

---

# Обзор современных языков программирования

---

## Scala

---

# Scala. Обзор

- Мультипарадигмальный язык, сочетающий возможности функционального и объектно-ориентированного программирования
  - Статически типизированный, типобезопасный
  - Поддерживает парадигму компонентного программирования
  - Реализован для платформ Java и .Net
-

---

# Scala. История

- Создан в 2001-2004 годах в Лаборатории методов программирования EPFL
  - За основу взяты две идеи
    - Должен быть масштабируемым
    - Должен унифицировать и обобщать объектно-ориентированное и функциональное программирование
  - Испытал влияние Java, C#, Smalltalk
-

---

# Scala. Ключевые аспекты

- Scala-программы во многом похожи на Java-программы
  - Включает единообразную объектную модель
  - Scala – это также функциональный язык
  - Позволяет производить декомпозицию объектов путем сравнения с образцом
  - Поддерживает параллелизм
-

---

# Scala. Области применения

- Web-приложения
  - Скрипты
  - Интеграционные приложения
  - Масштабируемые и высокопроизводительные серверные приложения
  - Мобильные приложения
  - Предметно-ориентированные языки (DSL)
-

# Scala. Примеры программ

```
object HelloWorld
{
    def main(args: Array[String]) : Unit =
    {
        println("Hello, world!")
    }
}
```

```
object HelloWorld extends App
{
    println("Hello, World!")
}
```

# Scala. Примеры программ

```
object MatchTest2 extends App
{
  def matchTest(x: Any): Any = x match
  {
    case 1 => "one"
    case "two" => 2
    case y: Int => "scala.Int"
  }
  println(matchTest("two"))
}
```

# Scala. Возможности

- В Scala можно создавать методы:

```
scala> def max(x: Int, y: Int): Int
      = if (x<y) y else x
max: (Int,Int)Int
```

- Тип возвращаемого значения можно опустить:

```
scala> def max2(x: Int, y: Int)
      = if (x<y) y else x
max2: (Int,Int)Int
```



---

# Scala. Возможности

- Scala хорошо масштабируется, поэтому подходит для создания скриптов

Файл helloarg.scala:

```
Println("Hello, " + args(0) + " !")
```

Запуск в интерпретаторе:

```
>scala helloarg.scala planet
```

Результат:

```
Hello, planet!
```

---

# Scala. Возможности

- Для обращения к методу вместо `.` можно использовать просто пробел, параметры указываются тоже через пробел

```
println("Hello, " + args(0) + "!").toUpperCase)
println("Hello, " + args(0) + "!" toUpperCase)
```

- Значения элементарных типов – тоже объекты

```
scala> 1 to 5
res0: Range.Inclusive = Range(1, 2, 3, 4, 5)
```

# Scala. Возможности

- Scala различает значения (`val`) и переменные (`var`)

```
scala> val msg = "Hello, world!"  
msg: java.lang.String = Hello, world!
```

```
scala>println(msg)  
Hello, world!  
Unnamed0: Unit = ()
```

```
scala>msg = "Goodbye, world!"  
<console>:5 error: assignment to non-variable  
    val unnamed4 = {msg = "Goodbye, world!";msg}
```

# Scala. Возможности

- Scala различает значения (`val`) и переменные (`var`)

```
scala> def sum(a: Int, b: Int): Int = {  
  | var result = 0  
  | for (i <- a to b) result += i  
  | result  
  | }  
sum: (Int,Int)Int
```

```
scala> sum(1, 5)  
res0: Int = 15
```

---

# Scala. Возможности

- Циклы – аналогичны C# и Java
- Поддерживаются лямбда-выражения

Файл pr.scala:

```
args.foreach(arg => print(arg))
```

Запуск в интерпретаторе:

```
> scala pr.scala Concise Is Nice
```

Результат:

```
ConciseIsNice
```

---

# Scala. Возможности

- Scala – объектно-ориентированный язык

```
val s = new String("Hello, world!")  
println(s)
```

```
val greetStrings = new[String] (3)  
greetStrings(0) = "Hello"  
greetStrings(1) = ", "  
greetStrings(2) = "world!\n"  
for (i <- to 2)  
  print(greetStrings(i))
```

# Scala. ВОЗМОЖНОСТИ

```
class Complex(r: Double, i: Double) {
  def real = r
  def image = i
  def magnitude = Math.sqrt(r*r + i*i)
  def + (y: Complex) = new Complex(this.real +
y.real, this.image + y.image)
  override def toString = real+" + i*" + image
}
object Main {
  def main(args:Array[String]) :Unit = {
    val first = new Complex(1, 5)
    val second = new Complex(2, 4)
    val sum = first + second
    println(sum)
  }
}
```

---

# Scala. ВОЗМОЖНОСТИ

```
class Complex(val real: Double, val image:
Double) extends Ordered[Complex] {
  def magnitude =Math.sqrt(real*real+image*image)
  def + (y: Complex) = new Complex(this.real +
y.real, this.image + y.image)
  def compare(y:Complex):Int = this.magnitude
compare y.magnitude
  override def toString = real+" + i*" + image
}
object Main {
  def main(args:Array[String]) :Unit = {
    val first = new Complex(1, 5)
    val second = new Complex(2, 4)
    if (first > second)
      println("First greater")
  }
```



# Scala. Возможности

- Scala поддерживает работу со списками

```
val x12 = List(1, 2)
val x34 = 3 :: 4 :: Nil
val x1234 = x12 ::: x34
println(x12 + " & " + x34 + " were not mutated")
println("Thus, " + x1234 + " is a new List")
```

Результат:

```
List(1, 2) & List(3, 4) were not mutated
Thus, List(1, 2, 3, 4) is a new List
```

# Scala. ВОЗМОЖНОСТИ

|   |                                      |
|---|--------------------------------------|
| <pre>val t = ("Scala", "Java",<br/>"Lama")</pre>                | Новый List[String] с<br>3 значениями |
| <pre>t.count(s =&gt; s.length == 4)</pre>                       | 2                                    |
| <pre>t.filter(s =&gt; s.length == 4)</pre>                      | List("Java", "Lama")                 |
| <pre>t.forall(s=&gt;s.endsWith("a"))</pre>                      | true                                 |
| <pre>t.foreach(s =&gt; print(s))</pre>                          | ВЫВОДИТ "ScalaJavaLama"              |
| <pre>t.foreach(print)</pre>                                     | ВЫВОДИТ "ScalaJavaLama"              |
| <pre>t.map(s =&gt; s+ "!")</pre>                                | List("Scala!", "Java!", "Lama!")     |
| <pre>t.sort((s,t) =&gt;<br/>s.charAt(0) &lt; t.charAt(0))</pre> | List("Java", "Lama",<br>"Scala")     |

---

# Обзор современных языков программирования

---

## Go

---

# Go. Обзор

- Компилируемый и многопоточный язык общего назначения, разработанный компанией Google
  - Статически типизированный
  - Создавался для того, чтобы помочь задействовать всю мощь современных многоядерных процессоров
  - Поддерживается многими ОС (включая Windows, Linux, Android)
-

---

# Go. Особенности

- Строгая типизация, доступен автоматический вывод типов
  - Полноценная поддержка указателей, но без возможности применять к ним арифметические операции, в отличие от C/C++
  - Строковый тип со встроенной поддержкой юникода
  - Автоматическое управление памятью со сборщиком мусора
-

---

# Go. Особенности

- Средства объектно-ориентированного программирования, но без поддержки наследования реализации (наследуются только интерфейсы)
  - Средства параллельного программирования: встроенные в язык потоки (`go routines`), взаимодействие потоков через каналы
  - лаконичный и простой синтаксис, основанный на Си
-

---

# Go. Особенности

- Существуют интерфейсы, которые не нужно явно имплементировать, а лишь достаточно реализовать методы интерфейса
  - Средства функционального программирования: неименованные функции, замыкания, передача функций в параметрах и возврат функциональных значений
-

---

# Go. Особенности

- Из языка сознательно исключены
    - Структурная обработка исключений (вместо этого рекомендуется использовать возврат ошибки как одного из результатов вызова функции и проверку его на месте вызова)
    - Наследование классов (заменяется механизмом встраивания)
    - Переопределение методов
    - Обобщённое программирование
-



---

# Go. Преимущества

- Простой синтаксис
  - Скорость и компиляция
  - Наличие сборщика мусора
  - Наследование
  - Параллелизм
  - Богатая стандартная библиотека
-

# Go. Примеры программ

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

```
type Point struct { x, y float }
func (p Point) Abs() float {
    return math.Sqrt(p.x*p.x + p.y*p.y)
}
```

```
func f(a, b int) (int, string) {
    return a+b, "сложение" }
```

---

# Go. Примеры программ

```
func incTwo(a, b int) (c, d int) {  
    c = a+1  
    d = b+1  
    return  
}
```

```
first, second := incTwo(1, 2)
```

---

# Go. Отложенные вызовы

```
func CopyFile(dstName, srcName string) (written
int64, err error) {
    src, err := os.Open(srcName)
    if err != nil {
        return
    } // иначе файл-источник успешно открыт
    defer src.Close()
    dst, err := os.Create(dstName)
    if err != nil {
        return
    }
    defer dst.Close()
    return io.Copy(dst, src)
}
```

---

# Go. Многопоточность

```
func server(i int) {  
    for {  
        print(i)  
        time.Sleep(10)  
    }  
}  
go server(1)  
go server(2)
```



---

# Обзор современных языков программирования

---

## Swift

---

# Swift. Обзор

- Разработка началась в 2010 году компанией Apple
  - Swift заимствовал идеи из Objective-C, Rust, Haskell, Ruby, Python, C#, CLU и других языков
  - Мультипарадигмальный язык программирования общего назначения, который был создан в первую очередь для разработчиков iOS, OS X, tvOS и watchOS
-

---

# Swift. Обзор

- Код, написанный на Swift, может работать вместе с кодом, написанным на языках программирования C, C++ и Objective-C в рамках одного и того же проекта
  - Swift - язык со строгой типизацией, имеется автоматическое выведение типа
  - Отсутствует неявное приведение типов
  - «Игровые площадки» – интерактивное выполнение кода
-



---

# Swift. Преимущества и недостатки

- Преимущества
    - Простой синтаксис
    - Интерактивность
    - Совместимость с Си
    - Высокая производительность
  - Недостатки
    - Акцент на скорости
    - Специализация только на OS X и iOS
-

# Swift. Примеры программ

```
println("Hello, world")
```

-----

```
class Shape {  
    var numberOfSides = 0  
    func simpleDescription() -> String {  
        return "A shape with \(numberOfSides) sides"  
    }  
}
```

# Swift. Примеры программ

```
func hasAnyMatches(list: Int[], condition: Int
-> Bool) -> Bool {
    for item in list {
        if condition(item) {
            return true
        }
    }
    return false
}
func lessThanTen(number: Int) -> Bool {
    return number < 10
}
var numbers = [20, 19, 7, 12]
hasAnyMatches(numbers, lessThanTen)
```

---

# Дополнительные материалы

1. Особенности языка Go:

<https://habr.com/ru/company/JetBrains-education/blog/495014/>

лекция "Введение в язык программирования Go»

2. Основы языка Swift:

<https://www.youtube.com/watch?v=Yh0jAl7ObmI>

3. Знакомство с Kotlin:

<https://www.youtube.com/watch?v=X83IR3UpK3Y>

Если возникнет желание попробовать что-нибудь написать на этих языках, можно использовать онлайн компилятор: <https://ideone.com/>

---