CodeToUML Test Cases

Use Case Id:	**UC_01**	**Use Case Name:**	Console – Individual selection Option	Test Date:	
Test Case Id:	**TC_01**	**Test Case Name:**	Input valid	Tester:	
Steps	**Inputs**	**Expected Output**	Actual Result	(Pass/Fail)	Comment
1.	Type the corresponding number to choose to select files individually	Console will let them know it will be doing individual file selection and ask the user to type in the file they would like to use with a warning to let them know it must be located in a specific folder.		Pass	

Use Case Id:	**UC_01**	**Use Case Name:**	Console – Individual file selection	Test Date:	
Test Case Id:	**TC_02**	**Test Case Name:**	Input Invalid	Tester:	
Steps	**Inputs**	**Expected Output**	Actual Result	(Pass/Fail)	Comment
1.	Type an invalid option that isn't for any of the listed options displayed by the console	Console will tell them that the input isn't accepted and try to put in the proper one again.		pass	
2.	Type a valid option corresponding to individual file selection	Console will proceed to perform individual file selection.		pass	

Use Case Id:	UC_02	Use Case Name:	Removing Unwanted Files	Test Date:	
Test Case Id:	TC_03	Test Case Name:	Remove File	Tester:	
Steps	Inputs	Expected Output	Actual Result	(Pass/Fail)	Comment
1.	Number to initiate file removal selection	Presents a list of files in a numbered list and prompts the user to input a number of the file they want to remove		Pass	
2.	User inputs a number to remove respective file and hits enter (done after successfully adding a file)	Program will then remove the file from the list of what to look through and present a new list of what remains, if any.		Pass	

Use Case Id:	UC_03	Use Case Name:	Add files individually (successful)	Test Date:	
Test Case Id:	TC_04	Test Case Name:	Individual selection	Tester:	
Steps	Inputs	Expected Output	Actual Result	(Pass/Fail)	Comment
1.	User selects the option to input files individually	Console prompts the user to input a file path to locate and add in the file		pass	
2.	User inputs “hello.txt” or “testing.txt” file path and hits enter	Program will show them the added file (or list of added files if done before), and ask the user if they wish to add more files, proceed with selection, or remove any unwanted files		pass	

Use Case Id:	**UC_03**	**Use Case Name:**	Add files individually (fail)	Test Date:	
Test Case Id:	**TC_06**	**Test Case Name:**	Individual selection	Tester:	
Steps	**Inputs**	**Expected Output**	Actual Result	(Pass/Fail)	Comment
1.	User selects option to add files individually	Program prompts the user to write in a file directory path		pass	
2.	User enters an incorrect path and hits enter (anything besides “hello.txt” or “testing.txt”)	Program shows an error message letting the user know that the directory path is invalid and asks the user to enter the correct path again (will repeat until user enters a valid path or chooses to cancel individual selection)		pass	

Use Case Id:	**UC_02**	**Use Case Name:**	Remove files (fail)	Test Date:	
Test Case Id:	**TC_06**	**Test Case Name:**	Remove files	Tester:	
Steps	**Inputs**	**Expected Output**	Actual Result	(Pass/Fail)	Comment
1.	User selects option to remove any unwanted files	Program displays list of current files selected in a numbered list		pass	

2.	User enters a number that is out of the range of the number of files present	Program shows an error message letting the user know that the input is invalid and they must choose from the available numbers displayed (will repeat until user chooses valid input or cancels file removal)		pass	

IGNORE!

Use Case Id:	**UC_04**	**Use Case Name:**	Add files recursively	Test Date:	
Test Case Id:	**TC_05**	**Test Case Name:**	Recursive selection: Single File	Tester:	
Steps	**Inputs**	**Expected Output**	Actual Result	(Pass/Fail)	Comment
1.	User selects option to input files recursively	Prompts user to enter topmost file path of project/folder			
2.	Type a file path with a single source code file and confirm with enter key	Prints “is valid” message. Program Proceeds as normal and finishes running			
3.	Check folders for a .txt file and open it in notepad	Text file should should only have one class formatted onto the file			
4.	Check image file	Only one class should be depicted in the image			

IGNORE!

Use Case Id:	**UC_04**	**Use Case Name:**	Add files recursively	Test Date:	
Test Case Id:	**TC_05**	**Test Case Name:**	Recursive selection: Multiple File	Tester:	
Steps	**Inputs**	**Expected Output**	Actual Result	(Pass/Fail)	Comment
1.	User selects option to input files recursively	Prompts user to enter topmost file path of project/folder			
2.	Type a file path with multiple source code files and confirm with enter key	Prints “is valid” message. Program Proceeds as normal and finishes running			
3.	Check folders for a .txt file and open it in notepad	Text file should should only have multiple classes matching number of number of files formatted onto the file			
4.	Check image file	Same number of classes should be depicted in the image			

IGNORE!

Use Case Id:	**UC_04**	**Use Case Name:**	Add files recursively	Test Date:	
Test Case Id:	**TC_05**	**Test Case Name:**	Recursive selection: Nested Folders	Tester:	
Steps	**Inputs**	**Expected Output**	Actual Result	(Pass/Fail)	Comment
1.	User selects option to input files recursively	Prompts user to enter topmost file path of project/folder			
2.	Type a file path with nested folders and source code files and confirm with enter key	Prints “is valid” message. Program Proceeds as normal and finishes running			
3.	Check folders for a .txt file and open it in notepad	Text file should should have multiple sections with the same total as the number of source code files formatted onto the file			
4.	Check image file	Multiple classes with the same total number of source code files should be depicted in the image			

IGNORE!

Use Case Id:	**UC_05**	**Use Case Name:**	Produce UML Diagram (May not work)	Test Date:	
Test Case Id:	**TC_06**	**Test Case Name:**	UML Diagram success	Tester:	
Steps	**Inputs**	**Expected Output**	Actual Result	(Pass/Fail)	Comment
1.	User confirms file selection to make a UML diagram	Program will produce a UML diagram and let the user know it has finished and show the directory path of where they can find the UML diagram			

IGNORE!

Use Case Id:	**UC_05**	**Use Case Name:**	Produce UML diagram (May not work)	Test Date:	
Test Case Id:	**TC_06**	**Test Case Name:**	UML diagram fail	Tester:	
Steps	**Inputs**	**Expected Output**	Actual Result	(Pass/Fail)	Comment
1.	User confirms file selection and confirms to produce a UML diagram	Program shows an error and lets the user know that a file is not valid (or may be corrupt) and shows the file name (possibly whole file directory for possible easy fine)			

2.	User will check the files and either remove or fix them and then re-attempt trying to produce a UML diagram	Program will produce a UML diagram and inform the user that it has finished and shows the file directory path of the UML diagram so the user can find it			