МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное автономное бюджетное образовательное учреждение высшего образования

Севастопольский государственный университет

кафедра Информационных систем

Институт информационных технологий и управления в технических системах

курс 1 группа ИC/м-11-о

Лисянский Александр Игоревич

09.04.02 Информационные системы (уровень магистра)

**ОТЧЁТ**

о лабораторном практикуме №3

«Исследование способов реализации классов и объектов в языке Scala»

Отметка о зачёте \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Руководитель практикума

Ст.преподаватель Строганов В.А.

(должность) (подпись) (инициалы, фамилия)

Севастополь

2016

**Цель работы**

Исследовать особенности реализации классов, объектов и трейтов в языке Scala. Реализовать программные модули для обработки сложных структур данных, используя объектно-ориентированный и функциональный подходы

**Задание на лабораторную работу**

3.1. Реализовать структуру бинарного дерева поиска (элементы в левом поддереве меньше, чем элемент в корне, а элементы правого поддерева больше, чем элемент в корне). Для реализации:

3.1.1. Написать трейт Tree, определяющий абстрактные методы getLeftSubtree: Tree, getRightSubtree: Tree и getNodeData: Int.

3.1.2. Написать класс Node, который реализует трейт Tree и представляет собой узел дерева.

3.1.3. Написать класс Leaf, который реализует трейт Tree и представляет собой лист дерева.

3.2. Для реализованной структуры дерева написать следующие функции:

3.2.1. Функцию printTree(Tree): Unit, которая выводит на экран дерево

3.2.2. Функцию insert(Int, Tree): Tree, которая принимает элемент для вставки и корень дерева, возвращает корень нового дерева со вставленным элементом (при этом изначальное дерево изменяться не должно).

3.2.3. Функцию contains(Int, Tree): Boolean, которая возвращает true или false в зависимости от того, содержится ли заданное число в дереве, или нет.

3.2.4. Функцию sum(Tree): Int, которая возвращает сумму всех элементов в дереве.

**Текст программы**

Трейт Tree

**package** main.scala  
  
*/\*\*  
 \* Created by user on 18.10.2016.  
 \*/***trait** Tree {  
 **def** getLeftSubtree: Tree  
 **def** getRightSubtree: Tree  
 **def** getNodeData: Int  
}

Класс и объект Node

**package** main.scala  
  
*/\*\*  
 \* Created by user on 18.10.2016.  
 \*/***class** Node **extends** Tree{  
 **private var** *left* : Tree = **null  
 private var** *right* : Tree = **null  
 private var** data : Int = 0  
  
 **def** getLeftSubtree: Tree = *left* **def** getRightSubtree: Tree = *right* **def** getNodeData: Int = data  
  
 **def this**(nodeData: Int) {  
 **this**()  
 **this**.data = nodeData  
 }  
  
 **def this**(leftSubtree: Tree, rightSubtree: Tree, nodeData: Int) {  
 **this**(nodeData)  
 **this**.*left* = leftSubtree  
 **this**.*right* = rightSubtree  
 }  
}  
  
**object** Node {  
  
 **def** apply: Node = **new** Node()  
 **def** apply(nodeData: Int): Node = **new** Node(nodeData)  
 **def** apply(leftSubtree: Tree, rightSubtree: Tree, nodeData: Int): Node = **new** Node(leftSubtree, rightSubtree, nodeData)  
  
}

Класс и объект Leaf

**package** main.scala  
  
*/\*\*  
 \* Created by user on 18.10.2016.  
 \*/***class** Leaf **extends** Tree {  
  
 **private var** *data*: Int = 0  
  
 **def this**(nodeData: Int) {  
 **this**()  
 **this**.*data* = nodeData  
 }  
  
 **def** getLeftSubtree: Tree = **null  
 def** getRightSubtree: Tree = **null  
 def** getNodeData: Int = *data*}  
  
**object** Leaf {  
  
 **def** apply: Leaf = **new** Leaf()  
 **def** apply(leafData:Int): Leaf = **new** Leaf(leafData)  
  
}

Функции обработки дерева

**package** main.scala  
  
*/\*\*  
 \* Created by user on 18.10.2016.  
 \*/***object** TreeOption {  
  
 **def** printTree(tree: Tree): Unit = {  
 **if**(tree != **null**) {  
 *printTree*(tree.getLeftSubtree)  
 *println*(tree.getNodeData)  
 *printTree*(tree.getRightSubtree)  
 }  
 }  
  
 **def** insert(nodeData: Int, tree: Tree): Tree = {  
 **if**(tree == **null**) *Leaf*(nodeData)  
 **else** {  
 **if** (nodeData <= tree.getNodeData) *Node*(*insert*(nodeData, tree.getLeftSubtree), tree.getRightSubtree, tree.getNodeData)  
 **else** *Node*(tree.getLeftSubtree, *insert*(nodeData, tree.getRightSubtree), tree.getNodeData)  
 }  
 }  
  
 **def** contains(nodeData: Int, tree: Tree): Boolean = {  
 **if**(tree == **null**) **false  
 else** {  
 **if**(nodeData == tree.getNodeData) **true  
 else** {  
 **if**(nodeData < tree.getNodeData) *contains*(nodeData, tree.getLeftSubtree)  
 **else** *contains*(nodeData, tree.getRightSubtree)  
 }  
 }  
 }  
  
 **def** sum(tree: Tree): Long = {  
 **if**(tree == **null**) 0  
 **else** *sum*(tree.getLeftSubtree) + *sum*(tree.getRightSubtree) + tree.getNodeData  
 }  
}

Основная программа

**package** main.scala  
  
*/\*\*  
 \* Created by user on 18.10.2016.  
 \*/***object** lab3 {  
  
 **def** main (args: Array[String]): Unit = {  
 **var** tree: Tree = *Leaf*(3)  
 tree = TreeOption.*insert*(23, tree)  
 tree = TreeOption.*insert*(72, tree)  
 tree = TreeOption.*insert*(5, tree)  
 tree = TreeOption.*insert*(92, tree)  
 tree = TreeOption.*insert*(261, tree)  
 tree = TreeOption.*insert*(0, tree)  
 tree = TreeOption.*insert*(12, tree)  
 tree = TreeOption.*insert*(303, tree)  
 tree = TreeOption.*insert*(922, tree)  
 TreeOption.*printTree*(tree)  
 *println*(TreeOption.*contains*(261, tree))  
 *println*(TreeOption.*contains*(1, tree))  
 *println*(TreeOption.*sum*(tree))  
 **val** tree1 : Tree = TreeOption.*insert*(1, tree)  
 *println*(TreeOption.*contains*(1, tree1))  
 *println*(**"----------------------"**)  
 TreeOption.*printTree*(tree)  
 *println*(**"---------Tree--------"**)  
 TreeOption.*printTree*(tree1)  
 }  
}

**Выводы**

В ходе выполнения лабораторной работы были получены теоретические знания по работе с кортежами и массивами данных в языке Scala, по заданию переписана функция processList с использованием хвостовой рекурсии, использованы полученные теоретические знания на практике для выполнения заданий, указанных в варианте.