МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждение образования «Полоцкий государственный университет»

Факультет информационных технологий
Кафедра технологий программирования

Лабораторная работа №7
по дисциплине: **«Объектно-ориентированные технологии программирования и стандарты проектирования»**

ВЫПОЛНИЛ студент группы 16 ИТ-3
Яблонский А.С.

ПРОВЕРИЛ преподаватель
Ярошевич П.В.

Полоцк, 2018 г.

Задание А, вариант 11

Используйте классы файлового ввода и вывода для выполнения задания.

Задание варианта: Разделить текст файла на предложения, и для каждого предложения вывести длину и используемые в нем знаки пунктуации.

Реализация :

Реализовывать задание лабораторной работы было принято на языке программирования kotlin. Он более удобные методы для работы с файлами. Kotlin имеет гибкий удобный синтаксис, что способствует более эффективному написанию кода.

Реализация метода чтения файла:

```
fun readContent(fileName: String) : String =
        File(fileName).inputStream().readBytes().toString(Charsets.UTF_8)
```

Методы деления содержимого файла на предложения и подсчет знаков препинания в строке:

```
val PUNCTUATION_LIST = listOf('.', ',', '!', '?', '-', ':', ';', '—')

fun main(args: Array<String>) {
    val content = readContent(FILENAME)
    val sentences = divideBySentence(content)

    sentences.forEachIndexed { index, s ->
        println(" ${index + 1}: '$s'")
        sentencePunctuation(s).forEach { char, count ->
            println("\t\t [$char]: $count")
        }
        println()
    }
}


fun sentencePunctuation(sentence: String): Map<Char, Int> {
    val resultMap = mutableMapOf<Char, Int>()
    PUNCTUATION_LIST.forEach { char ->
        sentence.count { it == char }.apply { this: Int
            if (this > 0) {
                // возвращать только не-нулевые значения
                resultMap[char] = this
            }
        }
    }
    return resultMap
}
```

```
val list = arrayListOf<String>()

val printMsg = { println("Enter new element (enter for continue): ") }
printMsg()

val sc = Scanner(System.`in`)
while (sc.hasNextLine()) {
    val s = sc.nextLine()
    if (s.isBlank()) break
    printMsg()
    list.add(s)
}
```

Тестирование

Для проверки корректности работы методов были разработаны следующие тесты:

```
@Test
fun divideBySentenceTestRusNumber() {
    val content = "Есть много вариантов Lorem Ipsum. Но большинство " +
            "из них имеет не всегда! 12т приемлемые модификации?"
    val sents = divideBySentence(content)

    assertEquals( expected: 3, sents.size)

    assertTrue(sents[0].startsWith( prefix: "Есть "))
    assertTrue(sents[0].endsWith( suffix: "Lorem Ipsum."))

    assertTrue(sents[1].startsWith( prefix: "Но большинс"))
    assertTrue(sents[1].endsWith( suffix: "не всегда!"))

    assertTrue(sents[2].startsWith( prefix: "12т приемле"))
    assertTrue(sents[2].endsWith( suffix: "фикации?"))
}

@Test
fun divideBySentenceTestDiffPunctuation() {
    val content = "Но большинство...    Из них имеет не всегда!?     12т" +
            " приемлемые модификации?"
    val sents = divideBySentence(content)

    assertEquals( expected: 3, sents.size)

    assertTrue( actual: sents[0] == "Но большинство...")

    assertTrue(sents[1].startsWith( prefix: "Из них "))
    assertTrue(sents[1].endsWith( suffix: "не всегда!?"))

    assertTrue(sents[2].startsWith( prefix: "12т приемлемые "))
    assertTrue(sents[2].endsWith( suffix: "модификации?"))
}
```

```
@Test
fun sentencePunctuationTest() {
    val content1 = "And, my signature – that's a promise, " +
            "from; me; to the: Hooli user!"
    val r1 = sentencePunctuation(content1)
    assertEquals( expected: 5, r1.size)
    assertTrue(r1.containsKey( key: ','))
    assertTrue(r1.containsKey( key: ';'))
    assertTrue(r1.containsKey( key: ':'))
    assertTrue(r1.containsKey( key: '!'))
    assertTrue(r1.containsKey( key: '–'))


    assertEquals( expected: 2, r1[','])
    assertEquals( expected: 2, r1[';'])
    assertEquals( expected: 1, r1[':'])
    assertEquals( expected: 1, r1['!'])
    assertEquals( expected: 1, r1['–'])
}
```

Задание B

В рамках данного задания необходимо переделать задание второй лабораторной работы для решения следующей задачи: добавление записей в список, вывод записей, сортировка записей по указанному полю и направлению, запись списка записей в файл и чтение записей из файла в список. Для хранения записей используйте бинарный файл.

Реализация :

Главное меню:

```
fun mainMenu(list: ArrayList<Company>, scanner: Scanner = Scanner(System.`in`)) {
    println("\n 1 - add")
    println(" 2 - sort")
    println(" 3 - print")
    println(" 4 - save")
    println(" 5 - load")
    val ta = {
        println(" Incorrect value, try again")
        mainMenu(list, scanner)
    }

    try {
        when (scanner.nextInt()) {
            1 -> addNew()
            2 -> sortList()
            3 -> printList(list)
            4 -> saveList()
            5 -> loadList()
            else -> ta()
        }
        mainMenu(list, scanner)
    } catch (ex: NumberFormatException) {
        ta()
    }
}
```

Методы сохранение и чтение списка:

```
fun loadList() {
    val content = File(FILENAME).readBytes().toString(Charsets.UTF_8)
    list.clear()
    content.split( ...delimiters: "---\n").forEach { it: String
        val c = Company()
        if (c.stringDecode(it.trim())) {
            list.add(c)
        }
    }
}

fun saveList() {
    var toWrite = ""
    list.forEach { it: Company
        println()
        println(it.stringEncode())
        toWrite += it.stringEncode()
        toWrite += "---\n"
    }
    toWrite = toWrite.trim( ...chars: '\n', '-')

    File(FILENAME).writeBytes(toWrite.toByteArray(Charsets.UTF_8))
}
```

Методы добавления нового элемента, сортировки и вывода списка на экран:

```
fun addNew() {
    val c = Company()
    c.readNew()
    list.add(c)
}

fun sortList() {
    val scanner = Scanner(System.`in`)
    print("Field ( Id:1 ; Name:2 ; Address:3 ; Tel:4 , Owner:5): ")
    val fId = scanner.nextInt()
    when {
        fId > 5 -> return
        fId < 1 -> return
    }

    print("Direction ( A-z:1 ; Z-a:2 ): ")
    val sId = scanner.nextInt()
    when {
        sId > 2 -> return
        sId < 1 -> return
    }

    when (fId) {
        1 -> list.sortBy { it.id }
        2 -> list.sortBy { it.name }
        3 -> list.sortBy { it.address }
        4 -> list.sortBy { it.tel }
        5 -> list.sortBy { it.owner }
        else -> throw IllegalArgumentException("Unknown field to sort")
    }

    if (sId == 2) {
        list.reverse()
    }

    printList(list)
}

private fun printList(list: List<Any>) {
    println("\n")
    list.forEach { println("=====\n$it") }
}
```

Реализация сортировки, подразумевает выбор поля сортировки и направления. Для сортировок используются встроенные функции языка Kotlin.

Вспомогательные методы класса элементов списка:

```
fun contains(string: String): Boolean {
    return when {
        id.toString().contains(string, ignoreCase: true) -> true
        name.contains(string, ignoreCase: true) -> true
        tel.contains(string, ignoreCase: true) -> true
        address.contains(string, ignoreCase: true) -> true
        owner.contains(string, ignoreCase: true) -> true
        createDate.toString().contains(string, ignoreCase: true) -> true
        Date(createDate).toString().contains(string, ignoreCase: true) -> true
        else -> false
    }
}

fun stringEncode(): String {
    var s = ""
    s += "$id\n"
    s += "$name\n"
    s += "$address\n"
    s += "$tel\n"
    s += "$owner\n"
    s += "$createDate\n"
    s.trim()
    return s
}

fun stringDecode(encoded: String): Boolean {
    val arr = encoded
            .split( ...delimiters: "\n")
            .map { it.trim() }
    if (arr.size != 6) return false
    return try {
        arr.forEachIndexed { index, s ->
            when (index) {
                0 -> id = s.toInt()
                1 -> name = s
                2 -> address = s
                3 -> tel = s
                4 -> owner = s
                5 -> createDate = s.toLong()
            }
        }
        true
    } catch (ex: NumberFormatException) {
        throw ex
        false
    }
}
```

Тестирование Задания В проводилось вручную - пользователем.

Вывод:

После выполнения данной лабораторной работы мне удалось получить практические навыки работы работы с файлами на языке программирования Kotlin. Получить дополнительный опыт работы с коллекциями и переопределением методов