Go

User Defined Types

In this lab we will explore Go structs.

1. Basic JSON file I/O

Enter the following program:

```
user@ubuntu:~/go/src/lab-program-construction$ cd

user@ubuntu:~$
```

```
user@ubuntu:~$ mkdir ~user/go/src/lab-user-defined-types

user@ubuntu:~$


user@ubuntu:~$ cd ~user/go/src/lab-user-defined-types/

user@ubuntu:~/go/src/lab-user-defined-types$


user@ubuntu:~/go/src/lab-user-defined-types$ vim user-defined-types.go
user@ubuntu:~/go/src/lab-user-defined-types$ cat user-defined-types.go

package main

import  "os"

func main() {
        buf := []byte(`[{"title":"Hello", "subject":"world"}]`)

        f, _ := os.Create("output.txt")
        f.Write(buf[:])
        f.Close()
}
user@ubuntu:~/go/src/lab-user-defined-types$
```

Test run the above program and verify that the expected file is created.

```
user@ubuntu:~/go/src/lab-user-defined-types$ go run user-defined-types.go

user@ubuntu:~/go/src/lab-user-defined-types$


user@ubuntu:~/go/src/lab-user-defined-types$ ls
```

```
output.txt  user-defined-types.go
user@ubuntu:~/go/src/lab-user-defined-types$



user@ubuntu:~/go/src/lab-user-defined-types$ cat output.txt && echo
[{"title":"Hello", "subject":"world"}]
user@ubuntu:~/go/src/lab-user-defined-types$
```

We can use the `jq` utility to display formatted JSON:

```
user@ubuntu:~/go/src/lab-user-defined-types$ sudo apt install jq
...

user@ubuntu:~/go/src/lab-user-defined-types$



user@ubuntu:~/go/src/lab-user-defined-types$ cat output.txt | jq .

[
  {
    "title": "Hello",
    "subject": "world"
  }
]
user@ubuntu:~/go/src/lab-user-defined-types$
```

Run `go doc` on the os package. Note that Create creates a File struct and returns a pointer to it. Now run `go doc` on the File struct type, `go doc os.File`. This lists the methods associated with the struct. As we will see shortly, functions can be given a receiver, a struct to operate on. This is Go's way of creating methods.

The File struct has various methods, including the two we used above, Write(), and Close().

2. Create a named struct type

Using the course materials as a guide, create a simple struct type called Car. Cars should have the following fields:

- Make
- Model
- Odo //short for odometer
- Year
- Color

Use appropriate types for each field.

Create an instance of the struct and display it to stdout.

3. Build a data structure

Create a program that creates a map with `year` keys. Initialize it with three years; 1969, 2011, & 2015. Each year key should have an array of cars from that year as its value. Create two sample cars for each year.

Using only the map[Year][]Car display all of the cars by year.

Add a second map with Make keys and Car values (map[Make]Car). Initialize this map with the same cars referenced in the Year map. Design the program such that each car is create only once.

Display the Cars by year using the year map and by Make using the Make map (enumerate via range).

4. Save to JSON

Modify the prior program such that it outputs cars by year in JSON format. When you have this working, modify the program to send the output to a file.

Congratulations you have completed the lab!!