Type is a tool that helps define new things. “Things” can be new datatypes, structures, methods.

Type aliases Let’s us give more meaningful names to existing types.

Interfaces lets us define a contract that types must fulfil. Eg: Any type that has a Speak() can be treated as a speaker.

If the function or variable name starts with a lowercase letter then it is treated as a private function or variable. If it is uppercase then as a public function or variable.

func NewTCPPeer(conn net.Conn, outbound bool) \*TCPPeer {

    return &TCPPeer{

        conn:     conn,

        outbound: outbound,

    }

}

Func: A keyword that defines a function in Go.

NewTCPPeer: Name of the function. New”function name” this indicates the it is creating a new instance of the function.

Conn, outbound: These are the parameters of type net.conn(represents a network connection) and bool(represents outbound or inbound).

\*TCPPeer: returns a pointer to TCPPeer struct. “\*” is used to return a reference, not the actual value.

&TCPPeer: Used to create a new instance of the TCPPeer struct and return a pointer to it. “&” is the address of the operator. Meaning we are returning the newly created TCPPeer rather than a copy.