

Go lang

Prattya Datta



Task organization

Tasks

Read project description	X							
Git Repo	X							
Create files	X							
Install Go	X							
Theoretical part								X
Http status code		X						

Return html			X					
Return xml				X				
Return json				X				
Return image					X			
Return plain text						X		
Documentation							X	X

Knowledge learned

Variables

```
var integer1 int = 15
```

```
var integer2 int8 = -25
```

```
var integer3 int32
```

```
var float1 float32 = 63.2
```

```
var string1 string = "Hello World!"
```

```
var boolean1 bool
```

```
var boolean2 bool = true
```

Pointers

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     i, j := 42, 2701
7
8     p := &i          // point to i
9     fmt.Println(*p) // read i through the pointer
10    *p = 21           // set i through the pointer
11    fmt.Println(i)   // see the new value of i
12
13    p = &j           // point to j
14    *p = *p / 37     // divide j through the pointer
15    fmt.Println(j)   // see the new value of j
16 }
17 |
```

Functions

```
1 package main
2
3 import "fmt"
4
5 func add(x int, y int) int {
6     return x + y
7 }
8
9 func main() {
10     fmt.Println(add(42, 13))
11 }
12
```

Conditionals

```
package main

import "fmt"

func main() {

    if 7%2 == 0 {
        fmt.Println("7 is even")
    } else {
        fmt.Println("7 is odd")
    }

    if 8%4 == 0 {
        fmt.Println("8 is divisible by 4")
    }

    if num := 9; num < 0 {
        fmt.Println(num, "is negative")
    } else if num < 10 {
        fmt.Println(num, "has 1 digit")
    } else {
        fmt.Println(num, "has multiple digits")
    }

}
```


Loops

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     sum := 0
7     for i := 0; i < 10; i++ {
8         sum += i
9     }
10    fmt.Println(sum)
11 }
12
```

Arrays and slices

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var a [2]string
7     a[0] = "Hello"
8     a[1] = "World"
9     fmt.Println(a[0], a[1])
10    fmt.Println(a)
11
12    primes := [6]int{2, 3, 5, 7, 11, 13}
13    fmt.Println(primes)
14 }
15
```

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     primes := [6]int{2, 3, 5, 7, 11, 13}
7
8     var s []int = primes[1:4]
9     fmt.Println(s)
10 }
11
```

Maps

```
package main

import "fmt"

func main() {

    m := make(map[string]int)

    m["k1"] = 7
    m["k2"] = 13

    fmt.Println("map:", m)

    v1 := m["k1"]
    fmt.Println("v1: ", v1)

    fmt.Println("len:", len(m))

    delete(m, "k2")
    fmt.Println("map:", m)

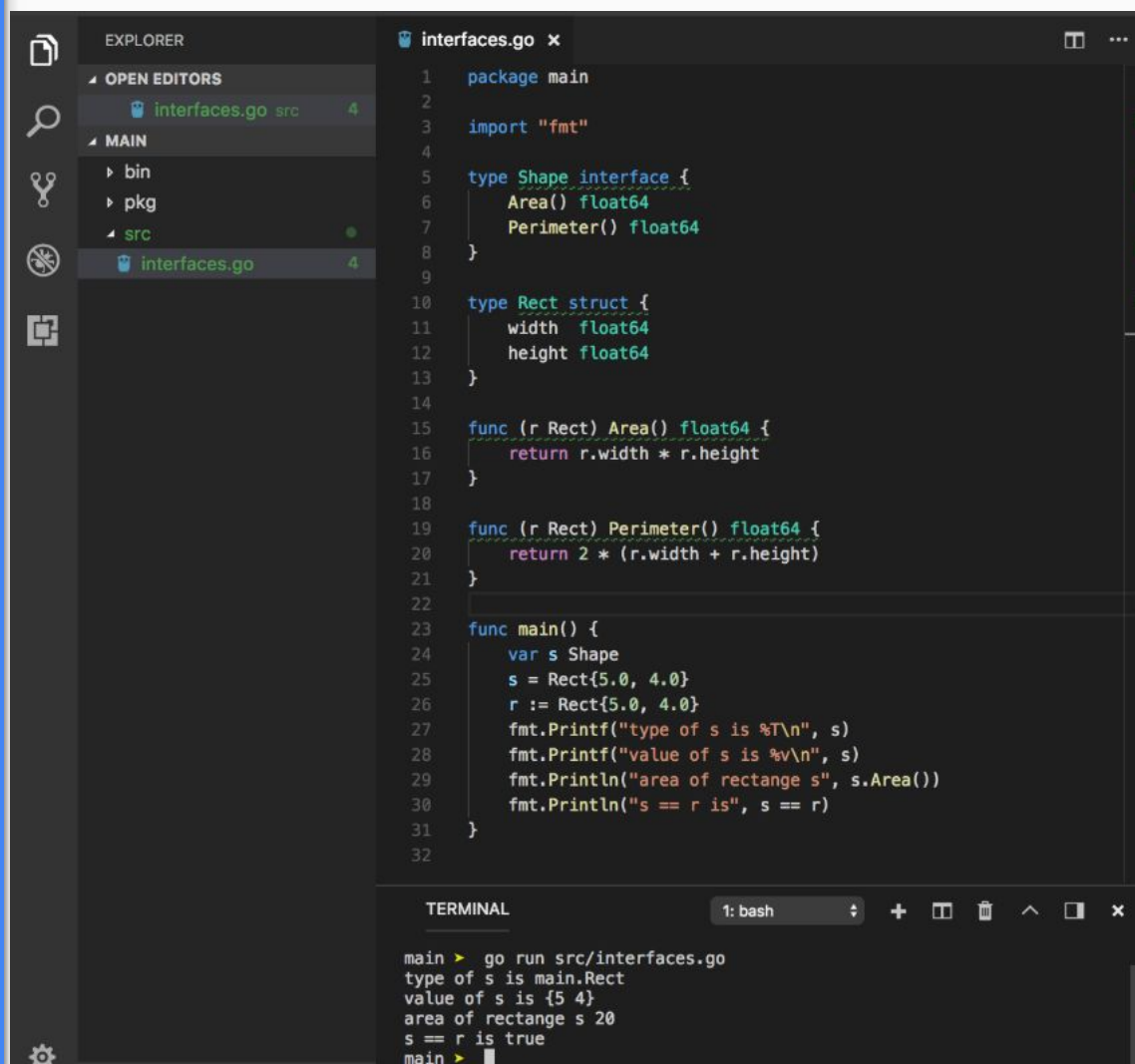
    _, prs := m["k2"]
    fmt.Println("prs:", prs)


    n := map[string]int{"foo": 1, "bar": 2}
    fmt.Println("map:", n)
}
```

Structs

```
1 package main
2
3 import "fmt"
4
5 type Vertex struct {
6     X int
7     Y int
8 }
9
10 func main() {
11     fmt.Println(Vertex{1, 2})
12 }
13
```

Interfaces



The screenshot displays a Go IDE interface. On the left, the EXPLORER sidebar shows a project structure with folders 'bin', 'pkg', and 'src'. The 'src' folder is expanded, showing the file 'interfaces.go'. The main editor window displays the code for 'interfaces.go'. The code defines a 'Shape' interface with 'Area()' and 'Perimeter()' methods, a 'Rect' struct with 'width' and 'height' fields, and a 'main' function that creates a 'Rect' object and prints its type, value, area, and whether it implements the 'Shape' interface. The terminal window at the bottom shows the output of running 'go run src/interfaces.go'.

```
EXPLORER
└─ OPEN EDITORS
   └─ interfaces.go src 4
      └─ MAIN
         ├── bin
         ├── pkg
         └─ src
            └─ interfaces.go 4
```

```
interfaces.go
1  package main
2
3  import "fmt"
4
5  type Shape interface {
6      Area() float64
7      Perimeter() float64
8  }
9
10 type Rect struct {
11     width  float64
12     height float64
13 }
14
15 func (r Rect) Area() float64 {
16     return r.width * r.height
17 }
18
19 func (r Rect) Perimeter() float64 {
20     return 2 * (r.width + r.height)
21 }
22
23 func main() {
24     var s Shape
25     s = Rect{5.0, 4.0}
26     r := Rect{5.0, 4.0}
27     fmt.Printf("type of s is %T\n", s)
28     fmt.Printf("value of s is %v\n", s)
29     fmt.Println("area of rectangle s", s.Area())
30     fmt.Println("s == r is", s == r)
31 }
32
```

```
1: bash
main > go run src/interfaces.go
type of s is main.Rect
value of s is {5 4}
area of rectangle s 20
s == r is true
main >
```

Mistakes

```
errors.go
1 package main
2
3 import (
4     "fmt"
5     "time"
6 )
7
8 type MyError struct {
9     When time.Time
10    What string
11 }
12
13 func (e *MyError) Error() string {
14     return fmt.Sprintf("at %v, %s",
15         e.When, e.What)
16 }
17
18 func run() error {
19     return &MyError{
20         time.Now(),
21         "it didn't work",
22     }
23 }
24
25 func main() {
26     if err := run(); err != nil {
27         fmt.Println(err)
28     }
29 }
30
```

Conclusions

Conclusions and recommendations

- GO-lang is very powerful and has ability to optimize codes greatly with its easy use of adding numerous developer functionalities, not possible in may other web development tools like pointers
- However, go still needs lot of development to counter the popularity of javascript