Satisfying Interfaces

This lesson discusses implicit interfaces and how Go interfaces are satisfied.

Implicit Interfaces

A type implements an interface by implementing the methods that it contains.

There is no explicit declaration of intent. The interfaces are satisfied implicitly through its methods.

Implicit interfaces decouple implementation packages from the packages that define the interfaces: neither depends on the other.

It also encourages the definition of **precise interfaces**, because you don't have to find every implementation and tag it with the new interface name.

Given below is an example implementation of Reader and Writer interfaces in Go:

```
package main

import (
        "fmt"
        "os"
)

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

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

type ReadWriter interface {
        Reader
```

```
	Writer
}


func main() {
	var w Writer

	// os.Stdout implements Writer
	w = os.Stdout

	fmt.Fprintf(w, "hello, writer\n")
}
```

Package io has Reader and Writer defined already so you don't have to.

Now that we have a clear idea of how interfaces in Go are implemented, we will look into how error messages can be displayed within them in the following lesson.