МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждение образования «Полоцкий государственный университет»

Факультет информационных технологий
Кафедра технологий программирования

Лабораторная работа №3
по дисциплине: **«Объектно-ориентированные технологии программирования и стандарты проектирования»**
на тему: «**ПЕРЕГРУЗКА ОПЕРАЦИЙ**»

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

ПРОВЕРИЛ — преподаватель
Ярошевич П.В.

Полоцк, 2018 г.

Вариант №11

Постановка задачи:

Получить практические навыки создания абстрактных типов данных и перегрузки операций. Определить и реализовать класс − абстрактный тип данных. Определить и реализовать операции над данными этого класса.

Реализация:

Реализовывать задание лабораторной работы было принято на языке программирования kotlin. Он поддерживает переопределение операторов, что необходимо для выполнения задания. Также он имеет гибкий удобный синтаксис, что способствует более эффективному написанию кода.

В результате работы был реализован класс *MyList*.

Согласно заданию варианта были переопределены следующие операторы:

```
operator fun plus(list: MyList): MyList {
    val mList = MyList( mList: this)
    mList.arrayList.addAll(list.arrayList)
    return mList
}


override fun equals(other: Any?): Boolean {
    if (this === other) return true
    if (javaClass != other?.javaClass) return false

    other as MyList

    if (arrayList != other.arrayList) return false

    return true
}
```

*оператор “+” и ==

Методы были предопределены согласно официальной документации языка.

Методы для ввода/вывода списка:

```
fun printAll() {
    println(this.toString())
}

override fun toString(): String {
    var rString = "CUSTOM LIST ::: SIZE ${this.size()}"
    this.arrayList.forEachIndexed { index, c ->
        rString += "\n[$index]: '$c'"
    }
    return rString
}

fun input() {
    val scanner = Scanner(System.`in`)
    var scannedString: String
    println("INPUT")
    while (true) {
        print("char[${this.size()}]: ")
        scannedString = scanner.nextLine()
        if (scannedString.isNotEmpty()) {
            this.arrayList.add(scannedString[0])
        } else {
            break
        }
    }
}
```

Также реализовано два конструктора: пустой конструктор, конструктор копирования:

```
constructor()

constructor(mList: MyList) {
    this.arrayList = ArrayList(mList.arrayList)
}
```

Тестирование:

Для тестирования использовалась библиотека JUnit.4. Код тестирования находится в классе MyListTest.kt.

Для тестирования корректности работы шаблонного класса MyList были реализованы следующие методы:

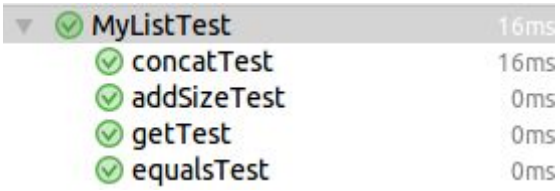
1. Тестирование основных операций над списком: добавление элемента, получение элемента по индексу, сравнение двух списков

```
@Test
fun addSizeTest() {
    val mList = MyList()
    mList.add('A')
    mList.add('B')
    mList.add('C')
    assertEquals( expected: 3, mList.size())
}

@Test
fun getTest() {
    val mList = MyList()
    mList.add('A')
    mList.add('B')
    mList.add('C')
    assertEquals( expected: 'A', mList[0])
    assertEquals( expected: 'B', mList[1])
    assertEquals( expected: 'C', mList[2])
}

@Test
fun equalsTest() {
    val mListA = MyList()
    mListA.add('A')
    mListA.add('B')
    mListA.add('C')

    val mListB = MyList()
    mListB.add('A')
    mListB.add('B')
    mListB.add('C')

    assertTrue(mListA == mListB)
    assertFalse(mListA != mListB)
}
```

2. Тестирование заданий варианта: оператор “+” и ==

```
@Test
fun concatTest() {
    val mListA = MyList()
    mListA.add('a')
    mListA.add('b')
    mListA.add('c')

    val mListB = MyList()
    mListB.add('A')
    mListB.add('B')
    mListB.add('C')


    val resultList = mListA + mListB
    assertEquals( expected: 3, mListA.size())
    assertEquals( expected: 3, mListB.size())
    assertEquals( expected: 6, resultList.size())

    assertEquals( expected: 'a', resultList[0])
    assertEquals( expected: 'c', resultList[2])
    assertEquals( expected: 'A', resultList[3])
    assertEquals( expected: 'C', resultList[5])
}

@Test
fun equalsTest() {
    val mListA = MyList()
    mListA.add('A')
    mListA.add('B')
    mListA.add('C')

    val mListB = MyList()
    mListB.add('A')
    mListB.add('B')
    mListB.add('C')

    assertTrue(mListA == mListB)
    assertFalse(mListA != mListB)
}
```

В результате тестирования все тесты были пройдены без ошибок



Для дополнительной проверки работы класса, были проверены методы ввода/вывода списка:

```
INPUT
char[0]: A
char[1]: c
char[2]: !
char[3]:
CUSTOM LIST ::: SIZE 3
[0]: 'A'
[1]: 'c'
[2]: '!'
```

Вывод:

В результате выполнения данной лабораторной работы мне удалось получить практические навыки создания классов и переопределения стандартных методов-операторов.