Methods

Methods are functions that give data the ability to exhibit behavior.

Notes

* Methods are functions that declare a receiver variable.
* Receivers bind a method to a type and can use value or pointer semantics.
* Value semantics mean a copy of the value is passed across program boundaries.
* Pointer semantics mean a copy of the values address is passed across program boundaries.
* Stick to a single semantic for a given type and be consistent.

Code Review

[Declare and receiver behavior](example1/example1.go)
[Value and Pointer semantics](example5/example5.go)
[Named typed methods](example2/example2.go)
[Function/Method variables](example3/example3.go)
[Function Types](example4/example4.go)

Interfaces

Interfaces provide a way to declare types that define only behavior. This behavior can be implemented by concrete types, such as struct or named types, via methods. When a concrete type implements the set of methods for an interface, values of the concrete type can be assigned to variables of the interface type. Then method calls against the interface value actually call into the equivalent method of the concrete value. Since any concrete type can implement any interface, method calls against an interface value are polymorphic in nature.

Notes

* The method set for a value, only includes methods implemented with a value receiver.
* The method set for a pointer, includes methods implemented with both pointer and value receivers.
* Methods declared with a pointer receiver, only implement the interface with pointer values.
* Methods declared with a value receiver, implement the interface with both a value and pointer receiver.
* The rules of method sets apply to interface types.
* Interfaces are reference types, don't share with a pointer.
* This is how we create polymorphic behavior in go.

Code Review

[Polymorphism](example1/example1.go)
[Method Sets](example2/example2.go)
[Address Of Value](example3/example3.go)
[Storage By Value](example4/example4.go)
[Type Assertions](example5/example5.go)
[Conditional Type Assertions](example6/example6.go)
[The Empty Interface and Type Switches](example7/example7.go)

Embedding

Embedding types provide the final piece of sharing and reusing state and behavior between types. Through the use of inner type promotion, an inner type's fields and methods can be directly accessed by references of the outer type.

Notes

* Embedding types allow us to share state or behavior between types.
* The inner type never loses its identity.
* This is not inheritance.
* Through promotion, inner type fields and methods can be accessed through the outer type.
* The outer type can override the inner type's behavior.

Code Review

[Declaring Fields](example1/example1.go)
[Embedding types](example2/example2.go)
[Embedded types and interfaces](example3/example3.go)
[Outer and inner type interface implementations](example4/example4.go)

Exporting

Packages contain the basic unit of compiled code. They define a scope for the identifiers that are declared within them. Exporting is not the same as public and private semantics in other languages. But exporting is how we provide encapsulation in Go.

Notes

* Code in go is compiled into packages and then linked together.
* Identifiers are exported (or remain unexported) based on letter-case.
* We import packages to access exported identifiers.
* Any package can use a value of an unexported type, but this is annoying to use.

Code Review

[Declare and access exported identifiers - Pkg](example1/counters/counters.go)
[Declare and access exported identifiers - Main](example1/example1.go)
[Declare unexported identifiers and restrictions - Pkg](example2/counters/counters.go)
[Declare unexported identifiers and restrictions - Main](example2/example2.go)
[Access values of unexported identifiers - Pkg](example3/counters/counters.go)
[Access values of unexported identifiers - Main](example3/example3.go)
[Unexported struct type fields - Pkg](example4/users/users.go)
[Unexported struct type fields - Main](example4/example4.go)
[Unexported embedded types - Pkg](example5/users/users.go)
[Unexported embedded types - Main](example5/example5.go)