Anonymous structures and fields in Golang

Anonymous structs in Golang are temporary structures with no names used for one-time purposes, while anonymous fields allow embedding of unnamed fields.

In Golang , structures (or structs) allow us to group elements of different types into a single unit, which is useful for modeling real-world entities. **Anonymous structures in Golang are temporary structures without names that are used for one-time purposes, while anonymous fields allow embedding of unnamed fields.**

For example:

```
package main import "fmt" // struct h?c sinh v?i c?u trúc và tr??ng ?
n danh type Student struct { struct { // C?u trúc bên trong ?
n danh cho thông tin cá nhân name string enrollment int } GPA float64 // Tr
??ng chu?
n } func main() { student := Student{ struct { name string enrollment int }{ nam
```

Syntax:

```
variable := struct { field1 dataType1 field2 dataType2 # C?u trúc ?
n danh // Tr??ng b? sung khi c?
n }{value1, value2} type StructName struct { dataType1 dataType2 # Tr??ng ?
n danh // Tr??ng ?n danh b? sung }
```

Anonymous Structures in Go

Anonymous structures in Go are defined without a name and are useful for creating temporary, disposable structures. Here is the syntax and code example.

Syntax:

```
variable := struct { field1 dataType1 field2 dataType2 // Các tr??ng b?
 sung khi c?n }{value1, value2}
```

For example:

```
package main import "fmt" // C?u trúc sinh viên v?i c?u trúc bên trong ?
n danh cho thông tin cá nhân type Student struct { personalDetails struct { // C
?u trúc ?
n danh bên trong cho thông tin cá nhân name string enrollment int } GPA float64 /
??ng chu?n } func main() { // Kh?i t?o c?
u trúc bên trong cho student student := Student{ personalDetails: struct { name s
?n giá tr?
 fmt.Println("Name:", student.personalDetails.name) fmt.Println("Enrollment:", st
```

Result:

```
Name: A Enrollment: 12345 GPA: 3.8
```

This code defines a Student structure with an anonymous *personalDetails* structure inside, storing the name and registration information. It then initializes *student* with values ??for these fields and prints them out.

Anonymous fields

Anonymous fields in Go allow you to define fields without explicit names, only their types. This is useful when fields naturally follow the type name.

Syntax

```
type StructName struct { dataType1 dataType2 // Additional anonymous fields }
```

For example:

```
package main import "fmt" // C?u trúc h?c sinh b?ng các tr??ng ?
n danh type Student struct { int // S? ??ng ký (tr??ng ?
n danh) string // Tên tr??ng ?n danh float64 // GPA (tr??ng ?
n danh) } func main() { // Kh?i t?o struct h?c sinh v?i các tr??ng ?
n danh student := Student{12345, "A", 3.8} // Hi?n giá tr?
 fmt.Println("Enrollment:", student.int) fmt.Println("Name:", student.string) fmt
```

Result:

```
Enrollment: 12345 Name: A GPA: 3.8
```

Here, the data types (*int*, *string*, *float64*) act as field names, so accessing the values ??depends on the types.

Important Points to Remember About Anonymous Fields in Golang

1. Unique requirement: You cannot use two fields of the same type in a structure. For example:

```
type InvalidStudent struct { int int // Error: duplicate type }
```

2. Combining named and anonymous fields: You can combine anonymous and named fields in a structure.

```
type Student struct { id int // Named field int // Anonymous field }
```

You finished reading the article "**Anonymous structures and fields in Golang**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.