```go
package clients

import "sync"

type InterfaceID interface {
    Next() uint64
    CurrentID() uint64
}

func NewID() *ID {
    return &ID{}
}
//Id class
type ID struct {
    id uint64
    lock sync.Mutex
}
//Next add microcontroller to server
//the process is locked for memory lock
func (i *ID) Next() uint64 {
    i.lock.Lock()
    //var ID int8
    i.id++

    defer i.lock.Unlock()
    return i.id
}
func (i *ID) CurrentID() uint64 {
    i.lock.Lock()
    defer i.lock.Unlock()
    return i.id
}

```

```
 1 package clients
 2
 3 import (
 4     "sync"
 5 )
 6 type cliMap map[uint64]*Client
 7 type InterfaceClient interface {
 8     PairClient(id InterfaceID) *Client
 9     GetAllClients() *map[uint64]*Client
10     GetAClient (id uint64) *Client
11 }
12
13 type Service struct {
14     m sync.Mutex
15     clients map[uint64]*Client
16 }
17
18 func Init() *Service {
19     return &Service{
20         clients: make(map[uint64]*Client),
21     }
22 }
23
24 func InterfaceConnection() (InterfaceID,
   InterfaceClient) {
25     idObject := InterfaceID(NewID())
26
27     server := InterfaceClient(Init())
28     return idObject,server
29 }
30
31
32 func (s *Service) PairClient(id InterfaceID) *
   Client {
33     s.m.Lock()
34     defer s.m.Unlock()
35     cli := new(Client)
36     cli.ID = id.Next()
37     //cli.name= "name"
38     s.clients[cli.ID] = cli
39     //c := make(cliMap)
40     //c[cli.ID] = cli
41     return s.clients[cli.ID]
42 }
```

```
43
44 func (s *Service) GetAllClients() *map[uint64]*
   Client {
45     return &s.clients
46 }
47
48 func (s *Service) GetAClient (id uint64) *Client{
49     return s.clients[id]
50 }
51
52 //func (s *Service) UpdateClient(ID uint64, key
   string, value interface{})  {
53 //  //m := make(map[uint64]*client)
54 //  //cli := new(client)
55 //
56 //  j := s.clients[ID]
57 //  j.
58 //  client[key] = value
59 //}
```

```go
package clients

import "time"

type Client struct {
    ID   uint64
    Name string
    Secure bool
    Duration time.Time
}

```

```go
package clients

import (
    "fmt"
    "github.com/stretchr/testify/assert"
    "sync"
    "testing"
)


func TestID_Next(t *testing.T) {
    idObject := InterfaceID(NewID())
    var wg sync.WaitGroup

    for i := 0; i < 50; i++ {
        wg.Add(1)
        go func() {
            id := idObject.Next()
            assert.IsType(t, uint64(5), id)
            //fmt.Println(id)
            wg.Done()
        }()
    }
    wg.Wait()
    assert.NotNil(t, idObject)
}

func TestID_CurrentID(t *testing.T) {
    idObject := InterfaceID(NewID())
    var wg sync.WaitGroup

    for i := 0; i < 10; i++ {
        wg.Add(1)
        go func() {
            _= idObject.Next()
            id := idObject.CurrentID()
            assert.IsType(t, uint64(5), id)
            fmt.Printf("[%v] value of ID : %v \n is
    ",i, id)
            wg.Done()
        }()
    }
    wg.Wait()
    assert.NotNil(t, idObject)
```

```
44
45 }
```

```go
package clients

import (
    "fmt"
    "github.com/stretchr/testify/assert"
    "sync"
    "testing"
)


func TestService_PairClient(t *testing.T) {
    idObject, server := InterfaceConnection()
    var wg sync.WaitGroup
    //typeMap := cliMap{}
    cliCheck := &Client{}
    for i := 0; i < 10; i++ {
        wg.Add(1)
        go func() {
            cli := server.PairClient(idObject)
            fmt.Printf("[%v] value of ID : %v \n is
 ", i,cli)
            assert.IsType(t, cliCheck, cli)
            wg.Done()
        }()
    }
    wg.Wait()
    assert.NotNil(t, server, idObject)

}

func TestService_GetAllClients(t *testing.T) {
    idObject, server := InterfaceConnection()
    var wg sync.WaitGroup
    mapSize := 10
    //typeMap := cliMap{}
    cliCheck := &Client{}
    for i := 0; i < mapSize; i++ {
        wg.Add(1)
        go func() {
            cli := server.PairClient(idObject)

            assert.IsType(t, cliCheck, cli)
            wg.Done()
        }()
```

```
44      }
45      wg.Wait()
46      clients := server.GetAllClients()
47      assert.NotNil(t, server, idObject)
48      for k,v := range *clients{
49          assert.NotNil(t, v, k)
50          fmt.Printf("value of ID {%v} is %v \n  ",k,
    v)
51          if k > uint64(mapSize) {
52
53              t.Error("error in map size")
54          }
55      }
56
57  }
58
59  func TestService_GetAClient(t *testing.T) {
60      idObject, server := InterfaceConnection()
61      var wg sync.WaitGroup
62      cliCheck := &Client{}
63      for i := 0; i < 10; i++ {
64          wg.Add(1)
65          go func() {
66              cli := server.PairClient(idObject)
67              //fmt.Printf("[%v] value of ID : %v \n
    is ", i,cli)
68              assert.IsType(t, cliCheck, cli)
69              wg.Done()
70          }()
71      }
72      wg.Wait()
73      client := server.GetAClient(3)
74      fmt.Println(client)
75      assert.NotNil(t, server, idObject, client)
76
77  }
```