7 - Channels and Go Routines

Website Status Checker

Non concurrent method

Main.go

```go
package main

import (
    "fmt"
    "net/http"
)

func main() {
    links := []string{
        "http://google.com",
        "http://facebook.com",
        "http://stackoverflow.com",
        "http://golang.org",
        "http://amazon.com",
    }

    for _, link := range links {
        checkLink(link)
    }
}

func checkLink(link string) {
    _, err := http.Get(link)
    if err != nil {
        fmt.Println(link, "might be down!")
        return
    }

    fmt.Println(link, "is up!")
}
```

Receiving Messages

main.go

```go
package main

import (
    "fmt"
    "net/http"
```

```
)

func main() {
    links := []string{
        "http://linkedin.com",
        "http://google.com",
        "http://facebook.com",
        "http://stackoverflow.com",
        "http://golang.org",
        "http://amazon.com",
    }

    c := make(chan string)

    for _, link := range links {
        go checkLink(link, c)
    }

    for i := 0; i < len(links); i++ {
        fmt.Println(<-c)
    }
}

func checkLink(link string, c chan string) {
    _, err := http.Get(link)
    if err != nil {
        fmt.Println(link, "might be down!")
        c <- "Might be down I think"
        return
    }

    fmt.Println(link, "is up!")
    c <- "Yep it's up"
}
```

Repeating Routines

main.go

```
package main

import (
    "fmt"
    "net/http"
)

func main() {
    links := []string{
        "http://linkedin.com",
        "http://google.com",
        "http://facebook.com",
        "http://stackoverflow.com",
        "http://golang.org",
        "http://amazon.com",
    }
```

```
17
18      c := make(chan string)
19
20      for _, link := range links {
21          go checkLink(link, c)
22      }
23
24      for {
25          go checkLink(<-c, c)
26      }
27  }
28
29  func checkLink(link string, c chan string) {
30      _, err := http.Get(link)
31      if err != nil {
32          fmt.Println(link, "might be down!")
33          c <- link
34          return
35      }
36
37      fmt.Println(link, "is up!")
38      c <- link
39  }
```

Alternative Loop Syntax

main.go

```
1  package main
2
3  import (
4      "fmt"
5      "net/http"
6  )
7
8  func main() {
9      links := []string{
10         "http://linkedin.com",
11         "http://google.com",
12         "http://facebook.com",
13         "http://stackoverflow.com",
14         "http://golang.org",
15         "http://amazon.com",
16     }
17
18     c := make(chan string)
19
20     for _, link := range links {
21         go checkLink(link, c)
22     }
23
24     for l := range c {
25         go checkLink(l, c)
26     }
27 }
```

```
28
29 func checkLink(link string, c chan string) {
30     _, err := http.Get(link)
31     if err != nil {
32         fmt.Println(link, "might be down!")
33         c <- link
34         return
35     }
36
37     fmt.Println(link, "is up!")
38     c <- link
39 }
40
```

Function Literals

main.go

```
1  package main
2
3  import (
4      "fmt"
5      "net/http"
6      "time"
7  )
8
9  func main() {
10     links := []string{
11         "http://linkedin.com",
12         "http://google.com",
13         "http://facebook.com",
14         "http://stackoverflow.com",
15         "http://golang.org",
16         "http://amazon.com",
17     }
18
19     c := make(chan string)
20
21     for _, link := range links {
22         go checkLink(link, c)
23     }
24
25     for l := range c {
26         go func(link string) {
27             time.Sleep(5 * time.Second)
28             checkLink(link, c)
29         }(l)
30     }
31 }
32
33 func checkLink(link string, c chan string) {
34     _, err := http.Get(link)
35     if err != nil {
36         fmt.Println(link, "might be down!")
37         c <- link
```

```
        return
    }

    fmt.Println(link, "is up!")
    c <- link
}
```