UNIVERSITÀ DEGLI STUDI
DI SALERNO

# PROGRAMMABLE CALCULATOR

Testing Document

13th DECEMBER 2023

DE LUCA A. - D'AGOSTINO M. - DE LUCA D. - FESTA R.

- GROUP 21 -

# Index

# 1 About

## 1.1 Team Information

| Name | ID |
|---|---|
| D'Agostino Marco | *0612705817* |
| De Luca Aniello *(group leader)* | *0612705805* |
| De Luca Daniele | *0612705654* |
| Festa Raffaele | *0612705535* |

## 1.2 Version

| Revision | Date | Summary of changes |
|---|---|---|
| **1.0** | 13/12/2023 | First release |
| | | |

# 2 Introduction

## 2.1 Purpose

The purpose of this project is to make a calculator app that uses a Stack to work with both complex and real numbers. By manipulating the Stack, the app enables users to carry out a range of mathematical operations, providing a flexible tool for numerical computation. Moreover, the app's interface shows the top elements of the Stack to simplify the input of complex expressions. Variable storage allows users to store and retrieve values when making calculations.

## 2.2 Document Scope

This document covers functional, integration, and system testing of the calculator application. It does not include performance or security testing.

## 2.3 Pass/Fail Criteria

Each test case will have specific pass/fail criteria, which will be detailed in the subsequent sections. Generally, a test is considered passed if the application behaves as expected and produces accurate results under the defined conditions.

## 2.4 Approach

The testing approach will be a combination of automated and manual testing strategies. Automated tests, primarily using JUnit for unit testing, will cover the majority of the functionalities, while manual testing will be employed for user interface validations.

# 3 Functional Test Cases

| FTC-1 | User click on Value or Operation buttons |
|---|---|
| **Precondition** | The user launches the calculator and visualizes the entire GUI interface and all its components. |
| **Flow of Events** | The User clicks the desired button |
| **Postcondition** | Clicked button’s value is displayed in the GUI Text Area |
| **Use Cases** | Enter Value (UC-1) |

| FTC-2 | User click on Execute button |
|---|---|
| **Precondition** | The user launches the calculator and visualizes the entire GUI interface and all its components. |
| **Flow of Events** | FTC-1<br>User clicks the Execute button on the GUI |
| **Postcondition** | Values in the Text Area are no longer displayed<br>Accepted values are processed and/or displayed in the GUI’s StackView |
| **Use Cases** | Execute Command (UC-2)<br>Commit to Stack (UC-4)<br>Perform Operation (UC-5) |

| FTC-3 | User click on StackManipulation buttons |
|---|---|
| **Precondition** | The user launches the calculator and visualizes the entire GUI interface and all its components. |
| **Flow of Events** | The user clicks on a number<br>FTC-2<br>The user clicks StackManupulation button on the GUI |
| **Postcondition** | The Stack manipulation is executed<br>StackView is refreshed. |
| **Use Cases** | Execute Stack Manipulation (UC-3) |

| FTC-4 | User click on Keychange buttons |
|---|---|
| **Precondition** | The user launches the calculator and visualizes the entire GUI interface and all its components. |
| **Flow of Events** | The user clicks Keychange button on the GUI |
| **Postcondition** | The Gui is refreshed showing a modified layout and the relating inteface, allowing the user to enter variables |
| **Use Cases** | Enter Value (UC-1) |

| FTC-5 | User click on Variables buttons |
|---|---|
| **Precondition** | The user launches the calculator and visualizes the entire GUI interface and all its components.<br>FTC-4 |
| **Flow of Events** | The user click on a variable button |
| **Postcondition** | The variable is displayed in the GUI Text Area |
| **Use Cases** | Enter Value (UC-1) |

# 4 Test Cases

## 4.1 Class ComplexNumber

| UTC-1.1 | Test ComplexNumber.getReal |
|---|---|
| **Test items** | class ComplexNumber, method getReal() |
| **Input** | ComplexNumber n = new ComplexNumber(3,4);<br>double out = n.getReal(); |
| **Oracle** | out == 3 |

| UTC-1.2 | testGetImaginary() |
|---|---|
| **Test items** | class ComplexNumber, method getImaginary() |
| **Input** | ComplexNumber n = new ComplexNumber(3,4);<br>double out = n.getImaginary(); |
| **Oracle** | out == 4 |

| UTC-1.3 | testAdd() |
|---|---|
| **Test items** | class ComplexNumber, method add() |
| **Input** | ComplexNumber n = new ComplexNumber(3,4);<br>ComplexNumber other = new ComplexNumber(2,3);<br>ComplexNumber out = n.add(other); |
| **Oracle** | out.getReal() == 5 && out.getImaginary() == 7 |

| UTC-1.4 | testSubtract() |
|---|---|
| **Test items** | class ComplexNumber, method subtract() |
| **Input** | ComplexNumber n = new ComplexNumber(3,4);<br>ComplexNumber other = new ComplexNumber(2,3);<br>ComplexNumber out = n.subtract(other); |
| **Oracle** | out.getReal() == 1 && out.getImaginary() == 1) |

| UTC-1.5 | testMultiply() |
|---|---|
| **Test items** | class ComplexNumber, method multiply() |
| **Input** | ComplexNumber n = new ComplexNumber(3,4);<br>ComplexNumber other = new ComplexNumber(2,3);<br>ComplexNumber out = n.multiply(other); |
| **Oracle** | out.getReal() == -6 && out.getImaginary() == 17 |

| **UTC-1.7** | **testDivide()** |
|---|---|
| **Test items** | class ComplexNumber, method divide() |
| **Input** | ComplexNumber n = new ComplexNumber(3,4);<br>ComplexNumber other = new ComplexNumber(2,3);<br>ComplexNumber out = n.divide(other); |
| **Oracle** | out.getReal() == 3.9 && out.getImaginary() == -0.3 |

| **UTC-1.8** | **testSquareRoot()** |
|---|---|
| **Test items** | class ComplexNumber, method squareRoot() |
| **Input** | ComplexNumber n = new ComplexNumber(3,4);<br>ComplexNumber out = n.squareRoot(); |
| **Oracle** | out.getReal() == 2 && out.getImaginary() == 1 |

| **UTC-1.9** | **testInvertSign()** |
|---|---|
| **Test items** | class ComplexNumber, method invertSign() |
| **Input** | ComplexNumber n = new ComplexNumber(3,4);<br>ComplexNumber out = n.invertSign(); |
| **Oracle** | out.getReal() == -3 && out.getImaginary() == 4 |

| **UTC-1.10.1** | **testComplexParse1()** |
|---|---|
| **Test items** | class ComplexNumber, method complexParse() |
| **Input** | ComplexNumber n = ComplexNumber.complexParse(“3+4j”); |
| **Oracle** | n.getReal() == 3 && n.getImaginary() == 4 |

| **UTC-1.10.2** | **testComplexParse2()** |
|---|---|
| **Test items** | class ComplexNumber, method complexParse() |
| **Input** | ComplexNumber n = ComplexNumber.complexParse(-3+4j”); |
| **Oracle** | n.getReal() == -3 && n.getImaginary() == 4 |

| UTC-1.10.3 | testComplexParse3() |
|---|---|
| **Test items** | class ComplexNumber, method complexParse() |
| **Input** | ComplexNumber n = ComplexNumber.complexParse(“3-4j”); |
| **Oracle** | n.getReal() == 3 && n.getImaginary() == -4 |

| UTC-1.10.4 | testComplexParse4() |
|---|---|
| **Test items** | class ComplexNumber, method complexParse() |
| **Input** | ComplexNumber n = ComplexNumber.complexParse(“0+4j”); |
| **Oracle** | n.getReal() == 0 && n.getImaginary() == 4 |

| UTC-1.10.5 | testComplexParse5() |
|---|---|
| **Test items** | class ComplexNumber, method complexParse() |
| **Input** | ComplexNumber n = ComplexNumber.complexParse(“3”); |
| **Oracle** | n.getReal() == 3 && n.getImaginary() == 0 |

| UTC-1.10.6 | testComplexParse6() |
|---|---|
| **Test items** | class ComplexNumber, method complexParse() |
| **Input** | ComplexNumber n = ComplexNumber.complexParse(“+j”); |
| **Oracle** | n.getReal() == 0 && n.getImaginary() == 1 |

| UTC-1.10.7 | testComplexParse7() |
|---|---|
| **Test items** | class ComplexNumber, method complexParse() |
| **Input** | ComplexNumber n = ComplexNumber.complexParse(“3+0j”); |
| **Oracle** | n.getReal() == 3 && n.getImaginary() == 0 |

| UTC-1.11 | Test ComplexNumber.toString |
|---|---|
| **Test items** | class ComplexNumber, method toString() |
| **Input** | ComplexNumber other = new ComplexNumber()<br>SimIO simIO = new SimIO();<br>simIO.captureOutput();<br>System.out.println(other); |
| **Oracle** | simIO.getCapturedOutput().trim() == “2,00 + 3,00j” |

## 4.2 Class StackNumber

| UTC-2.1 | Test StackNumber.pushNumber |
|---|---|
| **Test items** | class StackNumber, method pushNumber() |
| **Input** | StackNumber stackNumber = new StackNumber();<br>ComplexNumber n = new ComplexNumber(1,10);<br>stackNumber.pushNumber(n); |
| **Oracle** | n == stackNumber.peekNumber()<br>stackNumber.isEmpty == false |

| UTC-2.2 | Test StackNumber.getStackSize |
|---|---|
| **Test items** | class StackNumber, method getStackSize() |
| **Input** | StackNumber stackNumber = new StackNumber();<br>ComplexNumber n = new ComplexNumber(1,10);<br>stackNumber.pushNumber(n); |
| **Oracle** | Stacknumber.getStackSize() == 1 |

| UTC-2.3 | Test StackNumber.isEmpty |
|---|---|
| **Test items** | class StackNumber, method isEmpty() |
| **Input** | StackNumber stackNumber = new StackNumber();<br>ComplexNumber n = new ComplexNumber(1,10); |
| **Oracle** | stackNumber.getStackSize() == 0<br>stackNumber.isEmpty() = true |

| UTC-2.4 | Test StackNumber.peekNumber |
|---|---|
| **Test items** | class StackNumber, method peekNumber() |
| **Input** | StackNumber stackNumber = new StackNumber();<br>ComplexNumber n = new ComplexNumber(1,10);<br>stackNumber.pushNumber(n); |
| **Oracle** | Stacknumber.peekNumber() == n |

| UTC-2.5 | Test StackNumber.dropNumber |
|---|---|
| **Test items** | class StackNumber, method dropNumber() |
| **Input** | StackNumber stackNumber = new StackNumber();<br>ComplexNumber n = new ComplexNumber(1,10);<br>stackNumber.pushNumber(new ComplexNumber(20,32));<br>stackNumber.pushNumber(n); |
| **Oracle** | stackNumber.peekNumber != n |

| UTC-2.6 | Test StackNumber.clearNumber |
|---|---|
| **Test items** | class StackNumber, method clearNumber() |
| **Input** | StackNumber stackNumber = new StackNumber();<br>ComplexNumber n = new ComplexNumber(1,10);<br>stackNumber.pushNumber(n);<br>stackNumber.pushNumber(new ComplexNumber(20,32));<br>stackNumber.pushNumber(new ComplexNumber(24,25.4)); |
| **Oracle** | Stacknumber.isEmpty() == true |

| UTC-2.7 | Test StackNumber.dupNumber |
|---|---|
| **Test items** | class StackNumber, method dupNumber() |
| **Input** | StackNumber stackNumber = new StackNumber();<br>ComplexNumber n = new ComplexNumber(1,10);<br>stackNumber.pushNumber(new ComplexNumber(20,32));<br>stackNumber.pushNumber(n); |
| **Oracle** | stackNumber.dropNumber() == n<br>stackNumber.peekNumber() == n |

| UTC-2.8 | Test StackNumber.swapNumber |
|---|---|
| **Test items** | class StackNumber, method swapNumber() |
| **Input** | StackNumber stackNumber = new StackNumber();<br>ComplexNumber n = new ComplexNumber(1,10);<br>stackNumber.pushNumber(n);<br>stackNumber.pushNumber(new ComplexNumber(20,32)); |
| **Oracle** | Stacknumber.peekNumber() == n |

| UTC-2.9 | Test StackNumber.overNumber |
|---|---|
| **Test items** | class StackNumber, method overNumber() |
| **Input** | StackNumber stackNumber = new StackNumber();<br>ComplexNumber n = new ComplexNumber(1,10);<br>stackNumber.pushNumber(n);<br>stackNumber.pushNumber(new ComplexNumber(20,32)); |
| **Oracle** | Stacknumber.peekNumber() == n |

| UTC-2.10 | Test StackNumber.getNumber() |
|---|---|
| **Test items** | class StackNumber, method pushNumber() |
| **Input** | StackNumber stackNumber = new StackNumber();<br>ComplexNumber n = new ComplexNumber(1,10);<br>stackNumber.pushNumber(n); |
| **Oracle** | Stacknumber.getNumber(0) == n.toString() |

## 4.3 Class Execute

| UTC-3.1 | Test Execute.getVar |
|---|---|
| **Test items** | Class Execute, method getVar() |
| **Input** | Execute exe = new Execute(); |
| **Oracle** | exe.getVar() != null |

| UTC-3.2 | Test Execute.getStack |
|---|---|
| **Test items** | Class Execute, method getStack() |
| **Input** | Execute exe = new Execute(); |
| **Oracle** | exe.getStack() != null |

| UTC-3.3.1 | Test1 Execute.elaborateTextArea |
|---|---|
| **Test items** | Class Execute, method elaborateTextArea() |
| **Input** | Execute exe = new Execute();<br>ComplexNumber n2 = new ComplexNumber(14,22);<br>String str = "14+22j";<br>exe.elaborateTextArea(str);<br>ComplexNumber out = exe.getStack().peekNumber(); |
| **Oracle** | out == n2 |

| UTC-3.3.2 | Test2 Execute.elaborateTextArea |
|---|---|
| **Test items** | Class Execute, method elaborateTextArea() |
| **Input** | Execute exe = new Execute();<br>ComplexNumber n1 = new ComplexNumber(10,32);<br>ComplexNumber n2 = new ComplexNumber(14,22);<br>exe.getStack().pushNumber(n1);<br>exe.getStack().pushNumber(n2);<br>ComplexNumber sum = n1.add(n2);<br>String str = "+";<br>exe.elaborateTextArea(str);<br>ComplexNumber out = exe.getStack().peekNumber(); |
| **Oracle** | out == sum |

| UTC-3.3.3 | Test3 Execute.elaborateTextArea |
|---|---|
| **Test items** | Class Execute, method elaborateTextArea() |
| **Input** | Execute exe = new Execute();<br>ComplexNumber n2 = new ComplexNumber(14,22);<br>String str = ">A";<br>exe.elaborateTextArea(str);<br>ComplexNumber out = exe.getVar().searchVariable(str.charAt(1)); |
| **Oracle** | out == n2 |

## 4.4 Class Operation

| UTC-4.1 | Test Operation.perform |
|---|---|
| **Test items** | Class Operation, method perform() |
| **Input** | StackNumber numbers = new StackNumber();<br>ComplexNumber n2 = new ComplexNumber(9,0);<br>String operators = "√±";<br>ComplexNumber temp = n2.squareRoot().invertSign(); |
| **Oracle** | numbers.peekNumber() == temp |

| UTC-4.2 | Test Operation.permform |
|---|---|
| **Test items** | Class Operation, method perform() |
| **Input** | StackNumber numbers = new StackNumber();<br>ComplexNumber n1 = new ComplexNumber(5,4);<br>ComplexNumber n2 = new ComplexNumber(9,0);<br>String operators = "+";<br>ComplexNumber temp = n1.add(n2); |
| **Oracle** | numbers.peekNumber == temp |

| UTC-4.2.2 | Test Operation.permform |
|---|---|
| **Test items** | Class Operation, method perform() |
| **Input** | StackNumber numbers = new StackNumber();<br>ComplexNumber n1 = new ComplexNumber(5,4);<br>ComplexNumber n2 = new ComplexNumber(9,0);<br>String operators = "-";<br>ComplexNumber temp = n1.subtract(n2); |
| **Oracle** | numbers.peekNumber == temp |

| UTC-4.2.3 | Test Operation.permform |
|---|---|
| **Test items** | Class Operation, method perform() |
| **Input** | StackNumber numbers = new StackNumber();<br>ComplexNumber n1 = new ComplexNumber(5,4);<br>ComplexNumber n2 = new ComplexNumber(9,0);<br>String operators = "*";<br>ComplexNumber temp = n1.multiply(n2); |
| **Oracle** | numbers.peekNumber == temp |

| UTC-4.2.4 | Test Operation.permform |
|---|---|
| **Test items** | Class Operation, method permformBinaryCase() |
| **Input** | StackNumber numbers = new StackNumber();<br>ComplexNumber n1 = new ComplexNumber(5,4);<br>ComplexNumber n2 = new ComplexNumber(9,0);<br>String operators = "/";<br>ComplexNumber temp = n1.divide(n2); |
| **Oracle** | numbers.peekNumber == temp |

## 4.5 Class Variables

| UTC-5.1 | Test Variables.getVariables |
|---|---|
| **Test items** | Class Variables, method getVariables() |
| **Input** | Variables variables = new Variables(); |
| **Oracle** | Variables.getVariables() != null |

| UTC-5.2 | Test Variables.searchVariable |
|---|---|
| **Test items** | Class Variables, method searchVariable() |
| **Input** | Char tempchar = ‘A’<br>Variables variables = new Variables();<br>ComplexNumber tempnumber = new ComplexNumber(10,20);<br>variables.getVariables().put(tempchar,tempnumber); |
| **Oracle** | variables.searchVariable(tempchar) == tempnumber |

| UTC-5.3.1 | Test Variables.perform |
|---|---|
| **Test items** | Class Variables, method  perform() |
| **Input** | Variables variables = new Variables();<br>StackNumber numbers = new StackNumber();<br>ComplexNumber tempnumber = new ComplexNumber(10,20);<br>numbers.pushNumber(tempnumber);<br>variables.perform(">A",numbers); |
| **Oracle** | variables.searchVariables(‘A’) == tempnumber |

| UTC-5.3.2 | Test Variables.perform |
|---|---|
| **Test items** | Class Variables, method perform() |
| **Input** | Variables variables = new Variables();<br>StackNumber numbers = new StackNumber();<br>ComplexNumber tempnumber = new ComplexNumber(10,20);<br>numbers.pushNumber(tempnumber);<br>variables.getVariables().put('A',tempnumber);<br>variables.perform("<A",numbers); |
| **Oracle** | numbers.peekNumber() == variables.searchVariable(‘A’) |

| UTC-5.3.3 | Test Variables.perform |
|---|---|
| **Test items** | Class Variables, method perform() |
| **Input** | Variables variables = new Variables();<br>StackNumber numbers = new StackNumber();<br>ComplexNumber tempnumber = new ComplexNumber(10,20);<br>numbers.pushNumber(tempnumber);<br>variables.perform(">A",numbers);<br>variables.perform("+A",numbers); |
| **Oracle** | variables.searchVariable(‘A’)  == tempnumber.add(numbers.peekNumber()) |

| UTC-5.3.4 | Test Variables.perform |
|---|---|
| **Test items** | Class Variables, method perform() |
| **Input** | Variables variables = new Variables();<br>StackNumber numbers = new StackNumber();<br>ComplexNumber tempnumber = new ComplexNumber(10,20);<br>numbers.pushNumber(tempnumber);<br>variables.perform(">A",numbers);<br>variables.perform("-A",numbers); |
| **Oracle** | variables.searchVariable(‘A’) == tempnumber.subract(numbers.peekNumber()) |

| UTC-5.3.5 | Test Variables.perform |
|---|---|
| **Test items** | Class Variables, method perform() |
| **Input** | Variables variables = new Variables();<br>StackNumber numbers = new StackNumber();<br>ComplexNumber tempnumber = new ComplexNumber(10,20);<br>numbers.pushNumber(tempnumber);<br>variables.perform(">A",numbers);<br>variables.perform("*A",numbers); |
| **Oracle** | variables.searchVariable(‘A’) == tempnumber.multiply(numbers.peekNumber()) |

| UTC-5.3.6 | Test Variables.perform |
|---|---|
| **Test items** | Class Variables, method perform() |
| **Input** | Variables variables = new Variables();<br>StackNumber  numbers = new StackNumber();<br>ComplexNumber tempnumber = new ComplexNumber(10,20);<br>numbers.pushNumber(tempnumber);<br>variables.perform(">A",numbers);<br>variables.perform("/A",numbers); |
| **Oracle** | variables.searchVariable(‘A’)  == tempnumber.divide(numbers.peekNumber()) |

| UTC-5.3.7 | Test Variables.perform |
|---|---|
| **Test items** | Class Variables, method perform() |
| **Input** | Variables variables = new Variables();<br>StackNumber numbers = new StackNumber();<br>ComplexNumber tempnumber = new ComplexNumber(10,20);<br>numbers.pushNumber(tempnumber);<br>variables.perform(">A",numbers);<br>variables.perform("√A",numbers); |
| **Oracle** | variables.searchVariable('A') == numbers.peekNumber().squareRoot() |

| UTC-5.3.8 | Test Variables.perform |
|---|---|
| **Test items** | Class Variables, method perform() |
| **Input** | Variables variables = new Variables();<br>StackNumber numbers = new StackNumber();<br>ComplexNumber tempnumber = new ComplexNumber(10,20);<br>numbers.pushNumber(tempnumber);<br>variables.perform(">A",numbers);<br>variables.perform("±A",numbers); |
| **Oracle** | variables.searchVariable('A') == numbers.peekNumber().invertSign() |

# 5 Tracebility Matrix

The status of each requirement will be described in this traceability matrix.

| Tracebility Matrix – Group 21 – v1.3 | | | | |
|---|---|---|---|---|
| Req-ID | Design | Code | Test | Related Requirements |
| UI-1.1 | x | x | x | N/A |
| UI-1.2 | x | | x | N/A |
| IS-1.1 | x | | x | N/A |
| IF-1.1 | x | x | x | N/A |
| DF-1.1 | x | x | x | IF-1.1 |
| DF-1.2 | x | x | x | IF-1.1 |
| IF-1.2 | x | x | x | N/A |
| DF-1.3 | x | x | x | IF-1.2 |
| IF-1.3 | x | x | x | N/A |
| DF-1.4 | x | x | x | IF-1.3 |
| DF-1.5 | x | x | x | IF-1.3 |
| DF-1.6 | x | x | x | IF-1.3 |
| UI-2.1 | x | | x | N/A |
| IF-2.1 | x | x | x | N/A |
| UI-2.2 | x | | x | IF-2.1 |
| IS-2.1 | x | x | x | IF-2.1 |
| IS-2.2 | x | x | x | IF-2.1 |
| IS-2.3 | x | x | x | IF-2.1 |
| UI-3.1 | x | | x | N/A |
| IS-3.1 | x | | x | N/A |
| IF-3.1 | x | x | x | N/A |
| IF-3.2 | x | x | x | N/A |
| IF-3.3 | x | x | x | N/A |
| IF-3.4 | x | x | x | N/A |
| IF-3.5 | x | x | x | N/A |