Лабораторная работа № 2. «Введение в объектно-ориентированное программирование на языке Scala»

1 марта 2023 г.

Самохвалова Полина, ИУ9-62Б

Цель работы

Целью данной работы является изучение базовых объектно-ориентированных возможностей языка Scala.

Задание

Выполнение лабораторной работы заключается в разработке на языке Scala одного из классов, краткое описание которого приведено в таблице вариантов заданий. Объекты классов должны быть неизменяемые. Применение контейнерных классов из стандартной библиотеки языка Scala приветствуется. Кроме того, при выполнении лабораторной работы следует избегать использования рекурсии для замены циклов, т.к. у контейнерных классов для таких случаев имеется богатый набор методов (map, filter, flatMap, и т.д.).

Индивидуальный вариант

Последовательность, конструируемая как арифметическая прогрессия. Сумма двух последовательностей - знак «+». Умножение последовательности на число - знак «*». Обнуление всех членов последовательности, начиная с заданного номера - знак «/». Кроме того, в классе должен быть реализован метод «get», возвращающий i-й элемент последовательности.

Реализация

```
class Sequence private(val a0: Int, val d: Int, val list: List[Int]) {
  def this(s_a0: Int, s_d: Int) = this(s_a0, s_d, Nil)
```

```
  def +(b: Sequence): Sequence = {
    if (list.length < b.list.length) {
      new Sequence(a0 + b.a0, d + b.d, take(b.list.length).zip(b.list).map({ case (x, y) =>
    } else {
      new Sequence(a0 + b.a0, d + b.d, b.take(list.length).zip(list).map({ case (x, y) => x
    }
  }

  def *(k: Int): Sequence = {
    new Sequence(k * a0, k * d, list.map((x: Int) => k * x))
  }

  def /(n: Int): Sequence = {
    new Sequence(0, 0, take(n))
  }

  private def take(n: Int): List[Int] = {
    def tail(i: Int): List[Int] = if (i < n) { get(i) :: tail(i + 1) } else Nil
    tail(0)
  }

  def print_first_elements(s: String, c: Int): Unit = {
    println(s)
    println(take(c).mkString(" "))
    println()
  }

  def get(ind: Int): Int = {
    if (ind < list.length) {
      list(ind)
    } else {
      a0 + ind * d
    }
  }
}

class SequenceFactor (x: Int) {
  def *(s: Sequence) = s * x
}

object Lab2 {
  def main(args: Array[String]): Unit = {

    implicit def intToFactor(i: Int) = new SequenceFactor(i)

    val s1 = new Sequence(2, 2)
```

```
    s1.print_first_elements("s1 = new Sequence(2, 2)", 7)
    val s2 = new Sequence(3, 2)
    s2.print_first_elements("s2 = new Sequence(3, 2)", 7)
    val s3 = s2 / 3
    s3.print_first_elements("s3 = s2 / 3", 7)
    val s4 = s1 + s3
    s4.print_first_elements("s4 = s1 + s3", 7)
    val s5 = new Sequence(1, 1)
    s5.print_first_elements("s5 = new Sequence(1, 1)", 7)
    val s6 = s5 / 2
    s6.print_first_elements("s6 = s5 / 2", 7)
    val s7 = s4 + s6
    s7.print_first_elements("s7 = s4 + s6", 7)
    val s8 = 3 * s7
    s8.print_first_elements("s8 = 3 * s7", 7)
    val s9 = new Sequence(1, 1)
    s9.print_first_elements("s9 = new Sequence(1, 1)", 7)
    val s10 = s8 + s9
    s10.print_first_elements("s10 = s8 + s9", 7)
    println("s10.get(2)")
    println(s10.get(2))

  }
}
```

Вывод

В результате выполнения лабораторной работы были изучены базовые объектно-ориентированные возможности языка Scala, был разработан класс, представляющий собой последовательность, которая конструируется как арифметическая прогрессия. Операции с последовательностями: сумма двух последовательностей, умножение последовательности на число, обнуление всех членов последовательности, начиная с заданного номера. Также в классе был реализован метод «get», возвращающий i-й элемент последовательности