Object Orientation and Polymorphism

Michael VanSickle

@vansimke

Introduction

Methods

Interfaces

Generic Programming

Method

A ***method*** **in object-oriented programming is a function associated with an invocation and a variable.**

Method (computer programming). (2022, August 28). In *Wikipedia*.
https://en.wikipedia.org/wiki/Method_(computer_programming)

Functions vs Methods

```
// function
var i int

func isEven(i int) bool {
     return i%2==0
}

// method
type myInt int                          // need a type to bind method to.
                                        // DOESN'T HAVE TO BE A STRUCT
func (i myInt) isEven() bool {          // method receiver
     return int(i)%2==0
}
```

Methods indicate a tighter coupling between a function and a type

Functions vs Methods

```go
// function
var i int
func isEven(i int) bool {
    return i%2==0
}
ans := isEven(i)

// method
type myInt int
var mi myInt
func (i myInt) isEven() bool {
    return int(i)%2==0
}
ans = mi.isEven()
```

Method Receivers

```
type user struct {
    id       int
    username string
}

func (u user) String() string {                              // value receiver
    return fmt.Sprintf(“%v (%v)\n”, u.username, u.id)
}

func (u *user) UpdateName(n name) {                          // pointer receiver
    u.username = name
}
```

Use pointer receivers to share variable between caller and method

Demo

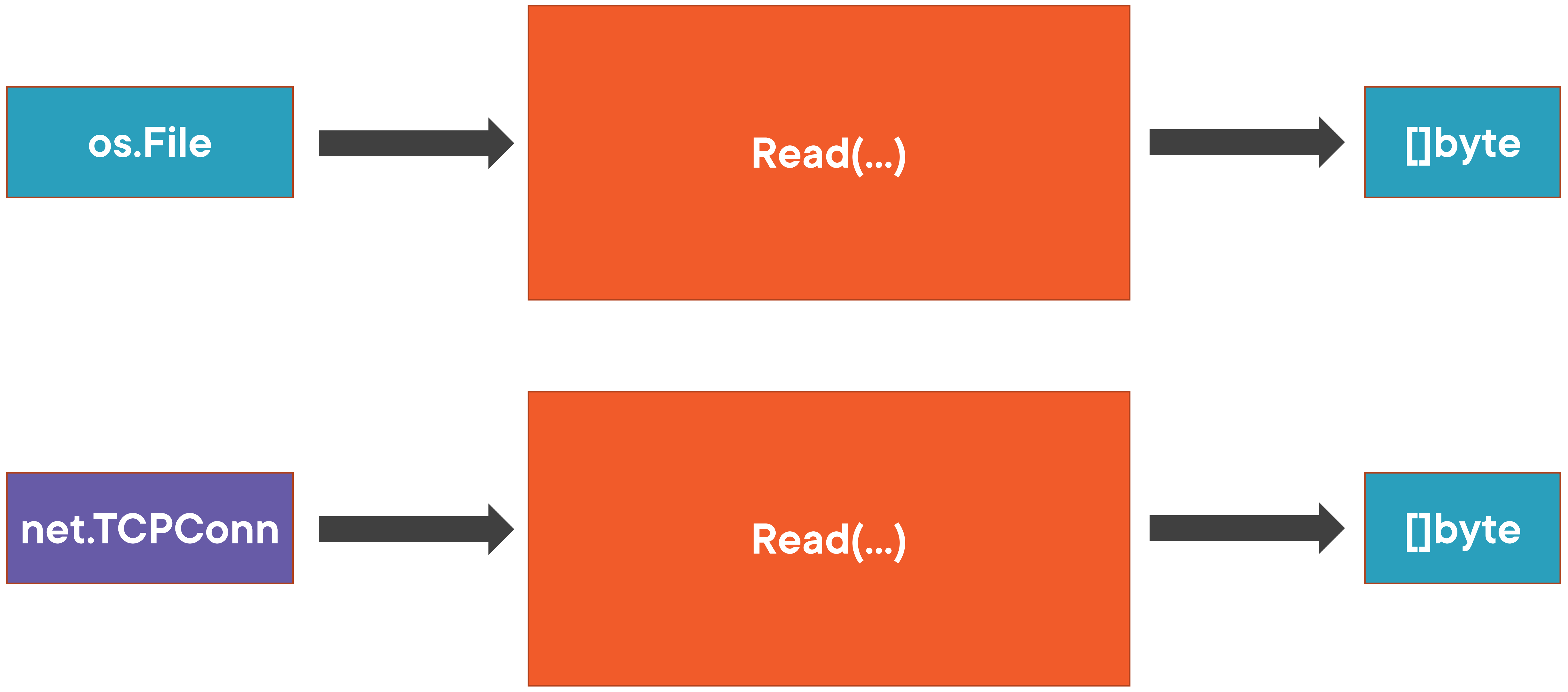
Methods

refactor course demo to use methods bound to types

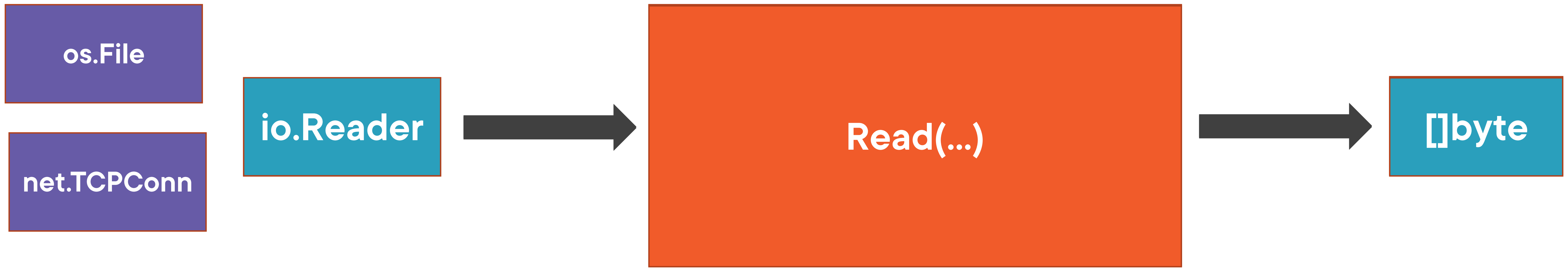
not necessarily better, just different!

Interfaces

Methods and Concrete Types



Methods and Interfaces



Interfaces

```
type Reader interface {
    Read([]byte) (int, error)
}

type File struct { ... }
func (f File) Read(b []byte) (n int, err error)

type TCPConn struct { ... }
func (t TCPConn) Read(b []byte) (n int, err error)

var f File
var t TCPConn

var r Reader
r = f
r.Read(...)                     // read from File
r = t
r.Read(...)                     // read from TCPConn
```

Type Assertions

```
type Reader interface {
    Read([]byte) (int, error)
}

type File struct { ... }
func (f File) Read(b []byte) (n int, err error)

var f File
var r Reader = f

var f2 File = r                    // error, Go can't be sure this will work
f2 = r.(File)                      // type assertion, panics upon failure
f2, ok := r.(File)                 // type assertion with comma okay, doesn't panic
```

Type Switches

```
var f File
var r Reader = f

var f2 File = r

switch v := r.(type) {
case File:
     // v is now a File object
case TCPConn:
     // v is now a TCPConn object
default:
     // this is selected if no types were matched
}
```

Demo

Interfaces

define interface and multiple concrete types that implement

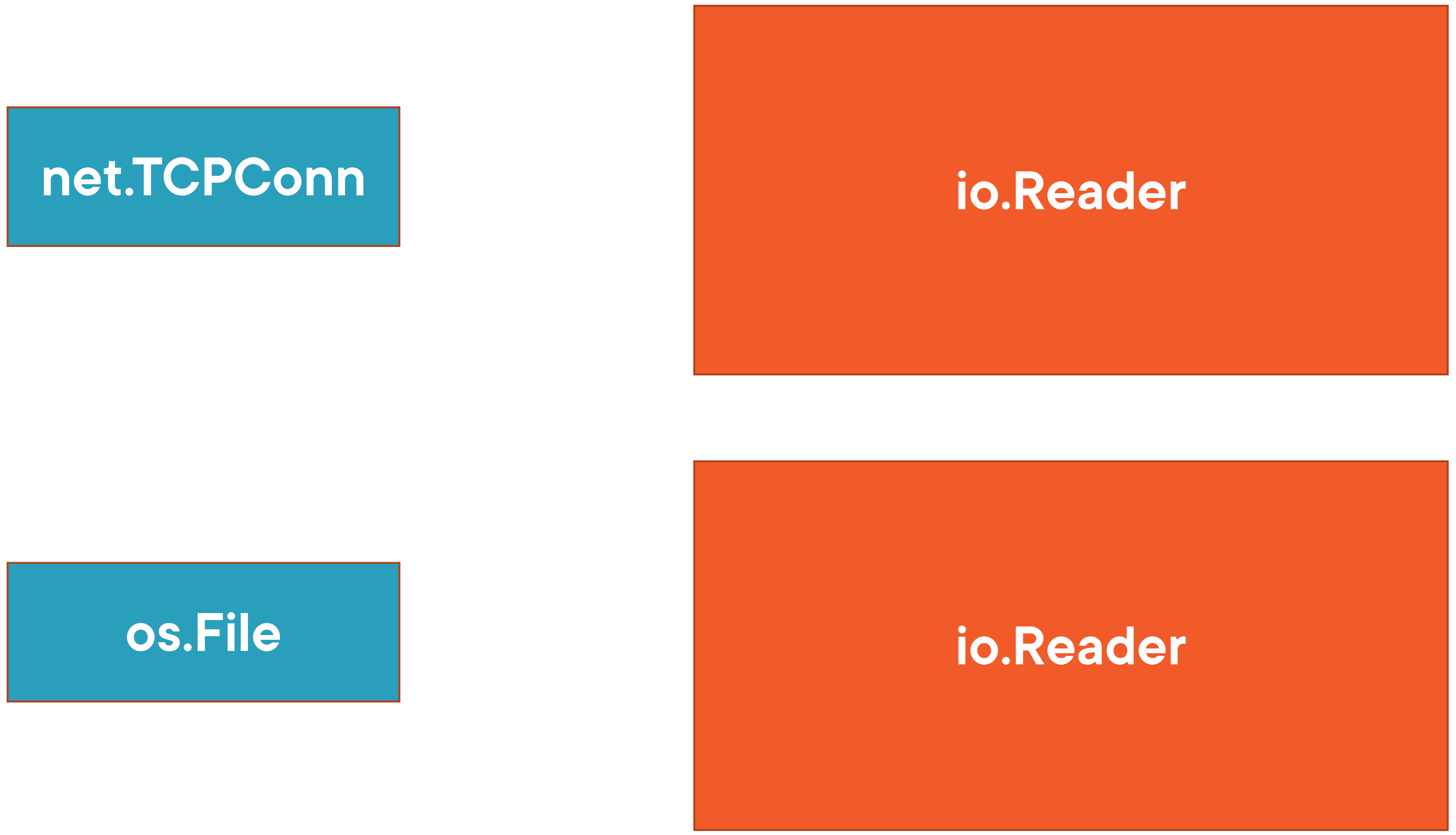
- generic example
- have menuItem type implement fmt.Printer interface?
- discuss structural typing

Generic Programming

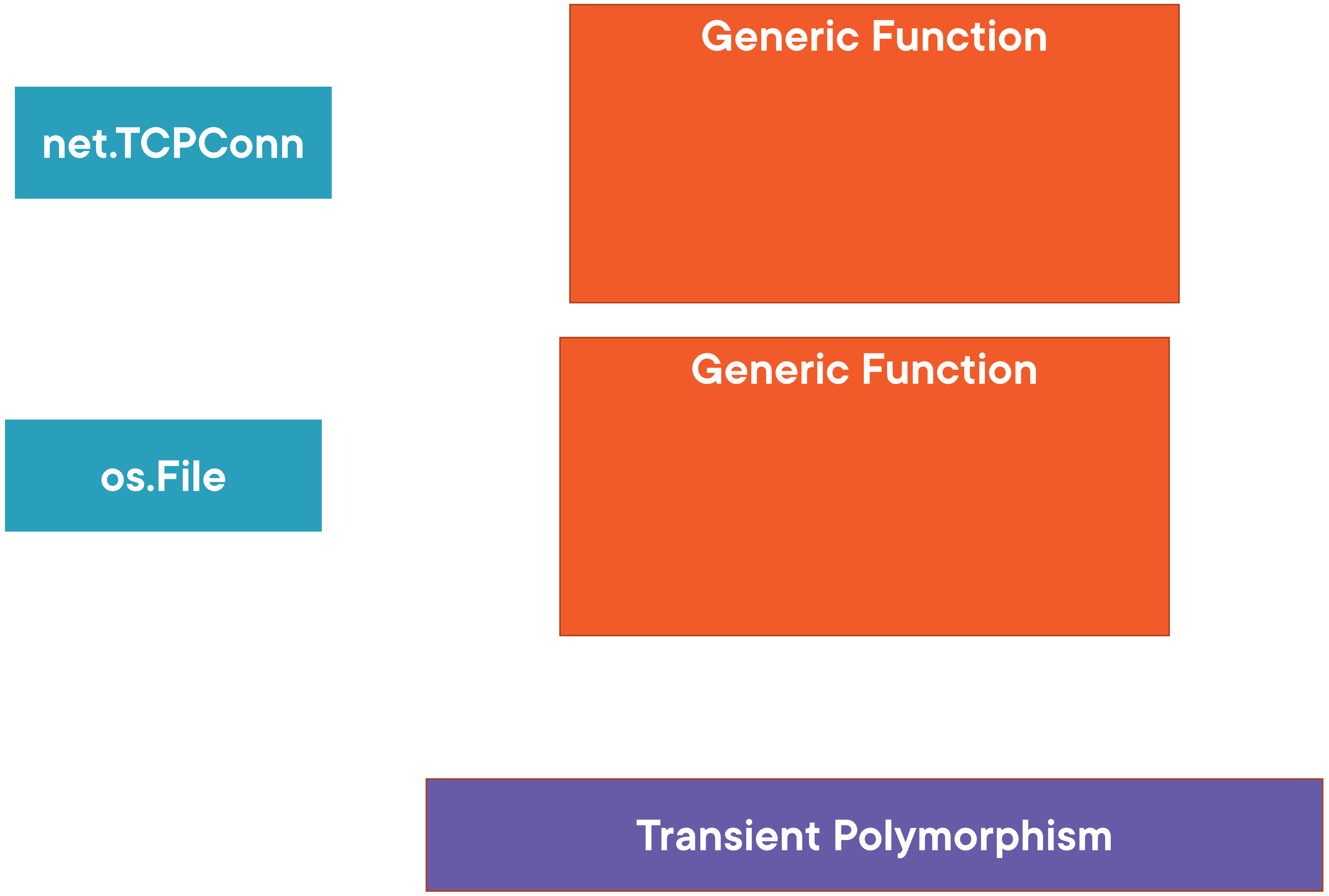
```
type Reader interface {
    Read([]byte) (int, error)
}

type File struct { ... }
func (f File)Read(b []byte) (n int, err error)

type TCPConn struct { ... }
func (t TCPConn) Read(b []byte) (n int, err error)

var f File
var t TCPConn

var r Reader
r = f
r = t
```

types lose identity!

Normal Interfaces



Generic Programming



Demo

Generics

generic clone for slice

Demo

Generics

generic clone for map

Demo

Generics

generic clone with type interface

```go
func foo[T any]() { ... }

func bar[T any, S any]() {...}

func baz[T any](in T) T {

     return in

}

fmt.Printf(“%T”, baz(3))    // int

fmt.Printf(“%T”, baz(true))  // bool


any

comparable
```

◀ create a function with a generic parameter ‘T’

◀ can use multiple generic types per function

◀ generics maintain type from consumer’s perspective

◀ matches any type, like interface{}

◀ matches types that can be compared

```go
type Addable interface {
    int | float64
}
func add[T Addable](){ ... }
```

◀ create a type interface

◀ used like other types as generic parameter

Useful Packages

golang.org/x/exp/constraints

golang.org/x/exp/slices

golang.org/x/exp/maps

Summary

Methods

Interfaces

Generic Programming