ODSOFT 2020/2021: CA- Part 1 self-assessment v1.0 TEAM ID: G_01_

—

Instructions: Please mark put an X on the non-greyed column that better reflects your project's item achievement. Fill the gaps (_____) when appropriate.
Analysis Grading: 0 – No Submission; 1 – Attempt; 2 – Achieves Sufficiently 50%; 3 – Achieves partially, includes justification for main options; 4 – Achieves completely with justifications and alternatives.
Implementation Grading: 0 – No Submission; 1 – Attempt; 2 – Achieves Sufficiently; 3 – Achieves with Minor Faults; 4 – Achieves completely including advanced topics.

Description	Analysis					Implementation				
	0	1	2	3	4	0	1	2	3	4
Overall Team Grading									X	
• Each component has, at least, one issue in Bitbucket										x
• Bitbucket issues are closed and refer to the commit									x	
• All source code is committed										x
• Commit messages are clear and well formed									x	
• The repository readme.md is well structured, clear and coherent with the work done										
• Issues have a clear explanation on the scope and differences with other issues								x		
• The documentation on how to debug the Client and the Server is clear, comprehensive and correct						x				
• Members have worked as a team										x
• Other issues:										
○ (Reuniões de trabalho fora do período letivo)										x
○ (_____)										
○ (_____)										
○ (_____)										
○ (_____)										
○ (_____)										
○ (_____)										
○ (_____)										

ODSOFT 2020/2021: CA- Part 1 self-assessment v1.0 TEAM ID: G_01_

Instructions: Please mark put an X on the non-greyed column that better reflects your project's item achievement. Fill the gaps (_____) when appropriate.
Analysis Grading: 0 – No Submission; 1 – Attempt; 2 – Achieves Sufficiently 50%; 3 – Achieves partially, includes justification for main options; 4 – Achieves completely with justifications and alternatives.
Implementation Grading: 0 – No Submission; 1 – Attempt; 2 – Achieves Sufficiently; 3 – Achieves with Minor Faults; 4 – Achieves completely including advanced topics.

Description	Analysis					Implementation				
	0	1	2	3	4	0	1	2	3	4
Component #1										
• The factorial operator is implemented					x				x	
• There is, at least, one unit-test on the factorial										x
• Javadoc of the factorial is generated using Gradle and includes the PlantUML diagram										x
• It is shown, at least, two ways of achieving PlantUML and Javadoc generation					x					x
• There is a PlantUML sequence diagram with the "Add Contact" operation									x	
• The overview page on Javadoc is comprehensive and has the correct sequence diagram									x	
• The project is built using Jenkins									x	
• Javadoc is not generated if the war file is not built									x	
• Jenkins configuration files are in the repository									x	
• The "View Warehouse Details" feature is developed				x					x	
• There are unit-tests on "View Warehouse Details"				x					x	
• There are integration tests on "View Warehouse Details"			x					x		
• There are mutation tests				x			x			
• There is a mutation execution report published			x				x			
• There is a mutation coverage report published			x				x			
• The Jenkins pipeline has all the tests			x					x		

ODSOFT 2020/2021: CA- Part 1 self-assessment v1.0 TEAM ID: G01 _

_

Instructions: Please mark put an X on the non-greyed column that better reflects your project's item achievement. Fill the gaps (_____) when appropriate.
Analysis Grading: 0 – No Submission; 1 – Attempt; 2 – Achieves Sufficiently 50%; 3 – Achieves partially, includes justification for main options; 4 – Achieves completely with justifications and alternatives.
Implementation Grading: 0 – No Submission; 1 – Attempt; 2 – Achieves Sufficiently; 3 – Achieves with Minor Faults; 4 – Achieves completely including advanced topics.

Description	Analysis					Implementation				
	0	1	2	3	4	0	1	2	3	4
Component #2										
• The third operator is implemented				x						x
• There is, at least, one unit-test on the third operator									x	
• Javadoc of the third operator is generated using Gradle and includes the PlantUML diagram									x	
• It is shown, at least, two ways of achieving PlantUML and Javadoc generation				x					x	
• There is a PlantUML sequence diagram with the "Delete Contact" operation								x		
• The overview page on Javadoc is comprehensive and has the correct sequence diagram								x		
• The project is built using Jenkins									x	
• Javadoc is not generated if the war file is not built									x	
• Jenkins configuration files are in the repository									x	
• The "Update Warehouse" feature is developed				x					x	
• There are unit-tests on "Update Warehouse"			x					x		
• There are integration tests on "Update Warehouse"			x					x		
• There are comprehensive unit-tests			x					x		
• There is a unit-test execution report published				x					x	
• There is a unit-test coverage report published				x					x	
• The Jenkins pipeline has all the tests		x					x			

ODSOFT 2020/2021: CA- Part 1 self-assessment v1.0 TEAM ID: G_ _

__

Instructions: Please mark put an X on the non-greyed column that better reflects your project's item achievement. Fill the gaps (_____) when appropriate.
Analysis Grading: 0 – No Submission; 1 – Attempt; 2 – Achieves Sufficiently 50%; 3 – Achieves partially, includes justification for main options; 4 – Achieves completely with justifications and alternatives.
Implementation Grading: 0 – No Submission; 1 – Attempt; 2 – Achieves Sufficiently; 3 – Achieves with Minor Faults; 4 – Achieves completely including advanced topics.

Description	Analysis					Implementation				
	0	1	2	3	4	0	1	2	3	4
Component #3										
• The double operator is implemented					x					x
• There is, at least, one unit-test on the double operator										x
• Javadoc of the double operator is generated using Gradle and includes the PlantUML diagram									x	
• It is shown, at least, two ways of achieving PlantUML and Javadoc generation					x					x
• There is a PlantUML sequence diagram with the "Update Contact" operation									x	
• The overview page on Javadoc is comprehensive and has the correct sequence diagram									x	
• The project is built using Jenkins									x	
• Javadoc is not generated if the war file is not built									x	
• Jenkins configuration files are in the repository										x
• The "Delete Warehouse" feature is developed					x					x
• There are unit-tests on "Delete Warehouse"			x					x		
• There are integration tests on "Delete Warehouse"				x					x	
• There are comprehensive integration tests			x					x		
• There is an integration test execution report published			x					x		
• There is an integration test coverage report published			x					x		
• The Jenkins pipeline has all the tests				x					x	

ODSOFT 2020/2021: CA- Part 1 self-assessment v1.0 TEAM ID: G_01_

_

Instructions: Please mark put an X on the non-greyed column that better reflects your project's item achievement. Fill the gaps (_____) when appropriate.
Analysis Grading: 0 – No Submission; 1 – Attempt; 2 – Achieves Sufficiently 50%; 3 – Achieves partially, includes justification for main options; 4 – Achieves completely with justifications and alternatives.
Implementation Grading: 0 – No Submission; 1 – Attempt; 2 – Achieves Sufficiently; 3 – Achieves with Minor Faults; 4 – Achieves completely including advanced topics.

Description	Analysis					Implementation				
	0	1	2	3	4	0	1	2	3	4
Component #4										
• The exponential operator is implemented				x					x	
• There is, at least, one unit-test on the exponential operator									x	
• Javadoc of the exponential operator is generated using Gradle and includes the PlantUML diagram									x	
• It is shown, at least, two ways of achieving PlantUML and Javadoc generation					x					x
• There is a PlantUML sequence diagram with the "View Contact" operation									x	
• The overview page on Javadoc is comprehensive and has the correct sequence diagram								x		
• The project is built using Jenkins									x	
• Javadoc is not generated if the war file is not built									x	
• Jenkins configuration files are in the repository										x
• The "Create Warehouse" feature is developed				x						x
• There are unit-tests on "Create Warehouse"			x					x		
• There are integration tests on "Create Warehouse"			x					x		
• There is comprehensive Javadoc documentation				x				x		
• The Javadoc is published				x					x	
• The war file is published				x					x	
• The Jenkins pipeline has all the tests				x					x	