A Step-By-Step Guide To Jubula – The Open Source Automated Functional Testing Tool

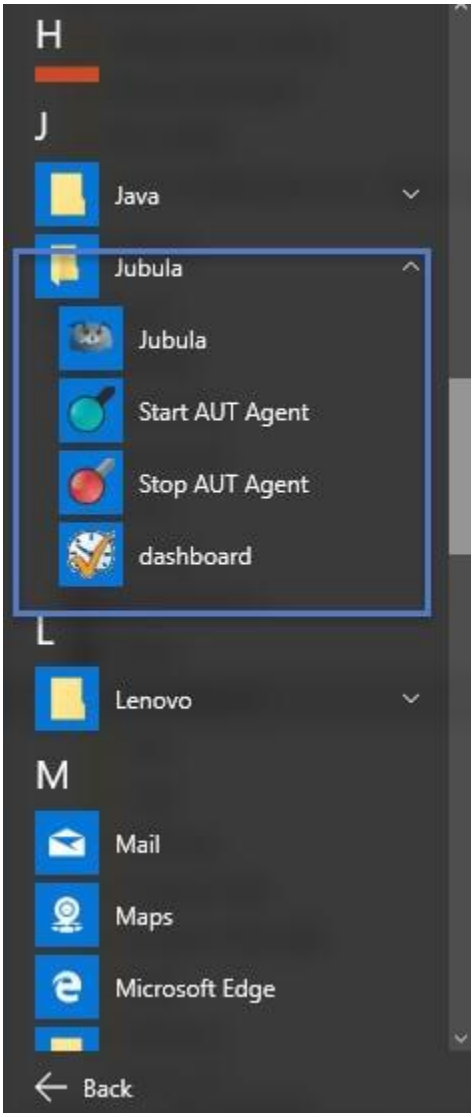
September 1, 2020

Today, we are going to learn a popular **open source functional testing tool – Jubula.**
It is an Eclipse Project for automated functional GUI testing for many applications. It's is helpful for writing automated tests from the user perspective with little or **no coding skills**, saving time and improving readability and test maintenance.
A wide range of open source automation tools are available in the market with the good amount of online help.
When it comes to ***Jubula*****,** the online help provides a lot of information on its built-in framework. This information is of great use to those non-technical testers who are not involved in coding and want to create automation scripts through Jubula's GUI.
But the technical automation testers wanting to create a customized framework using Jubula find it difficult to reach out the help.
This detailed tutorial is created with an aim to share the knowledge I have gained in Jubula to help you create a customized, robust and flexible automation testing framework.

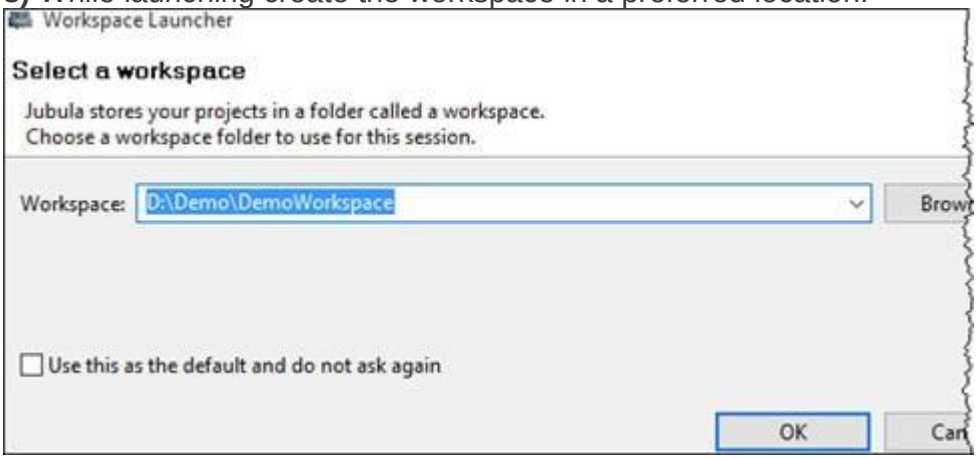
Let us first look into it's**s built-in framework** (This section of the tutorial will help you to understand the basics) and then proceed with **Building framework in Jubula using Java code.**

The Basics – built-in framework:

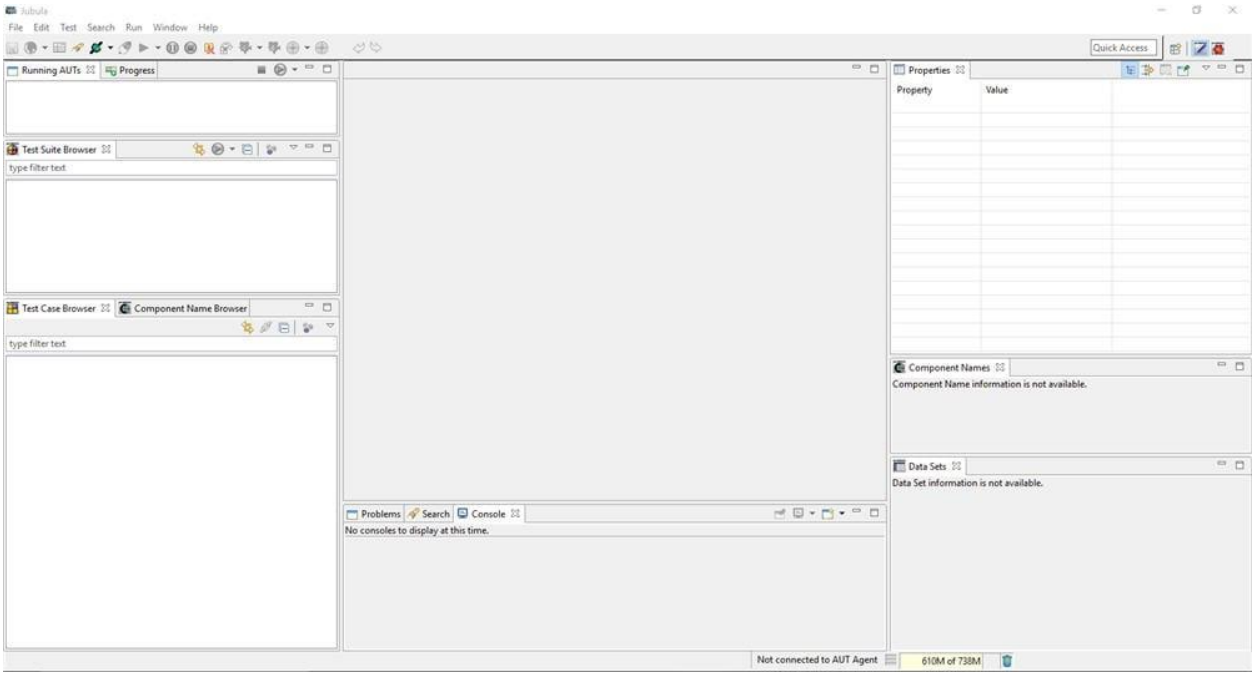
Installation and Launch:
(Note: click on any image for enlarged view)
1) Go to the download page here.
Note – You can check this download page for various options based on the features you are interested in.
2) Register and Login.
3) Click on **Download Installers.**
4) Click on the **download page** under **Jubula downloads** section.
5) Download the appropriate installer (as per the OS).
6) Install it using the downloaded **exe** file and save the folder in a preferred location (I have saved it in *C:\Program Files*).
7) Once the installation is completed, you should be able to launch the tool from 'All programs'.



8) While launching create the workspace in a preferred location.



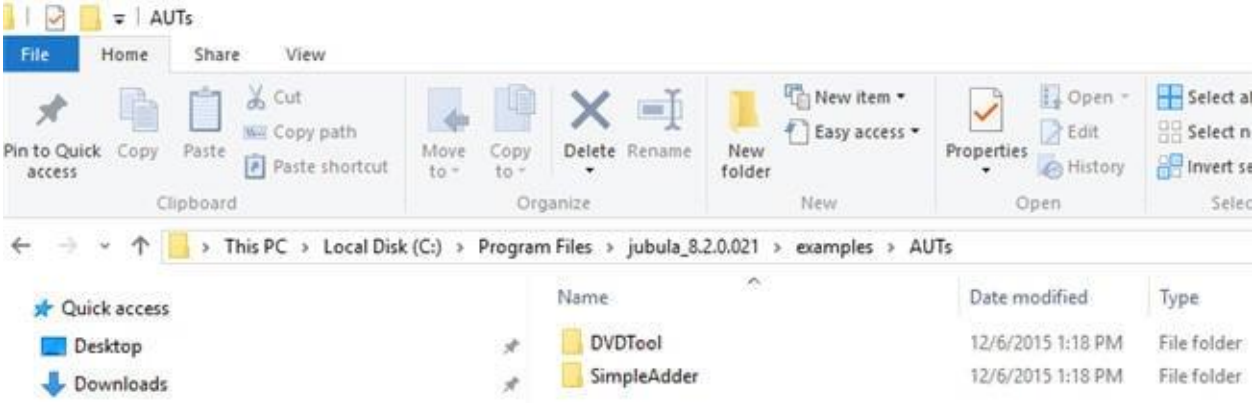
9) Home screen will look like below:



Sample AUTs:

One impressive thing about this tool is that it comes with sample AUTs (Applications Under Test). The AUTs will be present in <Installation Folder>\examples\AUTs.

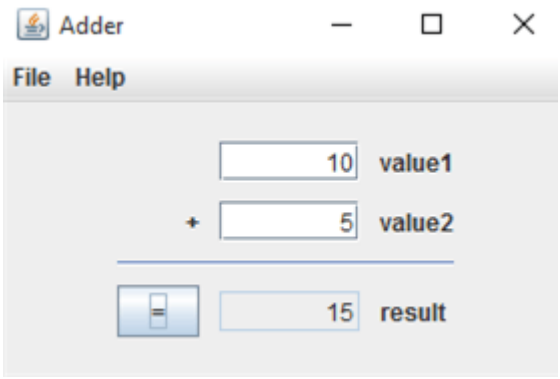
In this case, it is found in *C:\Program Files\jubula_8.2.0.021\examples\AUTs*



Demo Project

Now with Jubula installed and sample AUTs available, let us try to automate a ***simple addition functionality*** using a 'swing' application called 'SimpleAdder'.
This application can be invoked using the file:

'C: \ProgramFiles\jubula_8.8.0\examples\AUTs\SimpleAdder\swing\SimpleAdder.cmd' and looks like below:
The task is to enter 'value1', enter 'value2', click '=' button and verify the 'result'.



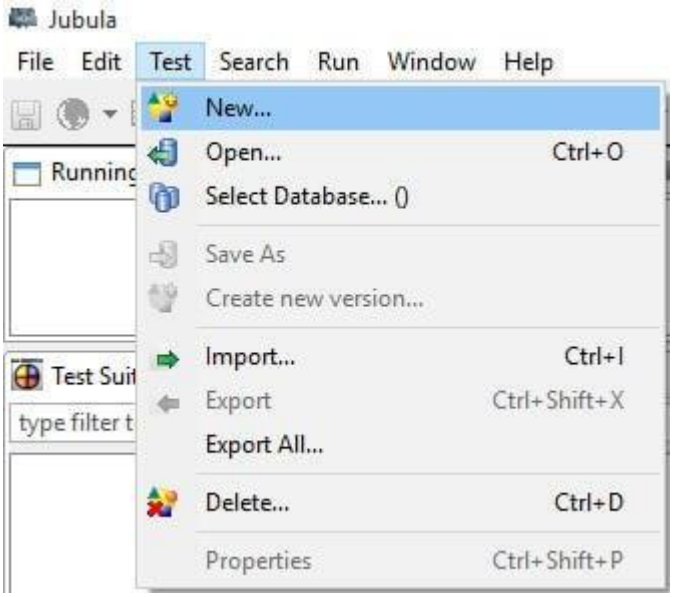
How to Test:

Below are the steps to be followed to automate the task:
Step 1 – Create project.
Step 2 – Create AUT.
Step 3 – Create test case, include test steps & map data.
Step 4 – Create test suite.
Step 5 – Assign AUT to the test suite.
Step 6 – Map the test case to the test suite.
Step 7 – Map logical test objects with technical object identifiers.
Step 8 – Run the test suite.
Let us see how to perform each step in detail:

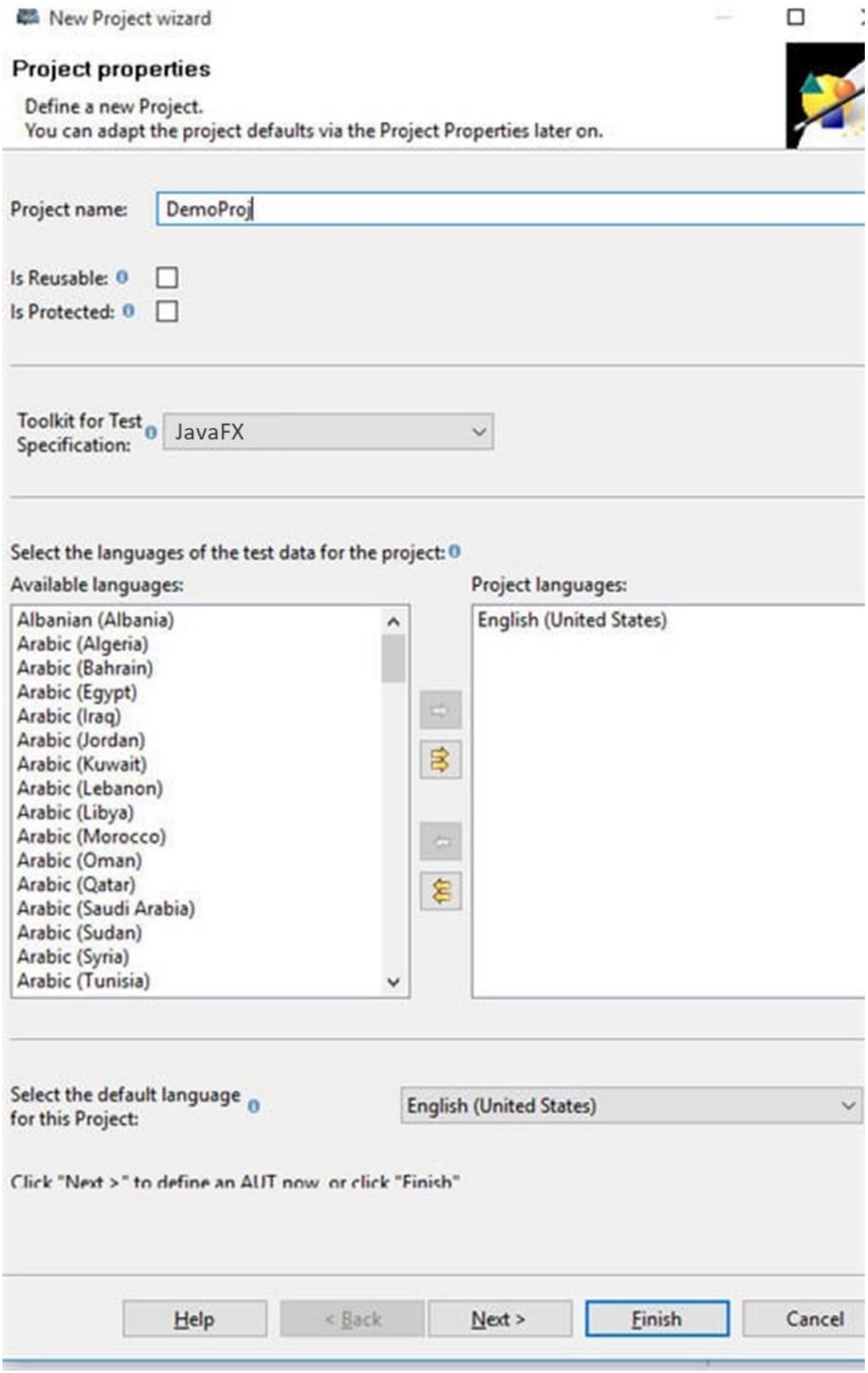
Step #1 – Create project

A Project in Jubula can be considered as a logical workspace where all required components are gathered to complete a testing task.

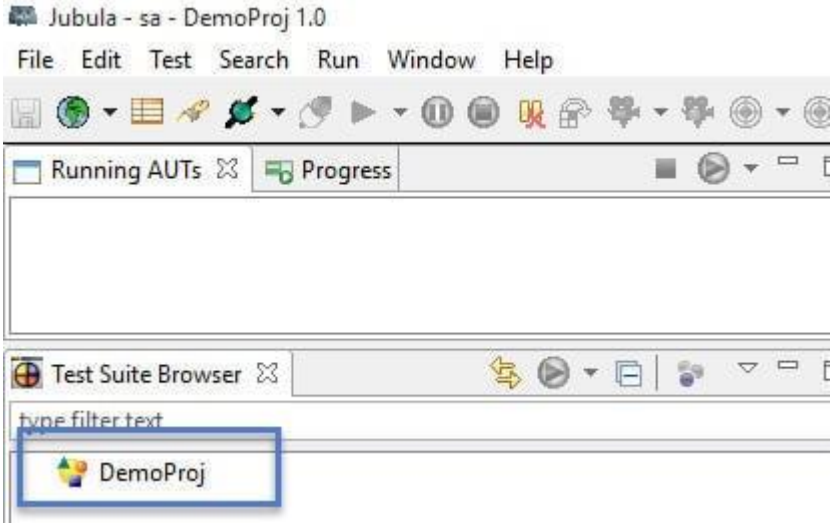
Creation of a project goes as below:
1) Go to Test > New.



2) Enter the name of the project, E.g. 'DemoProject' and Click 'Finish' (On clicking 'Next' you should be able to create AUT. But let us 'Finish' here and look at creating AUT in Step #2).


New Project wizard
Project properties
Define a new Project.
You can adapt the project defaults via the Project Properties later on.
Project name:
DemoProj
Is Reusable:
Is Protected:
Toolkit for Test Specification:
JavaFX
Select the languages of the test data for the project:
Available languages:
Albanian (Albania)
Arabic (Algeria)
Arabic (Bahrain)
Arabic (Egypt)
Arabic (Iraq)
Arabic (Jordan)
Arabic (Kuwait)
Arabic (Lebanon)
Arabic (Libya)
Arabic (Morocco)
Arabic (Oman)
Arabic (Qatar)
Arabic (Saudi Arabia)
Arabic (Sudan)
Arabic (Syria)
Arabic (Tunisia)
Project languages:
English (United States)
Select the default language for this Project:
English (United States)
Click "Next >" to define an AUT now, or click "Finish"
Help
< Back
Next >
Finish
Cancel

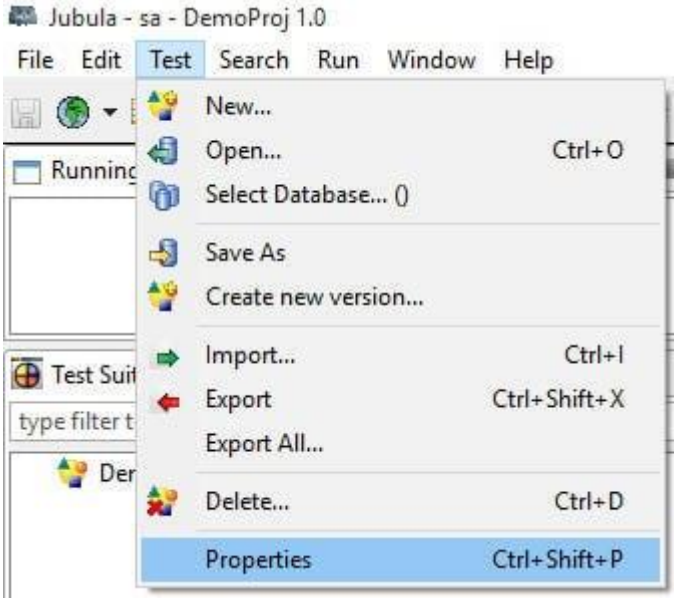
3) The Test Suite browser shows the created project.



Step #2 – Create AUT

An instance of the application under test (SimpleAdder) has to be created in Jubula for object mapping and to run the test suite.

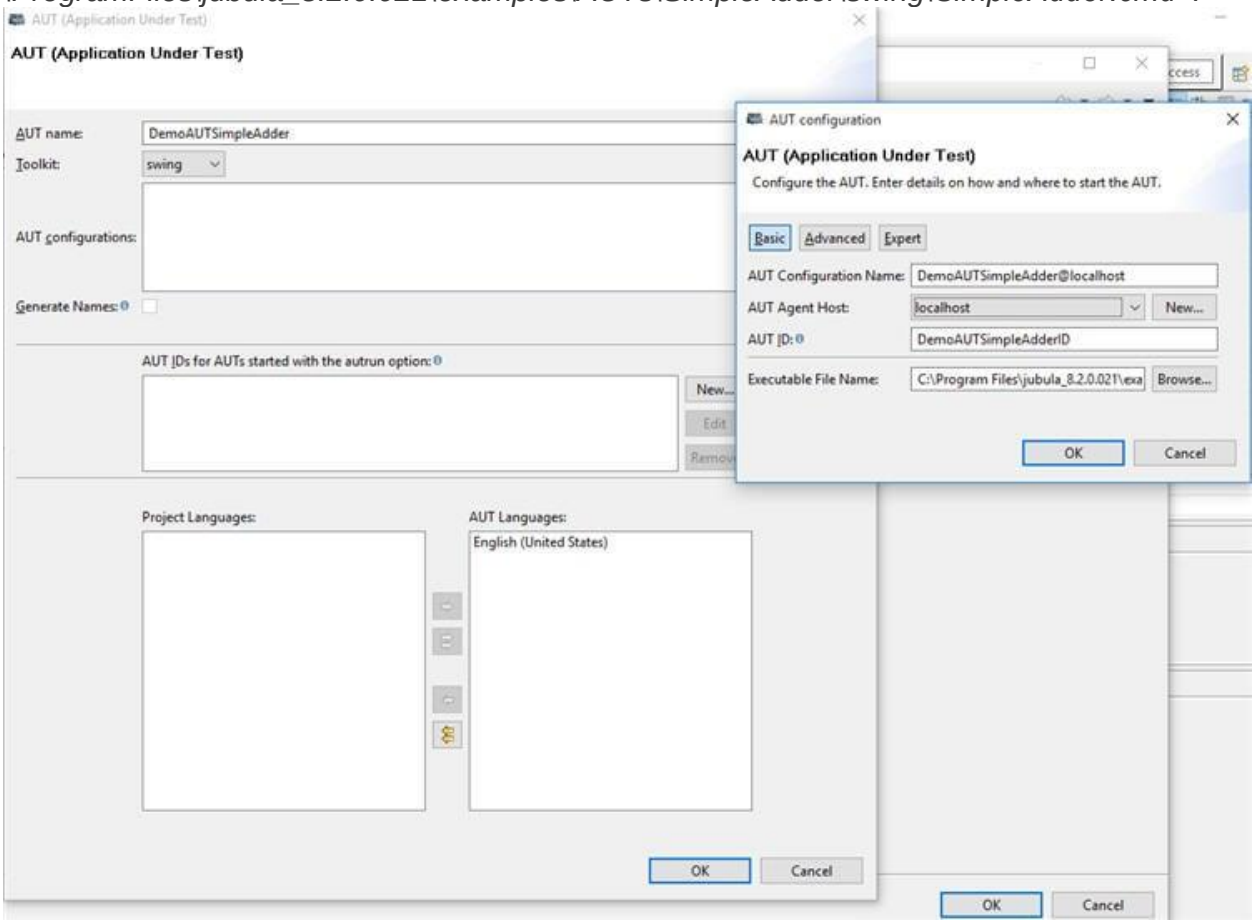
1) Go to Test > Properties.



2) Choose 'AUTs'.

3) Enter the AUT name (this can be any user defined value. E.g. DemoAUTSimpleAdder).
4) Choose the technology the AUT is developed on. (In this case, it is 'swing').
5) Click 'Add' under AUT configuration.

6) Enter AUT ID (this can again be any user defined value. E.g. DemoAUTSimpleAdderID).
7) Enter the executable file name i.e. the file invoking which the AUT will be opened. As mentioned earlier I have used *'C:\ProgramFiles\jubula_8.2.0.021\examples\AUTs\SimpleAdder\swing\SimpleAdder.cmd'* .



Please note that in order to invoke the AUT through Jubula, it must have been connected to 'AUT agents', There are two AUT agents Jubula can connect with:

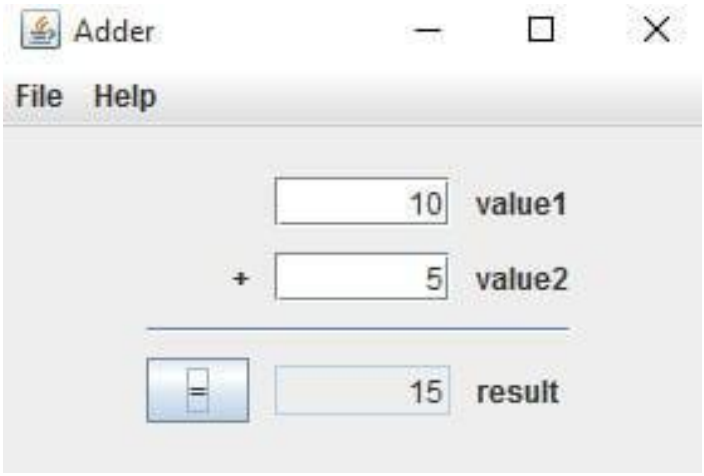
- Embedded AUT agent installed at 'localhost:60001'
- External AUT agent installed at 'localhost:60000'

Once Jubula is connected to either of the AUT agents, you should be able to invoke the application through it. The below screen shot shows how to connect to AUT agents. Here I am connecting to the embedded AUT agent.

The below screen shot shows how to connect to AUT agents. Here I am connecting to the embedded AUT agent.

Once AUT agent is connected with Jubula, the AUT (DemoAUTSimpleAdder) can be invoked as below:

The AUT will be invoked as below. The application can be kept running in the background. But, at this stage, I prefer to close the application to be comfortable in performing the rest of the steps.

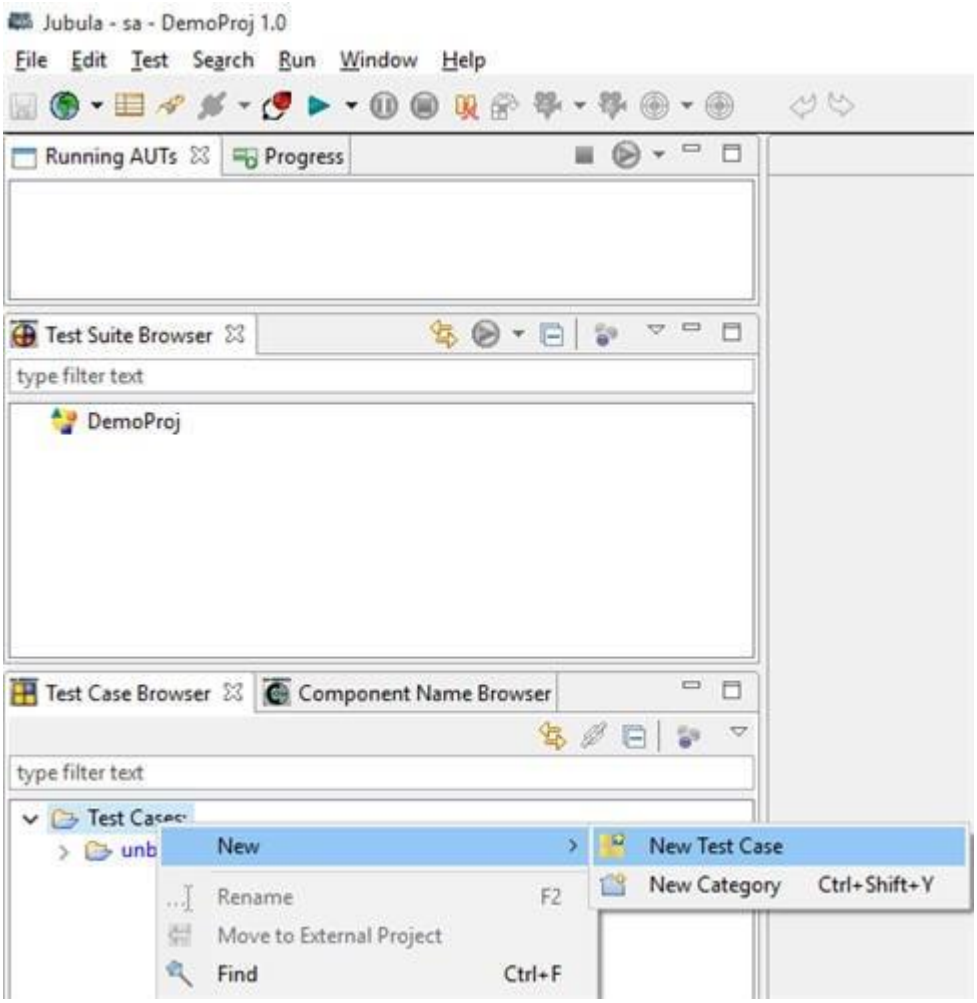


Step #3 – Create test case, include test steps and map data

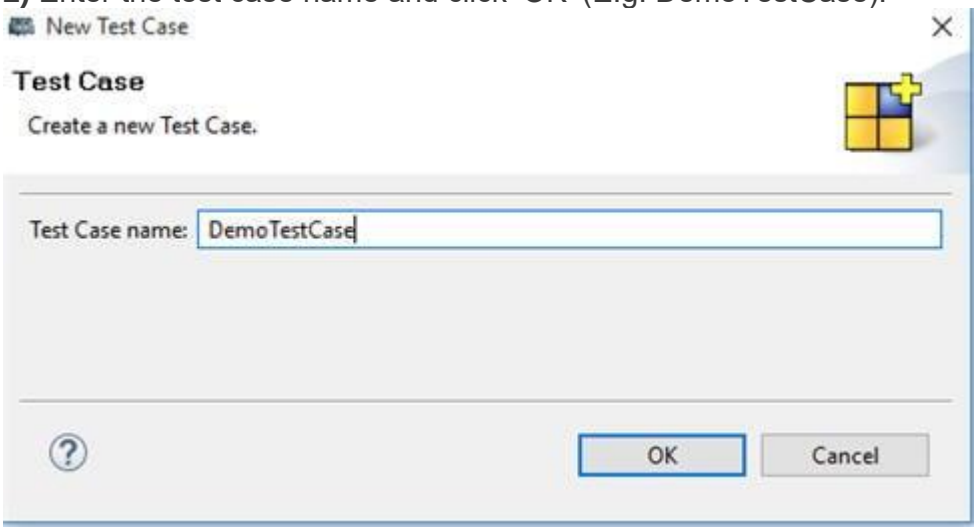
This is the crucial step where actual development of automation scripts happens (without coding).

There is a Test case browser at the bottom left part of the tool where user test cases can be developed.

1) Right click and move to New to create a new test case.

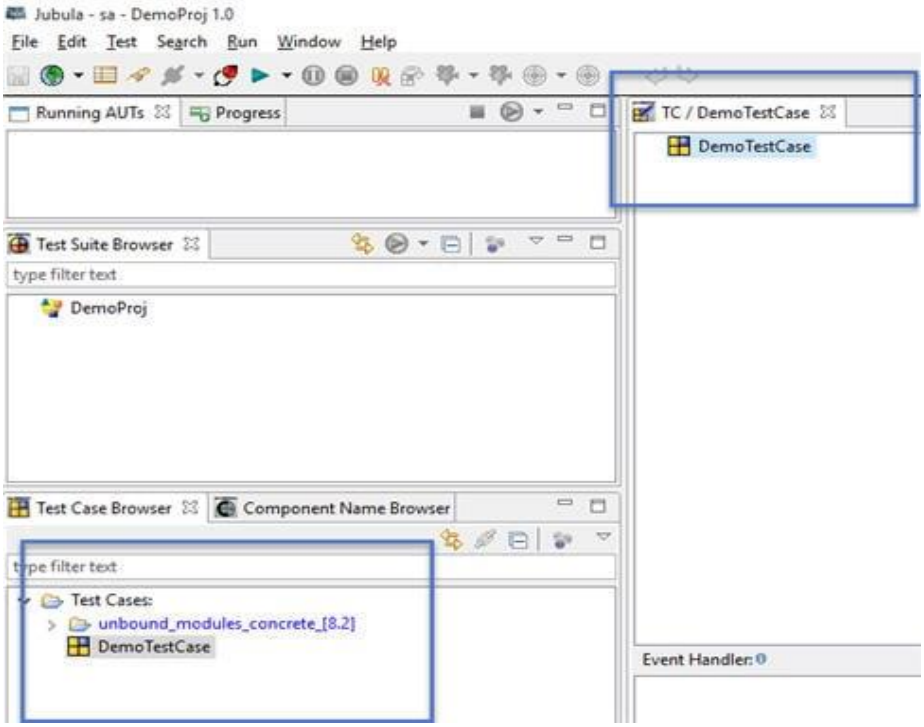


2) Enter the test case name and click 'OK' (E.g. DemoTestCase).



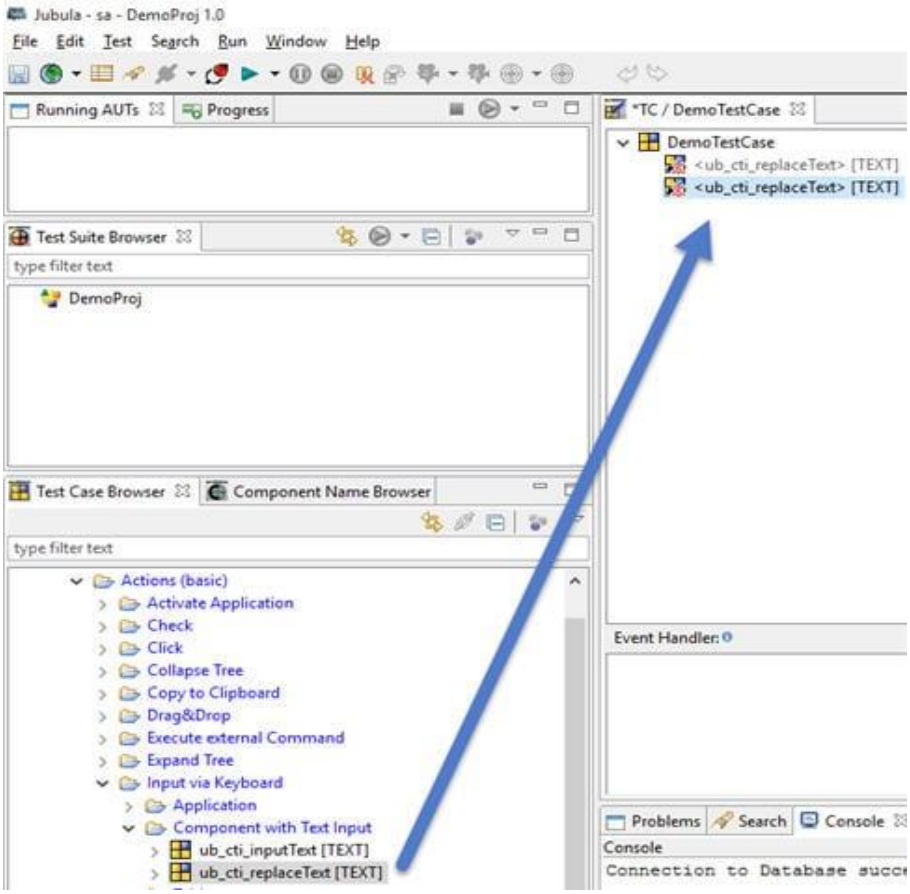
3) The test case browser now should have the user created test case along with Jubula's built in test cases.

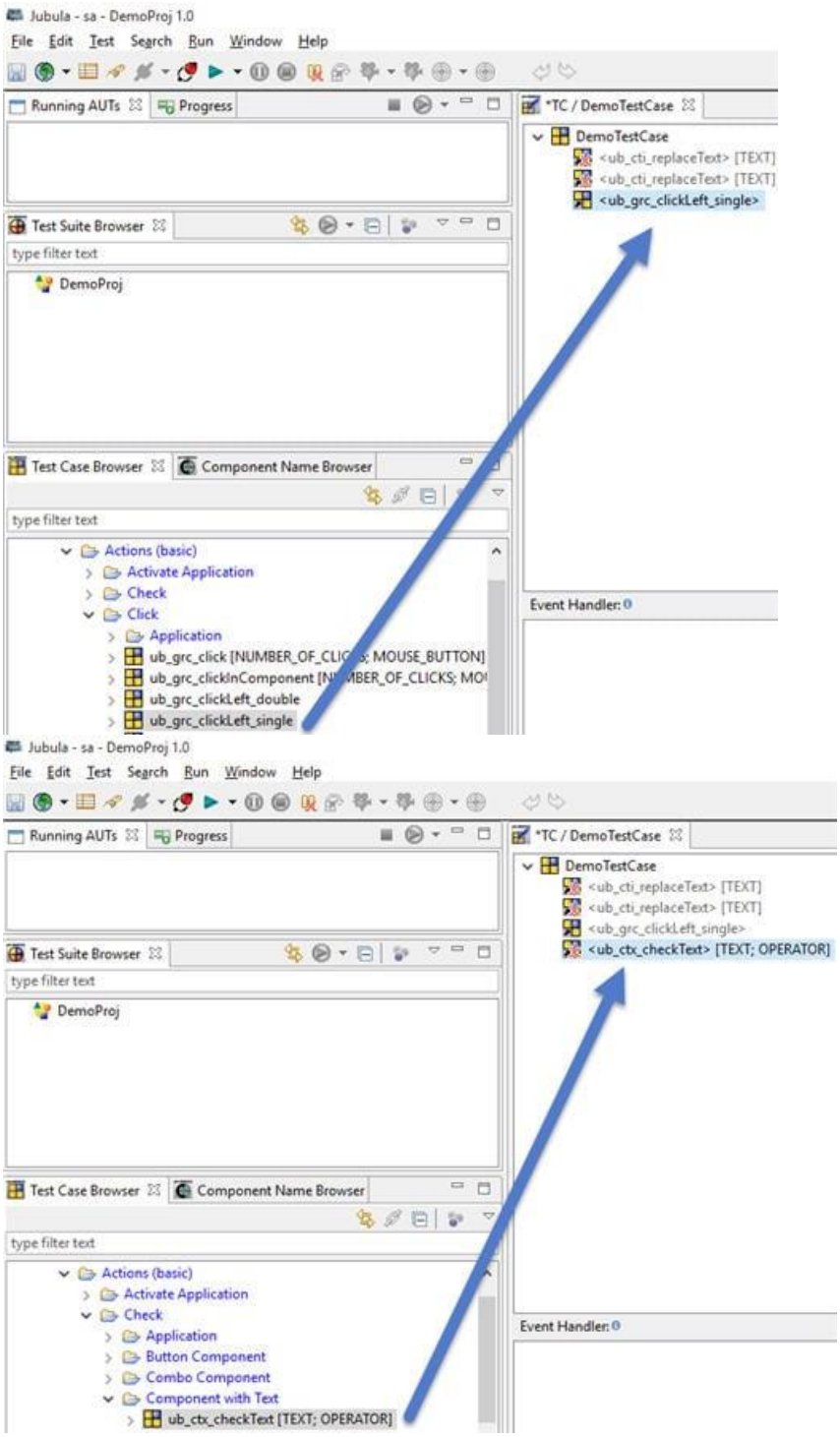
4) Double click on the created test case. The empty test case will be opened in the middle panel.



5) Drag and drop the appropriate test actions from 'Jubula's base actions' into the test case. As shown in the below screenshots:

- To enter value1 and value2 use 'Component with Text input – replace text'.
- To click equals button use 'Click left single'.
- To check the value use 'Component with Text input – check text'.

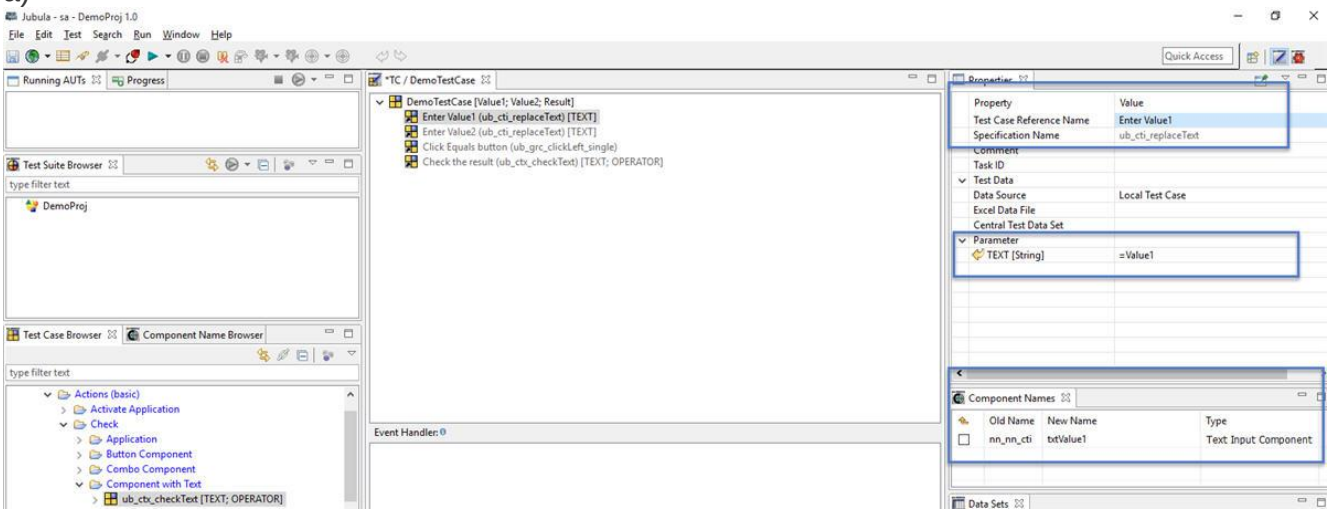




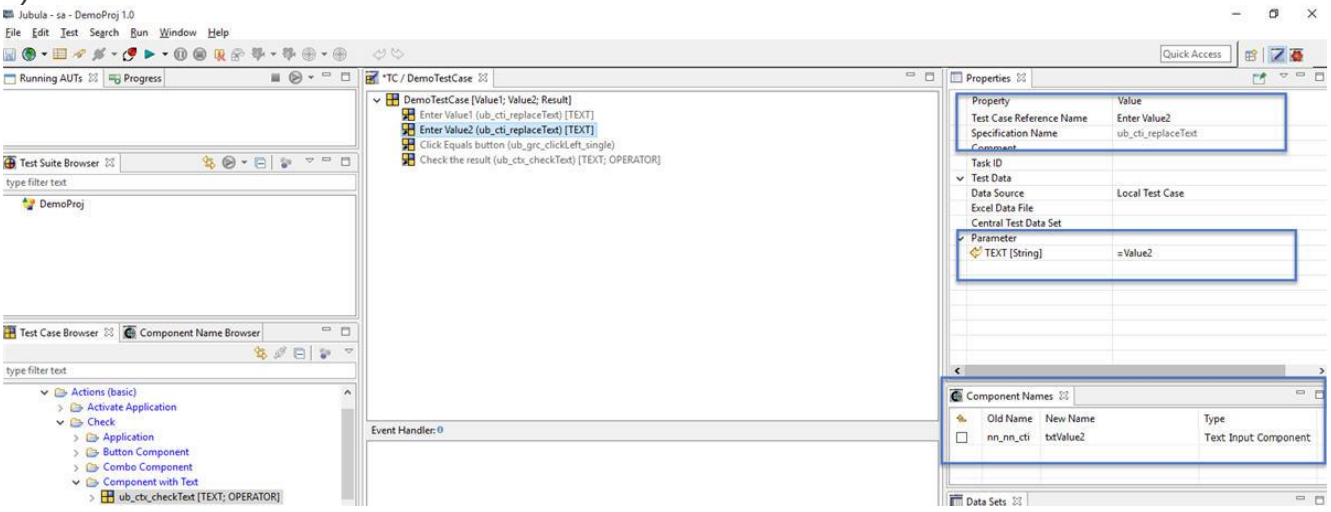
6) Now for each test step included enter the below as applicable (E.g. Clicking a button does not need any data):

- Test case reference name (Test step description).
- Component name (Logical name to which technical identifier will be mapped).
- Data – Data can be entered as direct value e.g. 10, 20 or parameterized using variables e.g. =Value1, =Value2 or fed through an excel sheet. In this case, I am parameterizing the data).

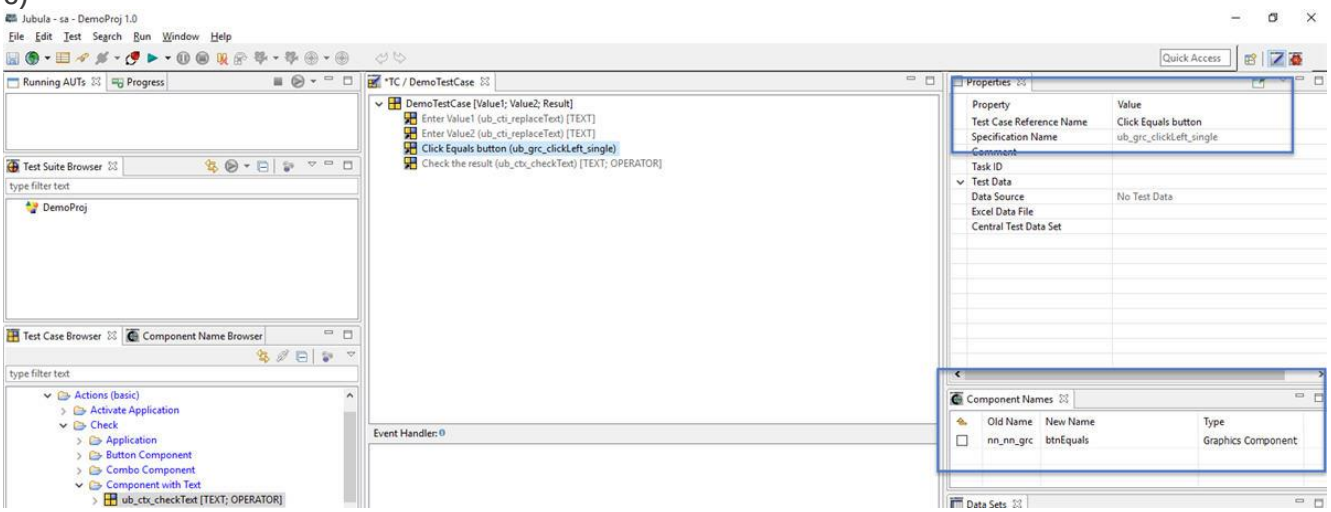
a)


Jubula - sa - DemoProj 1.0
File Edit Test Search Run Window Help
Quick Access
Running AUTs
Progress
*TC / DemoTestCase
DemoTestCase [Value1; Value2; Result]
Enter Value1 (ub_cti_replaceText) [TEXT]
Enter Value2 (ub_cti_replaceText) [TEXT]
Click Equals button (ub_grc_clickLeft_single)
Check the result (ub_ctx_checkText) [TEXT; OPERATOR]
Properties
Property
Value
Test Case Reference Name
Enter Value1
Specification Name
ub_cti_replaceText
Comment
Task ID
Test Data
Data Source
Local Test Case
Excel Data File
Central Test Data Set
Parameter
TEXT [String]
=Value1
Test Suite Browser
type filter text
DemoProj
Test Case Browser
Component Name Browser
Actions (basic)
Activate Application
Check
Application
Button Component
Combo Component
Component with Text
ub_ctx_checkText [TEXT; OPERATOR]
Event Handler:
Component Names
Old Name
New Name
Type
nn_nn_cti
txtValue1
Text Input Component
Data Sets

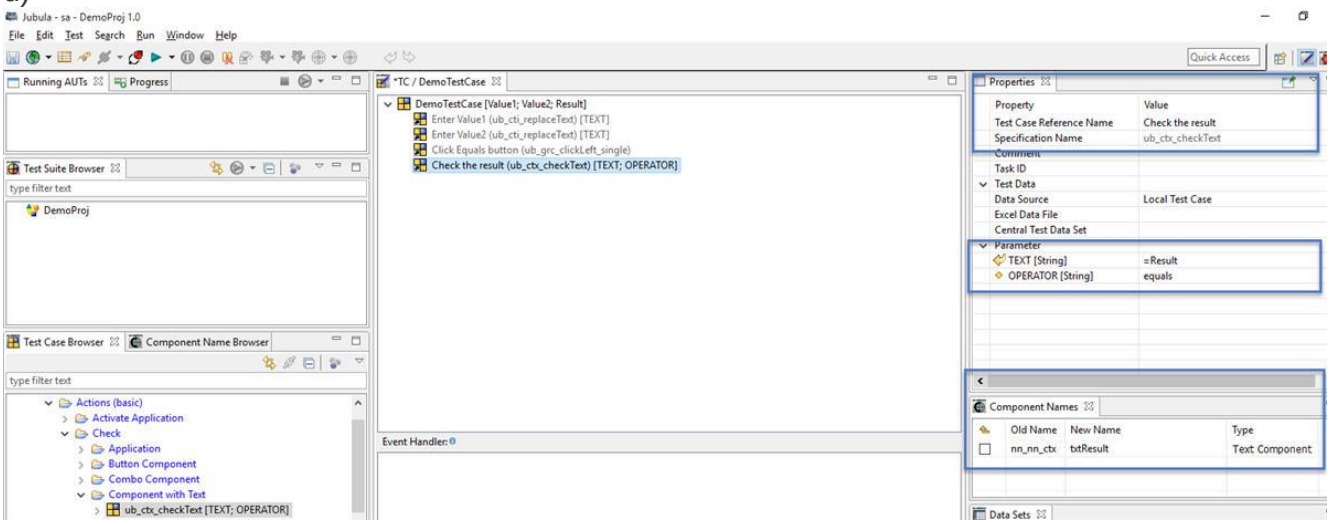
b)


Jubula - sa - DemoProj 1.0
File Edit Test Search Run Window Help
Quick Access
Running AUTs
Progress
*TC / DemoTestCase
DemoTestCase [Value1; Value2; Result]
Enter Value1 (ub_cti_replaceText) [TEXT]
Enter Value2 (ub_cti_replaceText) [TEXT]
Click Equals button (ub_grc_clickLeft_single)
Check the result (ub_ctx_checkText) [TEXT; OPERATOR]
Properties
Property
Value
Test Case Reference Name
Enter Value2
Specification Name
ub_cti_replaceText
Comment
Task ID
Test Data
Data Source
Local Test Case
Excel Data File
Central Test Data Set
Parameter
TEXT [String]
=Value2
Test Suite Browser
type filter text
DemoProj
Test Case Browser
Component Name Browser
Actions (basic)
Activate Application
Check
Application
Button Component
Combo Component
Component with Text
ub_ctx_checkText [TEXT; OPERATOR]
Event Handler:
Component Names
Old Name
New Name
Type
nn_nn_cti
txtValue2
Text Input Component
Data Sets

c)

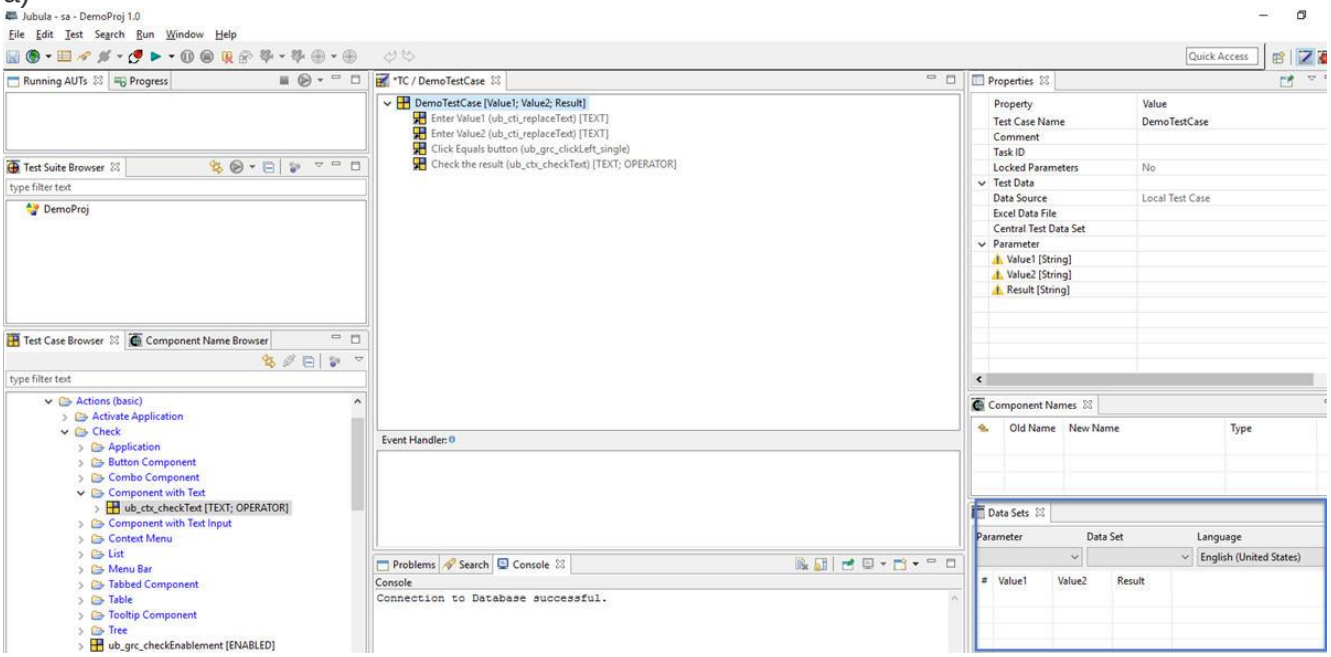

Jubula - sa - DemoProj 1.0
File Edit Test Search Run Window Help
Quick Access
Running AUTs
Progress
*TC / DemoTestCase
DemoTestCase [Value1; Value2; Result]
Enter Value1 (ub_cti_replaceText) [TEXT]
Enter Value2 (ub_cti_replaceText) [TEXT]
Click Equals button (ub_grc_clickLeft_single)
Check the result (ub_ctx_checkText) [TEXT; OPERATOR]
Properties
Property
Value
Test Case Reference Name
Click Equals button
Specification Name
ub_grc_clickLeft_single
Comment
Task ID
Test Data
Data Source
No Test Data
Excel Data File
Central Test Data Set
Test Suite Browser
type filter text
DemoProj
Test Case Browser
Component Name Browser
Actions (basic)
Activate Application
Check
Application
Button Component
Combo Component
Component with Text
ub_ctx_checkText [TEXT; OPERATOR]
Event Handler:
Component Names
Old Name
New Name
Type
nn_nn_grc
btnEquals
Graphics Component
Data Sets

d)

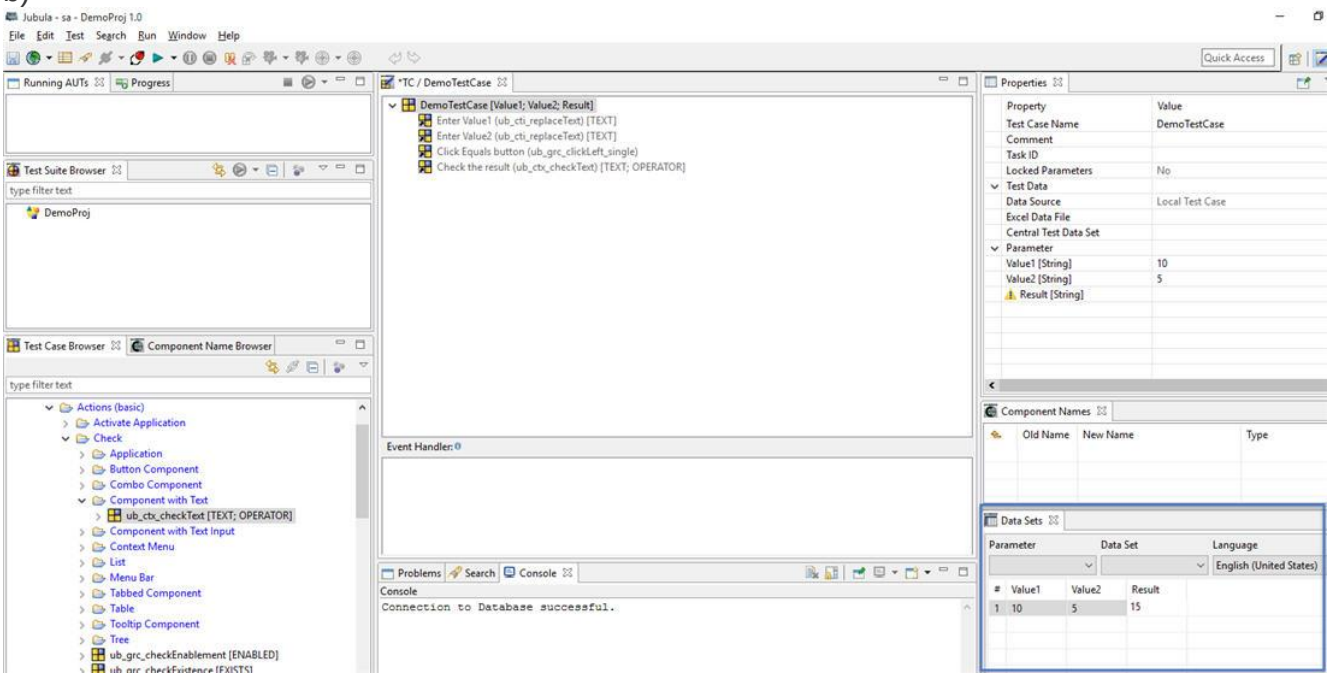


7) As mentioned in the previous step, when data values are parameterized, highlight the test case as below. You should be able to see a grid where the values for variables can be defined. Use ‘Add’ button to insert a row of data. You can have n number of rows of data for n number of execution iterations.

a)



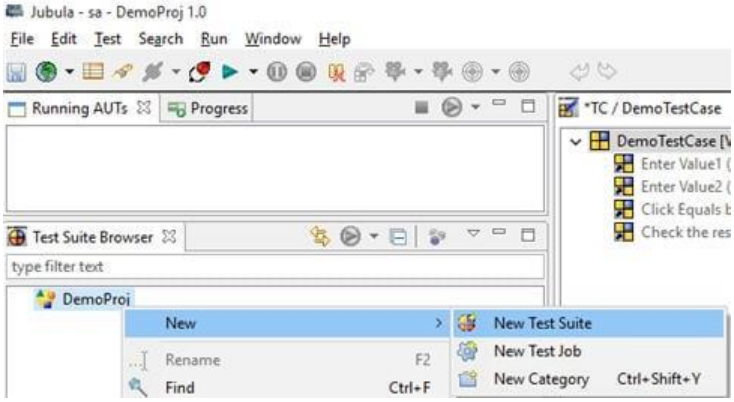
b)



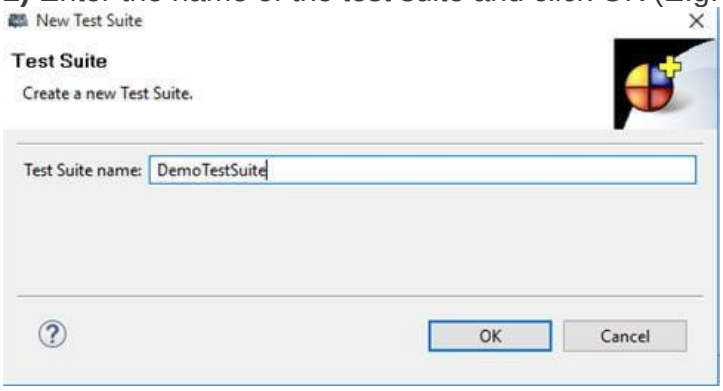
Step #4 – Create test suite

A Jubula's test suite is a runnable component under the project where user defined test cases are sequenced for execution.

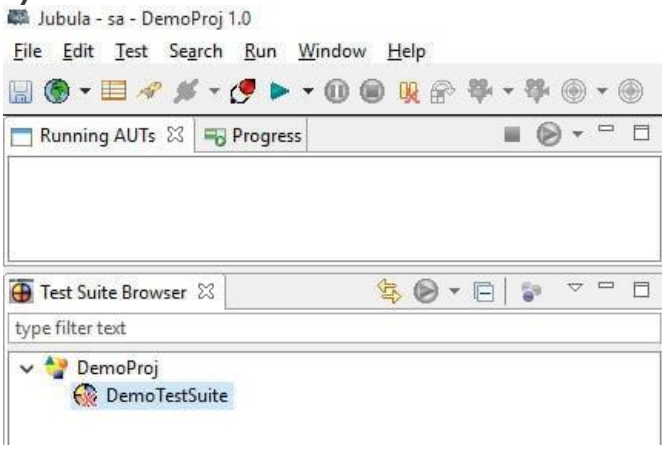
1) Right click on the project and move to New to create a new test suite.



2) Enter the name of the test suite and click OK (E.g. DemoTestSuite).



3) You should be able to see the created test suite under the project.



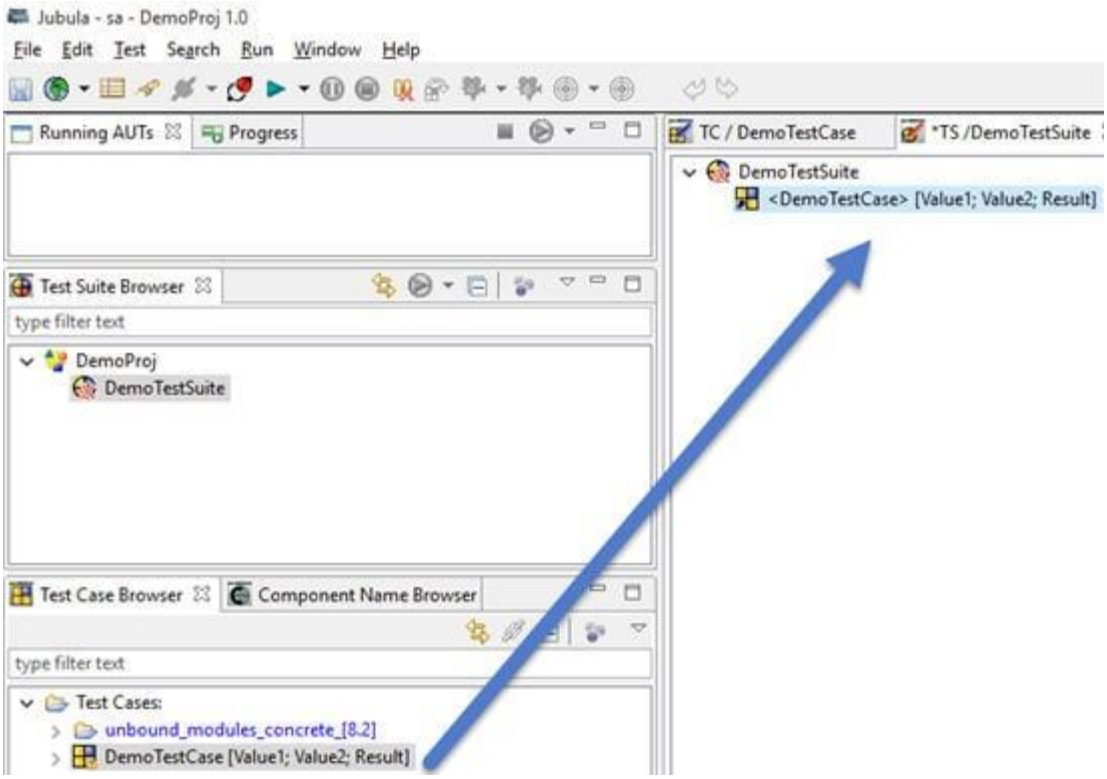
Step #5 – Assign AUT to the Test Suite

When there is just a single AUT defined in Jubula, the AUT will be automatically selected for the test suite. But when there are multiple AUTs, it is very important to make sure that the test suite runs on the correct AUT.

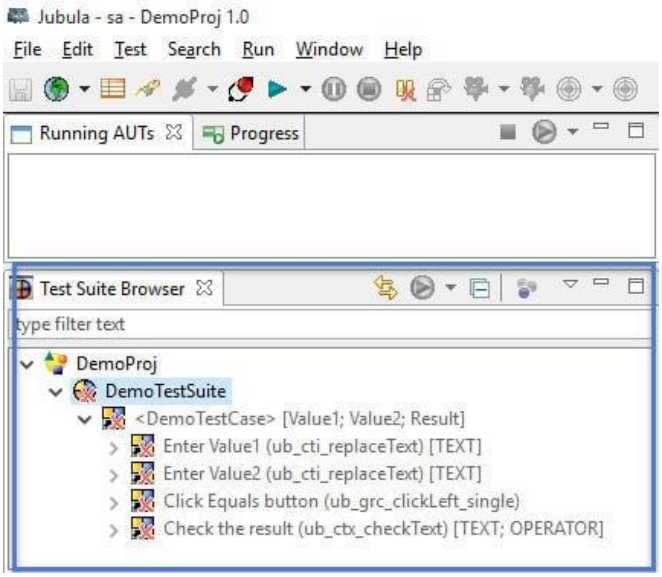
1) Double click on the test suite and highlight the same in the middle panel.
2) Select the AUT from the drop down.

Step #6 – Map the test case to the test Suite

Drag and drop the test case to the test suite. Multiple test cases can be sequenced under the test suite likewise.

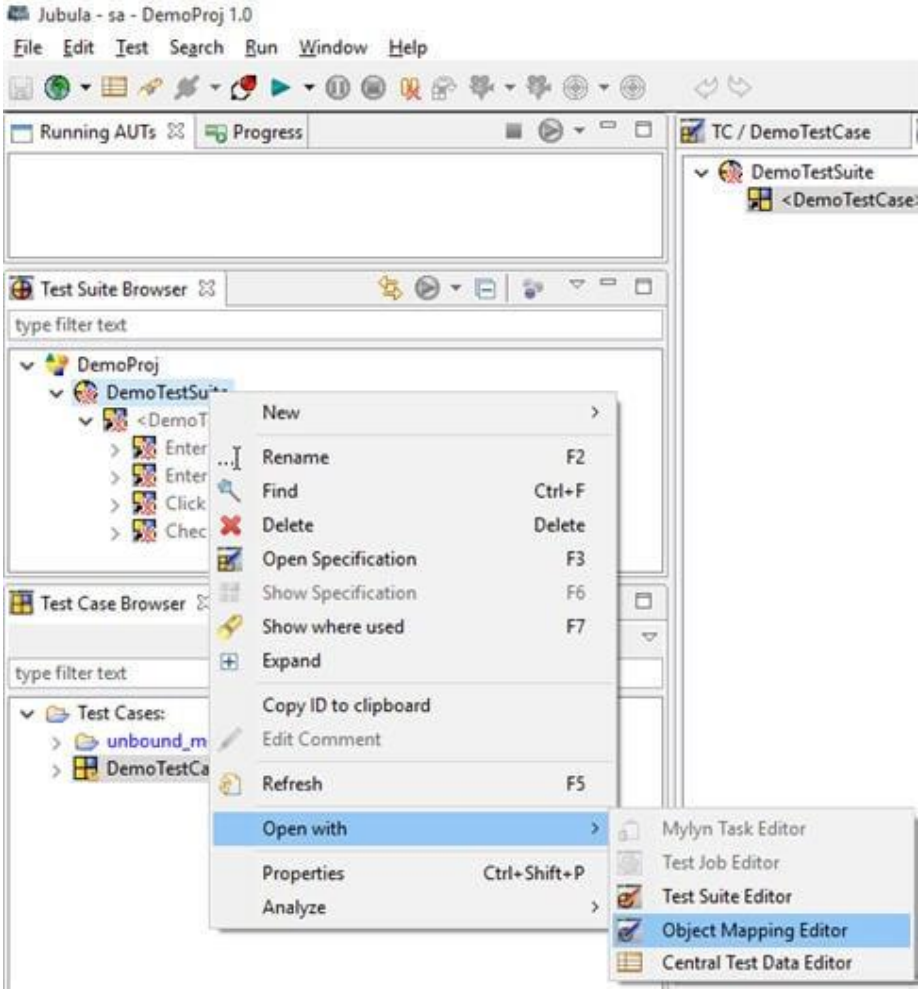


On saving, you should be able to see the test case under the test suite as below:

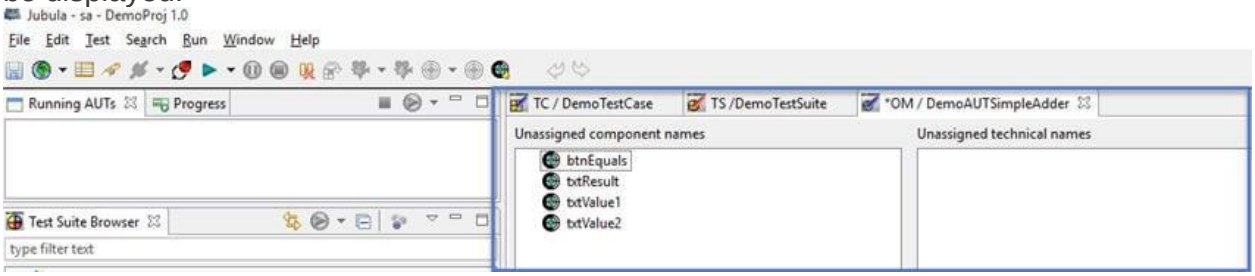


Step #7 – Map logical test objects with technical object identifiers

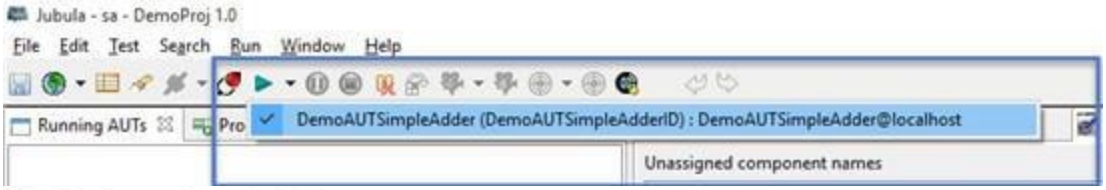
1) Right click on the test suite and open with object mapping editor.

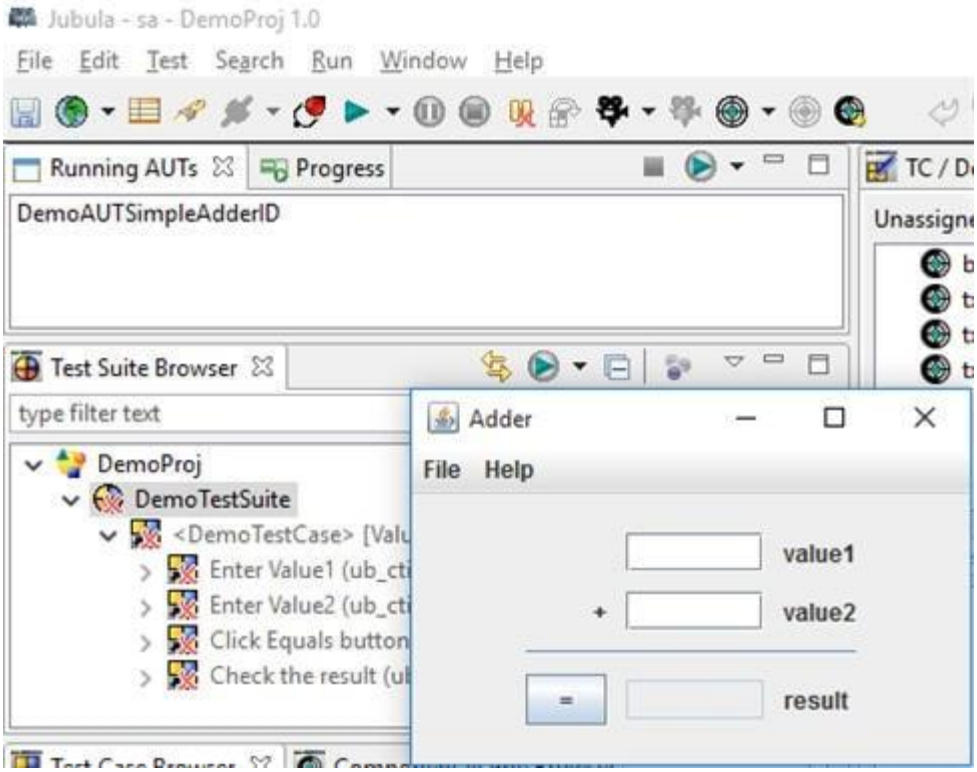


2) All the logical components within the test suite for which technical names are to be mapped will be displayed.



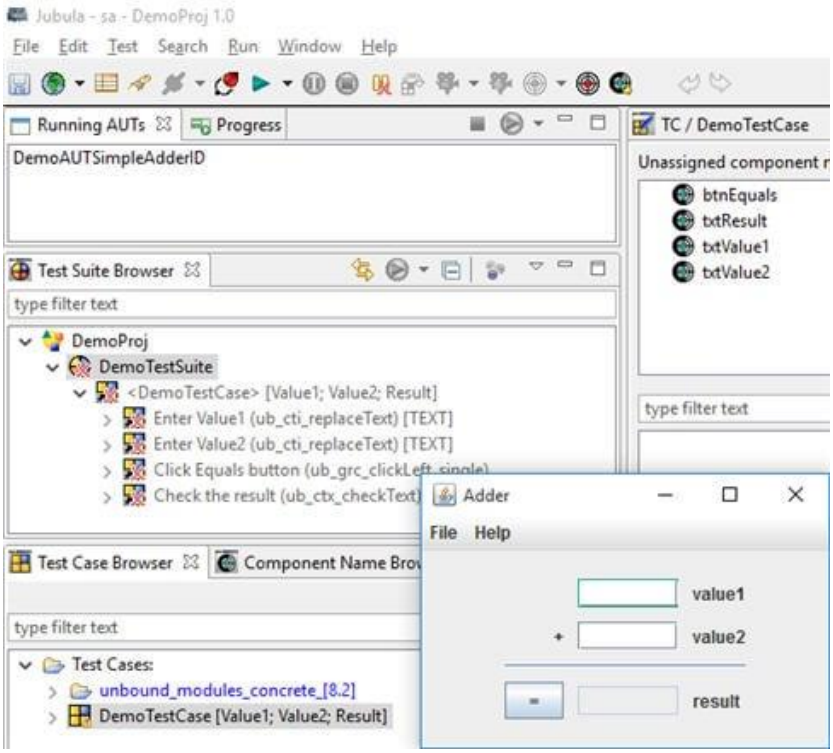
3) Invoke the AUT.



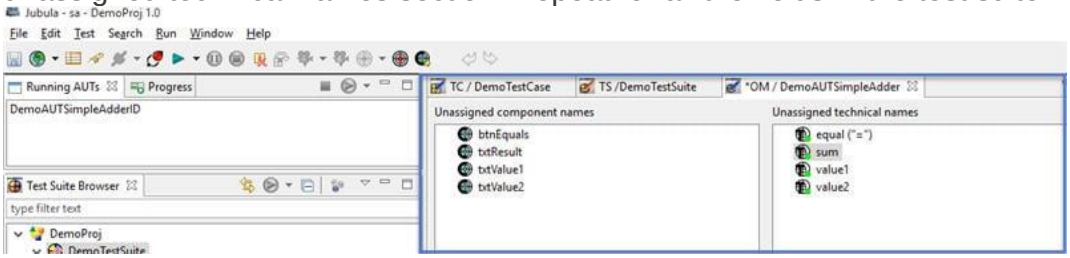


4) Click on the object mapping icon for the particular AUT.

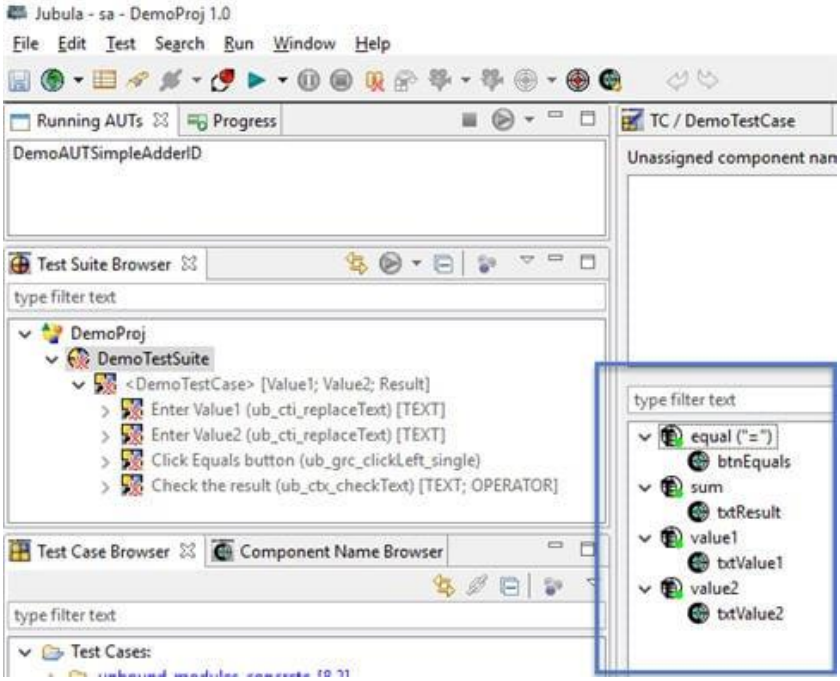
5) Move the cursor over the field for which you have to identify the technical name. The field will be highlighted in green.



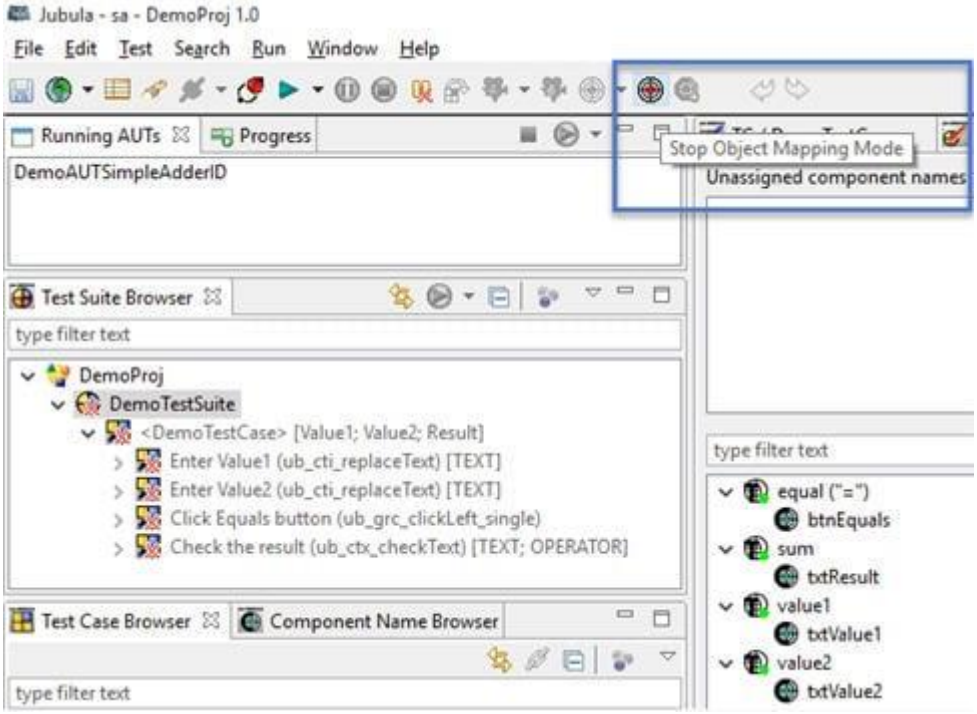
6) Press control + shift + Q to have the corresponding field's technical name under the unassigned technical names section. Repeat for all the fields in the test suite.



7) Now map unassigned component names with an unassigned technical name by simple drag and drop.



8) Save the workspace and quit from the object mapping mode.

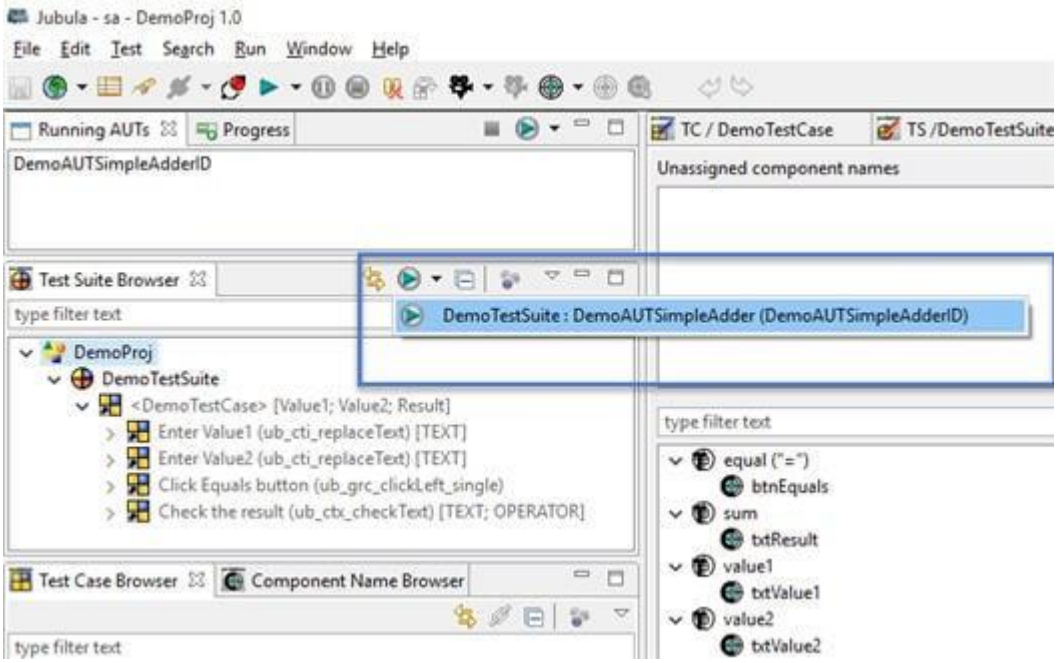


Step #8 – Run the test suite

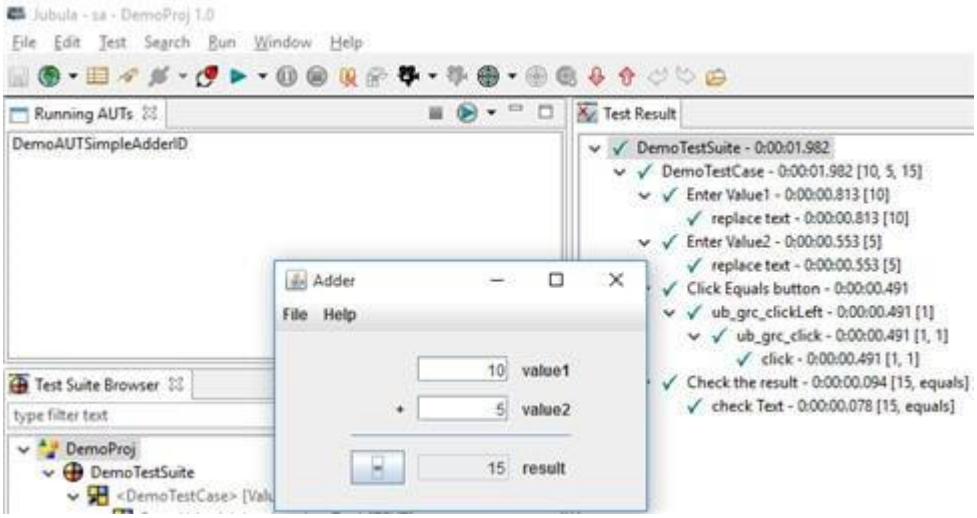
Now the test suite is ready to be run. Make sure the AUT is invoked using Jubula.

Click on the run button in the test suite browser.

(You can also activate the application as the first step. But when you don't have it ensure that you activate the application after running the test suite).



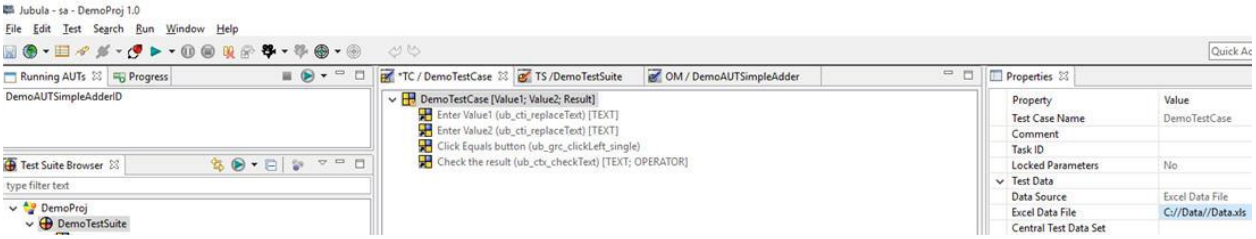
Results can be viewed as below:



Points to be noted

While feeding the test suite with data through Excel, provide the location of the excel file in the example format:

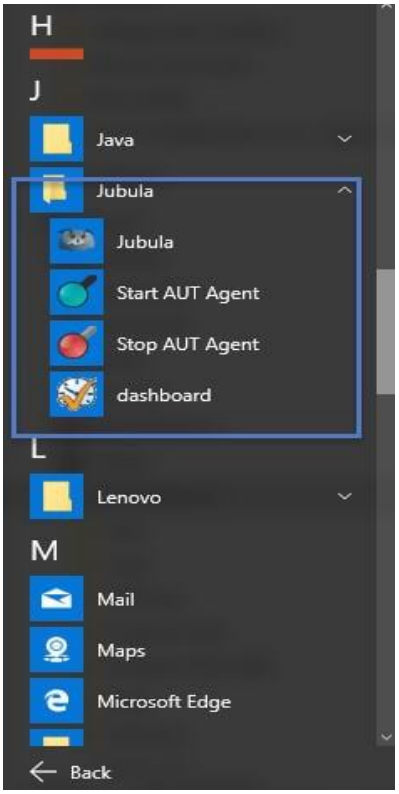
C://Data//Data.xls



In the above demonstration, the data is parameterized using the variables '=Value1', ='Value2' and ='Result'.

In such case when the values need to be passed through an excel file make sure that the file has the corresponding columns with names exactly matching the variables and sheet name set to the language chosen at the time of project creation.

External AUT agent (localhost: 60000) can be connected after starting the agent from 'All programs' where you can find options for starting and stopping the agent.



Multiple test suites can be created under a test job. This would be of help when the testing involves more than one application (Different AUTs can be assigned to different test suites).

This explains the basics to play around the tool. It is very important to get familiarized with Jubula's basic actions to automate complex functionalities and deal with various test objects.

Jubula can also automate applications developed with different technologies, not just swing application.