Module 4

INTERFACES & OBJECT-ORIENTED PATTERNS

INTERFACE

DEFINITION

- A type that specifies a set of method signatures but provides no implementations.
- It defines behavior without specifying how that behavior is implemented.

IMPLICIT IMPLEMENTATION

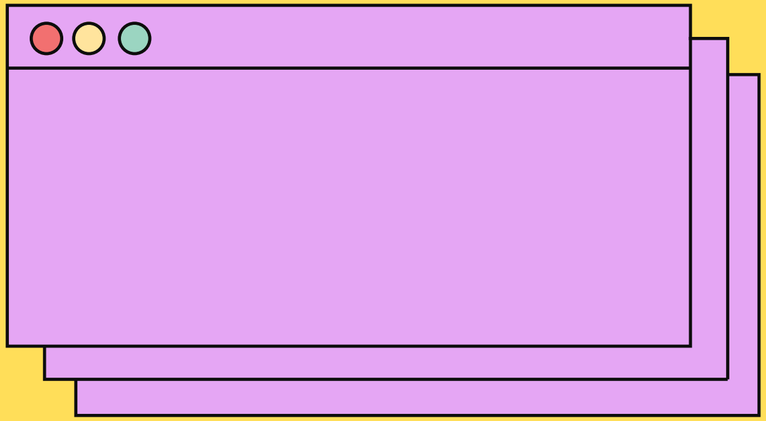
- Go interfaces are satisfied implicitly.
- Any type that implements all methods of an interface automatically satisfies that interface (no "implements" keyword required).

EMPTY INTERFACE

- An interface with no methods.
- It can hold values of any type since every type implements at least zero methods.

Type Assertion

- The operation value.(Type) that extracts the underlying value of a specific type from an interface value.



EMBEDDING

- Including one struct or interface type within another to inherit its fields and methods.

COMPOSITION

- A design principle where complex types are built by combining simpler ones, rather than using inheritance hierarchies.

Polymorphism

- The ability for different types to be treated as instances of the same type through a common interface.

Interface-based Polymorphism

- In Go, polymorphism is achieved by defining behavior through interfaces and implementing those interfaces for different types.

DESIGN PATTERNS

- Factory Pattern: A creational pattern that uses a function to create and return instances of different types that implement the same interface.

STRATEGY PATTERN

- A behavioral pattern that defines a family of algorithms, encapsulates each one, and makes them interchangeable at runtime.

STANDARD LIBRARY INTERFACES

- **io.Writer**: Interface for types that can write bytes.
- **io.Reader**: Interface for types that can read bytes.
- **http.Handler**: Interface for types that can serve HTTP requests.
- **sort.Interface**: Interface for types that can be sorted using the sort package.

THANK YOU