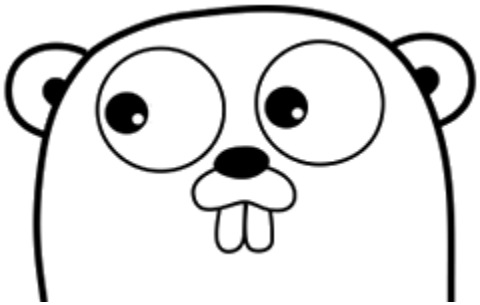
Разработка объектно-ориентированных систем программирования интегрированных в среду Eclipse

2. Разработка распознавателей объектно-ориентированных языков программирования

Владимир Юрьевич Романов,
Московский Государственный Университет им. М.В.Ломоносова
Факультет Вычислительной Математики и Кибернетики
vromanov@cmc.msu.ru,
vladimir.romanov@gmail.com



Язык программирования Go

Язык программирования **Go** фирмы **Google**

- **https://golang.org/**
- Обзор языка ***Go***
  https://tour.golang.org/welcome/1
- Описание языка **Go**
  https://golang.org/ref/spec

Введение в язык
GoLang

Пакеты и импорты

```
package main

import (
    "fmt"
    "math/rand"
)

func main() {
    fmt.Println("My favorite number is", rand.Intn(10))
}
```

```
package main;

import static java.lang.Math.random;

class Main {
        static public void main(String[] args) {
                System.out.println("My favorite number is" + random()*10);
        }
}
```

Функции

```
package main

import  "fmt”

func add(x  int, y  Int)  int  {
      return x + y
}

func mult(x , y  Int)  int  {
      return x * y
}
func main() {
      fmt.Println(  add(42, 13)  )
}
```

Множество результатов функции

```
package main

import "fmt"

func swap(x, y string) (string, string) {
	return y, x
}

func main() {
	a, b := swap("hello", "world")
	fmt.Println(a, b)
}
```

Именованные результаты функции

```
package main

import "fmt"

func split(sum int) (x, y int) {
	x = sum * 4 / 9
	y = sum - x
	return
}

func main() {
	fmt.Println(split(17))
}
```

Переменные

```
package main

import "fmt"

var  c, python, java bool

func main() {
    var  i  int
    fmt.Println(i, c, python, java)
}
```

Инициализаторы переменных

```
package main

import "fmt"

var i, j int = 1, 2

func main() {
    var c, python, java = true, false, "no!"
    fmt.Println(i, j, c, python, java)
}
```

Короткие объявления переменных

```
package main

import "fmt"

func main() {
    var i, j int = 1, 2
    k := 3
    c, python, java := true, false, "no!"

    fmt.Println(i, j, k, c, python, java)
}
```

Базовые типы

```
package main

import (
    "fmt"
    "math/cmplx"
)

var (
    ToBe   bool       = false
    MaxInt uint64     = 1<<64 - 1
    z      complex128 = cmplx.Sqrt(-5 + 12i)
)

func main() {
    fmt.Printf("Type: %T Value: %v\n", ToBe, ToBe)
    fmt.Printf("Type: %T Value: %v\n", MaxInt, MaxInt)
    fmt.Printf("Type: %T Value: %v\n", z, z)
}
```

Типы. Структуры

```
package main

import "fmt"

type Vertex struct {
    X int
    Y int
}

func main() {
    fmt.Println(Vertex{1, 2})
}
```

Типы. Поля структуры

```
package main

import "fmt"

type Vertex struct {
    X int
    Y int
}

func main() {
    v := Vertex{1, 2}
    v1 = Vertex{X: 1}
    v.X = 4
    fmt.Println( v.X )
}
```

Массивы

```
package main

import "fmt"

func main() {
    var a [2] string
    a[0] = "Hello"
    a[1] = "World"
    fmt.Println(a[0], a[1])
    fmt.Println(a)

    primes := [6]int{2, 3, 5, 7, 11, 13}
    fmt.Println(primes)
}
```

Срезы

```
package main

import "fmt"

func main() {
    primes := [6] int { 2, 3, 5, 7, 11, 13 }

    var s [ ] int = primes[1:4]
    fmt.Println(s)
}
```

Литералы срезов

```
package main

import "fmt"

func main() {
    q := [ ]int{2, 3, 5, 7, 11, 13}
    fmt.Println(q)

    r := [ ] bool {true, false, true, true, false, true}
    fmt.Println(r)
}
```

Карты

```
package main

import "fmt"

type Vertex struct {
    Lat,  Long   float64
}

var  m  map [string]  Vertex

func main() {
    m = make( map [string] Vertex )
    m ["Bell Labs"] = Vertex{
            40.68433,  -74.39967,
    }
    fmt.Println(m["Bell Labs"])
}
```

Литералы карты

```
package main

import "fmt"

type Vertex struct {
    Lat,  Long  float64
}

var m = map [string] Vertex {
    "Bell Labs" : Vertex { 40.68433,    -74.39967, },
    "Google"    : Vertex { 37.42202,  -122.08408, },
}

func main() {
    fmt.Println(m)
}
```

Функции - значения

```
package main
import (
    "fmt"
    "math"
)
func compute(fn func(float64, float64) float64) float64 {
    return fn(3, 4)
}
func main() {
    hypot := func(x, y float64) float64 {
        return math.Sqrt(x*x + y*y)
    }
    fmt.Println( hypot(5, 12) )

    fmt.Println( compute(hypot) )
    fmt.Println( compute(math.Pow))
}
```

Методы типов

```
package main
import (
    "fmt"
    "math"
)
type Vertex struct {
    X, Y float64
}

func (v Vertex) Abs() float64 {
    return math.Sqrt(v.X * v.X + v.Y * v.Y)
}

func main() {
    v := Vertex{3, 4}
    fmt.Println( v.Abs() )
}
```

Методы - функции

```
package main
import (
    "fmt"
    "math"
)
type Vertex struct {
    X, Y float64
}

func Abs(v Vertex) float64 {
    return math.Sqrt(v.X*v.X + v.Y*v.Y)
}

func main() {
    v := Vertex{3, 4}
    fmt.Println(Abs(v))
}
```

Интерфейсы

```
package main
import (
    "fmt"
    "math"
)
type Abser interface {
    Abs() float64
}

func main() {
    var a Abser
}
```

Неявная реализация интерфейса

```
package main
import "fmt“
type L interface {
    M()
}
type T struct {
    S string
}
// Тип T реализует интерфейс L,
// но нам не нужно это объявлять явно.
func (t T) M() {
    fmt.Println(t.S)
}
func main() {
    var  t  L = T {"hello"}
    t.M()
}
```