핵심! Go 프로그래밍

- 2일차

목차

- 포인터
- 구조체

포인터

- 다른 변수, 혹은 그 변수의 메모리 주소공간을 가리키는 변수
- 보통 c, c++등 하위레벨 언어에서 많이 사용됨
- 직접 메모리 주소를 조작할 수 있음 (golang은 안됨)
- * 키워드를 사용함

포인터

```go
package main

import "fmt"

func main(){
    var numPtr *int
    fmt.Println(numPtr)
    numPtr = new(int)
    fmt.Println(numPtr)
    i := 100
    numPtr = &i
    fmt.Println(*numPtr)
}
```

```
<nil>
0xc00000a0a8
100
```

```go
package main

import "fmt"

func main(){
    var p *[]int

    fmt.Println(p)

    a := make([]int, 5)
    fmt.Println(a)
    p = &a
    fmt.Println(*p)
    a[0] = 10
    a[1] = 20
    fmt.Println(*p)
}
```

```
<nil>
[0 0 0 0 0]
[0 0 0 0 0]
[10 20 0 0 0]
```

포인터

```go
package main

import "fmt"

func main(){
    var a =[5]int{1,2,3,4,5}
    fmt.Println(a)
    changeVal(a)
    fmt.Println(a)
    changeValPoint(&a)
    fmt.Println(a)
}
func changeVal(val [5]int){
    val[0] = 10

}
func changeValPoint(val *[5]int){
    val[0] = 10
}
```

```
[1 2 3 4 5]
[1 2 3 4 5]
[10 2 3 4 5]
```

```go
package main

import "fmt"

func main(){
    var p *int = new(int)

    p++
    p = 0x01

    fmt.Println(p)
}
```

```
# command-line-arguments
.\2day.go:8:3: invalid operation: p++ (non-numeric type *int)
.\2day.go:9:4: cannot use 1 (type int) as type *int in assignment
```

구조체

- 구조화된 데이터를 관리하는 자료구조
- 다양한 타입을 하나의 타입으로 관리 가능
- Class, Object와 비슷한 개념
- Struct 키워드를 사용

구조체

- Golang 구조체
  - Custom Data Type을 표현
  - 필드(맴버) 데이터만 가질 수 있음
  - OOP 구현을 위해 메서드를 분리해서 사용
  - 임베딩을 통한 상속 효과

- Golang 메서드
  - OOP 구현을 Golang의 고유 방식으로 지원
  - Value receiver, Pointer Receiver 방식

구조체

```go
package main

import "fmt"

type Person struct {
    gender bool
    age int
    name string
}
func main(){
    var p Person
    p.gender = true
    p.age = 33
    p.name = "taehoon"
    fmt.Println(p)
}
```

```
{true 33 taehoon}
```

```go
package main

import "fmt"

type Person struct {
    gender bool
    age int
    name string
}
func main(){

    var pp *Person
    pp = new(Person)
    fmt.Println(*pp)
    pp.gender = false
    pp.age = 23
    pp.name = "roh"
    fmt.Println(*pp)

}
```

```
<4 go setup calls>
{false 0 }
{false 23 roh}
```

구조체

```go
package main

import "fmt"

type Person struct {
    gender bool
    age int
    name string
}
func main(){
    p1 := Person{}
    fmt.Println(p1)
    p2 := new(Person)
    fmt.Println(*p2)
    p3 := make([]Person, 3)
    fmt.Println(p3)

}
```

```
{false 0 }
{false 0 }
[{false 0 } {false 0 } {false 0 }]
```

```go
package main

import "fmt"

type Person struct {
    gender bool
    age int
    name string
}
func main(){
    p1 := Person{}
    p1.name = "a";p1.age = 0; p1.gender = true;
    fmt.Println(p1)
    p2 := new(Person)
    p2 = &p1
    fmt.Println(*p2)
    p2.name = "b";p2.age = 1; p2.gender = false;
    fmt.Println(p1, *p2)
    p3 := make([]Person, 3)
    fmt.Println(p3)
    p3[0] = p1
    p3[1] = *p2
    fmt.Println(p3)
}
```

```
{true 0 a}
{true 0 a}
{false 1 b} {false 1 b}
[{false 0 } {false 0 } {false 0 }]
[{false 1 b} {false 1 b} {false 0 }]
```

구조체

```go
package main

import "fmt"

type Person struct {
    gender bool
    age int
    name string
}
func main(){
    p1 := Person{ gender: true,  age: 10,  name: "name"}
    fmt.Println(p1)
    p2 := Person{ gender: "test",  age: 1, name: false}
    fmt.Println(p2)

}
```

```
<3 go setup calls>
# command-line-arguments
.\2day.go:13:15: cannot use "test" (type untyped string) as type bool in field value
.\2day.go:13:25: cannot use false (type untyped bool) as type string in field value
```

```go
package main

import "fmt"

type Person struct {
    gender bool
    age    int
    name   string
}

func main() {
    p1 := Person{ gender: true,  age: 10,  name: "name"}
    fmt.Println(p1)
    p2 := Person{gender: false,
        age:  20,
        name: "ttt",
    }
    fmt.Println(p2)

}
```

```
{true 10 name}
{false 20 ttt}
```

구조체

```go
package main

import "fmt"

type Person struct {
    gender bool
    age    int
    name   string
}

func initPerson(gender bool, age int, name string) *Person{
    return &Person{ gender: gender,  age: age,  name: name}
}

func main() {
    p1 := Person{ gender: true,  age: 10,  name: "name"}
    fmt.Println(p1)

    p2 := initPerson( gender: true,  age: 10,  name: "test")
    fmt.Println(p2)

    p3 := &Person{ gender: false,  age: 20,  name: "test2"}
    fmt.Println(p3)

}
```

생성자(Constructor) 처럼 사용 가능

```
<4 go setup calls>
{true 10 name}
&{true 10 test}
&{false 20 test2}
```

구조체 메서드

```go
package main

import "fmt"

type Rectangle struct {
    width int
    height int
}

func (rect Rectangle) area() int{
    return rect.width * rect.width
}


func main() {
    rect := Rectangle{ width: 10,  height: 20}
    fmt.Println(rect.area())
}
```

```
100
```

```go
package main

import "fmt"

type Rectangle struct {
    width int
    height int
}
func (rect Rectangle) area() int{...}

func (rect *Rectangle) fix(w, h int) {
    rect.width = w
    rect.height = h
}


func main() {
    rect := Rectangle{ width: 10,  height: 20}
    fmt.Println(rect)
    rect.fix( w: 20,  h: 30)
    fmt.Println(rect)
}
```

```
{10 20}
{30 20}
```

구조체 메서드

```go
package main

import "fmt"

type Rectangle struct {
    width int
    height int
}
func (rect *Rectangle) fix(w, h int) {
    rect.width = w
    rect.height = h
}

func (rect Rectangle) fixVal(){
    rect.width = 100
    rect.height = 200
}


func main() {
    rect := Rectangle{ width: 10,  height: 20}
    fmt.Println(rect)
    rect.fix( w: 20,  h: 30)
    fmt.Println(rect)
    rect.fixVal()
    fmt.Println(rect)
}
```

```
{10 20}
{20 30}
{20 30}
```

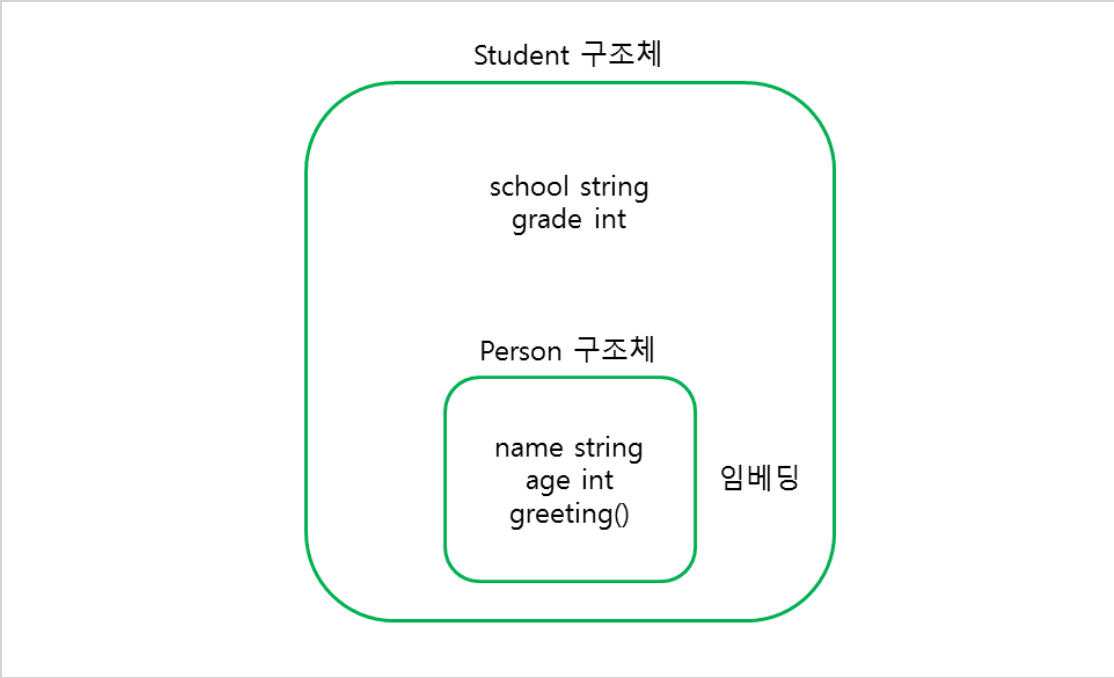
```go
package main

import "fmt"

type Rectangle struct {
    width int
    height int
}

func (_ Rectangle) dummy(){
    fmt.Println( a...: "dummy")
}


func main() {
    rect := Rectangle{ width: 10,  height: 20}
    fmt.Println(rect)
    rect.dummy()
}
```

```
{10 20}
dummy
```

구조체 임베딩

```go
package main

import "fmt"

type Person struct {
    name string
    age  int
}

func (p *Person) greeting() {
    fmt.Println( a...: "Hello~")
}

type Student struct {
    p      Person
    school string
    grade  int
}

func main() {
    var s Student
    s.p.greeting()
}
```

```
Hello~
```



출처: http://pyrasis.com/book/GoForTheReallyImpatient/Unit31

구조체 임베딩

```go
package main

import "fmt"

type Person struct {
    name string
    age  int
}

func (p *Person) greeting() {
    fmt.Println( a...: "Hello~")
}

type Student struct {
    Person
    school string
    grade  int
}

func main() {
    var s Student

    s.Person.greeting()
    s.greeting()
}
```

```
Hello~
Hello~
```

```go
package main

import "fmt"

type Person struct {
    name string
    age  int
}

func (p *Person) greeting() {
    fmt.Println( a...: "Hello~")
}

type Student struct {
    Person
    school string
    grade  int
}

func (s * Student) greeting(){
    fmt.Println( a...: "Hello Student~")
}

func main() {
    var s Student

    s.Person.greeting()
    s.greeting()
}
```

```
Hello~
Hello Student~
```