Golang Session

- Harsh Dusane

Topic : Golang struct

Golang struct

- A struct (short for "structure") is a collection of data fields with declared data types. Golang has the ability to declare and create own data types by combining one or more types, including both built-in and user-defined types. Each data field in a struct is declared with a known type, which could be a built-in type or another user-defined type.

Golang struct

- The declaration starts with the keyword type, then a name for the new struct, and finally the keyword struct. Within the curly brackets, a series of data fields are specified with a name and a type.

```
type identifier struct{
  field1 data_type
  field2 data_type
  field3 data_type
}
```

Declaration of a struct type

```
package main

import "fmt"

type rectangle struct {
        length  float64
        breadth float64
        color   string
}

func main() {
        fmt.Println(rectangle{10.5, 25.10, "red"})
}
```

Creating Instances of Struct Types

```
package main

import "fmt"

type rectangle struct {
        length  int
        breadth int
        color   string

        geometry struct {
                area      int
                perimeter int
        }
}

func main() {
        var rect rectangle // dot notation
        rect.length = 10
        rect.breadth = 20
        rect.color = "Green"

        rect.geometry.area = rect.length * rect.breadth
        rect.geometry.perimeter = 2 * (rect.length + rect.breadth)

        fmt.Println(rect)
        fmt.Println("Area:\t", rect.geometry.area)
        fmt.Println("Perimeter:", rect.geometry.perimeter)
}
```

Creating a Struct Instance Using a Struct Literal

```
package main
import "fmt"
type rectangle struct {
        length  int
        breadth int
        color   string
}
func main() {
        var rect1 = rectangle{10, 20, "Green"}
        fmt.Println(rect1)
        var rect2 = rectangle{length: 10, color: "Green"} // breadth value skipped
        fmt.Println(rect2)
        rect3 := rectangle{10, 20, "Green"}
        fmt.Println(rect3)
        rect4 := rectangle{length: 10, breadth: 20, color: "Green"}
        fmt.Println(rect4)
        rect5 := rectangle{breadth: 20, color: "Green"} // length value skipped
        fmt.Println(rect5)
}
```

Struct Instantiation using new keyword

```
package main

import "fmt"

type rectangle struct {
        length  int
        breadth int
        color   string
}

func main() {
        rect1 := new(rectangle) // rect1 is a pointer to an instance of rectangle
        rect1.length = 10
        rect1.breadth = 20
        rect1.color = "Green"
        fmt.Println(rect1)

        var rect2 = new(rectangle) // rect2 is an instance of rectangle
        rect2.length = 10
        rect2.color = "Red"
        fmt.Println(rect2)
}
```

Struct Instantiation Using Pointer Address Operator

```
package main
import "fmt"
type rectangle struct {
        length  int
        breadth int
        color   string
}
func main() {
        var rect1 = &rectangle{10, 20, "Green"} // Can't skip any value
        fmt.Println(rect1)
        var rect2 = &rectangle{}
        rect2.length = 10
        rect2.color = "Red"
        fmt.Println(rect2) // breadth skipped
        var rect3 = &rectangle{}
        (*rect3).breadth = 10
        (*rect3).color = "Blue"
        fmt.Println(rect3) // length skipped
}
```

Use Field Tags in the Definition of Struct Type

```
package main

import (
    "fmt"
    "encoding/json"
)

type Employee struct {
    FirstName  string `json:"firstname"`
    LastName   string `json:"lastname"`
    City string `json:"city"`
}

func main() {
    json_string := `
    {
        "firstname": "Rocky",
        "lastname": "Sting",
        "city": "London"
    }`

    emp1 := new(Employee)
    json.Unmarshal([]byte(json_string), emp1)
    fmt.Println(emp1)

    emp2 := new(Employee)
    emp2.FirstName = "Ramesh"
    emp2.LastName = "Soni"
    emp2.City = "Mumbai"
    jsonStr, _ := json.Marshal(emp2)
    fmt.Printf("%s\n", jsonStr)
}
```

Add Method to Struct Type

```
package main
import "fmt"
type Salary struct {
        Basic, HRA, TA float64
}
type Employee struct {
        FirstName, LastName, Email string
        Age                       int
        MonthlySalary             []Salary
}
func (e Employee) EmpInfo() string {
        fmt.Println(e.FirstName, e.LastName)
        fmt.Println(e.Age)
        fmt.Println(e.Email)
        for _, info := range e.MonthlySalary {
                fmt.Println("==================")
                fmt.Println(info.Basic)
                fmt.Println(info.HRA)
                fmt.Println(info.TA)
        }
        return "----------------------"
}
```

Assign Default Value for Struct Field

```
package main
import "fmt"
type Employee struct {
        Name string
        Age  int
}
func (obj *Employee) Info() {
        if obj.Name == "" {
                obj.Name = "John Doe"
        }
        if obj.Age == 0 {
                obj.Age = 25
        }
}
func main() {
        emp1 := Employee{Name: "Mr. Fred"}
        emp1.Info()
        fmt.Println(emp1)
        emp2 := Employee{Age: 26}
        emp2.Info()
        fmt.Println(emp2)
}
```

Find Type of Struct in Go Programming Language

```
package main
import (
        "fmt"
        "reflect"
)
type rectangle struct {
        length  float64
        breadth float64
        color   string
}
func main() {
        var rect1 = rectangle{10, 20, "Green"}
        fmt.Println(reflect.TypeOf(rect1))         // main.rectangle
        fmt.Println(reflect.ValueOf(rect1).Kind()) // struct
        rect2 := rectangle{length: 10, breadth: 20, color: "Green"}
        fmt.Println(reflect.TypeOf(rect2))         // main.rectangle
        fmt.Println(reflect.ValueOf(rect2).Kind()) // struct
        rect3 := new(rectangle)
        fmt.Println(reflect.TypeOf(rect3))         // *main.rectangle
        fmt.Println(reflect.ValueOf(rect3).Kind()) // ptr
        var rect4 = &rectangle{}
        fmt.Println(reflect.TypeOf(rect4))         // *main.rectangle
        fmt.Println(reflect.ValueOf(rect4).Kind()) // ptr
}
```

Comparing Structs with the Different Values Assigned to Data Fields

```
package main
import "fmt"
type rectangle struct {
        length  float64
        breadth float64
        color   string
}
func main() {
        var rect1 = rectangle{10, 20, "Green"}
        rect2 := rectangle{length: 20, breadth: 10, color: "Red"}
        if rect1 == rect2 {
                fmt.Println("True")
        } else {
                fmt.Println("False")
        }
        rect3 := new(rectangle)
        var rect4 = &rectangle{}
        if rect3 == rect4 {
                fmt.Println("True")
        } else {
                fmt.Println("False")
        }
}
```

Copy Struct Type Using Value and Pointer Reference

```
package main
import "fmt"
type rectangle struct {
        length  float64
        breadth float64
        color   string
}
func main() {
        r1 := rectangle{10, 20, "Green"}
        fmt.Println(r1)
        r2 := r1
        r2.color = "Pink"
        fmt.Println(r2)
        r3 := &r1
        r3.color = "Red"
        fmt.Println(r3)
        fmt.Println(r1)
}
```