ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

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

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3 |
| «Функциональное тестирование методом белого ящика» |
| по курсу: Управление качеством программного обеспечения |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4831 |  | 15.10.2020 |  | К.А.Корнющенков |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2020

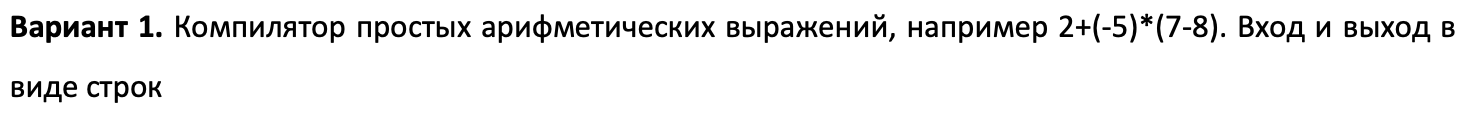
1. **Цель работы**

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

1. **Задание на лабораторную работу**

1 - Разработать функцию в соответствии со своим вариантом.

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



1. **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.charAt(i))) {  
 return false;  
 }  
 if (*isNumeric*(data.charAt(i)) | String.*valueOf*(data.charAt(i)).equals("+")  
 | String.*valueOf*(data.charAt(i)).equals("\*") | String.*valueOf*(data.charAt(i)).equals("/") | String.*valueOf*(data.charAt(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>> hardAction(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));  
 }  
}

1. **Тесты**

@org.junit.jupiter.api.Test  
void action\_white(){  
 String line = Main.*action*("-5+5+1-2");  
 Assert.*assertEquals*("Не вырные входные данные",line);  
}  
  
@org.junit.jupiter.api.Test  
void isNumeric\_white() {  
 boolean value = Main.*isNumeric*('q');  
 Assert.*assertEquals*(false,value);  
}  
  
@org.junit.jupiter.api.Test  
void Test\_checkData\_white() {  
 Boolean result = Main.*checkData*("4^5");  
 Assert.*assertEquals*(false,result);  
}

1. **Спецификация на тесты**

Функция action\_white (string)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя теста | Описание сценария | Входные данные | Выходные данные |
| action\_white(string) | Преобразование входных данных в строку, которое содержит чисто или сообщение: : “Не верные входные данные”, если входные данные были некорректно составлены | Строка: -5+5+1-2 | Строка: “Не верные входные данные” |

Функция isNumeric\_white (string)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя теста | Описание сценария | Входные данные | Выходные данные |
| isNumeric\_white (char) | Если символьный тип является числом, то возвращается true, в противном случае false | Символьный тип: ‘q’ | Результат вызова: false |

Функция checkData\_white (string)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя теста | Описание сценария | Входные данные | Выходные данные |
| checkData\_white (string) | Если строка удовлетворяет нашему алфавиту, то возвращается true, иначе false | Символьный тип: ‘4^5’ | Результат вызова: false |

1. **Вывод**

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