Go Programming - OOP Part II

Concepts of Programming Languages
24 October 2019

Johannes Weigend
Rosenheim Technical University

Interfaces, Embedding and Polymorphism

- Go supports embedding of other structs inside a struct
- In Java this is called delegation.
- Syntactically it is similar to inheritance in Java
- Access to embedded field is identical to a normal field inside a struct
- Polymorphism is only supported by using interfaces

Interfaces, Embedding and Polymorphism

```
func main() {
    var p = Point{1, 2}
    var cp = ColorPoint{Point{1, 2}, 3}

    fmt.Println(p)
    fmt.Println(cp)
    fmt.Println(cp.x) // access inherited field

    // p = cp does not work: No type hierarchy, no polymorphism

    // s is a interface and supports Polymorphism
    var s fmt.Stringer
    s = p
    fmt.Println(s.String())
    s = cp
    fmt.Println(s.String())
}
```

Run

Send mail with Go: Interfaces and polymorphism

```go
// Address is the address of the mail receiver.
type Address struct {
    Address string
}

// Sender is a interface to send mails.
type Sender interface {

    // Send an email to a given address with a  message.
    SendMail(address Address, message string)
}
```

- A example for service-oriented components

A type implements an interface when providing the required methods

```go
// Package smtp sends mails over the smtp protocol.
package smtp

import (
    "log"

    "github.com/jweigend/concepts-of-programming-languages/oop/mail"
)

// MailSenderImpl is a sender object.
type MailSenderImpl struct {
}

// SendMail sends a mail to a receiver.
func (m *MailSenderImpl) SendMail(address mail.Address, message string) {
    log.Println("Sending message with SMTP to " + address.Address + " message: " + message)
    return
}
```

- Import references fully qualified VC directories in $GOPATH/src

The Go interface can be used as in Java

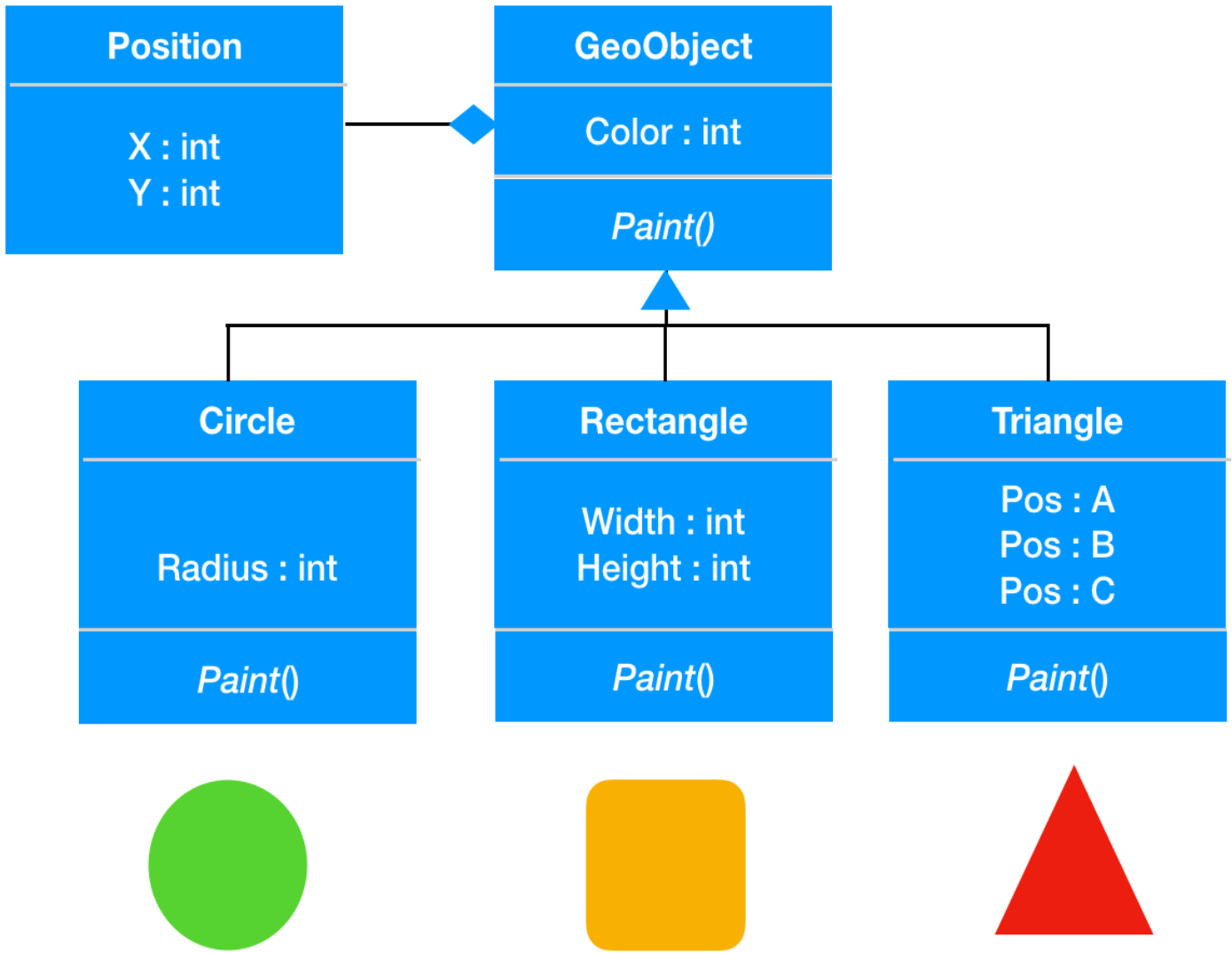
```
// Package client contains sample code for the mail components.
package client

import (
    "github.com/jweigend/concepts-of-programming-languages/oop/mail"
    "github.com/jweigend/concepts-of-programming-languages/oop/mail/util"
)

// Registry is the central configration for the service locator
var Registry = util.NewRegistry()

// SendMail sends a mail to a receiver.
func SendMail(address, message string) {

    // Create an implementation for the mail.Sender interface.
    var sender = Registry.Get("mail.Sender").(mail.Sender)

    mailaddrs := mail.Address{Address: address}
    sender.SendMail(mailaddrs, message)
}
```

Summary

- Several interfaces can be put together to form an interface
- Go does not support inheritance but type embedding (delegation without syntactic ballast)
- Go supports polymorphism only via interfaces, not through classes
- Interfaces with a method end with the ending "er" (Stringer, Writer, Reader...)

youtu.be/Ng8m5VXsn8Q?t=414 (https://youtu.be/Ng8m5VXsn8Q?t=414)

Exercise 3


Position
X : int
Y : int
GeoObject
Color : int
Paint()
Circle
Radius : int
Paint()
Rectangle
Width : int
Height : int
Paint()
Triangle
Pos : A
Pos : B
Pos : C
Paint()

Exercise

- Implement the UML diagram with Go
- The Paint() method should print the names and values of the fields to the console
- Allocate an array of polymorph objects and call Paint() in a loop

github.com/jweigend/concepts-of-programming-languages/blob/master/docs/exercises/Exercise3.md (https://github.com/jweigend/concepts-of-programming-languages/blob/master/docs/exercises/Exercise3.md)

Questions

- What is the difference between inheritance in Java and embedding in Go?
- How does Go support multiple inheritance? Is is supported for interfaces and types?

Thank you

Johannes Weigend
Rosenheim Technical University
johannes.weigend@qaware.de (mailto:johannes.weigend@qaware.de)
http://www.qaware.de (http://www.qaware.de)