ГУАП

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

acc.		Д.А.Кочин
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

«Функциональное тестирование методом черного ящика»

по курсу: Управление качеством программного обеспечения

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4831	08.10.2020	К.А.Корнющенков
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2020

1. Цель работы

Разработать функциональные тесты методом черного ящика для функции по соответствующему варианту.

2. Задание на лабораторную работу

1 - Разработать функцию в соответствии со своим вариантом.

2 - Разработать функциональные тесты для написанного кода методом черного ящика. Добиваться 100% прохождения тестов не нужно. Необходимо описать принципы выбора тестов.

Вариант 1. Компилятор простых арифметических выражений, например 2+(-5)*(7-8). Вход и выход в виде строк

3. Код программы

```
package com.company;

import java.util.LinkedList;
import java.util.Scanner;

class Pair<T,U> {
    public final T t;
    public final U u;

    public Pair(T t, U u) {
        this.t= t;
        this.u= u;
    }
}

public class Main {

    public static boolean isNumeric(char str) {
        try {
            Double.parseDouble(String.valueOf(str));
            return true;
        } catch(NumberFormatException e){
            return false;
        }
    }

    public static  Boolean checkData(String data){
        for (int i=0;i<data.length();i++){
            if ((i ==0 || i == data.length() - 1) && !isNumeric(data.cha-
rAt(i))) {
                return false;
            }
            if (isNumeric(data.charAt(i)) | String.valueOf(data.cha-
rAt(i)).equals("+")
                    | String.valueOf(data.charAt(i)).equals("*") |
String.valueOf(data.charAt(i)).equals("/") | String.valueOf(data.cha-
rAt(i)).equals("-")){
                continue;
            } else {
                return  false;
```

```
            }
        }
        return true;
    }

    public static Pair<LinkedList<Integer>, LinkedList<String>>
parseData(String data){
        LinkedList<Integer> number = new LinkedList<Integer>();
        LinkedList<String> action = new LinkedList<String>();
        String timeNumber = "";
        for (int i=0;i<data.length();i++){
            if (!isNumeric(data.charAt(i))){
                action.add(String.valueOf(data.charAt(i)));
                if (timeNumber != "") {
                    number.add(Integer.valueOf(timeNumber));
                    timeNumber = "";
                }
            } else{
                timeNumber += String.valueOf(data.charAt(i));
            }
        }
        if (timeNumber != "") {
            number.add(Integer.valueOf(timeNumber));
            timeNumber = "";
        }
        return new Pair(number,action);
    }

    public static Pair<LinkedList<Integer>, LinkedList<String>> hardAc-
tion(LinkedList<Integer> number, LinkedList<String> action){
        Integer i = 0;
        Boolean check = true;
        while (check){
            if (action.get(i).equals("*")) {
                number.set(i,number.get(i)*number.get(i+1));
                number.remove(i+1);
                action.remove(i+1-1);
                i = 0;
            }else if (action.get(i).equals("/")) {
                number.set(i,number.get(i)/number.get(i+1));
                number.remove(i+1);
                action.remove(i+1-1);
                i = 0;
            }
            i += 1;
            if (action.size() <= i){
                break;
            }
        }
        return new Pair(number,action);
    }

    public static String simpleAction(LinkedList<Integer> number,
LinkedList<String> action){
        Integer i = 0;

        while (!action.isEmpty()){
            if (action.get(i).equals("+")) {
                number.set(i,number.get(i)+number.get(i+1));
                number.remove(i+1);
                action.remove(i+1-1);
                i = 0;
            }else if (action.get(i).equals("-")) {
                number.set(i,number.get(i)-number.get(i+1));
```

```
                number.remove(i+1);
                action.remove(i+1-1);
                i = 0;
            }
        }
        return String.valueOf(number.get(0));
    }

    public static String action(String data){

        if (checkData(data)){
            Pair<LinkedList<Integer>, LinkedList<String>> parse =
parseData(data);
            parse = hardAction(parse.t,parse.u);
            return simpleAction(parse.t,parse.u);
        }else {
            return "Ошибка";
        }
    }

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Введите уравнение: ");
        String data = in.nextLine();
        System.out.println(action(data));
    }
}
```

4. Тесты

```
class MainTest {

    @org.junit.jupiter.api.Test
    void Test_checkData_1() {
        Boolean result = Main.checkData("4+5");
        Assert.assertEquals(true,result);
    }

    @org.junit.jupiter.api.Test
    void action_1(){
        String line = Main.action("5+5+1-2");
        Assert.assertEquals("9",line);
    }

    @org.junit.jupiter.api.Test
    void isNumeric() {
        boolean value = Main.isNumeric('2');
        Assert.assertEquals(true,value);
    }

    @org.junit.jupiter.api.Test
    void parseData() {
        Pair<LinkedList<Integer>, LinkedList<String>> parse =
Main.parseData("5+6");

        LinkedList<Integer> frs = new LinkedList<Integer>();
        frs.add(5);
        frs.add(6);

        LinkedList<String> scd = new LinkedList<String>();
        scd.add("+");
        Pair<LinkedList<Integer>, LinkedList<String>> eq = new Pair(frs,scd);
```

```
        Assert.assertEquals(eq.t,parse.t);
        Assert.assertEquals(eq.u,parse.u);
    }
}
```

5. Спецификация на тесты

Функция action (string)

Имя теста	Описание сценария	Входные данные	Выходные данные
action _1	Преобразование входных данных в строку, которое содержит чисто или ошибку	Строка: 5+5+1-2	Строка: 9
action_2	Преобразование входных данных в строку, которое содержит чисто или ошибку	Строка: 5+1*2	Строка: 7
action_3	Преобразование входных данных в строку, которое содержит чисто или ошибку	Строка: 5+4	Строка: 9
action_4	Преобразование входных данных в строку, которое содержит чисто или ошибку	Строка: +	Строка: не верные входные данные
action_5	Преобразование входных данных в строку, которое содержит чисто или ошибку	Строка: 5^2	Строка: не верные входные данные
action_6	Преобразование входных данных в строку, которое содержит чисто или ошибку	Строка: -5+2	Строка: не верные входные данные

6. Вывод

В ходе выполнения лабораторной работы были получены навыки написания тестов методом черного ящика.