Solution Review: Multiplication Table

This lesson contains the solution review for the exercise on Multiplication Table.

Here is the code to the challenge in the previous lesson:

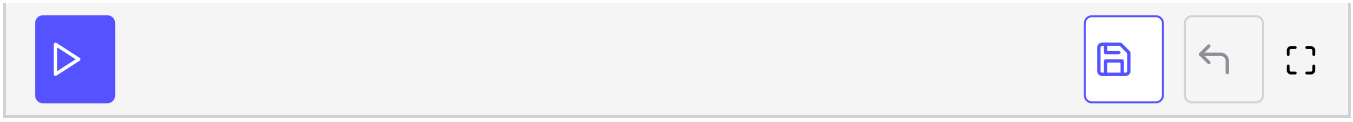
```
package main
import ( "fmt"
          "sync"
          "time")

func printTable(n int, wg *sync.WaitGroup) {
  for i := 1; i <= 12; i++ {
    fmt.Printf("%d x %d = %d\n", i, n, n*i)
    time.Sleep(50 * time.Millisecond)
  }
  wg.Done()
}

func main() {
  var wg sync.WaitGroup

  for number := 2; number <= 12; number++ {
    wg.Add(1)
    go printTable(number,&wg)
  }

  wg.Wait()
}
```



So we added a `WaitGroup` from the `sync` package on **line 15** named `wg`. At every iteration in the for-loop in the main routine, we increment the counter of `wg` on **line 18**. In the next step on **line 19**, we execute the `printTable` function in a goroutine and pass `number` and `wg` as input arguments. After launching all the goroutines for `number = 2` to `number = 12` in the for-loop which sets the counter of `wg` to `11`, we proceed to **line 22** and call `Wait()` on `wg`. This blocks the main routine until the counter of `wg` equals `0`. Hence, our main routine cannot exit until and unless we are done with the printing inside the `printTable` functions.

Let's see what's happening in the `printTable function`:

```
func printTable(n int, wg *sync.WaitGroup) {
  for i := 1; i <= 12; i++ {
    fmt.Printf("%d x %d = %d\n", i, n, n*i)
    time.Sleep(50 * time.Millisecond)
  }
  wg.Done()
}
```

We cause a delay of 50ms in each iteration of the for-loop on **line 9**. This implies that all the goroutines, when running concurrently, will print out their first iteration where `i` equals `1` before moving on to the next iteration. This ensures an order where all the prints with `i` equal `1` will be printed before `i` equals `2` and so on. Finally, we call `Done` on `wg` as we get done with the goroutine to decrement the counter of `wg` (**line 11**).

Hope everything makes sense now! In the next lesson, we will learn about mutexes in Go.