Министерство Образования, Культуры,

Исследований Республики Молдова

Технический Университет Молдовы

Департамент Программная Инженерия и Автоматика

**Отчёт**

по лабораторной работе №4

**по дисциплине «TS»**

Выполнил: ст.гр. TI-197

Шарафудинов Николай

Проверил: Скороходова Татьяна

Кишинёв – 2022

**Вариант 19. -** «Умножение комплексных чисел»

**Тема:** Автоматизация процесса тестирования программного продукта.

**Цель работы:** Разработка драйверов тестирования в соответствии с лабораторными работами №1 и №2.

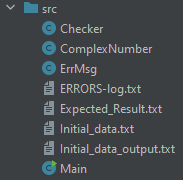
**Содержание отчета:**

1. Алгоритм работы драйвера тестирования.
2. Варианты тестирования (исходные данные, ожидаемые результаты).
3. Результат работы драйвера:
   1. вывод номера варианта тестирования;
   2. исходные данные;
   3. ожидаемые результаты;
   4. реальные результаты;
4. Приложение. Листинг программы

**Постановка задачи**

На вход принимаются значения действительных и мнимых частей двух комплексных чисел. Программа отображает результат на экран – произведение 2-х комплексных чисел.

**Структура проекта**



**Этап 1. Генерация входных данных**

В качестве входных данных были взяты числа в диапазоне [-10.0; 10.0], так же значения, выходящие за диапазон и не числовые значения.

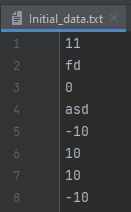
Так как в данной задаче количество обязательных поле 4 (Re1, Im1, Re2, Im2), соответственно всего вариантов проверки 2^m (где, m – количество обязательных полей) получим 16 вариантов. Так же в качестве не валидных данных добавлены.

**Этап 2. Алгоритм драйвера**

Алгоритм драйвера предельно прост. Работа драйвера заключается в последовательном считывании входных данных, отправке их тестируемому компоненту, получении результата и сравнении его с ожидаемым результатом. При этом если действительный результат не соответствует ожидаемому, то статус драйвера FAIL. В драйвере также необходимо логировать все результаты, результаты каждой итерации записывать в выходной файл и ее статус. Если результат хоть одной итерации FAIL, то драйвер должен продолжать свою работу. Таким образом в конце можно проанализировать все данные и выявить потенциальные дефекты.

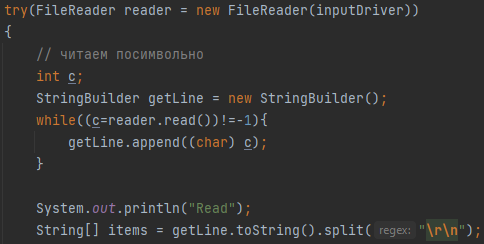
**1 ШАГ**

Создание файла с входными данными которые будут считываться драйвером



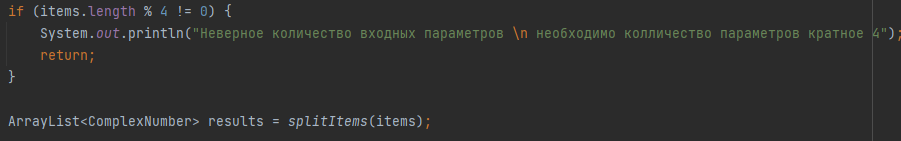
**2 ШАГ**

Чтение и обработка полученных значений из файла



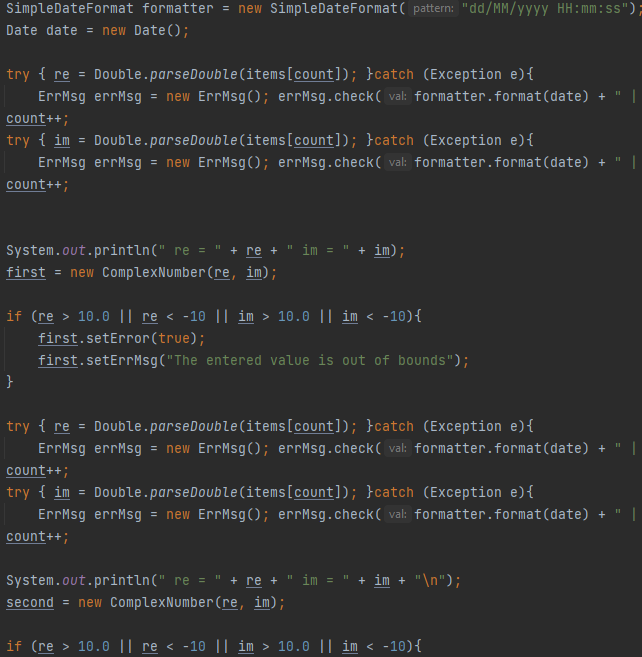
**3 ШАГ**

Проверив правильность количество тестируемых параметров вызываем функцию splitItems(items). Данная функция разделяет значения, полученные из файла в виде одной строки на отдельные элементы для записи их соответственно в комплексные числа.

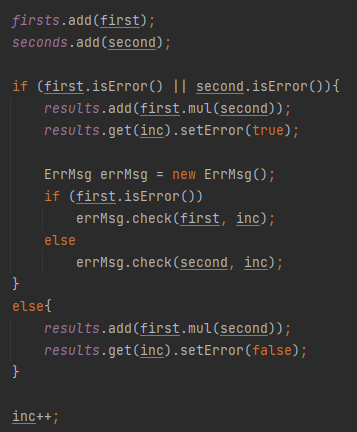


Внутри данной функции также производятся все проверки на то:

1. Правильно ли введено значение (проверка на то, что это число)
2. Проверка на принадлежность полученного числа к диапазону ([-10.0; 10.0])



Так же есть проверка на то была ли ошибка в первом комплексном числе или во втором и если она была, то в итоговый результат не считается так как в элементе была ошибка происходит переход к следующей итерации с новыми комплексными числами, а ошибка отправляется в отдельный файл, созданный для ошибок «ERRORS-log».



**Работа программы**

**1 ШАГ – запись входных данных**

11

fd

0

asd

-10

10

10

-10

0

0

10

-10

10

-10

0

100

-4

-4

7

-1

-4

-9

9

7

-6

6

9

2

6

-9

0

3

1

2

asd

5

**2 ШАГ – запись файла с ожидаемым результатом**

1. Result: 11.0 + 0.0i \* 0.0 + 0.0i = 0.0 + 0.0i

2. Result: -10.0 + 10.0i \* 10.0 + -10.0i = 0.0 + 200.0i

3. Result: 0.0 + 0.0i \* 10.0 + -10.0i = 0.0 + 0.0i

4. Result: 10.0 + -10.0i \* 0.0 + 0.0i = 0.0 + 0.0i

5. Result: -4.0 + -4.0i \* 7.0 + -1.0i = -32.0 + -24.0i

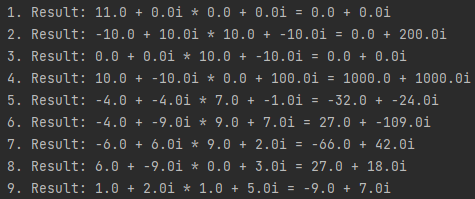
6. Result: -4.0 + -9.0i \* 9.0 + 7.0i = 27.0 + -109.0i

7. Result: -6.0 + 6.0i \* 9.0 + 2.0i = -66.0 + 42.0i

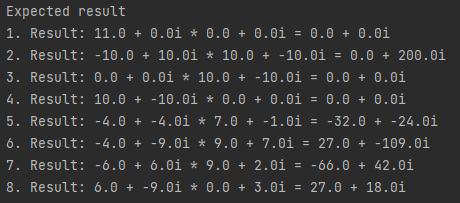
8. Result: 6.0 + -9.0i \* 0.0 + 3.0i = 27.0 + 18.0i

**3 ШАГ – запуск программы**

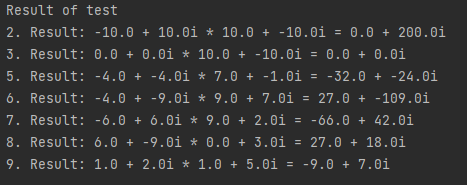
Изначально в консоли выводятся все варианты полученные от исходных данных



После этого выводится содержимое файла «Ожидаемый результат»



И за этим выводится что получилось на самом деле



Можно отметить, что нумерация местами идет не подряд что означает что в тех строках, которых тут нет произошла ошибка и их можно увидеть в «ERRORS-log».

Содержимое файла следующее

07/04/2022 00:31:57 | ERROR For input string: "fd" | Error Enter Im 2 | Error String 1 |

07/04/2022 00:31:57 | ERROR For input string: "asd" | Error Enter Im 4 | Error String 1 |

07/04/2022 00:31:57 | ERROR in 2 line the entered value is out of bounds (input element 6)

07/04/2022 00:31:57 | ERROR in 4 line the entered value is out of bounds (input element 16)

07/04/2022 00:31:57 | ERROR For input string: "asd" | Error Enter Re 35 | Error String 9 |

07/04/2022 00:31:57 | ERROR in 9 line the entered value is out of bounds (input element 34)

Здесь указана дата, время и тип ошибки. В случае введения не числа мы получим сообщение (ERROR For input string: "fd" | Error Enter Im 1 | Error String 0 |) содержащие:

1. что было введено (For input string: "fd")
2. для какой части комплексного числа (Enter Im)
3. в какой строке из файла входных данных это значение (Enter Im **1**)
4. а также номер строки примера, который должен был быть выведен на экран.

В случае, когда число за пределами [-10.0; 10.0] то выводится следующее сообщение (ERROR in 4 line the entered value is out of bounds (input element 16)) содержащие:

1. номер примера, в котором ошибка (ERROR in 4 line);
2. тип ошибки (the entered value is out of bounds);
3. из-за какого элемента в входных данных произошла эта ошибка (input element 16).

**Вывод**

В данной лабораторной работе был реализован драйвер позволяющий автоматизировать проверку приложения.

**Приложение. Листинг программы.**

ComplexNumber

public class ComplexNumber {  
 private double realPart;  
 private double imagePart;  
 private boolean error = false;  
 private String errMsg = "";  
  
 public void setErrMsg(String errMsg) {  
 this.errMsg = errMsg;  
 }  
  
 public String getErrMsg() {  
 return errMsg;  
 }  
  
 public void setError(boolean error) {  
 this.error = error;  
 }  
  
 public boolean isError() {  
 return error;  
 }  
  
 public ComplexNumber(double realPart, double imagePart) {  
 this.realPart = realPart;  
 this.imagePart = imagePart;  
 }  
  
 public double getRealPart() {  
 return realPart;  
 }  
  
 public double getImagePart() {  
 return imagePart;  
 }  
  
 public void setRealPart(double realPart) {  
 this.realPart = realPart;  
 }  
  
 public void setImagePart(double imagePart) {  
 this.imagePart = imagePart;  
 }  
  
 ComplexNumber mul (ComplexNumber a) {// умножить комплексные числа  
 double real2 = a.getRealPart();  
 double image2 = a.getImagePart();  
 double newReal = realPart \* real2 - imagePart \* image2;  
 double newImage = imagePart \* real2 + realPart \* image2;  
 return new ComplexNumber(newReal, newImage);  
 }  
  
 public String printComplexNumber(ComplexNumber f, ComplexNumber s){  
  
 return "Result: " + f.realPart + " + " + f.imagePart + "i \* " +  
 s.realPart + " + " + s.imagePart + "i = " +  
 this.realPart + " + " + this.imagePart + "i";  
 }  
}

ErrMsg

import java.io.FileWriter;  
import java.io.IOException;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
public class ErrMsg {  
  
 private String errMsg = "";  
  
 public ErrMsg() {  
 }  
  
 public ErrMsg(String errMsg) {  
 this.errMsg = errMsg;  
 }  
  
 public void check(ComplexNumber z, int position){  
  
 System.*out*.println();  
 String outDriver = "E:\\универ\\3.2\\SCO\_\_\_\_testS\\LAB4\\CODE\\src\\ERRORS-log.txt";  
  
 try(FileWriter writer = new FileWriter(outDriver, true))  
 {  
 if (z.isError())  
 this.errMsg += z.getErrMsg();  
  
 SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");  
 Date date = new Date();  
  
 // запись всей строки  
 StringBuilder text = new StringBuilder();  
 text.append(formatter.format(date) + " | ERROR in " + (position + 1) + " line " + errMsg).append("\n");  
  
 writer.write(text.toString());  
 writer.flush();  
 }  
 catch(IOException ex){  
 System.*out*.println(ex.getMessage());  
 }  
 }  
  
 public void check(String val){  
  
 System.*out*.println();  
 String outDriver = "E:\\универ\\3.2\\SCO\_\_\_\_testS\\LAB4\\CODE\\src\\ERRORS-log.txt";  
  
 try(FileWriter writer = new FileWriter(outDriver, true))  
 {  
 // запись всей строки  
 StringBuilder text = new StringBuilder();  
 text.append(val).append("\n");  
  
 writer.write(text.toString());  
 writer.flush();  
 }  
 catch(IOException ex){  
 System.*out*.println(ex.getMessage());  
 }  
  
 }  
  
}

Checker

import java.io.\*;  
  
public class Checker {  
 StringBuilder getLinef1;  
 StringBuilder getLinef2;  
 public void check() throws IOException {  
 String path1 = "E:\\универ\\3.2\\SCO\_\_\_\_testS\\LAB4\\CODE\\src\\Expected\_Result.txt";  
 String path2 = "E:\\универ\\3.2\\SCO\_\_\_\_testS\\LAB4\\CODE\\src\\Initial\_data\_output.txt";  
 try(FileReader reader = new FileReader(path1))  
 {  
 // читаем посимвольно  
 int c;  
 getLinef1 = new StringBuilder();  
 while((c=reader.read())!=-1) {  
 getLinef1.append((char) c);  
 }  
 }  
 try(FileReader reader = new FileReader(path2))  
 {  
 // читаем посимвольно  
 int c;  
 getLinef2 = new StringBuilder();  
 while((c=reader.read())!=-1) {  
 getLinef2.append((char) c);  
 }  
 }  
 catch (IOException e) {  
 e.printStackTrace();  
 }  
 String file1 = getLinef1.toString().replace("\r", "");  
 String file2 = getLinef2.toString().replace("\r", "");  
  
 System.*out*.println("\nExpected result\n" + file1);  
 System.*out*.println("Result of test\n" + file2);  
  
  
 if (file1.equals(file2))  
 System.*out*.println("All work good");  
 else  
 System.*out*.println("result is different");  
 }  
}

Main

import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
  
public class Main {  
  
 private static ArrayList<ComplexNumber> *results*;  
 private static ArrayList<ComplexNumber> *firsts*;  
 private static ArrayList<ComplexNumber> *seconds*;  
  
 public static void main(String[] args) throws Exception {  
  
 String inputDriver = "E:\\универ\\3.2\\SCO\_\_\_\_testS\\LAB4\\CODE\\src\\Initial\_data.txt";  
 String outDriver = "E:\\универ\\3.2\\SCO\_\_\_\_testS\\LAB4\\CODE\\src\\Initial\_data\_output.txt";  
 try(FileReader reader = new FileReader(inputDriver))  
 {  
 // читаем посимвольно  
 int c;  
 StringBuilder getLine = new StringBuilder();  
 while((c=reader.read())!=-1){  
 getLine.append((char) c);  
 }  
  
 System.*out*.println("Read");  
 String[] items = getLine.toString().split("\r\n");  
 *results* = new ArrayList<ComplexNumber>();  
 *firsts* = new ArrayList<ComplexNumber>();  
 *seconds* = new ArrayList<ComplexNumber>();  
  
 if (items.length % 4 != 0) {  
 System.*out*.println("Неверное количество входных параметров \n необходимо колличество параметров кратное 4");  
 return;  
 }  
  
 ArrayList<ComplexNumber> results = *splitItems*(items);  
  
 for (int i = 0; i < results.size(); i++) {  
 System.*out*.println((i+1) + ". " + results.get(i).printComplexNumber(*firsts*.get(i), *seconds*.get(i)));  
 }  
  
 try(FileWriter writer = new FileWriter(outDriver, false))  
 {  
 // запись всей строки  
 StringBuilder text = new StringBuilder();  
 for (int i = 0; i < results.size(); i++) {  
 if (!results.get(i).isError())  
 text.append( (i+1) + ". " + results.get(i).printComplexNumber(*firsts*.get(i), *seconds*.get(i))).append("\n");  
 }  
  
 writer.write(text.toString());  
 writer.flush();  
 }  
 catch(IOException ex){  
 System.*out*.println(ex.getMessage());  
 }  
  
 Checker checker = new Checker();  
 checker.check();  
 }  
 catch(Exception ex){  
 System.*out*.println(ex.getMessage());  
 }  
  
 }  
  
 private static ArrayList<ComplexNumber> splitItems(String[] items) throws Exception {  
  
 double range = 10;  
 int count = 0, inc = 0;  
 ComplexNumber first = null;  
 ComplexNumber second = null;  
 double re = 0, im = 0;  
 while (count != items.length){  
  
 SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");  
 Date date = new Date();  
  
 try { re = Double.*parseDouble*(items[count]); }catch (Exception e){  
 ErrMsg errMsg = new ErrMsg(); errMsg.check(formatter.format(date) + " | ERROR " + e.getMessage() + " | Error Enter Re " + (count + 1) + " | Error String " + (inc + 1) + " | " );}  
 count++;  
 try { im = Double.*parseDouble*(items[count]); }catch (Exception e){  
 ErrMsg errMsg = new ErrMsg(); errMsg.check(formatter.format(date) + " | ERROR " + e.getMessage() + " | Error Enter Im " + (count + 1) + " | Error String " + (inc + 1) + " | " );}  
 count++;  
  
  
 System.*out*.println(" re = " + re + " im = " + im);  
 first = new ComplexNumber(re, im);  
  
 if (re > 10.0 || re < -10 || im > 10.0 || im < -10){  
 first.setError(true);  
 first.setErrMsg("The entered value is out of bounds (input element " + count + ")");  
 }  
  
 try { re = Double.*parseDouble*(items[count]); }catch (Exception e){  
 ErrMsg errMsg = new ErrMsg(); errMsg.check(formatter.format(date) + " | ERROR " + e.getMessage() + " | Error Enter Re " + (count + 1) + " | Error String " + (inc + 1) + " | " );}  
 count++;  
 try { im = Double.*parseDouble*(items[count]); }catch (Exception e){  
 ErrMsg errMsg = new ErrMsg(); errMsg.check(formatter.format(date) + " | ERROR " + e.getMessage() + " | Error Enter Im " + (count + 1) + " | Error String " + (inc + 1) + " | " );}  
 count++;  
  
 System.*out*.println(" re = " + re + " im = " + im + "\n");  
 second = new ComplexNumber(re, im);  
  
 if (re > 10.0 || re < -10 || im > 10.0 || im < -10){  
 second.setError(true);  
 second.setErrMsg("The entered value is out of bounds (input element " + count + ")");  
  
 }  
  
 *firsts*.add(first);  
 *seconds*.add(second);  
  
 if (first.isError() || second.isError()){  
 *results*.add(first.mul(second));  
 *results*.get(inc).setError(true);  
  
 ErrMsg errMsg = new ErrMsg();  
 if (first.isError())  
 errMsg.check(first, inc);  
 else  
 errMsg.check(second, inc);  
 }  
 else{  
 *results*.add(first.mul(second));  
 *results*.get(inc).setError(false);  
 }  
  
 inc++;  
 }  
 return *results*;  
 }  
  
}

Initial\_data.txt

1  
fd  
0  
asd  
-10  
11  
10  
-10  
0  
0  
10  
-10  
10  
-10  
0  
100  
-4  
-4  
7  
-1  
-4  
-9  
9  
7  
-6  
6  
9  
2  
6  
-9  
0  
3  
1  
150  
asd  
5

Expected\_Result.txt

1. Result: 11.0 + 0.0i \* 0.0 + 0.0i = 0.0 + 0.0i  
2. Result: -10.0 + 10.0i \* 10.0 + -10.0i = 0.0 + 200.0i  
3. Result: 0.0 + 0.0i \* 10.0 + -10.0i = 0.0 + 0.0i  
4. Result: 10.0 + -10.0i \* 0.0 + 0.0i = 0.0 + 0.0i  
5. Result: -4.0 + -4.0i \* 7.0 + -1.0i = -32.0 + -24.0i  
6. Result: -4.0 + -9.0i \* 9.0 + 7.0i = 27.0 + -109.0i  
7. Result: -6.0 + 6.0i \* 9.0 + 2.0i = -66.0 + 42.0i  
8. Result: 6.0 + -9.0i \* 0.0 + 3.0i = 27.0 + 18.0i

Result.txt

1. Result: 1.0 + 0.0i \* 0.0 + 0.0i = 0.0 + 0.0i  
3. Result: 0.0 + 0.0i \* 10.0 + -10.0i = 0.0 + 0.0i  
5. Result: -4.0 + -4.0i \* 7.0 + -1.0i = -32.0 + -24.0i  
6. Result: -4.0 + -9.0i \* 9.0 + 7.0i = 27.0 + -109.0i  
7. Result: -6.0 + 6.0i \* 9.0 + 2.0i = -66.0 + 42.0i  
8. Result: 6.0 + -9.0i \* 0.0 + 3.0i = 27.0 + 18.0i