PES UNIVERSITY

CELEBRATING 50 YEARS

# Go Programming Language

## Reflection

Department of Computer Science and Engineering

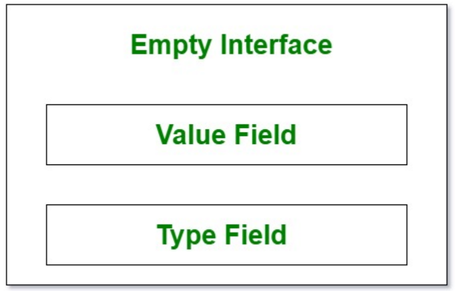
## Reflection

- Go provides a mechanism to update variables and inspect their values at run time, to call their methods, and to apply the operations intrinsic to their representation, all without knowing their types at compile time. This mechanism is called **reflection**.
- Reflection also lets us treat types themselves as first-class values
- Reflection is complex to reason about and not for casual use
- The **reflect package** offers all the required APIs/Methods for this purpose.
- Reflection is often termed as a method of **metaprogramming**

## Go Programming Language

## Empty interface

- The empty interface is extremely useful when we are declaring a function with unknown parameters and data types.
- Library methods such as Println, Printf take empty interfaces as arguments.
- The empty interface has certain hidden properties that give it functionality.
- The data is abstracted in the following way.



- An empty i o take in any argument and adapt to its value and data type. This includes but is not limited to **Structs and pointers to Structs.**

## Why Reflection?

- Sometimes we need to write a function capable of dealing uniformly with values of types that don’t satisfy a common interface, don’t have a known representation, or don’t exist at the time we design the function—or even all three.
- A familiar example is the formatting logic within fmt.Fprintf, which can usefully print an arbitrary value of any type, even a user-defined one.
- Without a way to inspect the representation of values of unknown types, we quickly get stuck.
- Often times the data passed to the empty interfaces are not primitives. They might be structs. We need to perform procedures on such data without knowing their type or the values present in it.
- In such a situation in order to perform various operations on the struct, such as interpreting the data present in it we need to know the types present in it as well as the number of fields. These problems can be dealt with during **run-time using reflection.**

## Reflect package

- Reflection is provided by the reflect package.
- It defines two important types, Type and Value.
- A Type represents a Go type. It is an interface with many methods for discriminating among types and inspecting their components, like the fields of a struct or the parameters of a function.
- The **reflect.TypeOf** function accepts any interface{} and returns its dynamic type as a reflect.Type
- A **reflect.Value** can hold a value of any type.
- The reflect.ValueOf function accepts any interface{} and returns a reflect.Value containing the interface's dynamic value.
- The inverse operation to reflect.ValueOf is the reflect.Value.Interface method. It returns an interface{} holding the same concrete value as the reflect.Value

## Reflect package

The reflect package offers us a number of other methods:

- **NumField():** This method returns the number of fields present in a struct. If the passed argument is not of the kind reflect.Struct then it panics.
- **Field():** This method allows us to access each field in the struct using an Indexing variable.
- **Copy(): This method** copies the contents of source into destination until either destination has been filled or source has been exhausted.
- **DeepEqual()**: This method returns True or False whether x and y are "deeply equal" or not. Array values are deeply equal when their corresponding elements are deeply equal. Struct values are deeply equal if their corresponding fields, both exported and un-exported, are deeply equal
- **Swapper()**: This method is used to swap the elements in the provided slice. You can use this function to reverse or sort the slice

## Go Programming Language
## Reflect package

The reflect package offers us a number of other methods:

- **FieldByIndex():** This method is used to get the nested field corresponding to index.
- **FieldByName()**: This method is used to get and set the struct field value by given field name.

PES UNIVERSITY

CELEBRATING 50 YEARS

## THANK YOU

**Suresh Jamadagni**

Department of Computer Science and Engineering