Compare Object Oriented Programming in Go with Eiffel

Concepts of Programming Languages - WS 18/19

Johann Neuhauser

University of Applied Sciences Rosenheim

10.01.2019

Table of Content

Introduction
Key characteristics

Go

- Garbage collection
- Static type system
- Multiple inheritance
- Polymorphism (interfaces)
- Very fast compilation times
- Pointer arithmetics

Eiffel

- Garbage collection
- Static type system
- Multiple inheritance
- Polymorphism
- Slow compilation times
- No pointers

Introduction
Hello World

Go

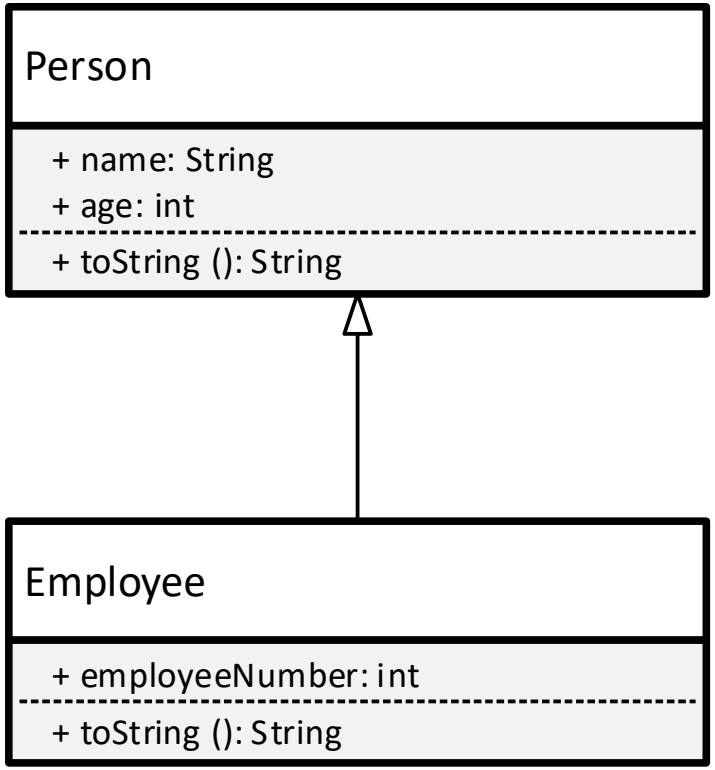
```
package main

import "fmt"

func main() {
    fmt.Println("Hello, World")
}
```

Eiffel

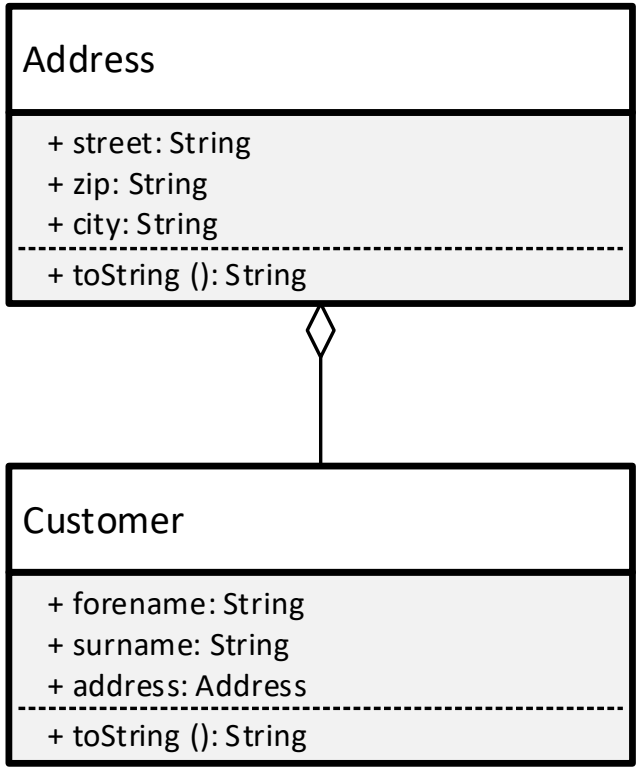
```
class
    HELLO_WORLD
create
    make
feature
    make
        do
            print ("Hello, World%N")
        end
end
```

Inheritance (and Composition)

Inheritance (Is a)



Composition (has a)



Inheritance (and Composition) Go

```
package main

import "fmt"

type Person struct {
	name string
	age  int
}

func (p Person) String() string {
	return fmt.Sprintf("%s (%d years)", p.name, p.age)
}
```

```
package main

import "fmt"

type Employee struct {
	Person
	employeeNumber int
}

func (e Employee) String() string {
	return fmt.Sprintf("%d: %s", e.employeeNumber, e.Person)
}
```

```
package main

import "fmt"

func main() {
	e := Employee{Person{"Thomas Bauer", 25}, 12346}
	fmt.Println(e)
	e.name = "Thomas Huber"
	fmt.Println(e)
}
```

Inheritance (and Composition) Eiffel

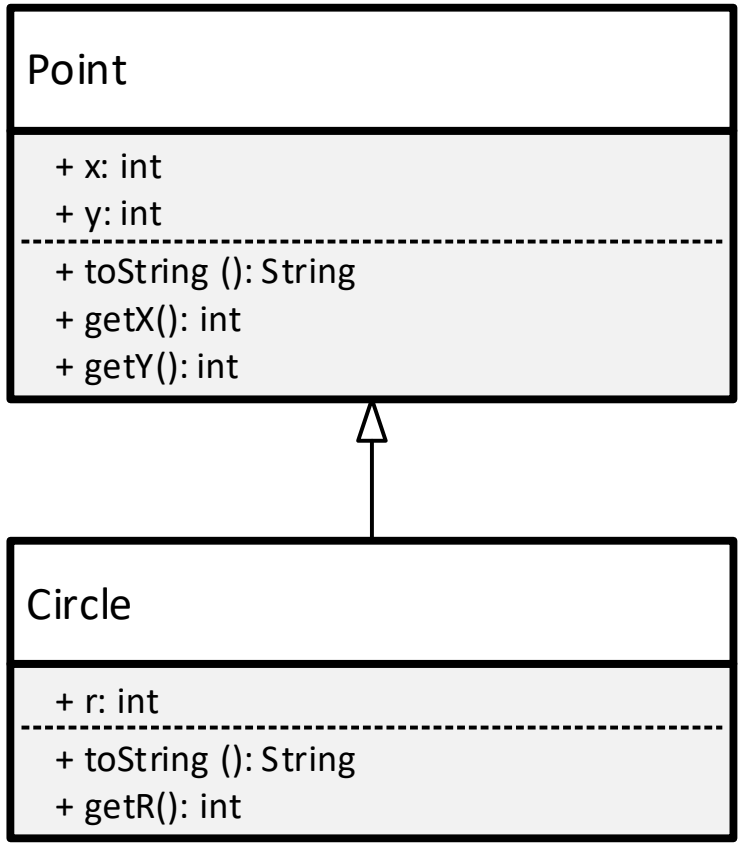
```
class PERSON
inherit
	ANY
		redefine
			out
		end
create
	make
feature {NONE}
	make (a_name: STRING; a_age: INTEGER)
		do
			set_name (a_name)
			set_age (a_age)
		end
feature
	name: STRING assign set_name
	age: INTEGER assign set_age
	set_name (a_name: STRING)
		do
			name := a_name
		end
	set_age (a_age: INTEGER)
		do
			age := a_age
		end
	out: STRING
		do
			Result := name.out + " (" + age.out + " years)"
		end
invariant
	non_negative_age: age >= 0
end
```

```
class
	EMPLOYEE
inherit
	PERSON
		rename
			make as person_make,
			out as person_out
		end
create
	make
feature {NONE}
	make (a_name: STRING; a_age, a_employee_number: INTEGER)
		do
			person_make (a_name, a_age)
			set_employee_number (a_employee_number)
		end
feature
	employee_number: INTEGER assign set_employee_number
	set_employee_number (a_employee_number: INTEGER)
		do
			employee_number := a_employee_number
		end
	out: STRING
		do
			Result := employee_number.out + ": " + person_out
		end
end
```

Inheritance (and Composition) Eiffel

```
class
	APPLICATION
create
	make
feature {NONE}
	make
		local
			e: EMPLOYEE
		do
			create e.make ("Thomas Bauer", 25, 12346)
			print (e.out + "%N")
			e.name := "Thomas Huber"
			print (e.out + "%N")
		end
end
```

Polymorphism



Polymorphism Go

```
package main

import "fmt"

type Point struct {
	x, y int
}

type IPoint interface {
	String() string
	GetX() int
	GetY() int
}

func (p *Point) String() string {
	return fmt.Sprintf("Point:\tx = %d\ty = %d", p.x, p.y)
}

func (p *Point) GetX() int {
	return p.x
}

func (p *Point) GetY() int {
	return p.y
}

func NewPoint(x, y int) *Point {
	return &Point{x, y}
}

func NewPointOrigin() *Point {
	return &Point{}
}
```

```
package main

import "fmt"

type Circle struct {
	Point
	r int
}

type ICircle interface {
	IPoint
	GetR() int
}

func (c *Circle) String() string {
	return fmt.Sprintf("Circle:\tx = %d\ty = %d\tr = %d", c.x, c.y, c.r)
}

func (c *Circle) GetR() int {
	return c.r
}

func NewCircle(x, y, r int) *Circle {
	if r < 0 {
		panic("Negative radius for circle not allowed")
	}
	return &Circle{Point{x, y}, r}
}

func NewCircleOrigin() *Circle {
	return &Circle{}
}

func NewCircleFromPoint(g *IPoint, r int) *Circle {
	if r < 0 {
		panic("Negative radius for circle not allowed")
	}
	return &Circle{Point{(*g).GetX(), (*g).GetY()}, r}
}
```

Polymorphism
Go

```
package main

import "fmt"

func main() {
	var myPoint IPoint
	var myCircle ICircle

	myPoint = NewPointOrigin()
	fmt.Println(myPoint)

	myPoint = NewCircleOrigin()
	fmt.Println(myPoint)

	myPoint = NewPoint(10, 15)
	fmt.Println(myPoint)

	myPoint = NewCircle(20, 25, 5)
	fmt.Println(myPoint)

	myCircle = NewCircle(30, 35, 10)
	fmt.Println(myCircle)

	myCircle = NewCircleFromPoint(&myPoint, 35)
	fmt.Println(myCircle)
}
```

Polymorphism Eiffel

```
class
	POINT
inherit
	ANY
		redefine
			out
		end
create
	make, make_origin
feature {NONE}
	make (a_x, a_y: INTEGER)
		do
			set_x (a_x)
			set_y (a_y)
		end
	make_origin
		do
		end
feature
	x: INTEGER assign set_x
	y: INTEGER assign set_y
	set_x (a_x: INTEGER)
		do
			x := a_x
		end
	set_y (a_y: INTEGER)
		do
			y := a_y
		end
	out: STRING
		do
			Result := "Point:%Tx = " + x.out + "%Ty = " + y.out
		end
end
```

```
class
	CIRCLE
inherit
	POINT
		rename
			make as point_make
		redefine
			make_origin,
			out
		end
create
	make, make_origin, make_from_point
feature {NONE}
	make (a_x, a_y, a_r: INTEGER)
		require
			non_negative_radius_argument: a_r >= 0
		do
			point_make (a_x, a_y)
			set_r (a_r)
		end
	make_origin
		do
		end
	make_from_point (a_p: POINT; a_r: INTEGER)
		require
			non_negative_radius_argument: a_r >= 0
		do
			set_x (a_p.x)
			set_y (a_p.y)
			set_r (a_r)
		end
feature
	r: INTEGER assign set_r
	set_r (a_r: INTEGER)
		require
			non_negative_radius_argument: a_r >= 0
		do
			r := a_r
		end
	out: STRING
		do
			Result := "Circle:%Tx = " + x.out + "%Ty = " + y.out + "%Tr
= " + r.out
		end
invariant
	non_negative_radius: r >= 0
end
```

Polymorphism Eiffel

```
class
	APPLICATION
create
	make
feature {NONE}
	make
		local
			my_point: POINT
			my_circle: CIRCLE
		do
			create my_point.make_origin
			print (my_point.out + "%N")

			create {CIRCLE} my_point.make_origin
			print (my_point.out + "%N")

			create my_point.make (10, 15)
			print (my_point.out + "%N")

			create {CIRCLE} my_point.make (20, 25, 5)
			print (my_point.out + "%N")

			create my_circle.make (30, 35, 10)
			print (my_circle.out + "%N")

			create my_circle.make_from_point (my_point, 35)
			print (my_circle.out + "%N")
		end
end
```

Thank you

Johann Neuhauser

University of Applied Sciences Rosenheim

johann.neuhauser@stud.fh-rosenehim.de

https://inf-git.fh-rosenheim.de/sINFjoneuh/kp-2018-compare_oop_in_go_with_eiffel