

apart 2
"First"

3
type Animal struct

id int
nombre string
edad int

4
func NewAnimal(id int, nombre string, edad
animal() return animal { id: id, nombre:
:nom, edad:edad })

5
func (a Animal) GetId() int {
return a.id }

fun (a Animal) GetNombre() string
return a.nombre

6
fun (a Animal) GetEdad() int {
return a.edad }

7

Metodo

func (a *Animal) setId(nuevoId int) {
a.id = nuevoId }

8

func (a *Animal) setNombre(nuevoNombre)
a.nombre = nuevoNombre

func Ca. (animal) setEdad (Creacion) int a.edad = edad

5.

func Ca (Animal) mostrarDatos () {

fmt. print ('id' + a.id, " nombre", a.nombre,
edad a.edad)

6.

from Ca. Animal) Comer (Comida string) {
fmt. printin C.nombre, 'a.nombre',
'la gesta comer', comida)

func Ca (Animal) comer (numero L int numero z int
int & return numero
acmez +

fun (a animal) toString() string d
return Fmt. Sprintf ("Id: %d; id, nombre
edads aid, a.nombre, a.edad)

(Main)

leon = NewAnimal (2, 'pedro', 5)
fmt. prints (f1 leon.)
leon.mostrarDatos ()

perro = NewAnimal (3, 'daniel', 4)
perro.comer ("Purina")

Fmt. Printin ("La suma es de", leon.suma
(2,6))

func calcular_poso_electrico_inicial

func calcular_poso_electrico_final

e

func calcular_poso_electrico (calcular_poso_electrico_final
* calcular_poso_electrico_inicial)

return calcular_poso_electrico

calcular_poso_electrico : calcular_poso_electrico
electricon_id, id,

g

g

func C.C. calcular_poso_electrico (calcular_poso_electrico_inicial)

s

func C.C. calcular_poso_electrico (calcular_poso_electrico_final)

f

func C.C. calcular_poso_electrico (calcular_poso_electrico_inicial)

f

C.C. calcular_poso_electrico = numero calculadora (posiva
calculadora (calculadora_poso_electrico))

boas !

func mult(float a, float b) {
 float result = a * b;
 return result;

func sum(float a, float b) {
 float result = a + b;
 return result;

func max(float a, float b) {
 if (a > b) return a;
 else return b;

func print(char c) {
 cout << c;

cout

return result;

return result;

func calculation(float a, float b) {
 float result = a + b;
 cout << "Result = " << result;

func calculator(float a, float b) {
 float result = a - b;

func calculator(float a, float b) {
 float result = a * b;

func calculator(float a, float b) {
 float result = a / b;

calculator = metacalculator

calculator

calculator = calculator

calculator = calculator

calculator = calculator

func mult(float a, float b) {
 float result = a * b;

func sum(float a, float b) {
 float result = a + b;

func max(float a, float b) {
 if (a > b) return a;
 else return b;

func min(float a, float b) {
 if (a < b) return a;
 else return b;

func print(char c) {
 cout << c;

cout

Calcular (θ , τ)

Type Person struct &

id int
edad int
genero bool

func GetId, edad int, genero bool) * person &
return person (id: id; edad: edad;
genero: genero)

func (P. person) GetID () int
return P. ~~id~~ id

5

func (P. person) Getedad () int &
return P. edad;

5

func (P. person) GetGenero () bool &
return P. genero

5

func (P. person) SetID (nuevoID INT) &
P. id = nuevoID

func (P. person) Setedad (nuevaEdad INT) &
P. edad = nuevaEdad.

Funt (P. x persona) n otratadot o n d

var g string
IF P. String

IF P. geneo d

g = "Masculino"

s e b e s

g = "Femenina"

S

Fmt.Println("10", id, 'edad', g)

Func (P. x persona) Resta (a int , b int) inf
resultado = a - b
return resultado

S

Funt (P. Persona) To json() atricog()
return Fmt.Sprintf("%d, edad, gmo")

type Samvel struct persona S

fun necesaria (id int, edad int
geneo bool) x'servir l

return Samvel (persona(id, edad, geneo))

type monica

type Maria struct {
 persona

s
func NecesariaMaria (int id, int edad, int
genero, bool) * Maria {
 return &Maria {persona(id, edad, genero), }

type Karol struct {

s func NecesarioKarol (int id, int edad,
genero, bool) * Karol {
 return &Karol {id, edad, genero}

type Pacna struct {
 persona

5
func NecesariaPacna (id int, color int, genero
bool) * Pacna {
 return &Pacna {persona
id}

Fmt. Print

Samuel = NecesarioSamuel (1, 25, true)
Samuel. mostrarDatos()

Fmt. Print ('ID' desea directa', samuel.id)

Maria = NecesariaMaria (21, false)
Fmt. Print (Maria). toJSON()

Karol = NecesarioKarol (3, 33, false)
Fmt. Print (Karol, to