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 |  | 18.10.2020 |  | К.А.Корнющенков |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2020

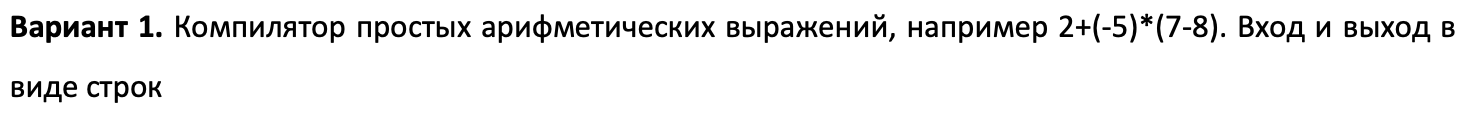
1. **Цель работы**

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**

class Pair<T,U> {  
 public final T t;  
 public final U u;  
  
 public Pair(T t, U u) {  
 this.t= t;  
 this.u= u;  
 }  
}  
  
enum ErrorEnum {  
 *Good*,  
 *ErrorSymbol*,  
 *ErrorFirstItemIsNotNumber*}  
  
public class Main {  
  
 public static boolean isNumeric(char str) {  
 try {  
 Integer.*parseInt*(String.*valueOf*(str));  
 return true;  
 } catch(NumberFormatException e){  
 return false;  
 }  
 }  
  
 public static ErrorEnum checkData(String data){  
 for (int i=0;i<data.length();i++){  
 if ((i == 0 || i == data.length() - 1) && !*isNumeric*(data.charAt(i))) {  
 return ErrorEnum.*ErrorFirstItemIsNotNumber*;  
 }  
 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 ErrorEnum.*ErrorSymbol*;  
 }  
 }  
 return ErrorEnum.*Good*;  
 }  
  
 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){  
 switch (*checkData*(data)){  
 case *Good*: {  
 Pair<LinkedList<Integer>, LinkedList<String>> parse = *parseData*(data);  
 parse = *hardAction*(parse.t, parse.u);  
 return *simpleAction*(parse.t, parse.u);  
 }  
 case *ErrorSymbol*:  
 return "Error - symbol";  
 case *ErrorFirstItemIsNotNumber*:  
 return "Error - first item is not number";  
 }  
 return null;  
 }  
  
 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**

class MainTest {  
  
 @org.junit.jupiter.api.Test  
 void action\_1(){  
 String line = Main.*action*("5+5\*1-2");  
 Assert.*assertEquals*("8",line);  
 }  
  
 @org.junit.jupiter.api.Test  
 void action\_2(){  
 String line = Main.*action*("5+1\*2");  
 Assert.*assertEquals*("7",line);  
 }  
  
 @org.junit.jupiter.api.Test  
 void action\_3(){  
 String line = Main.*action*("5/4");  
 Assert.*assertEquals*("1",line);  
 }  
  
 @org.junit.jupiter.api.Test  
 void action\_4(){  
 String line = Main.*action*("+5");  
 Assert.*assertEquals*("Error - first item is not number",line);  
 }  
  
 @org.junit.jupiter.api.Test  
 void action\_5(){  
 String line = Main.*action*("5^2");  
 Assert.*assertEquals*("Error - symbol",line);  
 }  
  
 @org.junit.jupiter.api.Test  
 void action\_6(){  
 String line = Main.*action*("-5+2");  
 Assert.*assertEquals*("Error - first item is not number",line);  
 }  
}

1. **Спецификация на тесты**

Функция action (string)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя теста | Описание сценария | Входные данные | Выходные данные |
| action \_1 | Входная строка корректная и нам вернется результат выражения | Строка: 5+5\*1-2 | Строка: 8 |
| action\_2 | Входная строка корректная и нам вернется результат выражения | Строка: 5+1\*2 | Строка: 7 |
| action\_3 | Входная строка корректная и нам вернется результат выражения | Строка: 5/4 | Строка: 1 |
| action\_4 | Входная строка начинается не с числа и из-за этого нам возвращается сообщение Error – first or last item is not number | Строка: +5 | Строка: Error – first or last item is not number |
| action\_5 | Входная строка содержит неизвестный символ (возведение в степень) и из-за этого нам возвращается сообщение Error - symbol | Строка: 5^2 | Строка: Error - symbol |
| action\_6 | Входная строка начинается не с числа и из-за этого нам возвращается сообщение Error – first or last item is not number | Строка: -5+2 | Строка: Error – first or last item is not number |

1. **Вывод**

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