

Assignment - ASE

COMPONENT - 1

PRAVESH PANSARI

Contents

Simple Programming Environment.....	2
Interface:.....	2
Functions:.....	3
Commands:	4
Program Design.....	5
UML Class Diagram	5
Version Control – GitHub.....	6
Repository:.....	6
Commits:	7
Test Cases.....	9
Summary:	9
All tests pass:.....	9
All tests fail:.....	9
Results:.....	10
Shape Factory Test – Rectangle	10
Shape Factory Test – Triangle	10
Command Test – Move to.....	11
Command Test – Draw To.....	11
Command Test – Reset Command.....	12
Command Test – Pen Command.....	12
Command Test – Create Variable	13
Command Test – While	13
Command Test – If statement	14
Command Test – Create Method.....	14
Command Test – Complex Expression.....	15

Simple Programming Environment

Interface:

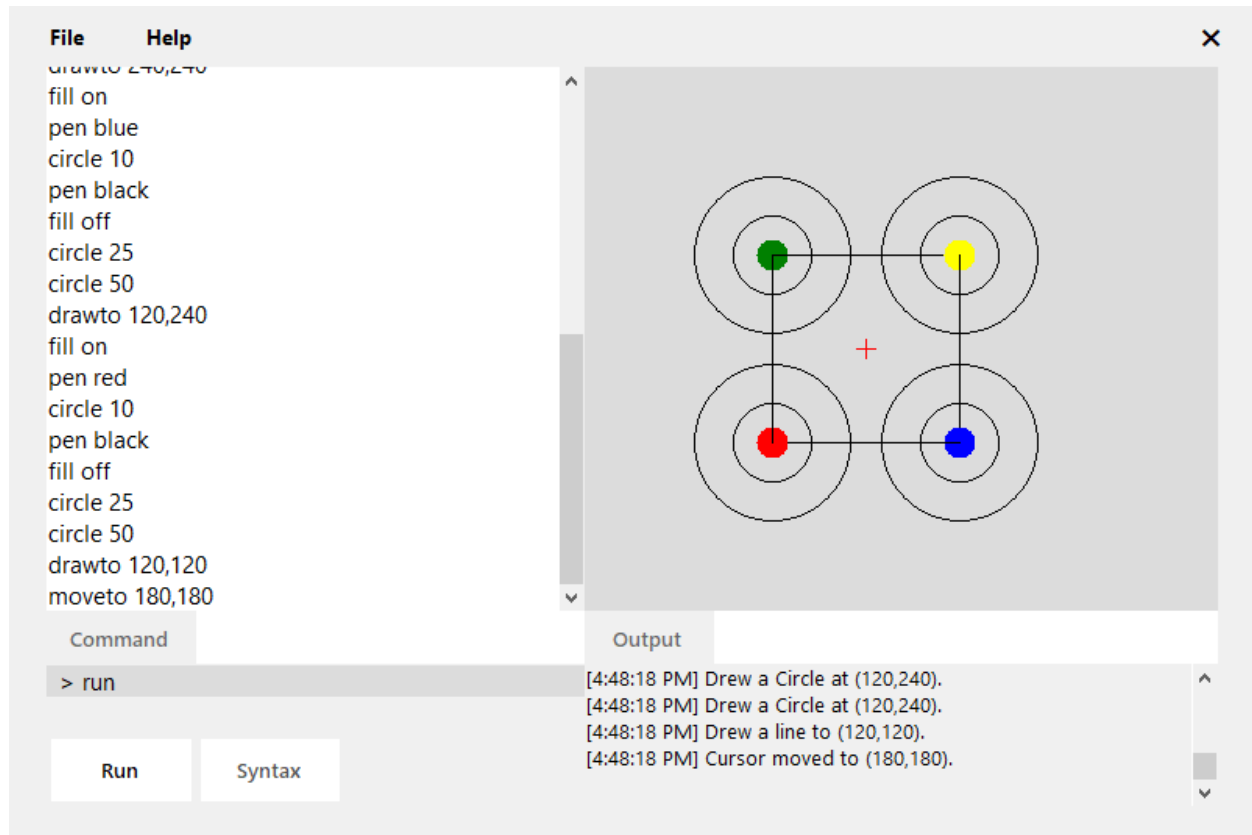


Figure 1: The GUI of Simple Programming Environment

This program creates a simple programming environment, where commands can be used to manipulate shapes and lines on an artboard. The artboard is the place where all drawings and the cursor are rendered. The menu allows access to different file operations, exit, and the about information. The current position of the cursor on the artboard is represented by the red cross. The command line runs single commands while the code editor can run multiple commands where every command is on a new line. Commands can be executed using the run button. The output box displays the log information and error information for the commands. The syntax button analyzes the code in the code editor and displays any errors it found in the log box.

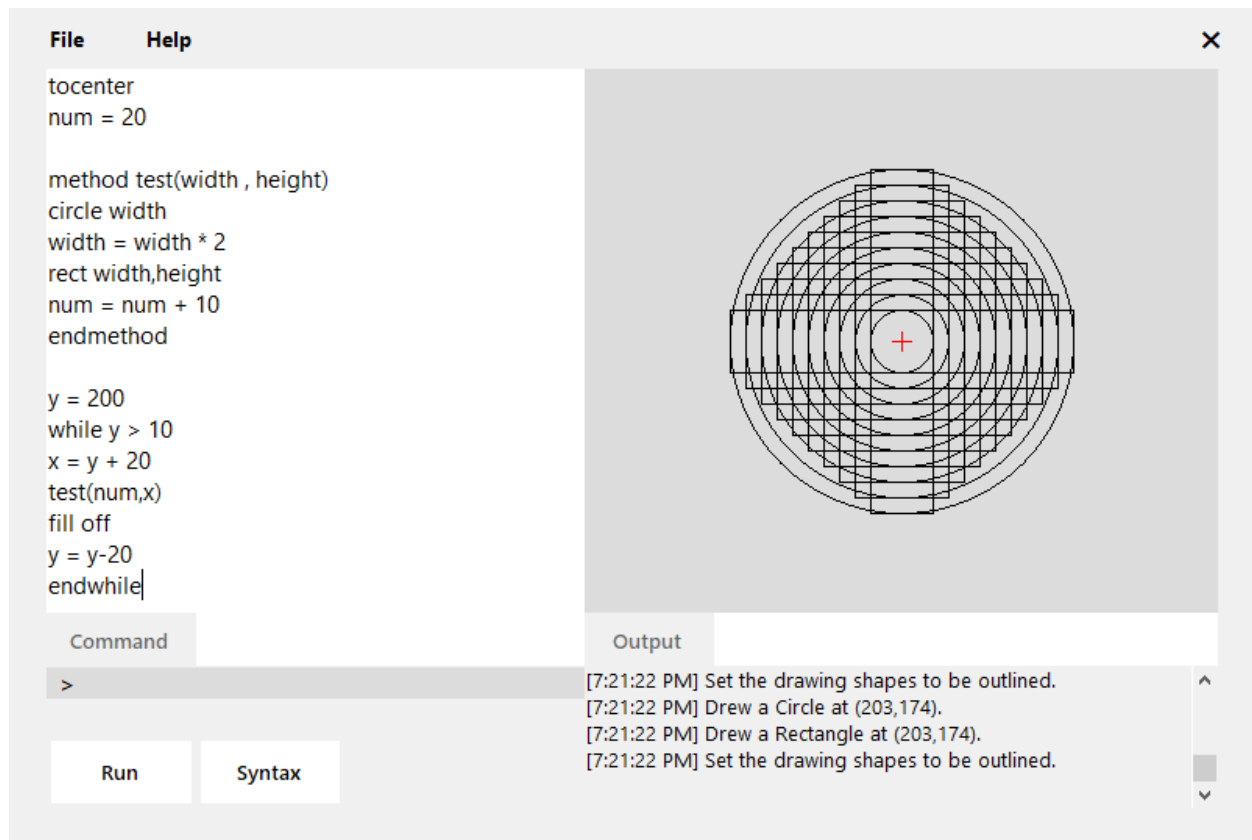


Figure 2: Making a complex drawing using advanced commands

Functions:

Four types of shapes can be drawn: rectangle, square, circle and triangle. And a line can be drawn given a destination position. The shapes can be drawn outlined or filled. The color of shapes and lines can be changed. The cursor can be reset to top left, moved to center or any position in the artboard. The board can be cleared, which removes all drawings on it. The file menu allows the user to create new file, save the code, open a file and exit the program. The help menu consists of information about the program.

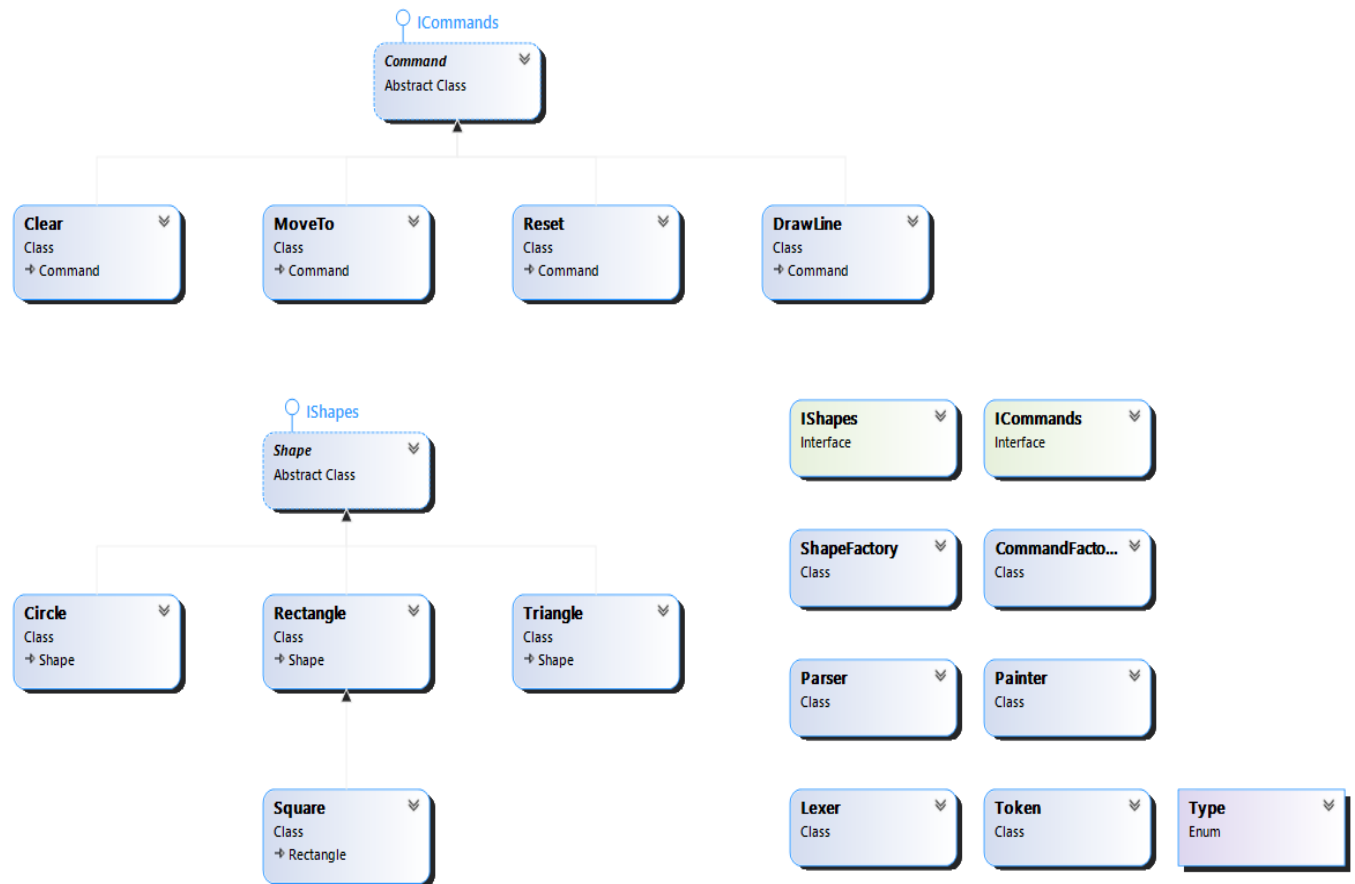
Variables can be declared in this program. They must contain only letters and can be assigned to integers only. Complex operations can be performed in the assignment statement including 5 types of mathematical operations. Commands can be run conditionally using if statements. They can be used as single line or a complete block of code. While loops can be used to redo commands until a stop condition. Methods can be declared and be called. Both parameterized and parameter less methods can be implemented. Then they can be called by passing appropriate parameters.

Commands:

- **drawto** int x, int y – Draws a line from current position to (x, y)
- **moveto** int x, int y – Moves the cursor to (x, y)
- **pen** string color – Changes the Pen color to color
- **fill** string input – Sets the fill of the shapes (on or off)
- **rect** int width, int height – Draws a rectangle of width and height
- **square** int size – Draws a square of length size
- **circle** int radius – Draws a circle of radius
- **triangle** int base, int height – Draws a triangle with base and height
- **clear** – Clears the drawing area
- **reset** – Reset the circle position to (0, 0)
- **tocenter** – Moves the cursor to center
- string test = int value – Creates a variable named test and assigns value to it
- **if** var **op** int x – Compares var against x using comparison operator op
- **while** var **op** int x – Loops till the condition var against x is true using op
- **method** example (param) – Creates a function example with parameter param

Program Design

UML Class Diagram



Version Control – GitHub

Repository:

Link: <https://github.com/praveshpansari/PLEnvironment>

The screenshot displays the GitHub repository page for `praveshpansari/PLEnvironment`. The repository is private and has 1 branch and 1 tag. The main branch is `master`. The repository contains a list of files and folders, including `Properties`, `SPL-Testing`, `_site`, `api`, `articles`, `testcases`, `.gitattributes`, `.gitignore`, `App.config`, `Assignment-ASE.cproj`, `Assignment-ASE.sln`, `AssignmentASE.xml`, `Circles.cs`, `Clear.cs`, `Command.cs`, `CommandFactory.cs`, `CommandInterface.cs`, `Commits.png`, `DocumentationASE1.docx`, `DrawLine.cs`, `Environment.Designer.cs`, `Environment.cs`, `Environment.resx`, `Lexer.cs`, `MoveTo.cs`, `Painter.cs`, `Parser.cs`, `Program.cs`, `README.md`, `Rectangle.cs`, `Reset.cs`, `Shape.cs`, `ShapeFactory.cs`, `ShapesInterface.cs`, `Square.cs`, `Triangle.cs`, `UMLClassDiagram.cd`, `UMLClassDiagram.png`, `docfx.json`, `index.md`, `log.txt`, `packages.config`, `test.xlsx`, and `toc.yml`.

The latest commit is by `praveshpansari` with the message "Implemented remaining test cases" and 33 commits. The commit history shows a series of updates to the repository, including adding XML comments, implementing test cases, and adding documentation.

On the right side of the page, there is an "About" section with the text: "Assignment of Advanced Software Engineering wherein we are expected to create a simple programming language environment." Below this, there is a "Releases" section with a "Release" button and a "Latest" label. The "Packages" section shows "No packages published" and a link to "Publish your first package". The "Languages" section shows a bar chart with the following data:

Language	Percentage
CSS	30.9%
HTML	29.1%
JavaScript	27.7%
C#	11.7%

Commits:

praveishpansari / PLEnvironment Private

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master

Commits on Dec 14, 2020

- Implemented remaining test cases [6bc3868](#)
- Added XML documentation and comments [4c7c3d7](#)
- Added Syntax Check button [39d72d3](#)
- Syntax Checking & Exception Handling Finished [c8d678c](#)

Commits on Dec 13, 2020

- Now supports modulo operator. Added some example code files [3b587e8](#)
- Added methods with parameters [8fc6273](#)

Commits on Dec 12, 2020

- Added functions [e3cf2e7](#)
- Added while loops [e551858](#)
- Added If statements [62d39ed](#)

Commits on Dec 11, 2020

- Fixed a bug in Lexer [e9c87d8](#)
- Added a variables [ba6b34e](#)

Commits on Dec 4, 2020

- Finished Component 1 & Documentation [f9acabf](#)
- Finished Adding comments [1a5ad02](#)
- Added XML comments [d117555](#)
- Moved the testing project here [9c138f1](#)

Commits on Dec 3, 2020

- Added Comments and Syntax check [141518c](#)
- Implement Tests [8a30cfc](#)

Commits on Dec 1, 2020	UI changes and added save,load,new praveshpansari committed 13 days ago	85ce25c	<>
Commits on Nov 30, 2020	Implemented Editor and added logging praveshpansari committed 14 days ago	97d8c5e	<>
Commits on Nov 29, 2020	Added Triangle & ClearUsed Shapefactory praveshpansari committed 15 days ago	64d8f66	<>
	Added Shapes Facotry praveshpansari committed 15 days ago	86f1f10	<>
	Now using the university advised method praveshpansari committed 15 days ago	ee23805	<>
	Implemented Parser praveshpansari committed 15 days ago	f0f93d7	<>
Commits on Nov 28, 2020	Added Documentation for Lexer praveshpansari committed 16 days ago	f9c46ad	<>
	Completed Lexer praveshpansari committed 16 days ago	59ceb7f	<>
Commits on Nov 27, 2020	Added Parser praveshpansari committed 17 days ago	2ec79f0	<>
	Added Lexer praveshpansari committed 17 days ago	7d8c976	<>
	Created the collection of some commands praveshpansari committed 17 days ago	3975fb2	<>
Commits on Nov 16, 2020	Created ReadMe praveshpansari committed 28 days ago	Verified bbe4267	<>
	Added Command Interface praveshpansari committed 28 days ago	9eb1b91	<>
	Initations praveshpansari committed 28 days ago	d21ab85	<>
	Add project files. praveshpansari committed 28 days ago	1d6e575	<>
	Add gitignore and gitattributes. praveshpansari committed 28 days ago	179cb7a	<>

Test Cases

Summary:

All tests pass:

Test	Duration	Group Summary
<ul style="list-style-type: none"> ✓ SPL-Testing (15) <ul style="list-style-type: none"> ✓ SPL_Testing (15) <ul style="list-style-type: none"> ✓ CommandTest (11) <ul style="list-style-type: none"> ✓ TestParseCommandColor ✓ TestParseCommandDrawTo ✓ TestParseCommandFillOff ✓ TestParseCommandFillOn ✓ TestParseCommandIf ✓ TestParseCommandMethod ✓ TestParseCommandMoveTo ✓ TestParseCommandReset ✓ TestParseCommandVariable ✓ TestParseCommandWhile ✓ TestParseExpression ✓ ShapeFactoryTest (4) <ul style="list-style-type: none"> ✓ TestShapeFactoryCircle ✓ TestShapeFactoryRectangle ✓ TestShapeFactorySquare ✓ TestShapeFactoryTriangle 		SPL-Testing Tests in group: 15 ⌚ Total Duration: 180 ms Outcomes ✓ 15 Passed

All tests fail:

Test	Duration	Group Summary
<ul style="list-style-type: none"> ✗ SPL-Testing (15) <ul style="list-style-type: none"> ✗ SPL_Testing (15) <ul style="list-style-type: none"> ✗ CommandTest (11) <ul style="list-style-type: none"> ✗ TestParseCommandColor ✗ TestParseCommandDrawTo ✗ TestParseCommandFillOff ✗ TestParseCommandFillOn ✗ TestParseCommandIf ✗ TestParseCommandMethod ✗ TestParseCommandMoveTo ✗ TestParseCommandReset ✗ TestParseCommandVariable ✗ TestParseCommandWhile ✗ TestParseExpression ✗ ShapeFactoryTest (4) <ul style="list-style-type: none"> ✗ TestShapeFactoryCircle ✗ TestShapeFactoryRectangle ✗ TestShapeFactorySquare ✗ TestShapeFactoryTriangle 	1 ms	SPL-Testing Tests in group: 15 ⌚ Total Duration: 292 ms Outcomes ✗ 15 Failed

Results:

Shape Factory Test – Rectangle

Test Case ID	SF_01	Test Case Description	Test for the shape factory			
Created By	Pravesh Pansari			Version	1.0	
Tester's Name	Pravesh Pansari	Date Tested	4th December,	Test Case (Pass/Fail)	Pass	
S #	Prerequisites:		S #	Test Data		
1	Access to Visual Studio 2019		1	GetShape parameter = "rect"		
2	Access to Shape Interface and Shape classes		2			
3	Access to Shape Factory		3			
4			4			
Test Scenario	Verify whether the required shape can be generated from the factory					
Step #	Step Details	Expected Results	Actual Results		Pass / Fail / Not executed / Suspended	
1	Create a ShapeFactory object	ShapFactory Object	As Expected		Pass	
2	Create a Shape object	Shape object created	As Expected		Pass	
3	Use the getShape() Method	Rectangle Shape	As Expected		Pass	

Shape Factory Test – Triangle

Test Case ID	SF_02	Test Case Description	Test for the shape factory			
Created By	Pravesh Pansari			Version	1.0	
Tester's Name	Pravesh Pansari	Date Tested	4th December,	Test Case (Pass/Fail)	Pass	
S #	Prerequisites:		S #	Test Data		
1	Access to Visual Studio 2019		1	GetShape parameter = "triangle"		
2	Access to Shape Interface and Shape classes		2			
3	Access to Shape Factory		3			
4			4			
Test Scenario	Verify whether the required shape can be generated from the factory					
Step #	Step Details	Expected Results	Actual Results		Pass / Fail / Not executed / Suspended	
1	Create a ShapeFactory object	ShapFactory Object	As Expected		Pass	
2	Create a Shape object	Shape object created	As Expected		Pass	
3	Use the getShape() Method	Triangle Shape Created	As Expected		Pass	

Command Test – Move to

Test Case ID	C_01	Test Case Description	Test for Move to command		
Created By	Pravesh Pansari			Version	1.0
Tester's Name	Pravesh Pansari	Date Tested	4th December,	Test Case (Pass/Fail)	Pass
S #	Prerequisites:		S #	Test Data	
1	Access to Visual Studio 2019		1	parseCommand(input,0), where input = "moveto	
2	Access to Command Interface, Command		2		
3	Access to Command Factory		3		
4	Access to Parser,Painter classes		4		
Test Scen:	Verify whether the x position and the y position of the cursor have been moved				
Step #	Step Details	Expected Results	Actual Results		Pass / Fail / Not executed / Suspended
1	Create a Painter Object	Painter object created	As Expected		Pass
2	Create a Parser Object	Parser object created	As Expected		Pass
3	Use the parseCommand() method	Cursor moved to	As Expected		Pass

Command Test – Draw To

Test Case ID	C_02	Test Case Description	Test for Draw to command		
Created By	Pravesh Pansari			Version	1.0
Tester's Name	Pravesh Pansari	Date Tested	4th December,	Test Case (Pass/Fail)	Pass
S #	Prerequisites:		S #	Test Data	
1	Access to Visual Studio 2019		1	parseCommand(input,0), where input = "drawto	
2	Access to Command Interface, Command		2		
3	Access to Command Factory		3		
4	Access to Parser,Painter classes		4		
Test Scenario	Verify whether the x position and the y position of the cursor have been moved				
Step #	Step Details	Expected Results	Actual Results		Pass / Fail / Not executed / Suspended
1	Create a Painter Object	Painter object created	As Expected		Pass
2	Create a Parser Object	Parser object created	As Expected		Pass
3	Use the parseCommand() method	Cursor moved to	As Expected		Pass

Command Test – Reset Command

Test Case ID	C_03	Test Case Description	Test for reset command			
Created By	Pravesh Pansari			Version	1.0	
Tester's Name	Pravesh Pansari	Date Tested	4th December,	Test Case (Pass/Fail)	Pass	
S #	Prerequisites:			S #	Test Data	
1	Access to Visual Studio 2019			1	parseCommand(input,0), where input = "reset"	
2	Access to Command Interface, Command			2		
3	Access to Command Factory			3		
4	Access to Parser,Painter classes			4		
Test Scen:	Verify whether the x position and the y position of the cursor have been set to top left					
Step #	Step Details	Expected Results	Actual Results		Pass / Fail / Not executed / Suspended	
1	Create a Painter Object	Painter object created	As Expected		Pass	
2	Create a Parser Object	Parser object created	As Expected		Pass	
3	Use the parseCommand() method	Cursor reset to top left	As Expected		Pass	

Command Test – Pen Command

Test Case ID		C_04	Test Case Description		Test for pen command			
Created By		Pravesh Pansari			Version		1.0	
Tester's Name		Pravesh Pansari	Date Tested		4th December,		Test Case (Pass/Fail)	
							Pass	
S #		Prerequisites:		S #		Test Data		
1		Access to Visual Studio 2019		1		parseCommand(input,0), where input = "pen red"		
2		Access to Command Interface, Command		2				
3		Access to Command Factory		3				
4		Access to Parser,Painter classes		4				
Test Scenario		Verify whether the pen color has been set to red						
Step #		Step Details		Expected Results		Actual Results		Pass / Fail / Not executed / Suspended
1		Create a Painter Object		Painter object created		As Expected		Pass
2		Create a Parser Object		Parser object created		As Expected		Pass
3		Use the parseCommand() method		Pen color set to red		As Expected		Pass

Command Test – Create Variable

Test Case ID	C_05	Test Case Description	Test for var command			
Created By	Pravesh Pansari			Version	1.1	
Tester's Name	Pravesh Pansari	Date Tested	14th December, 2020	Test Case (Pass/Fail)	Pass	
S #	Prerequisites:		S #	Test Data		
1	Access to Visual Studio 2019		1	parseUsingLexer("num = 5",0)		
2	Access to Command Interface, Command		2			
3	Access to Command Factory		3			
4	Access to Parser,Painter,Lexer classes		4			
Test Scenario	Verify whether the pen color has been set to red					
Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended		
1	Create a Painter Object	Painter object created	As Expected	Pass		
2	Create a Parser Object	Parser object created	As Expected	Pass		
3	Use parseUsingLexer() method	Create variable "num" with value 5	As Expected	Pass		

Command Test – While

Test Case ID	C_06	Test Case Description	Test for while command			
Created By	Pravesh Pansari			Version	1.1	
Tester's Name	Pravesh Pansari	Date Tested	14th December, 2020	Test Case (Pass/Fail)	Pass	
S #	Prerequisites:		S #	Test Data		
1	Access to Visual Studio 2019		1	parseEditor(input) where input is "num = 5\r\nwhile num > 3\r\nnum = num - 1\r\nendwhile"		
2	Access to Command Interface, Command					
3	Access to Command Factory					
4	Access to Parser,Painter, Lexer classes					
Test Scenario	Verify whether num has been decreased to 3 from 5					
Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended		
1	Create a Painter Object	Painter object created	As Expected	Pass		
2	Create a Parser Object	Parser object created	As Expected	Pass		
3	Use parseEditor() method	Num decrease to 3	As Expected	Pass		

Command Test – If statement

Test Case ID	C_07	Test Case Description	Test for if command			
Created By	Pravesh Pansari			Version	1.1	
Tester's Name	Pravesh Pansari	Date Tested	14th December, 2020	Test Case (Pass/Fail)	Pass	
S #	Prerequisites:		S #	Test Data		
1	Access to Visual Studio 2019		1	parseUsingIf(input,0) where input is "if 5 > 20"		
2	Access to Command Interface, Command					
3	Access to Command Factory					
4	Access to Parser,Painter, Lexer classes					
Test Scenario	Verify whether 5 is greater than 20 in if statement returns false					
Step #	Step Details	Expected Results	Actual Results		Pass / Fail / Not executed / Suspended	
1	Create a Painter Object	Painter object created	As Expected		Pass	
2	Create a Parser Object	Parser object created	As Expected		Pass	
3	Use parseUsingIf() method	Returns false	As Expected		Pass	

Command Test – Create Method

Test Case ID	C_08	Test Case Description	Test for method command			
Created By	Pravesh Pansari			Version	1.1	
Tester's Name	Pravesh Pansari	Date Tested	14th December, 2020	Test Case (Pass/Fail)	Pass	
S #	Prerequisites:		S #	Test Data		
1	Access to Visual Studio 2019		1	parseEditor(input) where input is "method example()\r\n\r\n20\r\n\r\nendmethod"		
2	Access to Command Interface, Command					
3	Access to Command Factory					
4	Access to Parser,Painter, Lexer classes					
Test Scenario	Verify whether method example has been created					
Step #	Step Details	Expected Results	Actual Results		Pass / Fail / Not executed / Suspended	
1	Create a Painter Object	Painter object created	As Expected		Pass	
2	Create a Parser Object	Parser object created	As Expected		Pass	
3	Use parseEditor() method	Method created example	As Expected		Pass	

Command Test – Complex Expression

Test Case ID	C_09	Test Case Description	Test for expressions			
Created By	Pravesh Pansari			Version	1.1	
Tester's Name	Pravesh Pansari	Date Tested	14th December, 2020	Test Case (Pass/Fail)	Pass	
S #	Prerequisites:		S #	Test Data		
1	Access to Visual Studio 2019		1	parseUsingLexer(input,0) where input is "x = 15"		
2	Access to Command Interface, Command		2	parseUsingLexer(input,0) where input is		
3	Access to Command Factory			"num = 5 + x *2"		
4	Access to Parser,Painter, Lexer classes					
Test Scenario	Verify whether num has been set to 40					
Step #	Step Details	Expected Results	Actual Results		Pass / Fail / Not executed / Suspended	
1	Create a Painter Object	Painter object created	As Expected		Pass	
2	Create a Parser Object	Parser object created	As Expected		Pass	
3	Use parseUsingLexer() method	Variable x created	As Expected		Pass	
4	Use parseUsingLexer() method	Variable num created with value 40	As Expected		Pass	