Знакомство со Scala

Четвертое занятие

Параметризация типов

- Мощный инструмент для борьбы с дублированием кода
- Более «правильный» подход, чем в Java generics
- Параметрический полиморфизм высших порядков (ППВП): **class A[Coll[_], T <: Coll[Int]]**
- Поддержка вариантности
- Вывод типов спасает от громоздких конструкций

Вывод типов, примеры

```
case class MyItems[T](l: List[T])
val a = MyItems(List("a", "b")) // #1

case class MyHigherRankItems[T, CC[X]](l: CC[T])
val b = MyHigherRankItems(List("a", "b")) // #2

def myfunc[T >: String <: Any](param: T) {} // #3

class SetBuilder [A, C <: Set[A] with SetLike[A, C]] // #4
```

Вариантность

```
class ReadOnly[+T] {
  // covariant position
  def value: T = ...

  // contravariant position, compile error
  def error(param: T): Unit = ...
}

val a: ReadOnly[Any] = new ReadOnly[String]

// возвращает String, но тип «расслаблен» до Any
a.value
```

Вариантность

```
class WriteOnly[-T] {
  // covariant position, compile error
  def error: T = ...

  // contravariant position
  def accept(param: T): Unit = ...
}

val a: WriteOnly[String] = new WriteOnly[Any]

// принимает String, но на самом деле
// умеет обрабатывать Any
accept(“abc”)
```

Вариантность

```
class Function[-T, +R] // T => R

val a: Any => String = _.toString

val b: Any => Any = a // covariance of R
val anyParam: Any = 1
val anyResult: Any = b(anyParam)

val c: Int => String = a // contravariance of T
val intParam: Int = 1
val strResult: String = b(intParam)
```

Библиотека коллекций

- Переписана в версии 2.8
- Активно используется ППВП и неявные параметры
- Самая мощная для статических ООП языков
- Mutable и Immutable коллекции
- Параллельные коллекции
- Распределённые коллекции – в разработке

Методы коллекций

- map - преобразование:

```
List(“1”, “2”, “3”).map(s => s.toInt * 2) // List(2,4,6)
```

- foreach - для каждого:

```
List(1,2,3).foreach(println(_))
```

- filter - фильтр:

```
List(1,2,3,4).filter(_ % 2 == 0)  // List(2,4)
```

- sortWith - сортировка:

```
List(2,1,3).sortWith(_ < _) // List(1,2,3)
```

- mkString - собрать в строку:

```
List(1,2,3).mkString(“;”) // “1;2;3”
```

Методы коллекций

- find - найти:

```
List(1,2,3).find(_ % 2 == 1) // Some(1)
```

- take - первые N элементов:

```
List(1,2,3).take(2) // List(1,2)
```

- flatMap - преобразование с распаковкой:

```
List(1,2,3).flatMap(List(0,_)) // List(0,1,0,2,0,3)
```

- sliding - список подпоследовательностей:

```
List(1,2,3,4).sliding(2).toList // List(List(1, 2),
List(2, 3), List(3, 4))
```

Методы коллекций

- zip - список пар, пример:

```
List(1,2,3).zip(List(4,5,6)) // List((1,4),(2,5),(3,6))
```

- zipWithIndex - список пар вида (элемент коллеции, порядковый номер)

```
List(“a”, “b”, “c”).zipWithIndex // List((“a”, 0), (“b”,1),(“c”,2))
```

- foldLeft - собрать список в значение при помощи начального значения и функции, значение может быть другого типа, пример:

```
List(“1”,”2”,”3”).foldLeft(5)(_ + _.toInt) // 11 (5 + 1 + 2 + 3)
```

Класс **Option**

```
abstract class Option[T]
object None extends Option
class Some[+T] extends Option[T]

val name:Option[String] = request.getParameter("name")
name.get                        // Exception, если None
if (name.isEmpty) doSomething(name.get)
name.getOrElse(“unknown name”)
name.map(print)

def getUser(name:String):Option[User] =
{ if (authorized)  Some(new User) else None}
val cards = name.flatMap(getUser(_)).map(_.getCards).getOrElse(Nil)
```

Кортежи (Tuples)

```
val tuple = (1, "Test", false)
if (tuple._3 && tuple._1 == 1) tuple._2

def dies = (Random.nextInt(6), Random.nextInt(6))
println(“Sum of dies: “+(dies._1+dies._2))

def funcs = Array[(Int, Int)=>Int](_*_, _+_)
def funcs2 = Array[((Int, Int)=>Int, String)]((_*_,"*"), (_+_, "+"))

for (func <- funcs2) if (func._1(2, 3) == 6)
println(2+func._2+3+”=6”)
```

Методы работы с Map

```
val map1 = Map(("Denis", 5), ("Oleg", 4))
val map2 = Map("Denis" -> 5, "Oleg" -> 4)
map1("Denis")       // 4
map1("Ivan")        // throws exception
map1.get("Ivan").getOrElse(0)

map1.map(elem => println("Name: "+elem._1+" Code: "+elem._2))

val map3 = map1 ++ map2 ++ Map("Ivan"->3)
```

Mutable Set и Map

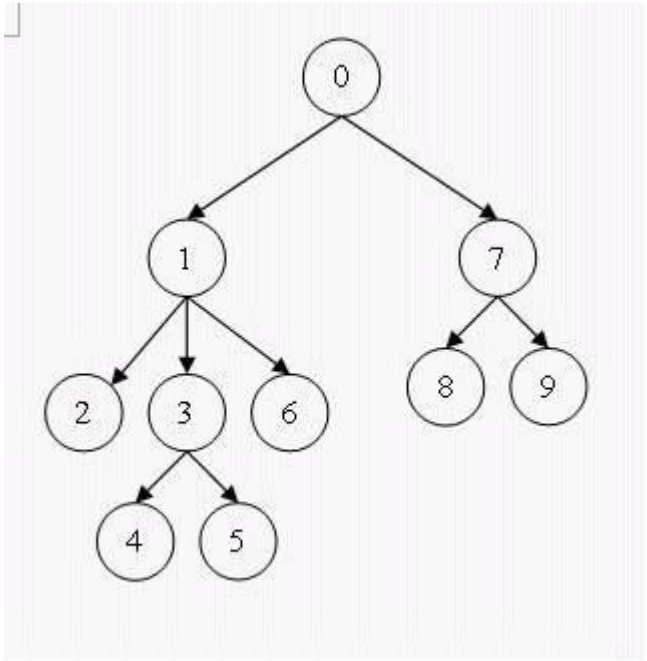
```
val map = Map ("Denis" -> 5)
map("Oleg") = 4                    // compilation error, because immutable by default

import scala.collection.mutable
val mSet = mutable.Set(1, 2, 3)
mSet += 10
mSet -= 2
mSet ++= Set(11, 12)

import scala.collection.mutable.Map
val mMap = Map("Denis" -> 5)
val mMap = mutable.Map("Denis" -> 5)
mMap("Oleg") = 4
mMap += "Ivan"-> 3
mMap -= "Denis"
```

Задание №3-1

Распечатать бинарное представление дерева:
Вход: 1101101001001101 00
Выход:
0
1 7
2 3 6 8 9
4 5



Начинаем с корня дерева.
1 – спускаемся на уровень ниже (самый левый еще не пройденный)
0 – дошли до листа, возвращаемся на уровень выше

Пример:

11111101001000111001000101 1000

0
1
2 13 14
3 9 15
4 10 12
5 8 11
6 7

Задание №3-2

Напишите программу, которая заменяет все гласные прописные и строчные буквы в тексте на цифры:
О – 0 А – 1 И – 2 Е – 3 У – 4
Ё – 5 Я – 6 Ы – 7 Э - 8 Ю – 9

Программа принимает на вход предложение. На выходе программа печатает преобразованную строку. В предложении могут встречаться пробелы, поэтому программа должна собрать входные параметры в строку.

Дополнительно: попробуйте уместить код функции main() в одну строку в 220 символов при условии, что используются 4-х символьные идентификаторы и нет незначащих пробелов.

Задание №3-3

Попробуйте решить на Scala задания из:
http://ccfit.nsu.ru/~den/2001/semestr2/Terminal2-3.doc

Для чтения из файла можете использовать стандартные Java классы из пакета java.io