1/16/2022

GRAPHICAL PROGRAMMING LANGUAGE

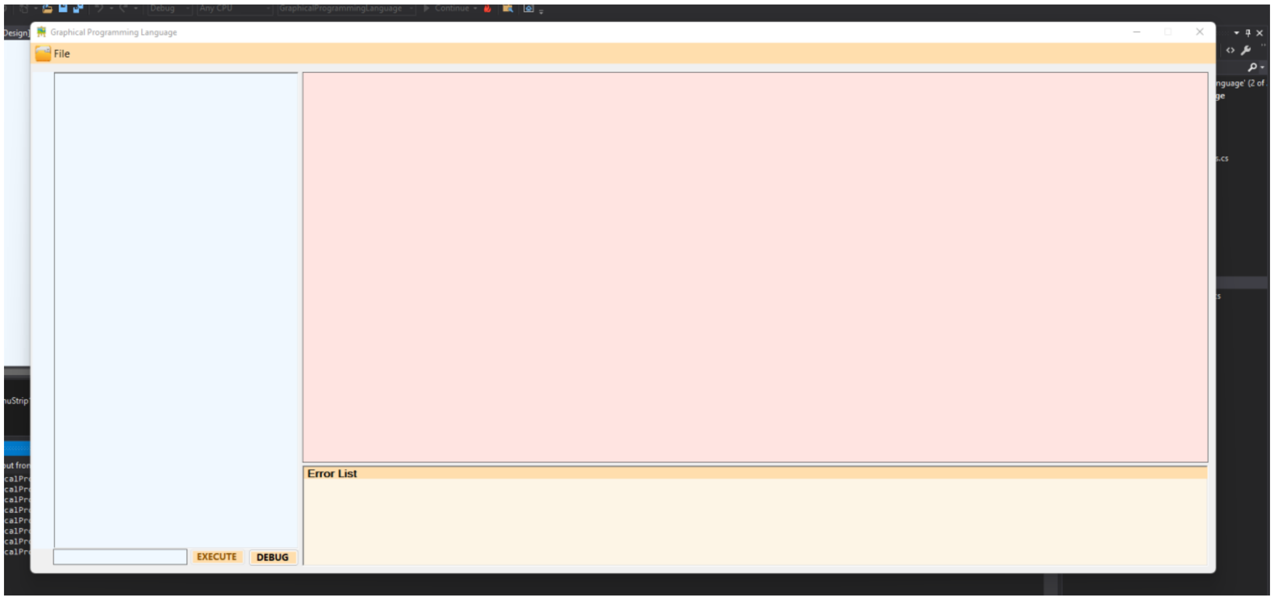
COMPONENT 1 AND 2

saugat thapa
77227277

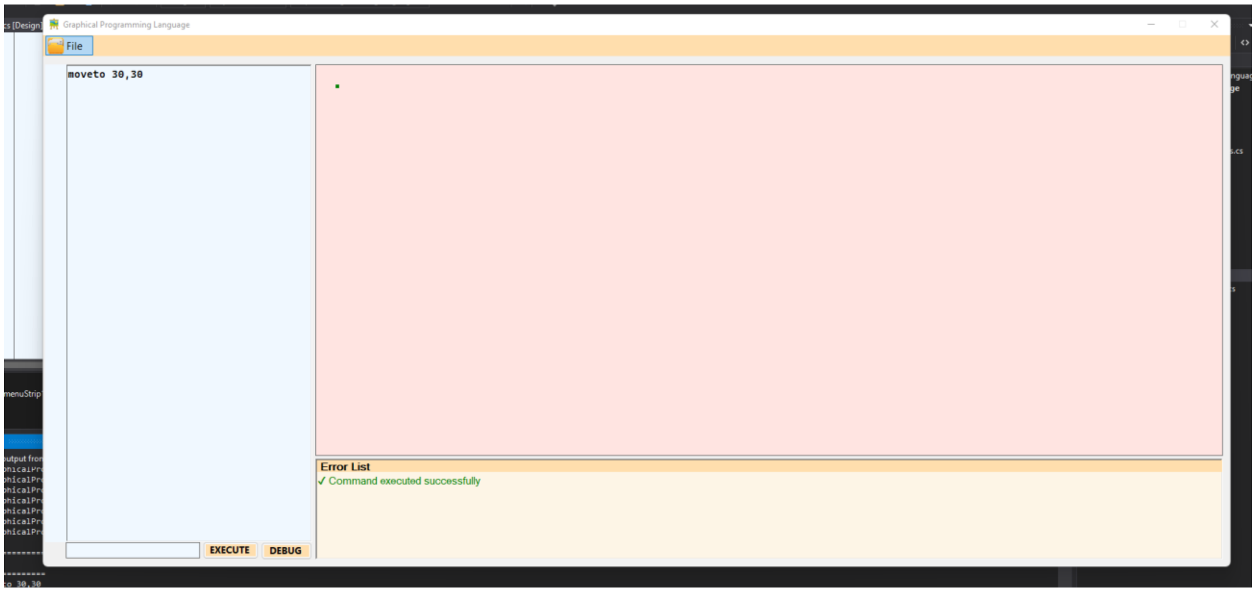
Contents

COMPONENT 1

The Interface



MOVETO

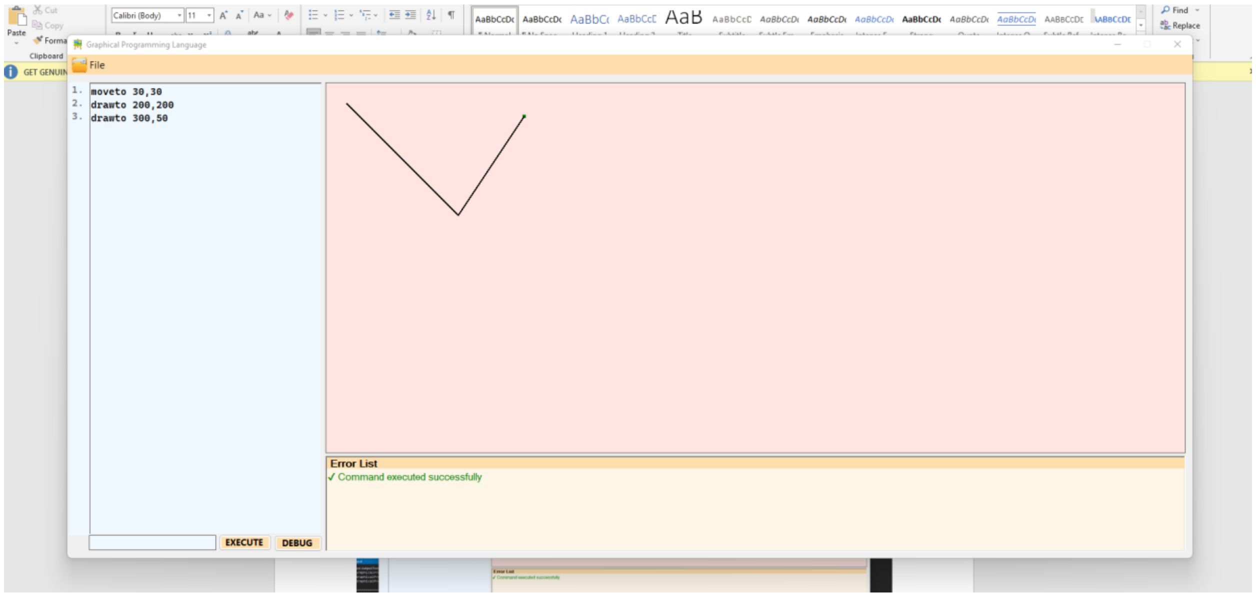


```
moveto 30,30
```

✓ Command executed successfully

DRAWTO

Draws line from previous MOVETO or DRAWTO position

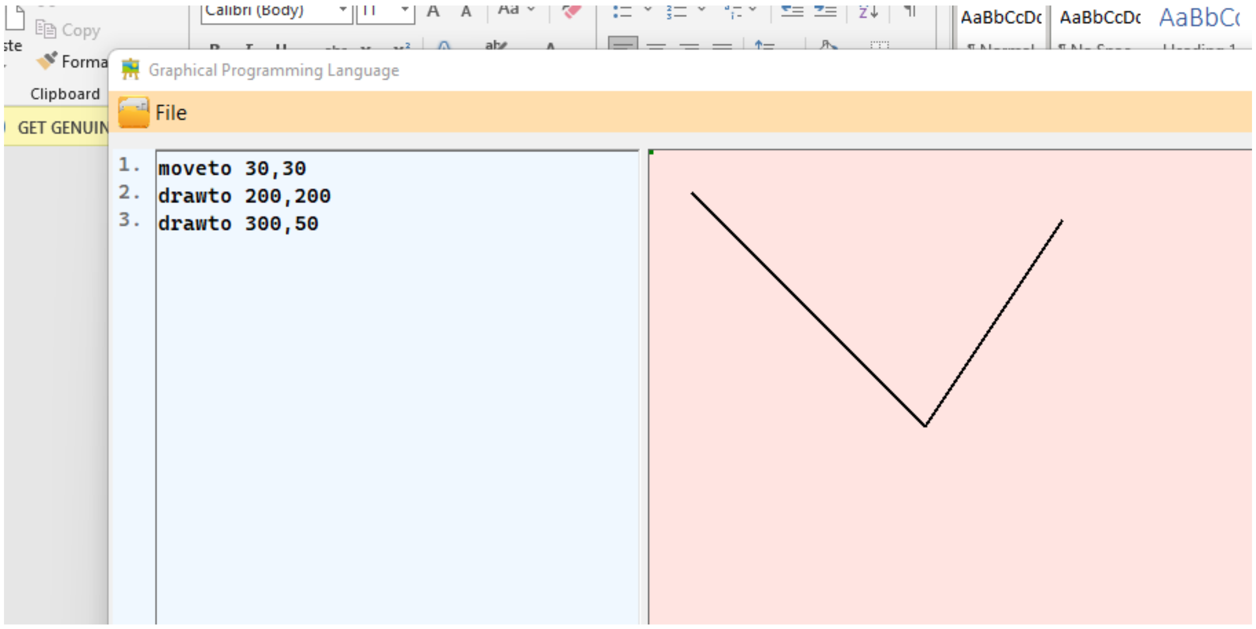


```
moveto 30,30
drawto 200,200
drawto 300,50
```

```
Error List
✓ Command executed successfully
```

RESET

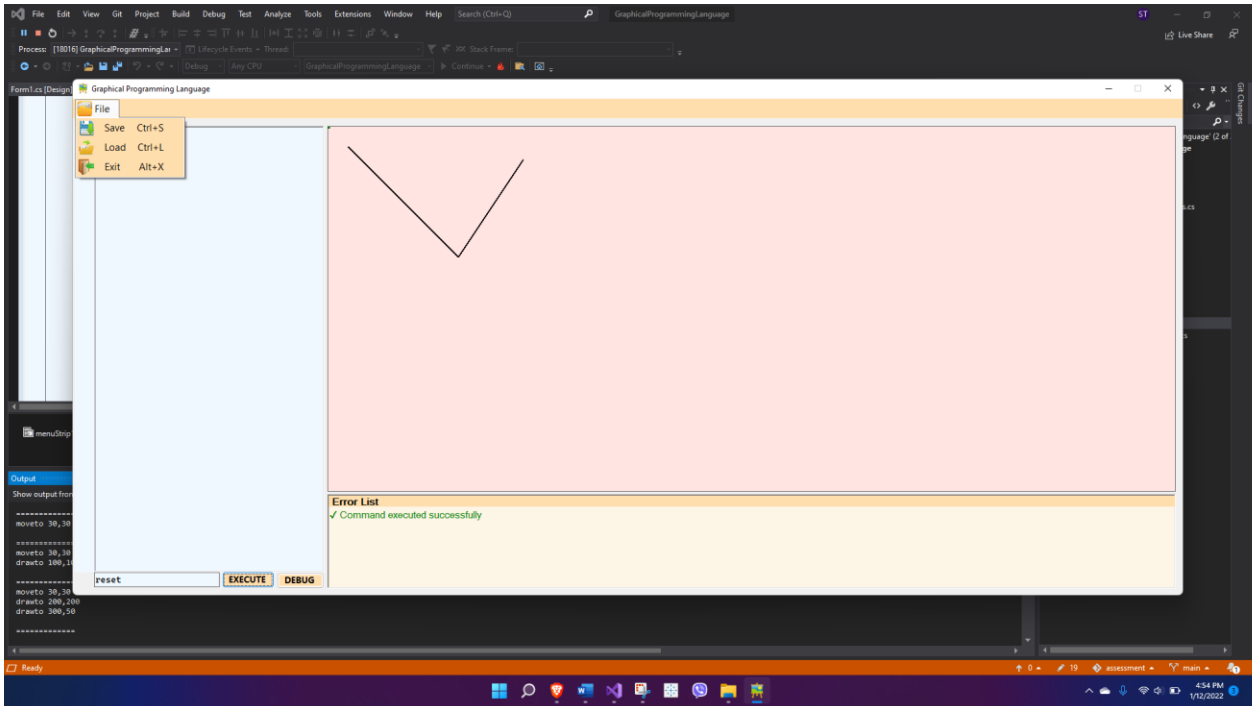
Moves the MOVETO object to top left corner of the screen



```
moveto 30,30
drawto 200,200
drawto 300,50
```

FILE MENU

To SAVE, LOAD and also EXIT the application. Also includes shortcuts for all items. If a file is already loaded on the app, we can save it directly to the same file name in the same location just by Clicking the SAVE menu item. But if no file is loaded , on clicking save it will give u the option to save the file as a new file.

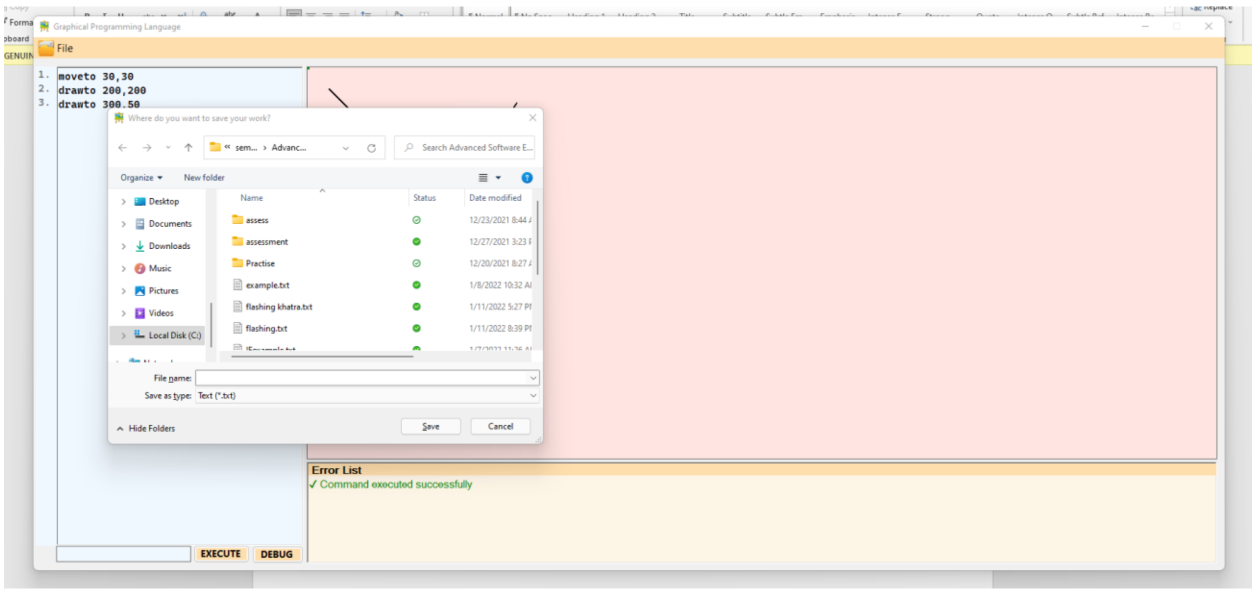


```
reset
```

```
moveto 30,30
moveto 30,30
drawto 100,1
moveto 30,30
drawto 200,200
drawto 300,50
```

```
Error List
✓ Command executed successfully
```

Saving a new file:

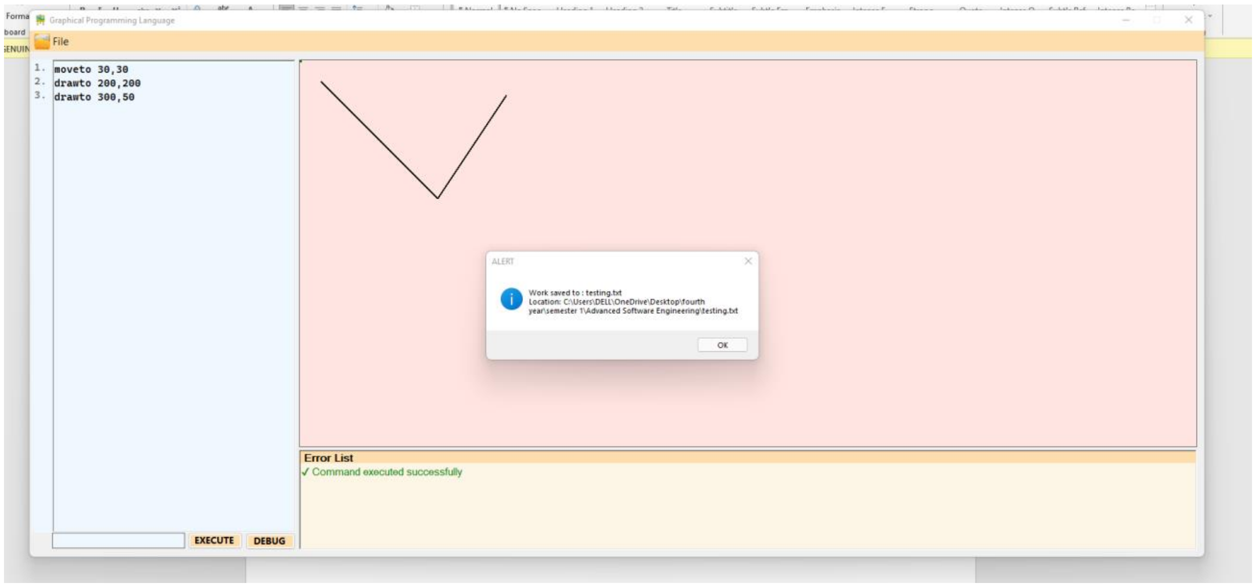


```
moveto 30,30
drawto 200,200
drawto 300,50
```

Name	Date modified
assess	12/23/2021 8:44
assessment	12/27/2021 3:23
Practise	12/20/2021 8:27
example.txt	1/8/2022 10:32
flashing khatra.txt	1/11/2022 5:27
flashing.txt	1/11/2022 8:39

```
Error List
✓ Command executed successfully
```

File saved message along with the location

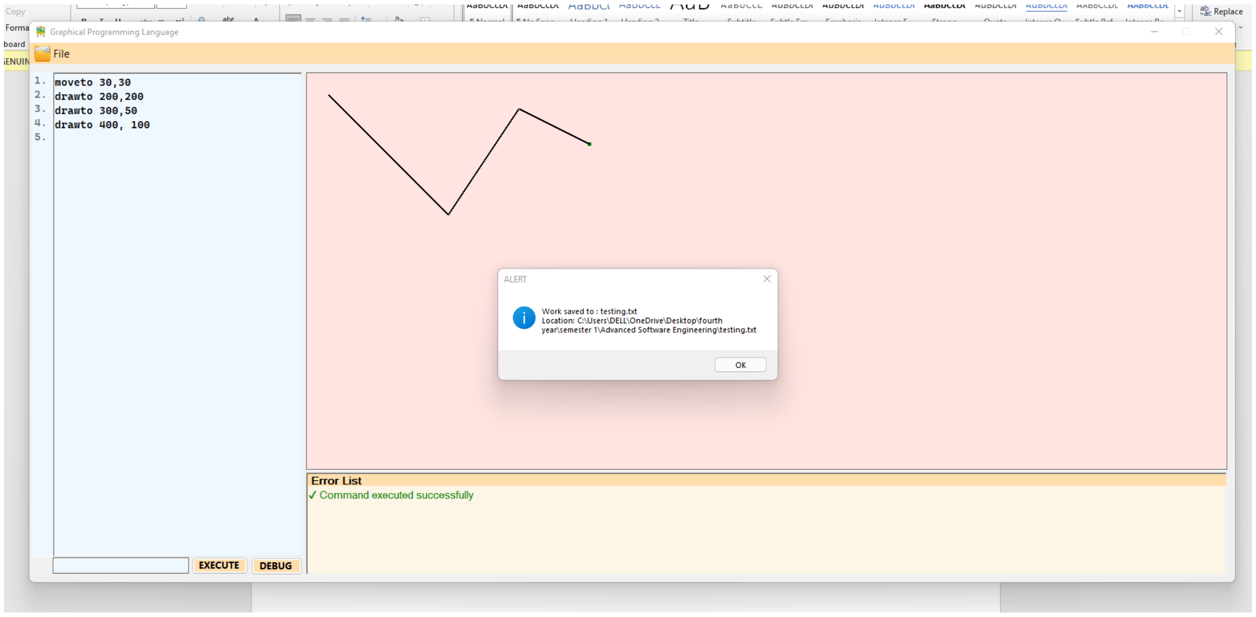


```
moveto 30,30
drawto 200,200
drawto 300,50
```

Work saved to : testing.txt
Location: C:\Users\DELL\OneDrive\Desktop\fourth year\semester 1\Advanced Software Engineering\testing.txt

Error List
✓ Command executed successfully

Saving something new on a preloaded file

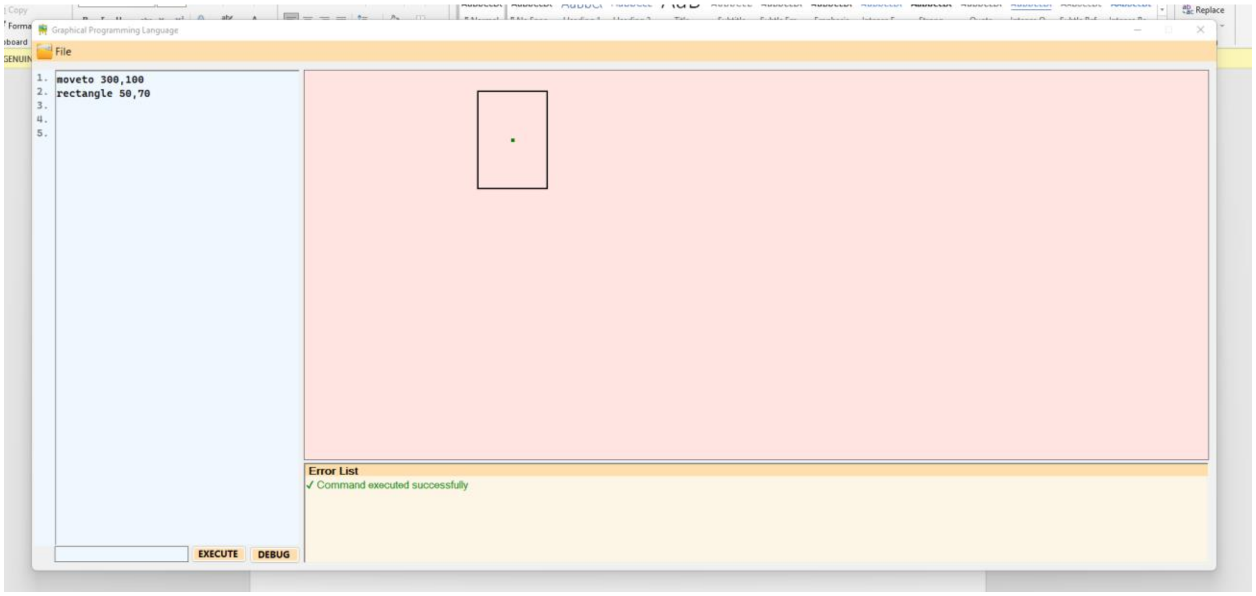


```
moveto 30,30
drawto 200,200
drawto 300,50
drawto 400, 100

```

Work saved to : testing.txt
Location: C:\Users\DELL\OneDrive\Desktop\fourth year\semester 1\Advanced Software Engineering\testing.txt

Error List
✓ Command executed successfully

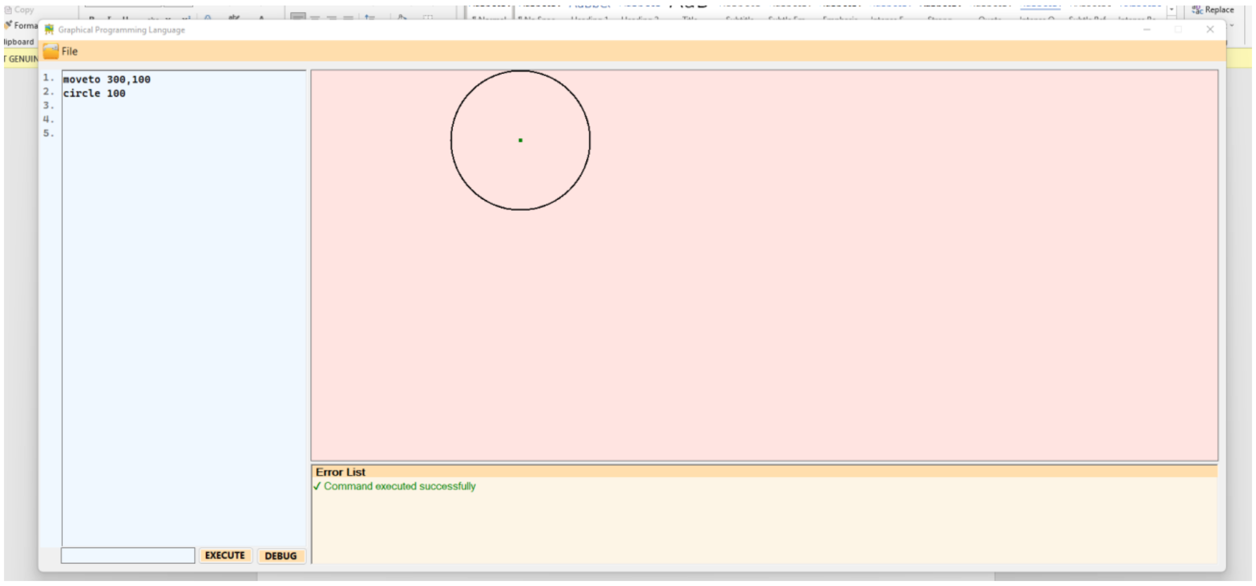
RECTANGLE



```
moveto 300,100
rectangle 50,70
```

```
✓ Command executed successfully
```

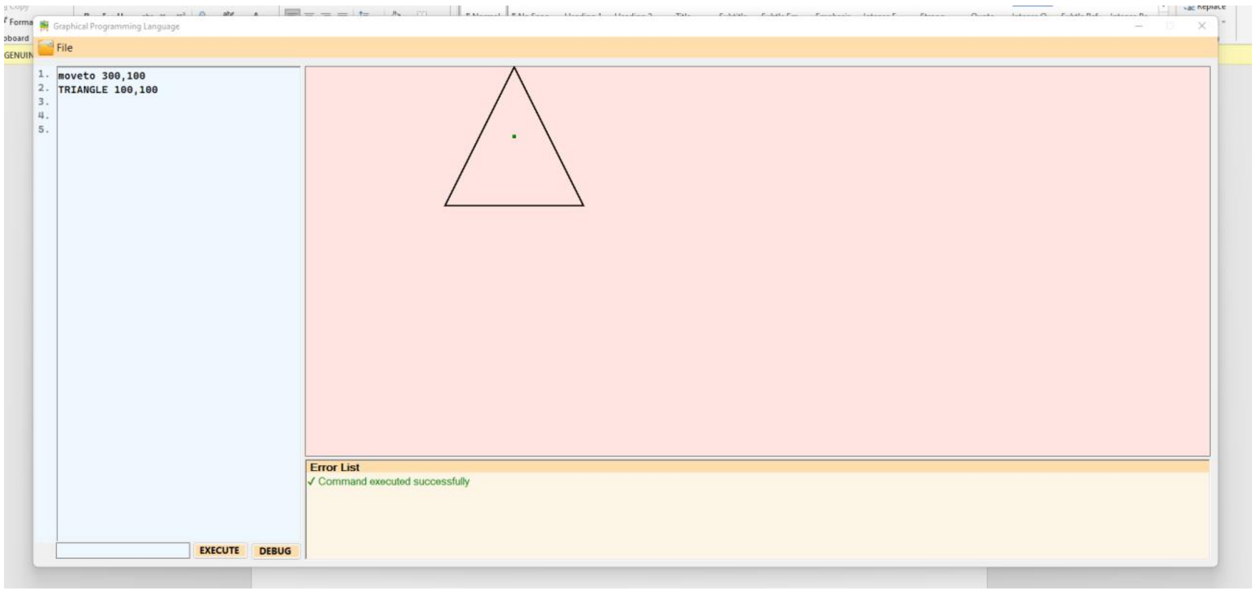
CIRCLE



```
moveto 300,100
circle 100
```

```
✓ Command executed successfully
```

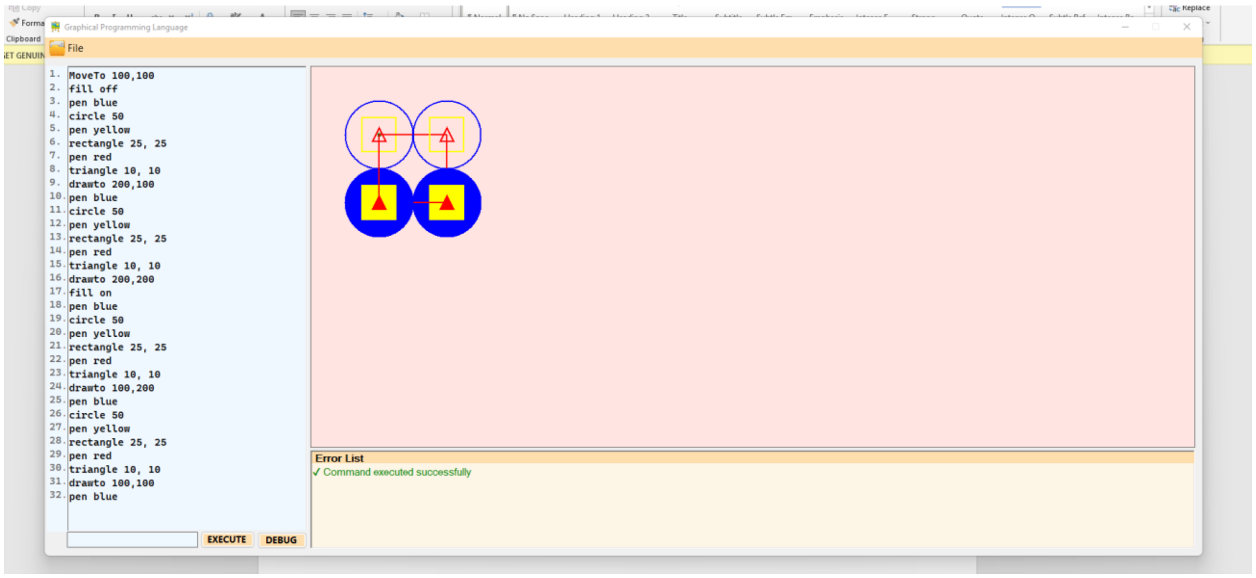
TRIANGLE



```
moveto 300,100
TRIANGLE 100,100
```

```
✓ Command executed successfully
```

PEN COLOR AND FILL

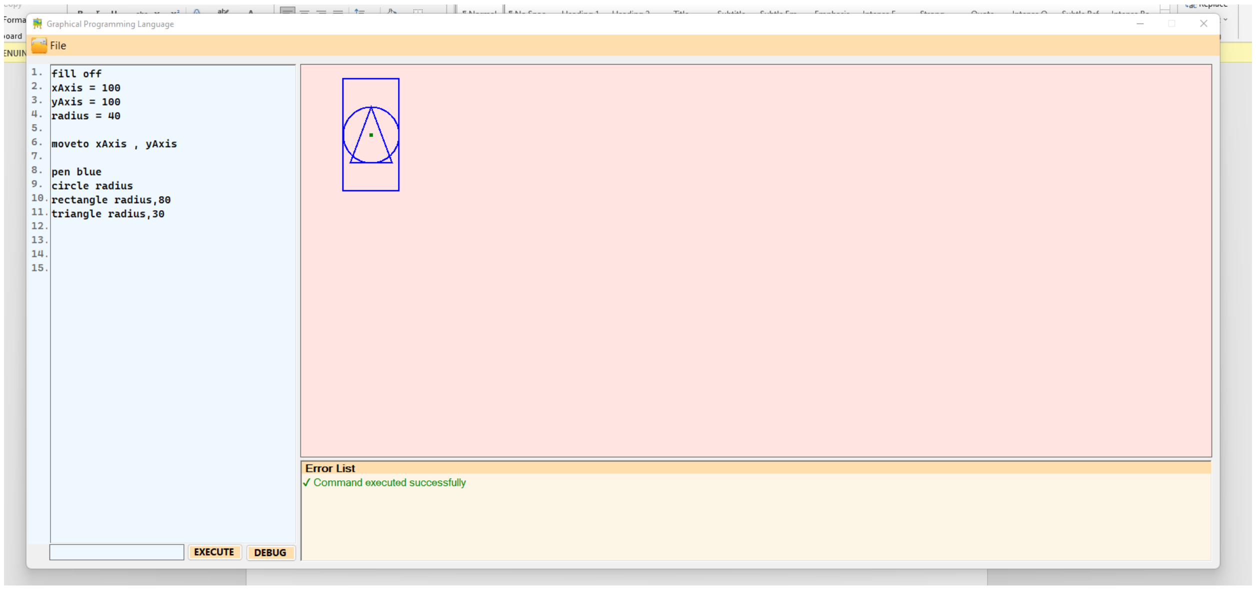


```
MoveTo 100,100
fill off
pen blue
circle 50
pen yellow
rectangle 25, 25
pen red
triangle 10, 10
drawto 200,100
pen blue
circle 50
pen yellow
rectangle 25, 25
pen red
triangle 10, 10
drawto 200,200
fill on
pen blue
circle 50
pen yellow
rectangle 25, 25
pen red
triangle 10, 10
drawto 100,200
pen blue
circle 50
pen yellow
rectangle 25, 25
pen red
triangle 10, 10
drawto 100,100
pen blue
```

```
✓ Command executed successfully
```

COMPONENT 2

VARIABLES

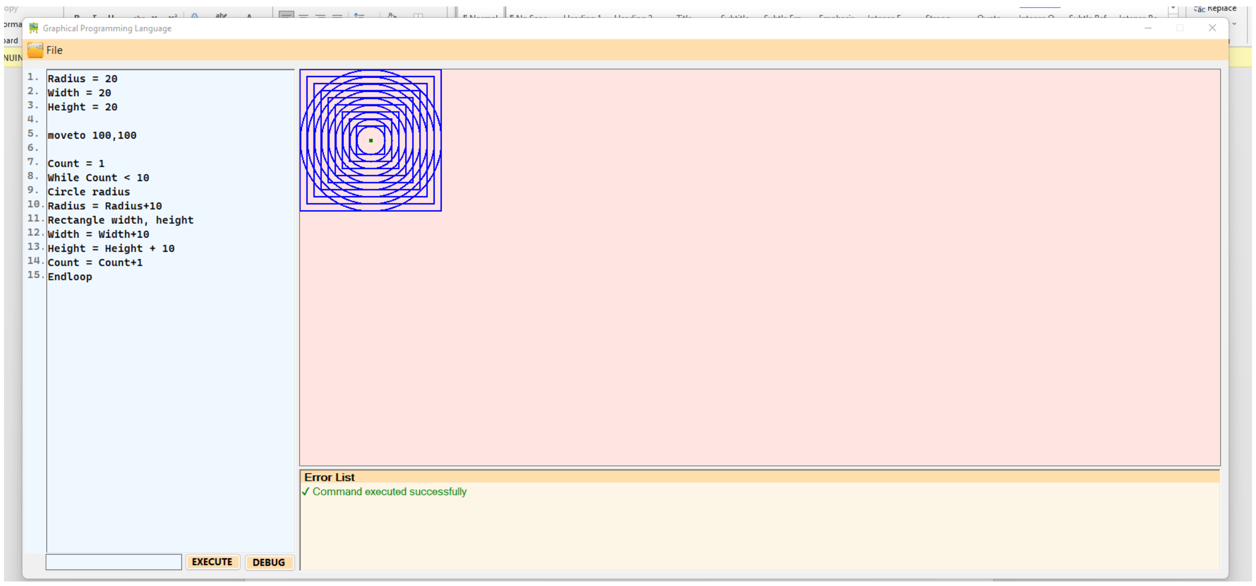


```
fill off
xAxis = 100
yAxis = 100
radius = 40

moveto xAxis , yAxis

pen blue
circle radius
rectangle radius,80
triangle radius,30
```

Error List

✓ Command executed successfully

LOOP

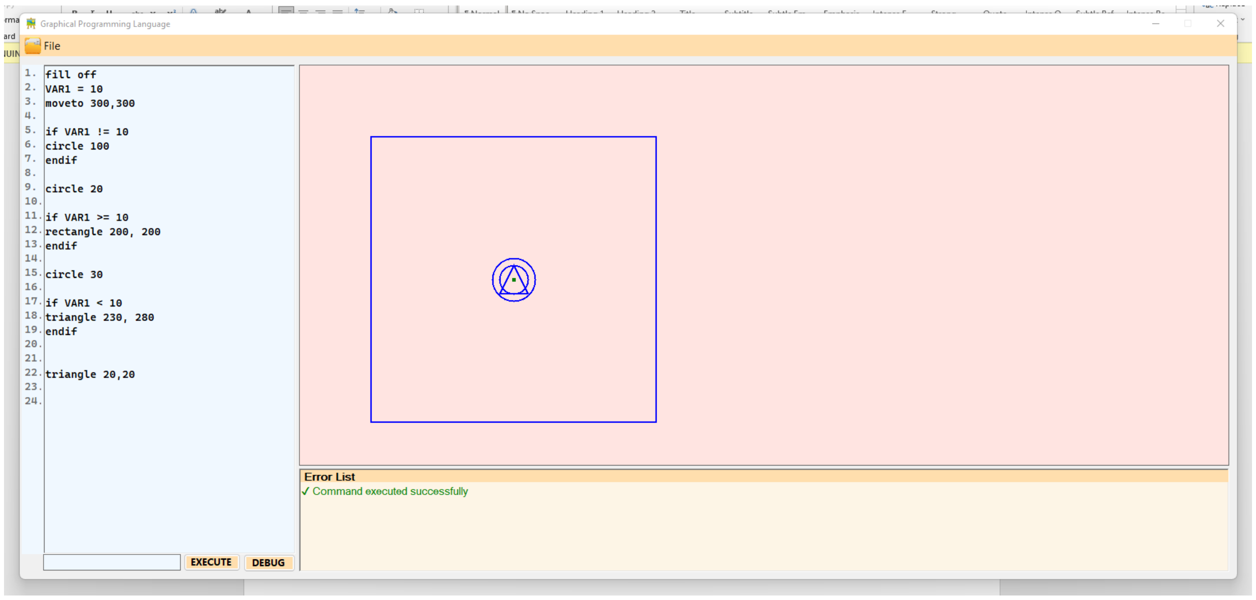


```
Radius = 20
Width = 20
Height = 20

moveto 100,100

Count = 1
While Count < 10
Circle radius
Radius = Radius+10
Rectangle width, height
Width = Width+10
Height = Height + 10
Count = Count+1
Endloop
```

Error List

✓ Command executed successfully

IF STATEMENT with ENDIF block

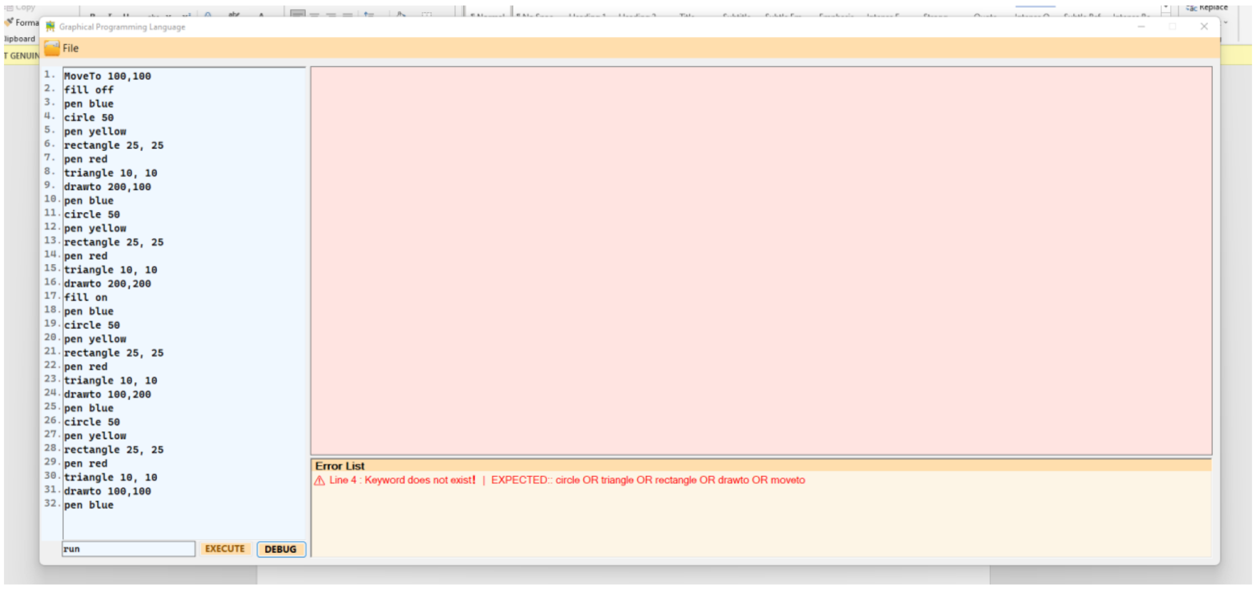


```
fill off
VAR1 = 10
moveto 300,300

if VAR1 != 10
circle 100
endif

circle 20

if VAR1 >= 10
rectangle 200, 200
endif

circle 30

if VAR1 < 10
triangle 230, 280
endif


triangle 20,20
```

```
Error List
✓ Command executed successfully
```

Syntax Checking Before the program is run

As part of the requirement the EXECUTE button remains disabled until all the errors are solved, which can be viewed by pressing the DEBUG button. If an error is found later then the EXECUTE button is again disabled

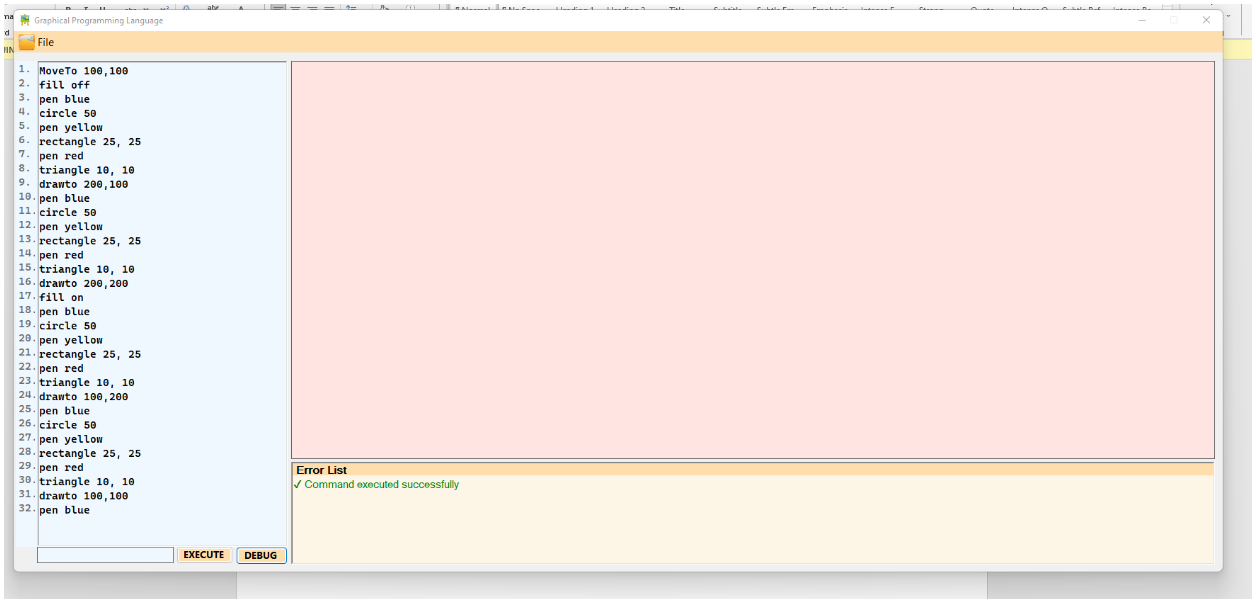


```
1.  MoveTo 100,100
2.  fill off
3.  pen blue
4.  cirle 50
5.  pen yellow
6.  rectangle 25, 25
7.  pen red
8.  triangle 10, 10
9.  drawto 200,100
10. pen blue
11. circle 50
12. pen yellow
13. rectangle 25, 25
14. pen red
15. triangle 10, 10
16. drawto 200,200
17. fill on
18. pen blue
19. circle 50
20. pen yellow
21. rectangle 25, 25
22. pen red
23. triangle 10, 10
24. drawto 100,200
25. pen blue
26. circle 50
27. pen yellow
28. rectangle 25, 25
29. pen red
30. triangle 10, 10
31. drawto 100,100
32. pen blue
```

```
run
```

```
Error List
⚠ Line 4 : Keyword does not exist! | EXPECTED:: circle OR triangle OR rectangle OR drawto OR moveto
```

EXECUTE button reenabled once all errors are fixed

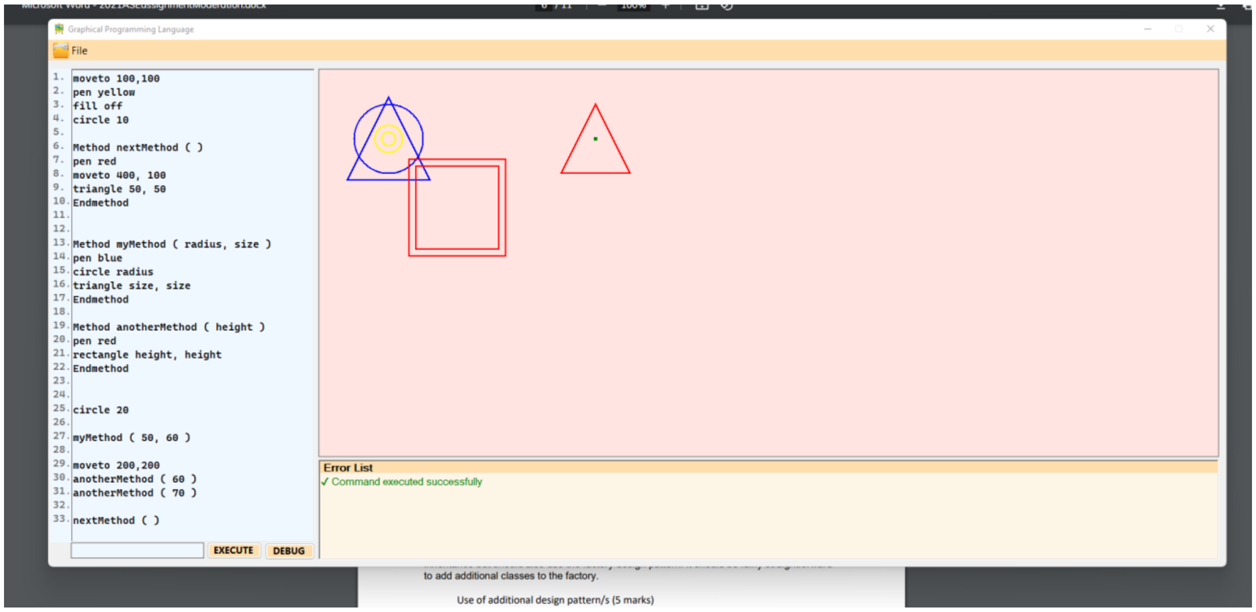


```
MoveTo 100,100
fill off
pen blue
circle 50
pen yellow
rectangle 25, 25
pen red
triangle 10, 10
drawto 200,100
pen blue
circle 50
pen yellow
rectangle 25, 25
pen red
triangle 10, 10
drawto 200,200
fill on
pen blue
circle 50
pen yellow
rectangle 25, 25
pen red
triangle 10, 10
drawto 100,200
pen blue
circle 50
pen yellow
rectangle 25, 25
pen red
triangle 10, 10
drawto 100,100
pen blue
```

```
Error List
✓ Command executed successfully
```

METHODS

Works with parameters and without. Multiple calls can be made to the same method. Also works with multiple parameters.



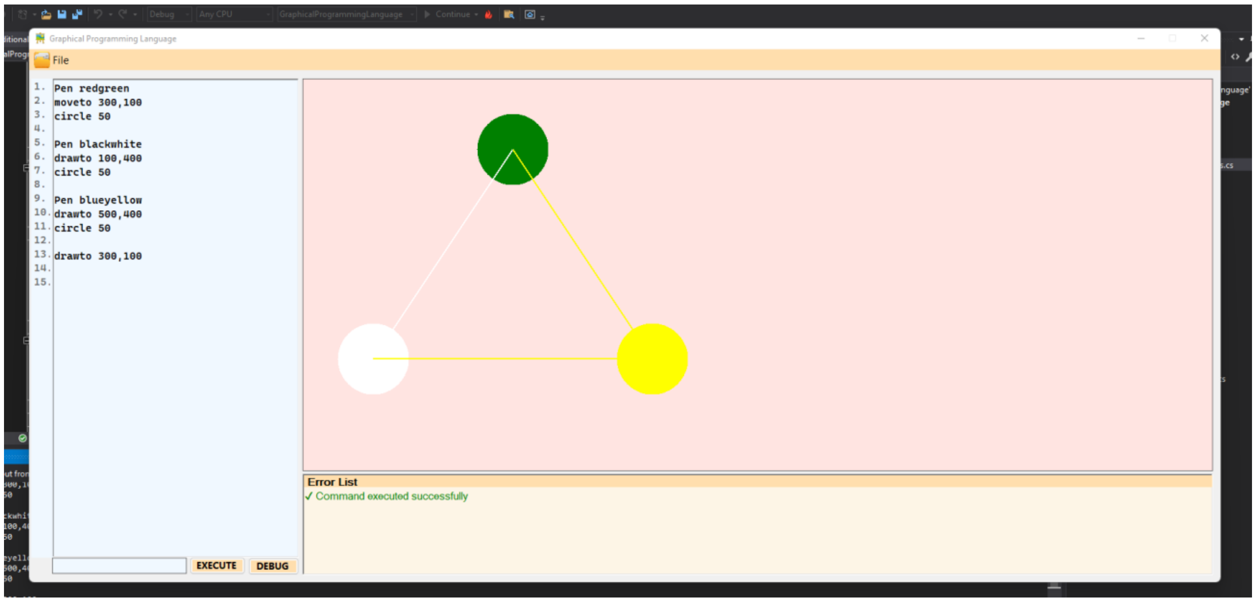
```
moveto 100,100
pen yellow
fill off
circle 10

Method nextMethod ( )
pen red
moveto 400, 100
triangle 50, 50
Endmethod


Method myMethod ( radius, size )
pen blue
circle radius
triangle size, size
Endmethod

Method anotherMethod ( height )
pen red
rectangle height, height
Endmethod


circle 20

myMethod ( 50, 60 )

moveto 200,200
anotherMethod ( 60 )
anotherMethod ( 70 )

nextMethod ( )
```

```
Error List
✓ Command executed successfully
```

to add additional classes to the factory.

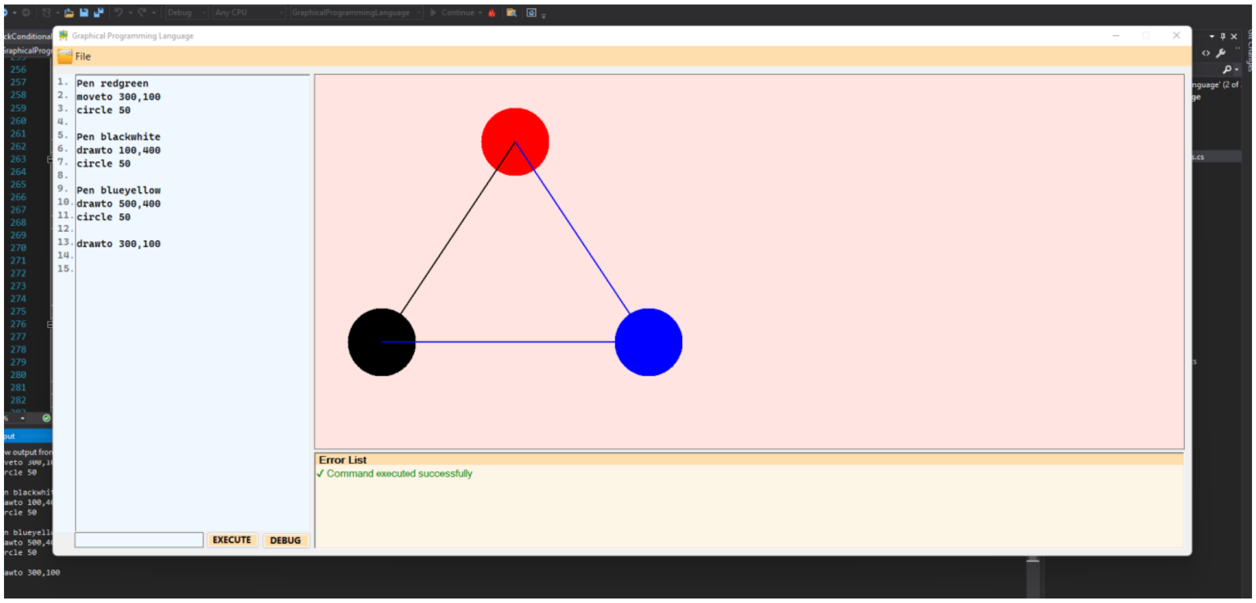
Use of additional design pattern/s (5 marks)

FLASHING COLOR



```
Pen redgreen
moveto 300,100
circle 50

Pen blackwhite
drawto 100,400
circle 50

Pen blueyellow
drawto 500,400
circle 50

drawto 300,100
```

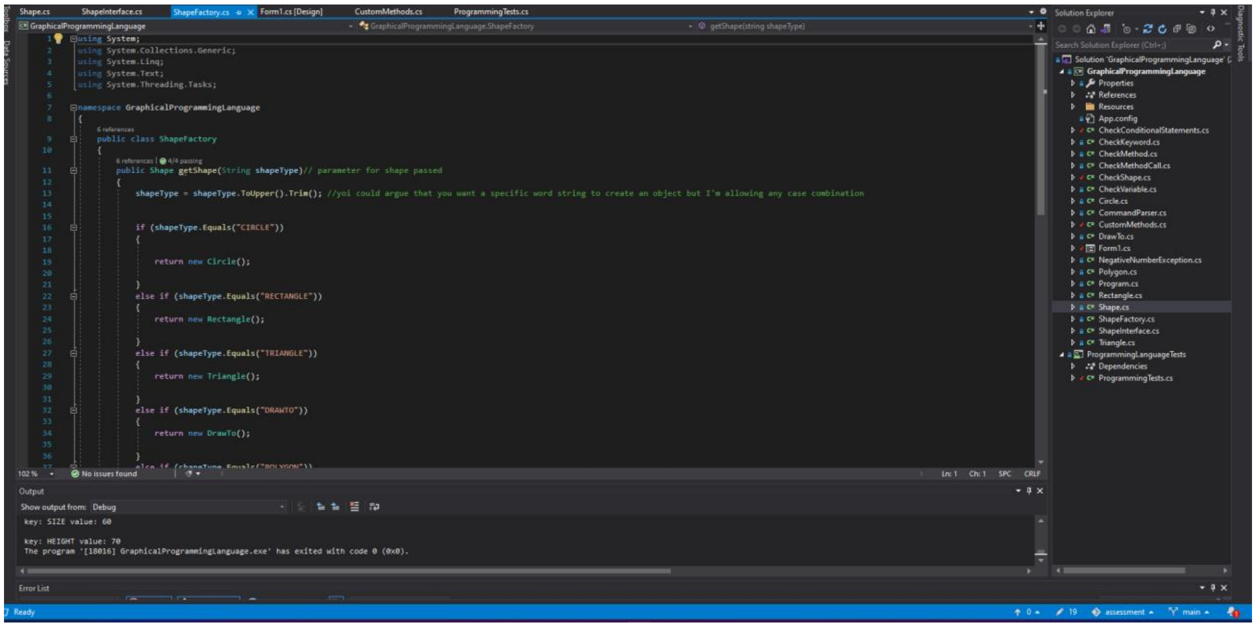
Error List
✓ Command executed successfully



```
Pen redgreen
moveto 300,100
circle 50

Pen blackwhite
drawto 100,400
circle 50

Pen blueyellow
drawto 500,400
circle 50

drawto 300,100
```

Error List
✓ Command executed successfully

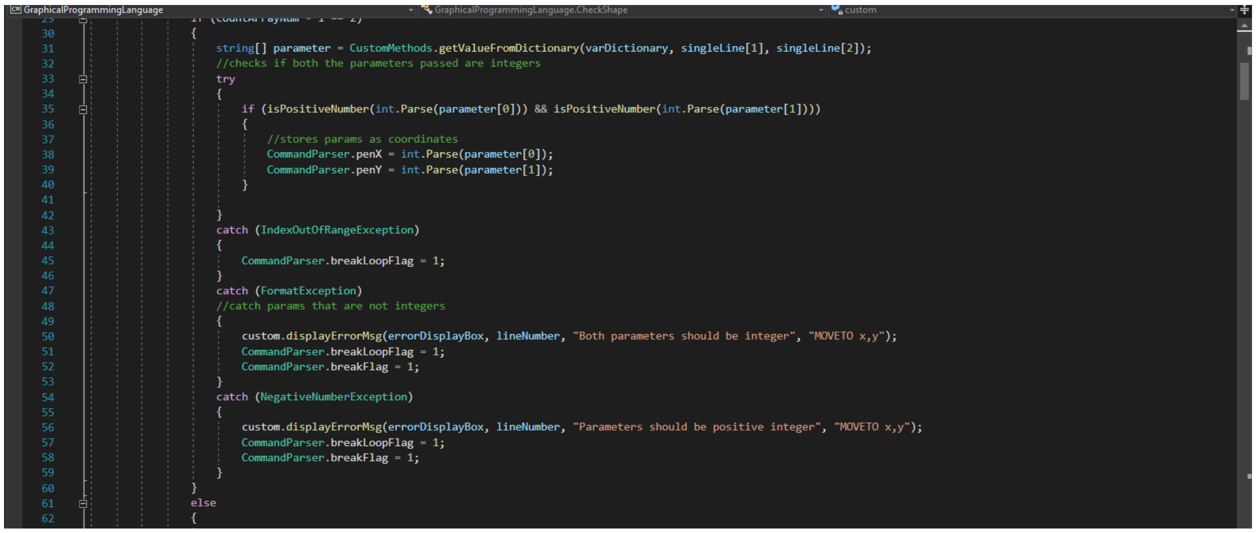
Use of FACTORY DESIGN PATTERN



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GraphicalProgrammingLanguage
{
    public class ShapeFactory
    {
        public Shape getShape(String shapeType)// parameter for shape passed
        {
            shapeType = shapeType.ToUpper().Trim(); //yoi could argue that you want a specific word string to create an object but I'm allowing any case combination


            if (shapeType.Equals("CIRCLE"))
            {

                return new Circle();

            }
            else if (shapeType.Equals("RECTANGLE"))
            {
                return new Rectangle();

            }
            else if (shapeType.Equals("TRIANGLE"))
            {
                return new Triangle();

            }
            else if (shapeType.Equals("DRAWTO"))
            {
                return new DrawTo();

            }
            else if (shapeType.Equals("POLYGON"))
```

```
key: SIZE value: 60

key: HEIGHT value: 70
The program '[18016] GraphicalProgrammingLanguage.exe' has exited with code 0 (0x0).
```

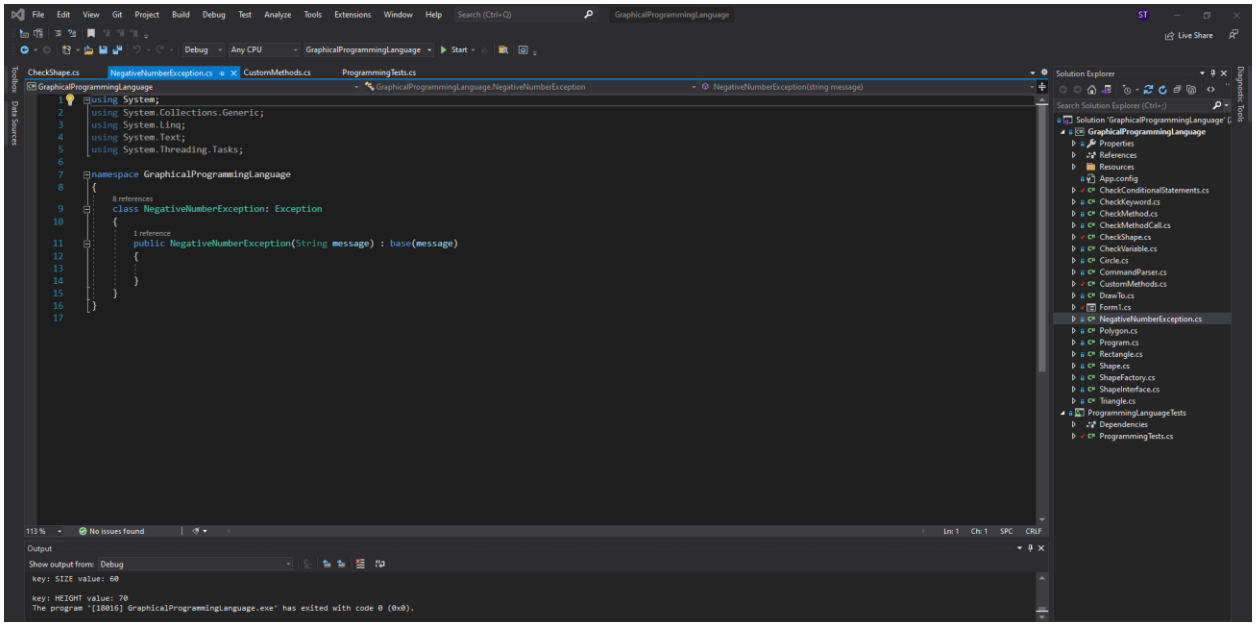
Use of EXCEPTION handling



```
            {
                string[] parameter = CustomMethods.getValueFromDictionary(varDictionary, singleLine[1], singleLine[2]);
                //checks if both the parameters passed are integers
                try
                {
                    if (isPositiveNumber(int.Parse(parameter[0])) && isPositiveNumber(int.Parse(parameter[1])))
                    {
                        //stores params as coordinates
                        CommandParser.penX = int.Parse(parameter[0]);
                        CommandParser.penY = int.Parse(parameter[1]);
                    }

                }
                catch (IndexOutOfRangeException)
                {
                    CommandParser.breakLoopFlag = 1;
                }
                catch (FormatException)
                //catch params that are not integers
                {
                    custom.displayErrorMsg(errorDisplayBox, lineNumber, "Both parameters should be integer", "MOVETO x,y");
                    CommandParser.breakLoopFlag = 1;
                    CommandParser.breakFlag = 1;
                }
                catch (NegativeNumberException)
                {
                    custom.displayErrorMsg(errorDisplayBox, lineNumber, "Parameters should be positive integer", "MOVETO x,y");
                    CommandParser.breakLoopFlag = 1;
                    CommandParser.breakFlag = 1;
                }
            }
            else
            {
```

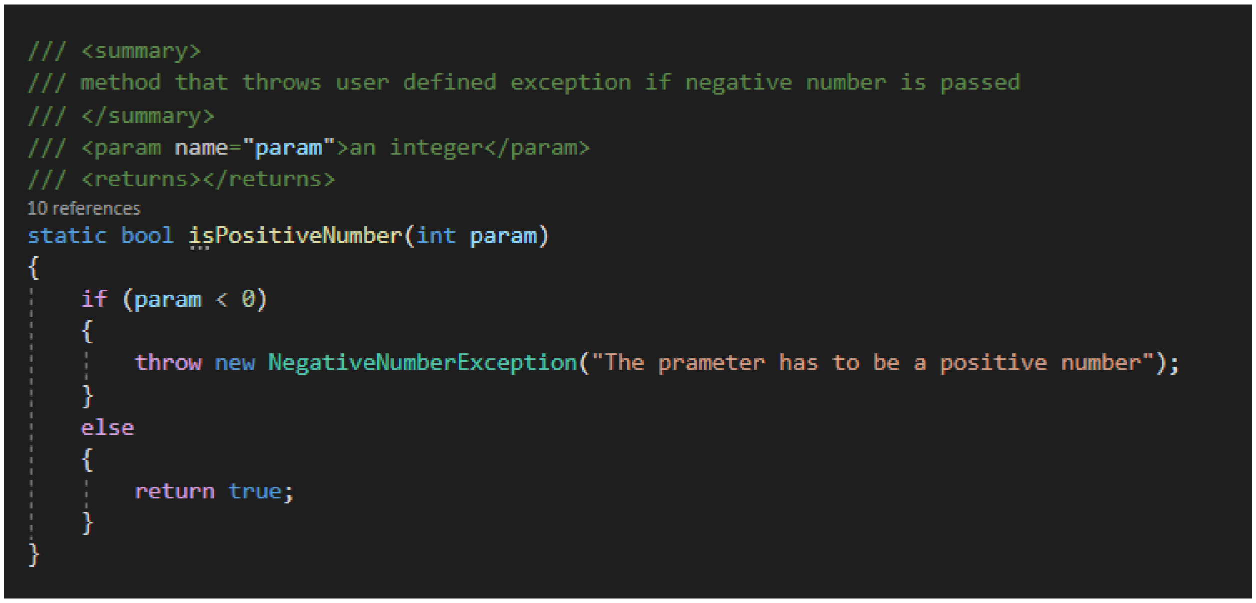
Use of user generated exceptions

Throws exceptions for negative numbers



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GraphicalProgrammingLanguage
{
    8 references
    class NegativeNumberException: Exception
    {
        1 reference
        public NegativeNumberException(String message) : base(message)
        {

        }
    }
}
```

```
key: SIZE value: 60

key: HEIGHT value: 70
The program '[18016] GraphicalProgrammingLanguage.exe' has exited with code 0 (0x0).
```

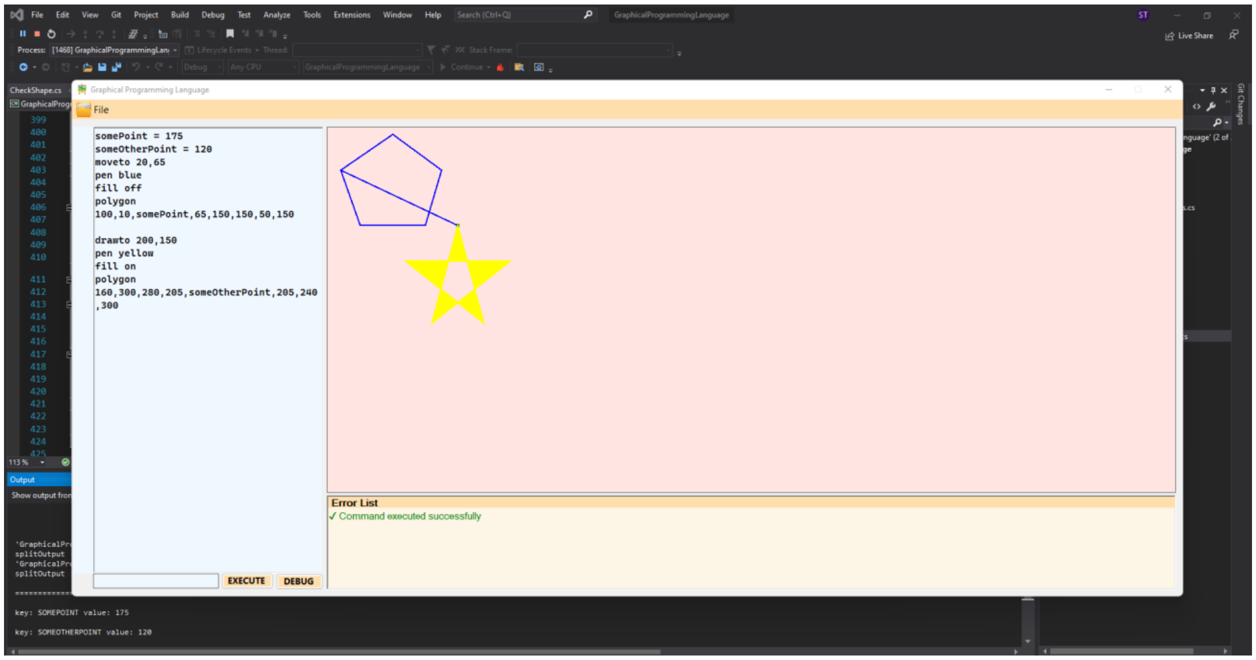


```
/// <summary>
/// method that throws user defined exception if negative number is passed
/// </summary>
/// <param name="param">an integer</param>
/// <returns></returns>
10 references
static bool isPositiveNumber(int param)
{
    if (param < 0)
    {
        throw new NegativeNumberException("The prameter has to be a positive number");
    }
    else
    {
        return true;
    }
}
```

ADDITIONAL FUNCTIONALITY

Complex Shape (POLYGON)

Takes pairs of X and Y coordinates to draw the shapes



```
somePoint = 175
someOtherPoint = 120
moveto 20,65
pen blue
fill off
polygon
100,10,somePoint,65,150,150,50,150

drawto 200,150
pen yellow
fill on
polygon
160,300,280,205,someOtherPoint,205,240
,300
```

✓ Command executed successfully

```
key: SOMEPOINT value: 175

key: SOMEOTHERPOINT value: 120
```

Testing

For Component 1

Test for `ShapeFactory_Return_Shape()`

Input	Expected	Actual	Result
Circe	ArgumentException	ArgumentException	Pass
Circle	No Exception is thrown	ArgumentException	Fail

Test for `Check_if_Possible_Command ()`

Input	Expected	Actual	Result
moveto	`possibleCommands` contains input	`possibleCommands` contains input	Pass
forloop	`possibleCommands` contains input	`possibleCommands` does not contains input	Fail

Test for `Set_Rectangle_Width_Height()`

Input	Expected	Actual	Result

`50, 60`	rect.width = 50
	rect.height = 60
		rect.width = 50
		rect.height = 60
			Pass

50	rect.width = 50 rect.height =	Throws System.IndexOutOfRangeException	Fail

Test for `Return_Circle ()`

Input	Expected	Actual	Result
`circle`	`IsTrue`	IsTrue	Pass
`triangle`	`IsTrue`	`IsFalse`	Fail

Test for `Return_Triangle ()`

Input	Expected	Actual	Result
`triangle`	`IsTrue`	`IsTrue`	Pass
`rectangle`	`IsTrue`	`IsFalse`	Fail

For Component 2

Test for `Check_if_Valid_Variable ()`

Input	Expected	Actual	Result
VAR2	30	30	Pass
VAR3	30	40	Fail

Test for `Compare_Operands ()`

Input	Expected	Actual	Result

`num1 = 20;`
`opperator = "<";`
`num2 = 100;`
	IsTrue	IsTrue	Pass

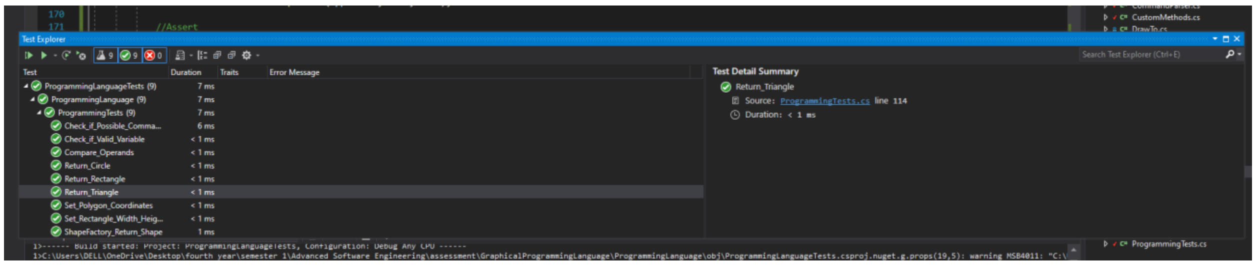
`num1 = 100;`
`opperator = "==";`
`num2 = 100;`
	IsTrue	IsFalse	Fail

Test for `Set_Polygon_Coordinates()`

Input	Expected	Actual	Result
`int[] polyArray = new int[] { 20, 65, 100, 10 }`	poly.polyArray = Shape.polyArray	poly.polyArray = Shape.polyArray	Pass

Test for `Test_User_Generated_Exception()`

Input	Expected	Actual	Result
-20	NegativeNumberException	NegativeNumberException	Pass
20	NegativeNumberException	No exception thrown	Fail



```
//Assert
```

Test Explorer

Test	Duration	Traits	Error Message
ProgrammingLanguageTests (9)	7 ms		
ProgrammingLanguage (9)	7 ms		
ProgrammingTests (9)	7 ms		
Check_if_Possible_Comma...	6 ms		
Check_if_Valid_Variable	< 1 ms		
Compare_Operands	< 1 ms		
Return_Circle	< 1 ms		
Return_Rectangle	< 1 ms		
Return_Triangle	< 1 ms		
Set_Polygon_Coordinates	< 1 ms		
Set_Rectangle_Width_Heig...	< 1 ms		
ShapeFactory_Return_Shape	1 ms		

Test Detail Summary

Return_Triangle

Source: ProgrammingTests.cs line 114

Duration: < 1 ms

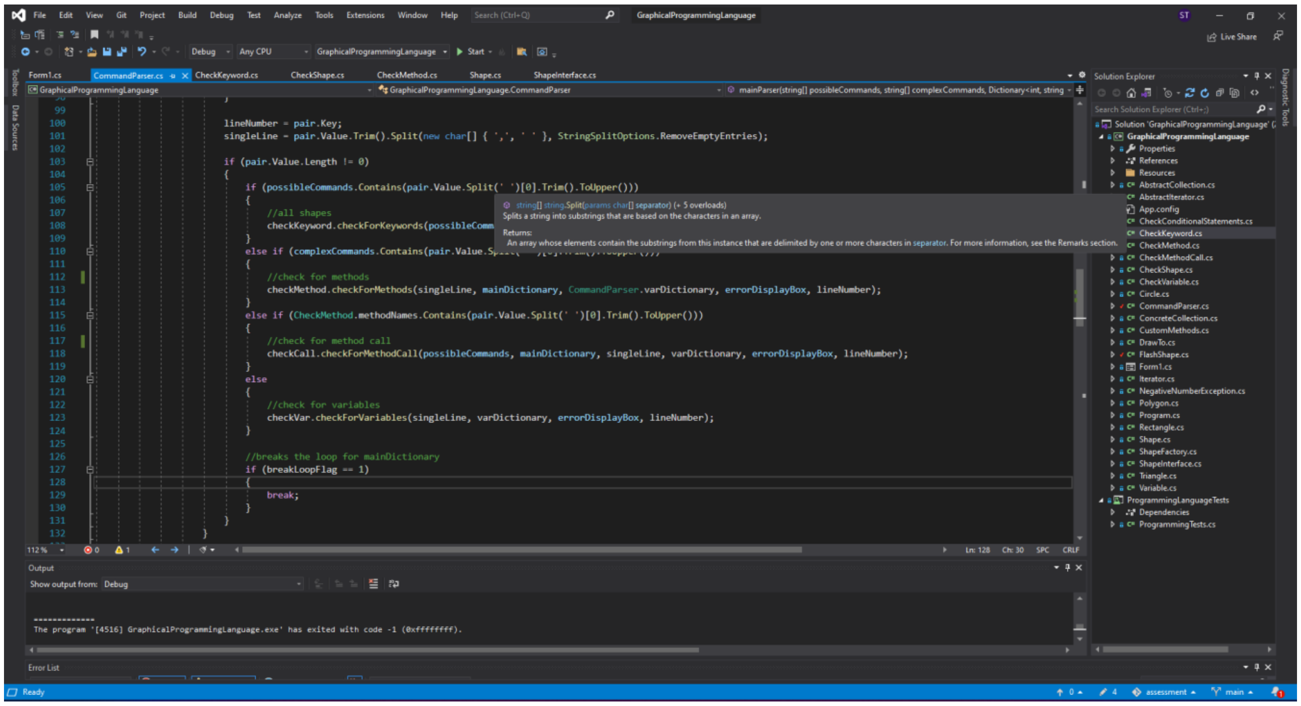
```
1>------ Build started: Project: ProgrammingLanguageTests, Configuration: Debug Any CPU ------
1>C:\Users\DELL\OneDrive\Desktop\fourth year\semester 1\Advanced Software Engineering\assessment\GraphicalProgrammingLanguage\ProgrammingLanguage\obj\ProgrammingLanguageTests.csproj.nuget.g.props(19,5): warning MSB4011: "C:\
```

ADDITIONAL DESIGN PATTERNS

Façade Design Pattern

The class Command Parser acts as a façade class as the method mainParser() inside the class, when called in Form1, checks for all the commands the user may have entered like METHODS, METHODCALLS, VARIABLES, SHAPES and so on as show below.

Since a single method i.e. mainParser is making all these calls but making it seem like only one method has been called, we can say that Façade design pattern has been implemented.



```
lineNumber = pair.Key;
singleLine = pair.Value.Trim().Split(new char[] { ',', ' ' }, StringSplitOptions.RemoveEmptyEntries);

if (pair.Value.Length != 0)
{
    if (possibleCommands.Contains(pair.Value.Split(' ')[0].Trim().ToUpper()))
    {
        //all shapes
        checkKeyword.checkForKeywords(possibleComm
    }
    else if (complexCommands.Contains(pair.Value.S
    {
        //check for methods
        checkMethod.checkForMethods(singleLine, mainDictionary, CommandParser.varDictionary, errorDisplayBox, lineNumber);
    }
    else if (CheckMethod.methodNames.Contains(pair.Value.Split(' ')[0].Trim().ToUpper()))
    {
        //check for method call
        checkCall.checkForMethodCall(possibleCommands, mainDictionary, singleLine, varDictionary, errorDisplayBox, lineNumber);
    }
    else
    {
        //check for variables
        checkVar.checkForVariables(singleLine, varDictionary, errorDisplayBox, lineNumber);
    }

    //breaks the loop for mainDictionary
    if (breakLoopFlag == 1)
    {
        break;
    }
}
}
```

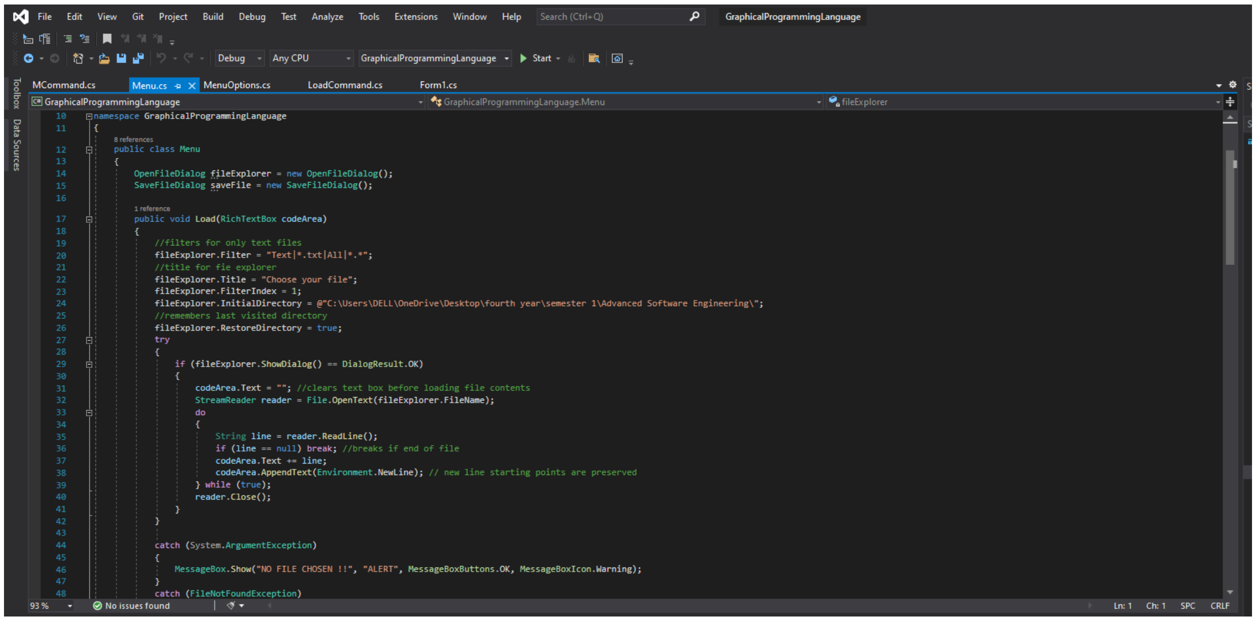
string[] string.Split(params char[] separator) (+ 5 overloads)
Splits a string into substrings that are based on the characters in an array.
Returns:
An array whose elements contain the substrings from this instance that are delimited by one or more characters in separator. For more information, see the Remarks section.

```
=============
The program '[4516] GraphicalProgrammingLanguage.exe' has exited with code -1 (0xffffffff).
```

Command Design Pattern

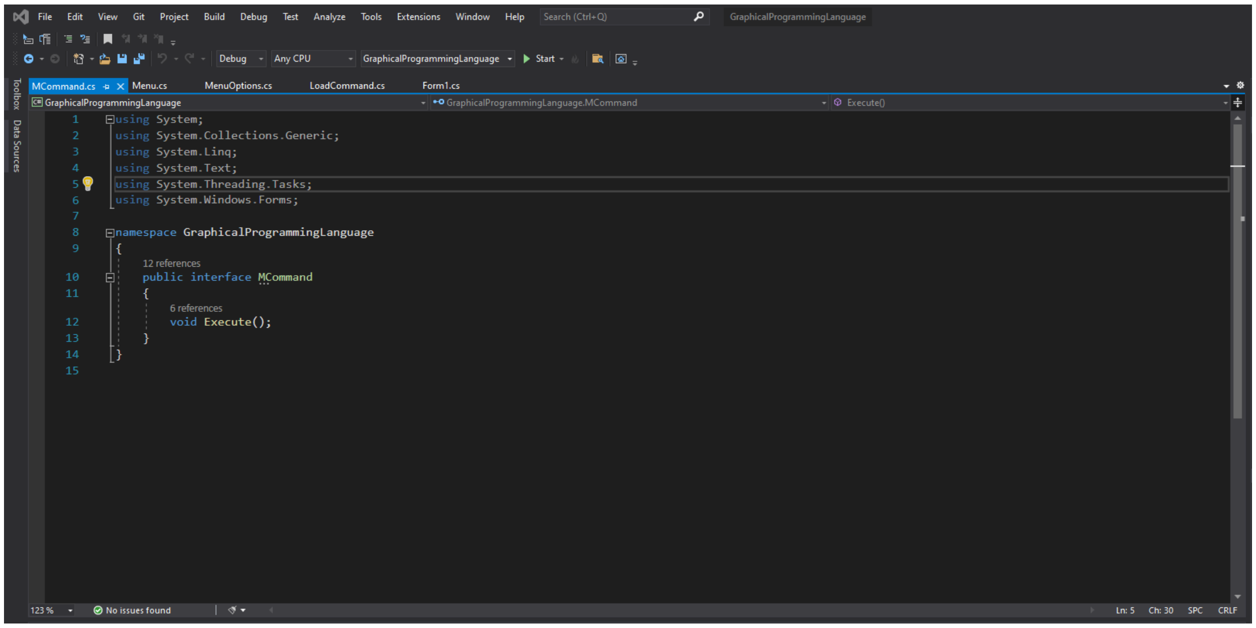
The Command Design Pattern was use to make execution commands for the menu options which includes Save, Close and Load commands for a file.

Creating the receiver object



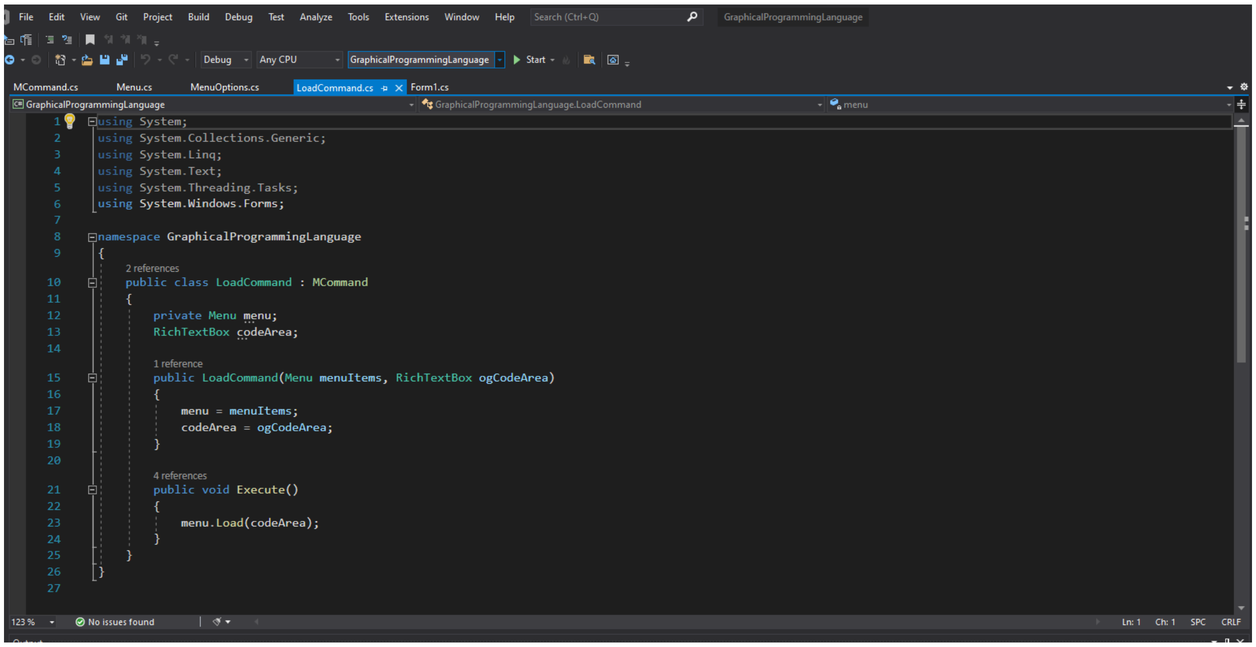
```
namespace GraphicalProgrammingLanguage
{
    8 references
    public class Menu
    {
        OpenFileDialog fileExplorer = new OpenFileDialog();
        SaveFileDialog saveFile = new SaveFileDialog();

        1 reference
        public void Load(RichTextBox codeArea)
        {
            //filters for only text files
            fileExplorer.Filter = "Text|*.txt|All|*.*";
            //title for fie explorer
            fileExplorer.Title = "Choose your file";
            fileExplorer.FilterIndex = 1;
            fileExplorer.InitialDirectory = @"C:\Users\DELL\OneDrive\Desktop\fourth year\semester 1\Advanced Software Engineering\";
            //remembers last visited directory
            fileExplorer.RestoreDirectory = true;
            try
            {
                if (fileExplorer.ShowDialog() == DialogResult.OK)
                {
                    codeArea.Text = ""; //clears text box before loading file contents
                    StreamReader reader = File.OpenText(fileExplorer.FileName);
                    do
                    {
                        String line = reader.ReadLine();
                        if (line == null) break; //breaks if end of file
                        codeArea.Text += line;
                        codeArea.AppendText(Environment.NewLine); // new line starting points are preserved
                    } while (true);
                    reader.Close();
                }
            }

            catch (System.ArgumentException)
            {
                MessageBox.Show("NO FILE CHOSEN !!", "ALERT", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
            catch (FileNotFoundException)
```

Creating an interface which is used to execute a command



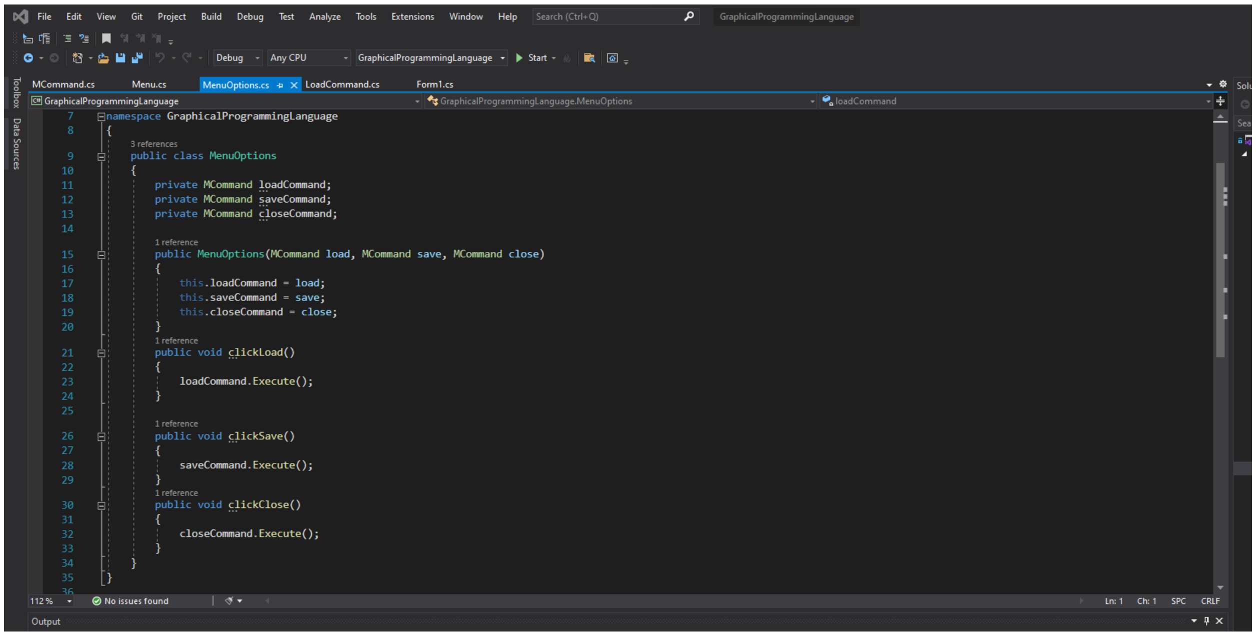
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GraphicalProgrammingLanguage
{
    12 references
    public interface MCommand
    {
        6 references
        void Execute();
    }
}
```

Creating command classes that implement the above interface



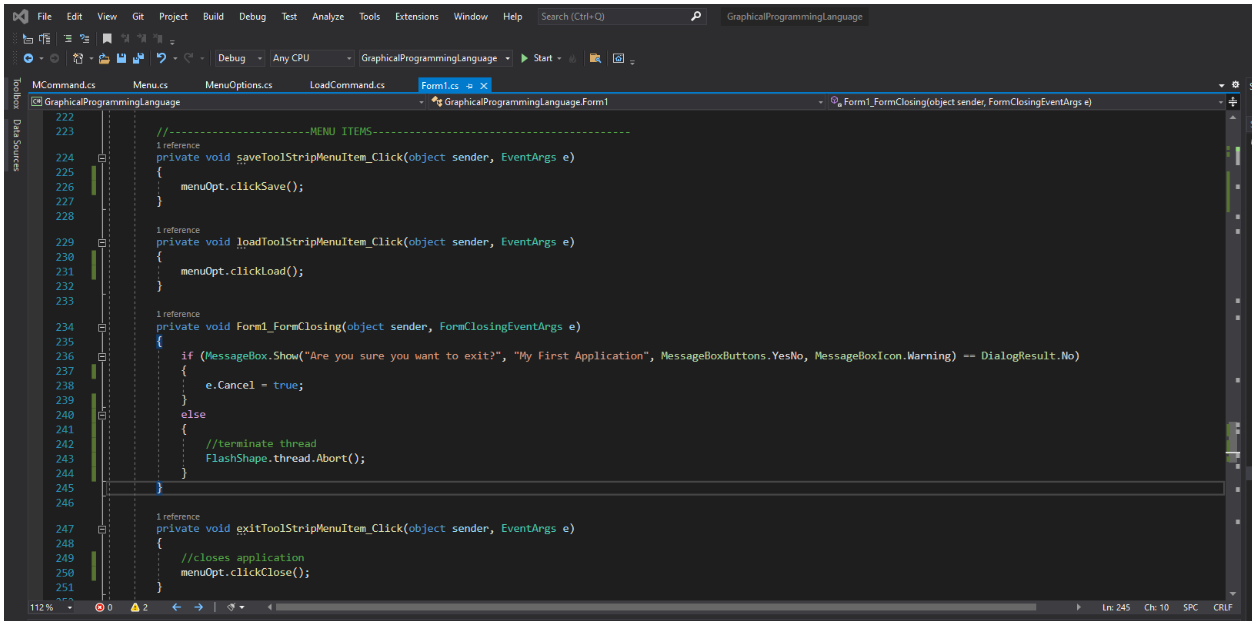
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GraphicalProgrammingLanguage
{
    2 references
    public class LoadCommand : MCommand
    {
        private Menu menu;
        RichTextBox codeArea;

        1 reference
        public LoadCommand(Menu menuItems, RichTextBox ogCodeArea)
        {
            menu = menuItems;
            codeArea = ogCodeArea;
        }

        4 references
        public void Execute()
        {
            menu.Load(codeArea);
        }
    }
}
```

Creating the invoker. The invoker object does not depend on the concrete command or receiver classes. It passes the request to a receiver indirectly by executing a command.



```
namespace GraphicalProgrammingLanguage
{
    3 references
    public class MenuOptions
    {
        private MCommand loadCommand;
        private MCommand saveCommand;
        private MCommand closeCommand;

        1 reference
        public MenuOptions(MCommand load, MCommand save, MCommand close)
        {
            this.loadCommand = load;
            this.saveCommand = save;
            this.closeCommand = close;
        }
        1 reference
        public void clickLoad()
        {
            loadCommand.Execute();
        }

        1 reference
        public void clickSave()
        {
            saveCommand.Execute();
        }
        1 reference
        public void clickClose()
        {
            closeCommand.Execute();
        }
    }
}
```

Usage



```
        //----------------------MENU ITEMS-----------------------------------------
        1 reference
        private void saveToolStripMenuItem_Click(object sender, EventArgs e)
        {
            menuOpt.clickSave();
        }

        1 reference
        private void loadToolStripMenuItem_Click(object sender, EventArgs e)
        {
            menuOpt.clickLoad();
        }

        1 reference
        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (MessageBox.Show("Are you sure you want to exit?", "My First Application", MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.No)
            {
                e.Cancel = true;
            }
            else
            {
                //terminate thread
                FlashShape.thread.Abort();
            }
        }

        1 reference
        private void exitToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //closes application
            menuOpt.clickClose();
        }
```