**Product Backlog**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Priority | Product Backlog Items | User Story # | User Stories | Story Points | State |
| **1** | Insert Complex Number | **1** | AS A user I WANT TO insert a complex number SO I CAN use them as operands | 21 | Done |
| **1** | Addition two numbers | **2** | AS A user I WANT TO add two complex numbers SO I CAN have the result I want | 5 | Done |
| **1** | Subtraction two numbers | **3** | AS A user I WANT TO subtract two complex numbers SO I CAN have the result I want | 5 | Done |
| **1** | Multiply two numbers | **4** | AS A user I WANT TO multiply two complex numbers SO I CAN have the result I want | 5 | Done |
| **1** | Divide two numbers | **5** | AS A user I WANT TO divide two complex numbers SO I CAN have the result I want | 5 | Done |
| **1** | Square root of a number | **6** | AS A user I WANT TO calculate the root of a complex number SO I CAN have the result I want | 5 | Next Sprint |
| **1** | Invert sign of a number | **7** | AS A user I WANT TO invert the sign of a complex number SO I CAN have the result I want | 5 | Next Sprint |
| **1** | Visualize the last 12 elements of the stack | **8** | AS A user I WANT TO visualize the last 12 elements on the stack SO I CAN interact with the calculator | 21 | Done |
| **2** | Clear the numbers stack | **9** | AS A user I WANT TO clear the numbers stack SO I CAN restart the calculator | 1 | Next Sprint |
| **2** | Remove the last element of numbers stack | **10** | AS A user I WANT TO remove the last element the numbers stack SO I CAN add a different number | 1 | Next Sprint |
| **2** | Copy the last element of numbers stack | **11** | AS A user I WANT TO copy the last element the numbers stack SO I CAN avoid typing it in again | 2 | Next Sprint |
| **2** | Swap the last two elements of numbers stack | **12** | AS A user I WANT TO swap the last two elements of the numbers stack SO I CAN avoid having to delete them and insert them again | 2 | Next Sprint |
| **2** | Push a copy of the second to last element of numbers stack | **13** | AS A user I WANT TO push a copy of the second to last element of the numbers stack SO I CAN avoid having to type it in myself again | 2 | Next Sprint |
| **3** | Save the last element of the numbers stack in a variable | **14** | AS A user I WANT TO save the last element of the numbers stack in one of the 26 variables SO I CAN use it later | 21 | Next Sprint |
| **3** | Push the value of a variable on the numbers stack | **15** | AS A user I WANT TO push the value of one of the 26 variables on the stack SO I CAN perform an operation with it | 3 | Next Sprint |
| **3** | Adds the top element of the numbers stack to the value of the variable | **16** | AS A user I WANT TO add the top element of the numbers stack to the value of a variable SO I CAN increment the value of the variable | 5 | Next Sprint |
| **3** | Subtract the top element of the numbers stack from the value of the variable | **17** | AS A user I WANT TO subtract the top element of the numbers stack from the value of a variable SO I CAN decrement the value of the variable | 5 | Next Sprint |
| **4** | Define a new user-defined operation with a name | **18** | AS A user I WANT TO define an operation and give it a name SO I CAN refer to it | 21 |  |
| **4** | Define the sequence of operations of a user-defined operation | **19** | AS A user I WANT TO define the sequence of the operations of a user-defined operation SO I CAN specify what it does | 21 |  |
| **4** | Invoke a user-defined operation | **20** | AS A user I WANT TO invoke a user-defined operation SO I CAN perform its sequence of operations | 34 |  |
| **4** | Delete a user-defined operation | **21** | AS A user I WANT TO delete a user-defined operation SO I CAN delete it from the memory of the calculator | 3 |  |
| **4** | Modify the name of a user-defined operation | **22** | AS A user I WANT TO modify the name of a user-defined operation SO I CAN call it in a different way | 5 |  |
| **4** | Modify the sequence of operations of a user-defined operation | **23** | AS A user I WANT TO modify the sequence of operations of a user-defined operation SO I CAN edit its behaviour | 5 |  |
| **4** | Save to a file the existing user-defined operations | **24** | AS A user I WANT TO save all the operations I defined SO I CAN invoke them even after I close the calculator and reopen it | 13 |  |
| **4** | Load from a file the user-de  fined operations | **25** | AS A user I WANT TO load all the operations I saved in the file SO I CAN invoke them | 21 |  |
| **5** | Save the current values of the 26 variables on the variable stack | **26** | AS A user I WANT TO save the current values of the variables SO I CAN restore them later | 21 |  |
| **5** | Restore for all variables the last values saved on the variable stack | **27** | AS A user I WANT TO restore the last saved values of the variables SO I CAN re-use them | 5 |  |
| **6** | Calculate the modulus of a complex number | **28** | AS A user I WANT TO calculate the modulus of a complex number SO I CAN perform the operation | 1 |  |

**Sprint Backlog**

**First Sprint**

**Estimated Velocity:** 51

**First Sprint SP:** 47

**Goal of the sprint:** Create the first delivery of the project which consists of:

* the graphic interface with a text field where the user can insert numbers and operations and a text area that shows the last 12 elements of the stack
* the possibility for the user to insert a complex number on the stack
* checking if the inserted input is valid

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User Story # | Product Backlog Item | Task | Story Points | Assigned to member |
| 1 | Insert Complex Number | US:1.1 Create Class ComplexNumber | 1 | Corvino Luigi |
| 1 | Insert Complex Number | US:1.2 Make utility methods for class ComplexNumber | 1 | Corvino Luigi |
| 1 | Insert Complex Number | US 1.3: Creation of a JFrame | 3 | Esposito Alfonso |
| 1 | Insert Complex Number | US 1.4: Create an input TextField in JFrame and check inputs | 8 | Esposito Alfonso |
| 1 | Insert Complex Number | US 1.5 Make the button of insert complex number | 3 | Esposito Alfonso |
| 1 | Insert Complex Number | US 1.6 Make insert complex number method for ProgrammableCalculatorController Class | 5 | Alexander De Santis |
| 2 | Addition two numbers | US 2.1: Create class ComplexNumberOperations | 1 | Corvino Luigi |
| 2 | Addition two numbers | US 2.2: Make addition method for ComplexNumberOperations Class | 1 | Corvino Luigi |
| 2 | Adding two numbers | US 2.3 Make add complex number method for ProgrammableCalculatorController Class | 3 | Corvino Luigi |
| 8 | Visualize the last 12 elements of the stack | US 8.1: Create an output TextArea in JFrame | 3 | Alfonso Esposito |
| 8 | Visualize the last 12 elements of the stack | US 8.2: Create Class NumbersStack | 1 | Luigi Corvino |
| 8 | Visualize the last 12 elements of the stack | US 8.3 Create ProgrammableCalculatorController class | 2 | Alexander De Santis |
| 8 | Visualize the last 12 elements of the stack | US 8.6 Make update method for the JFrame | 5 | Luigi Corvino |
| 8 | Visualize the last 12 elements of the stack | US 8.7 Make utility methods for the ProgrammableCalculatorController | 5 | Alexander De Santis |
| 8 | Visualize the last 12 elements of the stack | US 8.4 Create utility elaborateInput for the ProgrammableCalculatorController | 5 | Alexander De Santis |

**Acceptance criteria**

* **US1:** Given I have accessed the user interface when I insert a number into the text field then I can see the number in the text area which represents the calculator’s stack.
* **US2**:Given I have accessed the user interface and inserted two numbers in the stack when I insert the sum operator then I can see the result in the text area.
* **US3** : Given I have accessed the user interface and inserted two numbers in the stack when I insert the subtraction operator then I can see the result in the text area.
* **US4:** Given I have accessed the user interface and inserted two numbers in the stack when I insert the multiply operator then I can see the result in the text area.
* **US5**: Given I have accessed the user interface and inserted two numbers in the stack when I insert the division operator then I can see the result in the text area.
* **US6**: Given I have accessed the user interface and inserted a number in the stack when I insert the square root operator then I can see the result in the text area.
* **US7** Given I have accessed the user interface and inserted a number in the stack when I insert the invert sign operator then I can see the result in the text area.
* **US8**: Given I have accessed the user interface when I am using the calculator then I can see the last 12 elements of the calculator’s stack.
* **US9**: Given I have accessed the user interface when I insert the clear command then I can empty the calculator’s stack and look for the modifications in the text area.
* **US10**: Given I have accessed the user interface when I insert the drop command then I can
* remove the last element from the calculator’s stack and look at the modifications in the text area.
* **US11**: Given I have accessed the user interface when I insert the dup command then I can create a copy of the last element in the top of the calculator’s stack and look at the modifications in the text area.
* **US12**: Given I have accessed the user interface when I insert the swap command then I can swap the last two elements in the calculator’s stack and look at the modifications in the text area.
* **US13**: Given I have accessed the user interface when I insert the over command then I can create a copy of the second last element in the top of the calculator’s stack and look at the modifications in the text area.
* **US14**: Given I have accessed the user interface when I insert the >x command then I assign the value of the last number on the stack to the variable x
* **US15**: Given I have accessed the user interface when I insert the <x command then I push the value of the variable x on the stack
* **US16:** Given I have accessed the user interface when I insert the +x command then I add the value of the last number on the stack to the value of x
* **US17:** Given I have accessed the user interface when I insert the -x command then I subtract the value of the last number on the stack from the value of x
* **US18:** Given I have accessed the user interface when I insert the “new” command then I can define a new operation and give it a name
* **US19:** Given I have defined a new operation when I have given it a name then I can insert the sequence of operations that have to be executed when it is invoked
* **US20:** Given I have defined an operation when I write its name in the text field then its sequence of operations are executed
* **US21:** Given I have defined an operation and I have accessed the user interface when I insert “del userDefinedOperationName” then I can delete the operation from the calculator
* **US22:** Given I have defined an operation and I have accessed the user interface when I insert “edit userDefinedOperationName” then I can edit its name
* **US23:** Given I have defined an operation and have accessed the user interface when I insert “edit userDefinedOperationName” then I can edit its sequence of operations
* **US24:** Given I have defined a set of operations and have accessed the user interface when I insert the command “save operations” then I can save the operations on a file
* **US25:** Given I have accessed the user interface and executed the command “save operations” in one of the previous sessions then all my operations are loaded
* **US26**: Given I have accessed the user interface when I insert the command “save” then I can save all the current variables’ values
* **US27**: Given I have accessed the user interface when I insert the command “restore” then I can restore the values of the current variables to the ones they had when I inserted the command “save” the last time
* **US28**: Given I have accessed the user interface when I insert the command “mod” then I can calculate the modulus of the last complex number on the stack

**Definition of Done**

* The user story implementation meets all the acceptance criteria
* The unit tests were written, executed and passed
* The code of the user story has been reviewed in the code-review session

**Sprint Backlog**

**Second Sprint**

**Estimated** **Velocity:** 62

**Sprint Story points:** 54

**Goal of the second Sprint:** Adding the following features to the first sprint release software:

* square root and invert sign operations
* stack operations
* usage of variables
* variables operations

|  |  |  |  |
| --- | --- | --- | --- |
| User Story # | Product Backlog Item | Story Points | Assigned to member |
| 6 | Square root of a number | 5 |  |
| 7 | Invert sign of a number | 5 |  |
| 9 | Clear the numbers stack | 2 |  |
| 10 | Remove the last element of numbers stack | 2 |  |
| 11 | Copy the last element of numbers stack | 2 |  |
| 12 | Swap the last two elements of numbers stack | 2 |  |
| 13 | Push a copy of the second to last element of numbers stack | 2 |  |
| 14 | Save the last element of the numbers stack in a variable | 21 |  |
| 15 | Push the value of a variable on the numbers stack | 3 |  |
| 16 | Adds the top element of the numbers stack to the value of the variable | 5 |  |
| 17 | Subtract the top element of the numbers stack from the value of the variable | 5 |  |