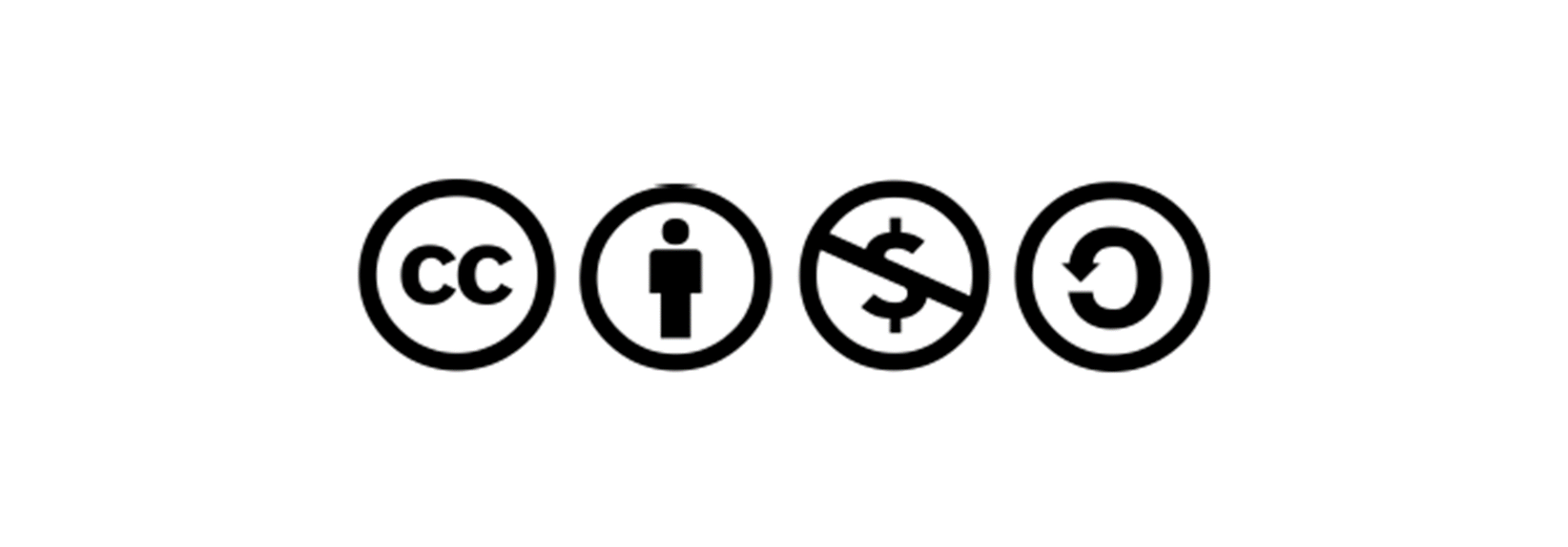
****

**Progetto di Software Engineering**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Software Design Document**

## **Gruppo 23**

A cura di:

Aniello Cione Mat.0622701628

Michela Corso Mat. 0622701841

Roberta Leo Mat.0622701831

Anno accademico: 2021/2022

## **1.1** **INTRODUCTION**

## **1.1** **Purpose**

This software design document describes the architecture and system design of a Scientific Programmable Calculator based on complex numbers.

### **1.1.2 Users.**

The scientific calculator is specially designed for any person who needs to perform operations on complex numbers.

### **1.1.3 Location**

The application will be launched by any user in possession of a personal computer. It will not be necessary to perform installations, as the application will be portable. It is not supported on mobile devices.

## **1.2** **Scope**

The main scope of the software is to create a user-friendly interface, allowing the user to perform even with complex numbers or not in an easy way.

Goals:

1. To view the last inserted elements through a text area.

2. To save partial values in variables for use in another session.

3. To introduce new user-defined operations

**1.3 Definitions and Acronyms**

**Complex Number**

A complex number is defined as “z = a + bj”. The complex number will therefore be defined by a real part “Re (z) = a” and “Im (z) = b”. The term j indicates "the imaginary unit": a complex number such that j² = -1.

Suppose you want to calculate the result of the equation x² + 1 = 0. Such an equation would have no solution in the field of Real numbers, however in the field of complex numbers, there is a solution. There is no real number that squared to -1.

Hence the second degree equation: x² + 1 = 0 admits solutions in particular, it will admit the two distinct complex roots x₁ = + j, x₂ = -j.

**2.0** **DEVELOPMENT TECHNOLOGIES**

The application follows the object-oriented paradigm. The development language is Java.

The development environment used is Apache NetBeans.

Testing Libraries: JUnit.

**3.0** **SYSTEM ARCHITECTURE**

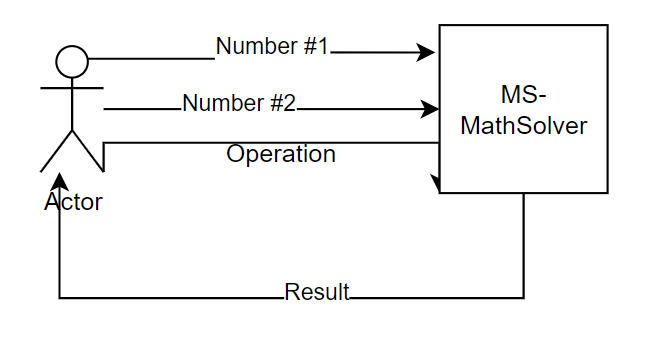
## **3.1** **Architectural Design**

A system based on Event-Driven architecture was developed for the calculator. The system components interact with the outside and communicate with each other through events. No client-server architecture has been defined, as the application was designed to work only locally.

Based on the events generated by the user interacting with the graphical interface, a series of internal events are generated to solve the task requested by the user. An example of the events potentially generated by the user are the insertion of complex numbers and operations within a text box or mouse click to save user-defined operations.

In the following diagrams the way in which the various components interact with each other will be explained.

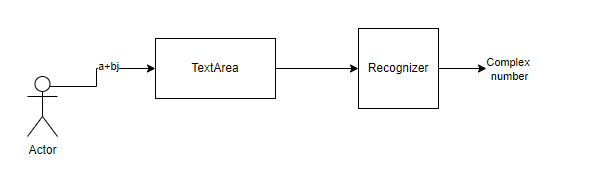
**Operation Flow**



**3.1.1 Complex number**

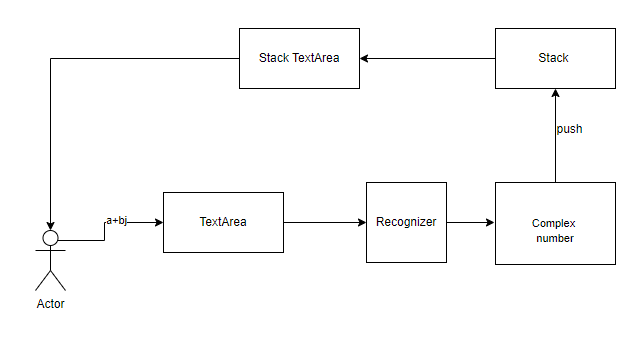
The first module corresponds to the insertion of a complex number in the textarea within the interface of the scientific calculator, in Cartesian notation.

The external event, in this case, corresponds to this entry, while the relative response is identified with the interpretation of the complex number by the Recognizer.



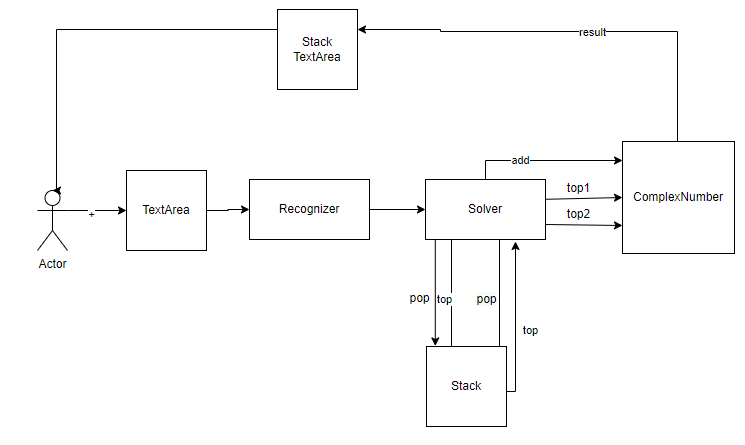
**3.1.2 Saving on memory**

Once the number has been entered into the textarea, after being interpreted as a complex number, as a response to the event, it is saved on the stack through a push. With the memory update, the display is also updated where the elements present within the stack are visible.

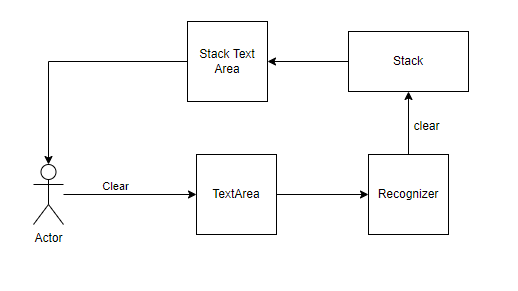


**3.1.3 Predefined operations execution**

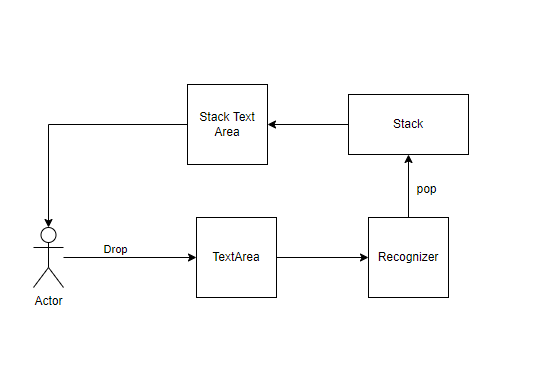
Once an operation to be performed within the textarea has been entered, the corresponding response is equivalent to the execution by the calculator of this operation which aims to produce the desired result. This result will be appended to the top of the stack and shown on the screen, together with the previous results.

From the user's point of view, the application appears to be a **black box**, once values have been entered and an operation is requested, the result is obtained. The application hides the implementation details. When an operation is requested, the string is passed to a recognizer. The recognizer, discovering that it is a predefined operation, passes the operation itself to the solver. The solver removes the last two values present within the stack and performs the operation on the numbers, obtaining the result and saving it on the stack. The stack will update the text area associated with it.

**3.1.4 Stack Operations**

The user can insert predefined operations within the text area that perform operations on the stack.

The **clear** operation, once recognized by the recognizer, allows you to delete all the elements from the stack. Then, the clear method of the stack is called and the textArea is updated.

After recognizing the **DROP** command, the last element of the stack is taken and the penultimate element is read. We proceed by updating the text Area relating to the stack.

**4.0** **HUMAN INTERFACE DESIGN**

When the user wants to carry out an operation, he inserts one number at a time within the text area, which will be inserted into the stack.

The user can choose to enter numbers using the buttons provided by the graphic interface or can enter numbers from the keyboard.

The user can enter decimal numbers using the "." made available from the graphic interface or by typing the number from the keyboard.

The insertion of numbers within the stack can be done using the "push" button made available by the graphical interface, or through the "Enter" key on the keyboard.

Subsequently, when defining the operation to be carried out on the numbers entered, the result of the operation will be shown within the stack.

When the user defines a binary operation, this will be performed on the last two numbers present within the stack; instead, when the user defines a unary operation, this will be carried out on the last element present within the stack.