Лабораторная работа № 3 «Обобщённые классы в Scala»

20 марта 2024 г.

Илья Афанасьев, ИУ9-61Б

Цель работы

Целью данной работы является приобретение навыков разработки обобщённых классов на языке Scala с использованием неявных преобразований типов.

Индивидуальный вариант

Класс `StackMachine[T]`, представляющий неизменяемый снимок состояния стековой машины, оперирующей значениями типа `T`. В случае, если `T` — числовой тип, в стековой машине должны быть реализованы арифметические операции. В `StackMachine[Boolean]` должны присутствовать логические операции.

Реализация

```
class StackMachine[T](final val items: List[T]) {
  private def performBinary(op: (T, T) => T) =
    items match {
      case x :: y :: tail => Some(StackMachine(op(x, y) :: tail))
      case _ => None
    }

  private def performUnary(op: T => T) =
    items match {
      case x :: tail => Some(StackMachine(op(x) :: tail))
      case _ => None
    }

  def add()(implicit ops: Numeric[T]) = performBinary(ops.plus)
  def sub()(implicit ops: Numeric[T]) = performBinary(ops.minus)
  def mul()(implicit ops: Numeric[T]) = performBinary(ops.times)
```

```
  def div()(implicit ops: Divisible[T]) = performBinary(ops.div)

  def not()(implicit ops: Logical[T]) = performUnary(ops.not)
  def and()(implicit ops: Logical[T]) = performBinary(ops.and)
  def or()(implicit ops: Logical[T]) = performBinary(ops.or)
}

trait Divisible[T] {
  def div(x: T, y: T): T
}

object Divisible {
  implicit def FractionalDivisible[T](implicit frac: Fractional[T]): Divisible[T] =
    new Divisible[T] {
      def div(x: T, y: T): T = frac.div(x, y)
    }

  implicit def IntegralDivisible[T](implicit integ: Integral[T]): Divisible[T] =
    new Divisible[T] {
      def div(x: T, y: T): T = integ.quot(x, y)
    }
}

trait Logical[T] {
  def not(x: T): T
  def and(x: T, y: T): T
  def or(x: T, y: T): T
}

object Logical {
  implicit final val BooleanOps: Logical[Boolean] = {
    new Logical[Boolean] {
      def not(x: Boolean): Boolean = !x
      def and(x: Boolean, y: Boolean): Boolean = x & y
      def or(x: Boolean, y: Boolean): Boolean = x | y
    }
  }
}

object Main extends App {
  val i1 = StackMachine(List(1, 3, -7))
  val i2 = i1.add().get
  val i3 = i2.sub().get

  println(i2.items)
  println(i3.items)
```

```
  val f1 = StackMachine(List(2.5, 0.5, 3.0, 1.4))
  val f2 = f1.div().get;
  val f3 = f2.mul().get;

  println(f2.items)
  println(f3.items)

  val b1 = StackMachine(List(true, true, false, true))
  val b2 = b1.not().get
  val b3 = b2.or().get
  val b4 = b3.and().get

  println(b2.items)
  println(b3.items)
  println(b4.items)
}
```

Тестирование

Результат запуска программы:

```
List(4, -7)
List(11)
List(5.0, 3.0, 1.4)
List(15.0, 1.4)
List(false, true, false, true)
List(true, false, true)
List(false, true)
```

Вывод

В результате выполнения лабораторной работы я приобрёл навыки разработки обобщённых классов на языке Scala с использованием неявных преобразований типов. Неявные преобразования в Scala - мощный выразительный инструмент языка, и их грамотное использование позволяет писать код качественнее и лаконичнее.