Golang meetup #15

7 August 2018

Thach Le

Who am I?

- @runi, @runititi, @1BanNamGiauTen
- Python, NodeJS, Go
- Backend at SunnySoft
- 3.5 years with Go

runikitkat.com (http://runikitkat.com)

Agenda

- Go drawbacks
- The cures
- Code generator
- Example

Go drawbacks

No generics support

- Parametric polymorphism
- Type specificed later

```
class ArrayAlg
{
    public static <T> T getMiddle(T []a)
    {
        return a[a.length / 2];
    }
}
```

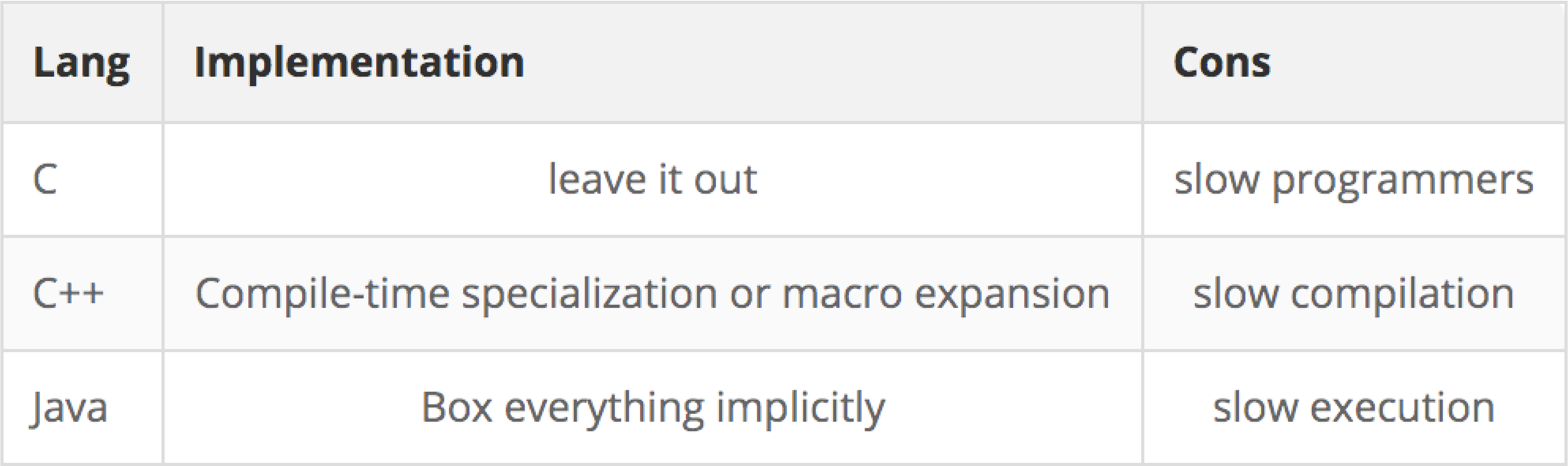
Error everywhere

```
a, err := DoA()
if err != nil {
    // handle
}

b, err := DoB(a)
if err != nil {
    // handle
}
```

No function overloading

```
function Person[] FindPersons(string nameOfPerson) { ... }
function Person[] FindPersons(date dateOfBirth) { ... }
function Person[] FindPersons(int age, string dogsName) { ... }

function Person[] FindPersonsByName(string nameOfPerson) { ... }
function Person[] FindPersonsByDOB(date dateOfBirth) { ... }
function Person[] FindPersonsByAgeAndDogsName(int age, string dogsName) { ... }
```

The generic dimplema



Lang	Implementation	Cons
C	leave it out	slow programmers
C++	Compile-time specialization or macro expansion	slow compilation
Java	Box everything implicitly	slow execution

- Do you want slow programmers, slow compilers and bloated binaries, or slow execution time?

The cures

Copy & paste

- Go does have generics: Slices, maps, array ...

Copy & paste

```
// IntSlice attaches the methods of Interface to []int, sorting in increasing order.
type IntSlice []int

func (p IntSlice) Len() int           { return len(p) }
func (p IntSlice) Less(i, j int) bool { return p[i] < p[j] }
func (p IntSlice) Swap(i, j int)      { p[i], p[j] = p[j], p[i] }

// Sort is a convenience method.
func (p IntSlice) Sort() { Sort(p) }

// Float64Slice attaches the methods of Interface to []float64, sorting in increasing order
// (not-a-number values are treated as less than other values).
type Float64Slice []float64

func (p Float64Slice) Len() int           { return len(p) }
func (p Float64Slice) Less(i, j int) bool { return p[i] < p[j] || isNaN(p[i]) && !isNaN(p[j]) }
func (p Float64Slice) Swap(i, j int)      { p[i], p[j] = p[j], p[i] }
```

interface{}

- Duck-typing
- Closer to Java Object
- Type assert
- Runtime

```go
package main

func doSomething(a interface{}) {
    switch a.(type) {
    case int:
    case string:
    }

}

func main() {
    doSomething([]int{2, 3})
    doSomething([]string{"a", "b"})
}
```

Code generator

- Turing completeness: a code, to write code
- Separate with go build.

Code generator

Methods

- Go template: Rely on `text/template`
- Go generate: Allow to run general commands by scanning for special comments `//go:generate`
- Reading code: `go/types`, `go/scanner`, `go/parser`, `go/ast`

Talk is cheap, show me the examples

Bonus

The ORM dimplema

```
const sqlCreateCriterion = `
INSERT INTO product_manager.criteria(product_id, variable, data_source, comparison,  list_values, active
) VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?);`

func createCriterion() (Criterion, error) {
    criterion := Criterion{
        ProductID:  productID,
        Variable:   variable,
        DataSource: dataSource,
        Comparison: comparison,
        ListValues: listValues,
        Active:     true,
    }
    result, err := db.Exec(sqlCreateCriterion, criterion.ProductID, criterion.Variable,
        criterion.DataSource, criterion.Comparison,
        stringListValues, criterion.Active)
    if err != nil {
        return nil, err
    }
    criterion.ID, err = result.LastInsertId()
    if err != nil {
        return nil, err
    }
}
```

The ORM dimplema

```go
func createCriterion() (Criterion, error) {
    criterion := Criterion{
        ProductID:  productID,
        Variable:   variable,
        DataSource: dataSource,
        Comparison: comparison,
        ListValues: listValues,
        Active:     true,
    }
    err := db.Insert(criterion)
    if err != nil {
        return nil, err
    }

    return criterion, nil
}
```

Introduce sql-gen library

- Code-first modelgen
- Performance
- Developer friendly

Thank you

Thach Le
thach@kakaolabs.com (mailto:thach@kakaolabs.com)