz }

func main() {
    foos := []Foo{
        {0, 1},
        {4, 2},
        {2, 10},
        {0, 0},
    }
    sort.Sort(ByBarBaz(foos))
    fmt.Println(foos)
}
```

```go
type Reader interface {
  Read(p []byte) (n int, err error)
}

type Writer interface {
  Write(p []byte) (n int, err error)
}
```

Some sample scenarios

- Mocking: Especially with databases
- API: Interacting with various packages (maintaining various versions)
- Abstract data type: Building SQL CRUD library

Conclusion

- Do not limit yourself
- Start with concrete structures
- Postel’s Law
  - “Be conservative with what you do, be liberal with what you accept from others”

Useful Links

- https://medium.com/@rakyll/interface-pollution-in-go-7d58bccec275
- https://blog.chewxy.com/2018/03/18/golang-interfaces/
- https://www.ardanlabs.com/blog/2016/10/avoid-interface-pollution.html
- https://medium.com/@matryer/golang-advent-calendar-day-seventeen-io-reader-in-depth-6f744bb4320b