

Project 1: Token, Parser, and Drawings

Victoria Lee

UMGC: University of Maryland Global Campus

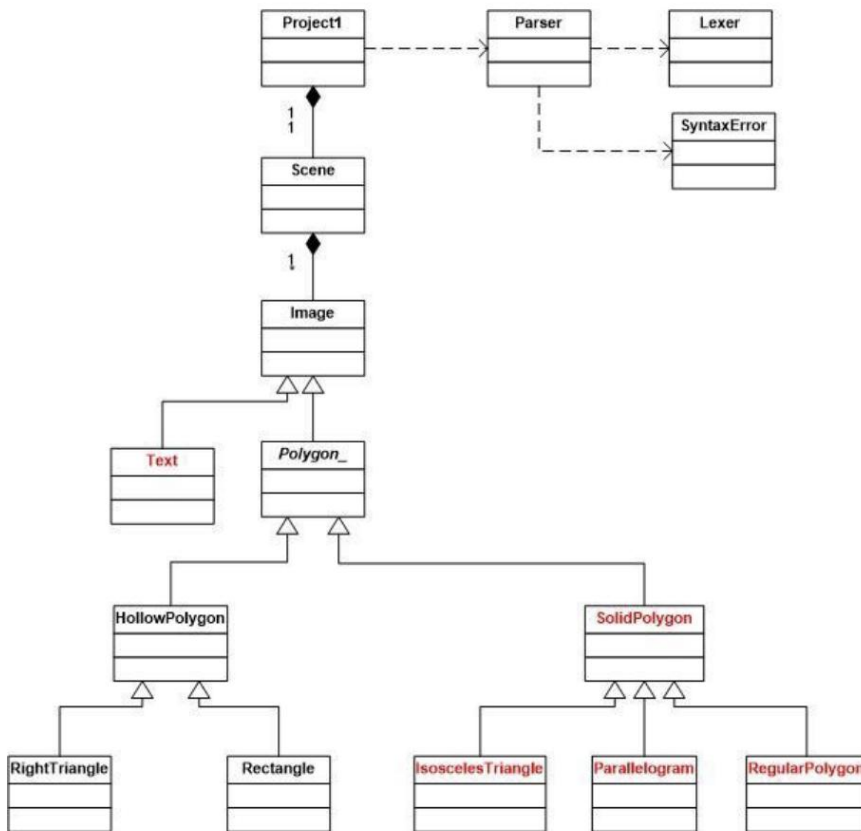
CMSC 330: Advanced Programming Languages

9/7/2024

UML Diagrams:

Note: The Main Method is in Project1 Class. This allows it to run the program smoothly. The package is project1. The UML diagrams below include the package.

UML Diagram:

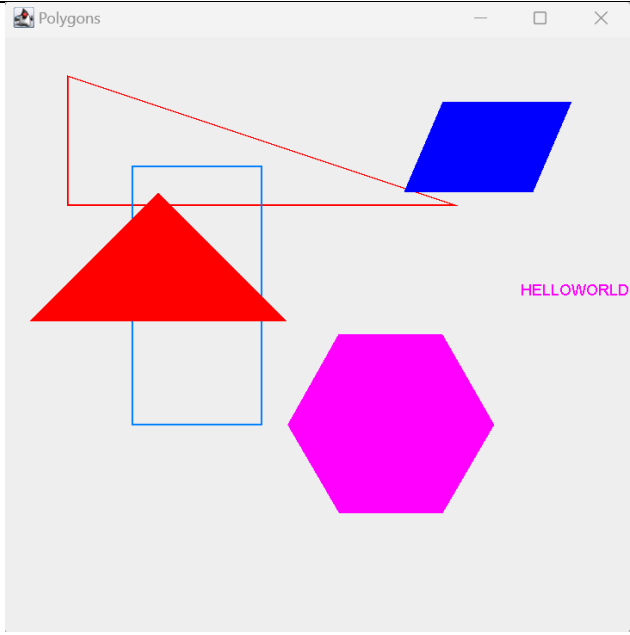


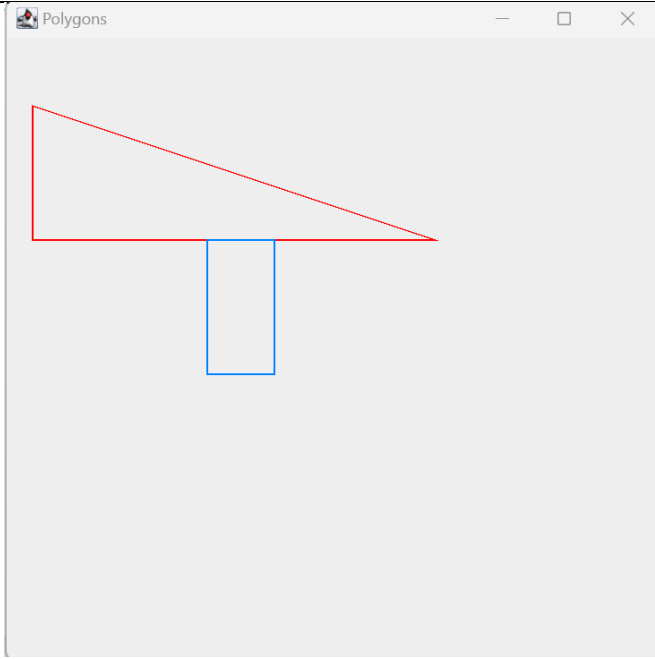
Developer's Guide, Test Cases, and Lessons Learned:

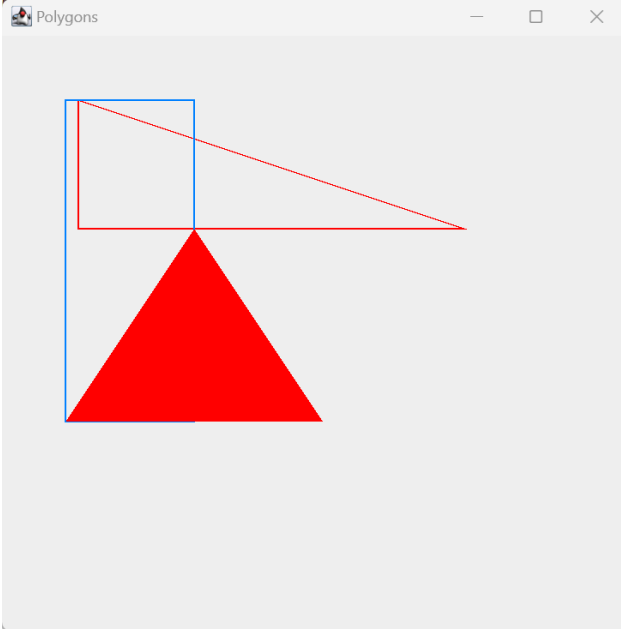
You can import the files using the new project and import all of the classes. Make sure to have a package called “cmisc_330_Project_1_Skeleton” in the “src” file. An alternative way is to create the project and import all the files including the package I already included from the “.ZIP” file. You can compile the file and execute the program by going to the Main class, right-clicking on the class, and selecting the run option. It should pop up the File Manager and then once you select a scene file it will pop up the drawings from that file.

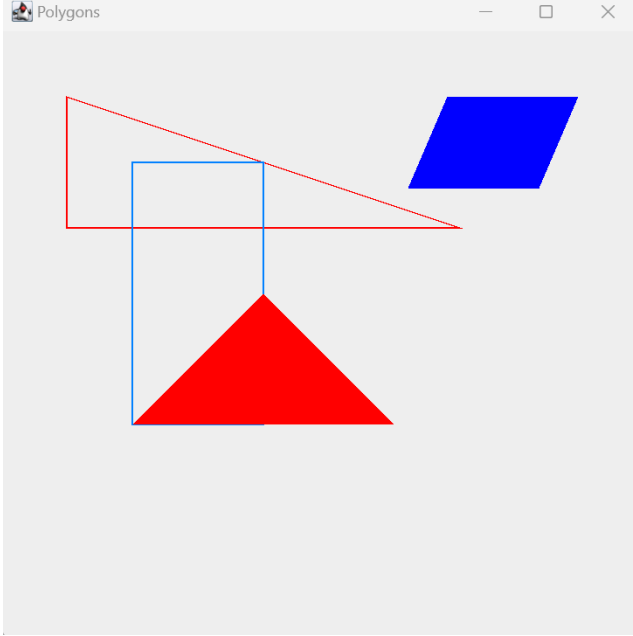
Table 1 (below): *Developer's guide describing compiling and executing the program.*

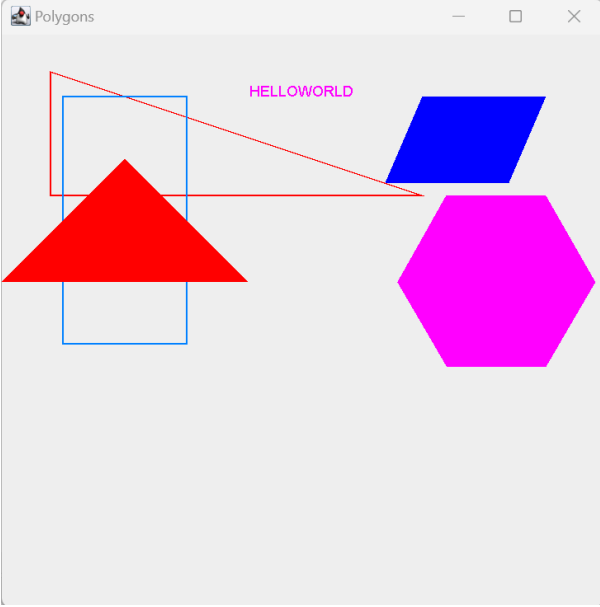
Documentation includes Lessons learned at the end.

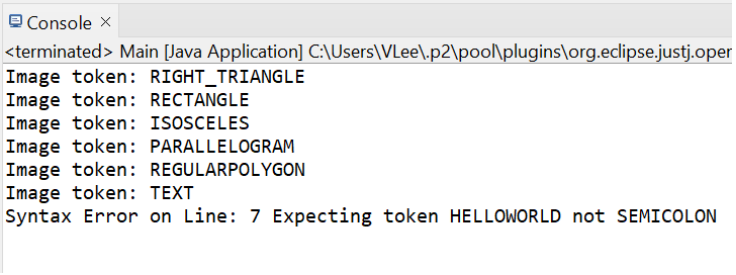
Test #	Description	Screenshot	PASS / FAIL Flag
1.	<p>This is to test the given output with some changes to the color on the text. This should pop up all the required polygons.</p> <p>The File is scene.txt</p> <p>Scene Polygons (500, 500)</p> <p>RightTriangle Color (255, 0, 0) at (50, 30) Height 100 Width 300;</p> <p>Rectangle Color (0, 128, 255) at (100, 100) Height 200 Width 100;</p>		PASS

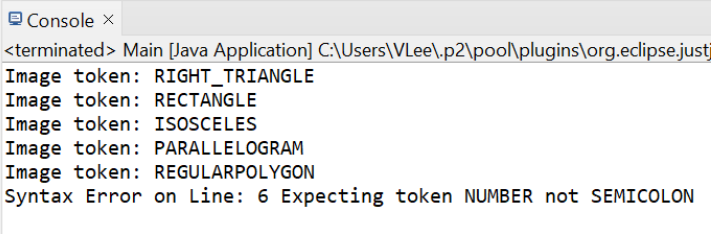
	<p><u>Isosceles</u> Color (255, 0, 0) at (120, 120) Height 100 Width 200;</p> <p>Parallelogram Color (0, 0, 255) at (340, 50) (440, 120) Offset 30;</p> <p>RegularPolygon Color (255, 0, 255) at (300, 300) Sides 6 Radius 80;</p> <p>Text Color (255, 0, 255) at (400, 200) HELLOWORLD;</p> <p>End.</p>		
2.	<p>This is to test the default given of two polygons. It should appear two polygons.</p> <p>The File is scene2.txt</p> <p>Scene Polygons (500, 500)</p> <p>RightTriangle Color (255, 0, 0) at (20, 50) Height 100 Width 300;</p> <p>Rectangle Color (0, 128, 255) at (150, 150) Height 100 Width 50;</p> <p>End.</p>		PASS

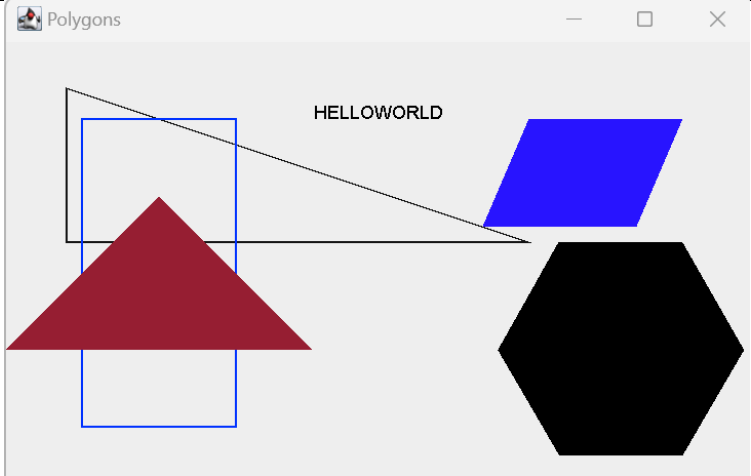
3.	<p>This is to test out three polygons. It should appear three polygons.</p> <p>The File is scene3.txt</p> <p>Scene Polygons (500, 500)</p> <p>RightTriangle Color (255, 0, 0) at (60, 50) Height 100 Width 300;</p> <p>Rectangle Color (0, 128, 255) at (50, 50) Height 250 Width 100;</p> <p><u>Isosceles</u> Color (255, 0, 0) at (150, 150) Height 150 Width 200;</p> <p>End.</p>		PASS

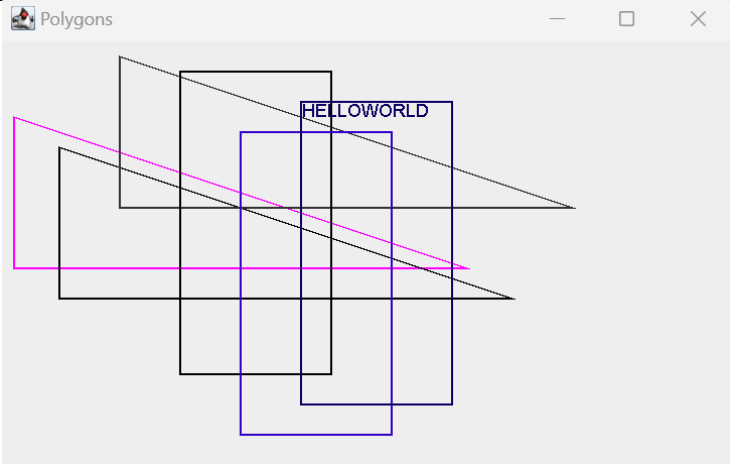
<p>4.</p>	<p>This is to test out four polygons. It should appear as four polygons.</p> <p>The File is scene4.txt</p> <p>Scene Polygons (500, 500)</p> <p>RightTriangle Color (255, 0, 0) at (50, 50) Height 100 Width 300;</p> <p>Rectangle Color (0, 128, 255) at (100, 100) Height 200 Width 100;</p> <p><u>Isosceles</u> Color (255, 0, 0) at (200, 200) Height 100 Width 200;</p> <p>Parallelogram Color (0, 0, 255) at (340, 50) (440, 120) Offset 30;</p> <p>End.</p>	 <p>The screenshot shows a window titled 'Polygons' with a light gray background. It contains four distinct polygons: a red right triangle on the left, a blue rectangle in the center, a red isosceles triangle at the bottom center, and a blue parallelogram on the right. The polygons are drawn with solid colors and no borders.</p>	<p>PASS</p>
-----------	---	--	-------------

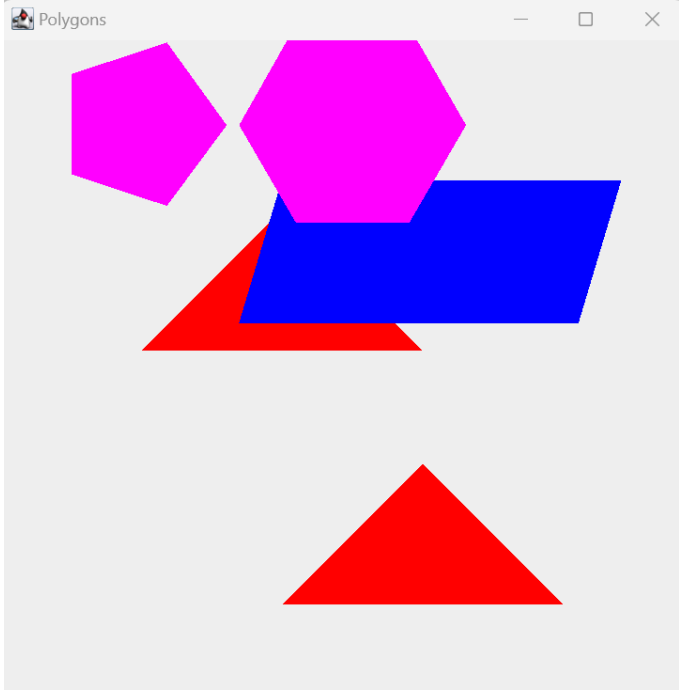
<p>5.</p>	<p>This is to test 4 polygons and a text. It should appear 4 polygons and a text at the top in different locations.</p> <p>The File is scene5.txt</p> <p>Scene Polygons (500, 500)</p> <p>RightTriangle Color (255, 0, 0) at (40, 30) Height 100 Width 300;</p> <p>Rectangle Color (0, 128, 255) at (50, 50) Height 200 Width 100;</p> <p><u>Isosceles</u> Color (255, 0, 0) at (100, 100) Height 100 Width 200;</p> <p>Parallelogram Color (0, 0, 255) at (340, 50) (440, 120) Offset 30;</p> <p>RegularPolygon Color (255, 0, 255) at (400, 200) Sides 6 Radius 80;</p> <p>Text Color (255, 0, 255) at (200, 50) HELLOWORLD;</p> <p>End.</p>		<p>PASS</p>
-----------	--	--	-------------

6.	<p>This is to test wrong input in the scenes like forgetting the HELLOWORLD in Text. It should pop up the error and state that it is looking for HELLOWORLD.</p> <p>The File is scene6.txt</p> <p>Scene Polygons (500, 500)</p> <p>RightTriangle Color (255, 0, 0) at (40, 30) Height 100 Width 300;</p> <p>Rectangle Color (0, 128, 255) at (50, 50) Height 200 Width 100;</p> <p><u>Isosceles</u> Color (255, 0, 0) at (100, 100) Height 100 Width 200;</p> <p>Parallelogram Color (0, 0, 255) at (340, 50) (440, 120) Offset 30;</p> <p>RegularPolygon Color (255, 0, 255) at (400, 200) Sides 6 Radius 80;</p> <p>Text Color (255, 0, 255) at (200, 50);</p>	 <pre> Console × <terminated> Main [Java Application] C:\Users\VLee\p2\pool\plugins\org.eclipse.justj.oper Image token: RIGHT_TRIANGLE Image token: RECTANGLE Image token: ISOSCELES Image token: PARALLELOGRAM Image token: REGULARPOLYGON Image token: TEXT Syntax Error on Line: 7 Expecting token HELLOWORLD not SEMICOLON </pre>	PASS

	End.		
7.	<p>This is to test wrong input in the scenes like forgetting the radius number in regular polygon. It should pop up the error and state that it is looking for a number value missing.</p> <p>The File is scene7.txt</p> <p>Scene Polygons (500, 500)</p> <p>RightTriangle Color (255, 0, 0) at (40, 30) Height 100 Width 300;</p> <p>Rectangle Color (0, 128, 255) at (50, 50) Height 200 Width 100;</p> <p><u>Isosceles</u> Color (255, 0, 0) at (100, 100) Height 100 Width 200;</p> <p>Parallelogram Color (0, 0, 255) at (340, 50) (440, 120) Offset 30;</p> <p>RegularPolygon Color (255, 0, 255) at (400, 200) Sides 6 Radius ;</p>	 <pre> Console × <terminated> Main [Java Application] C:\Users\VLee\p2\pool\plugins\org.eclipse.just Image token: RIGHT_TRIANGLE Image token: RECTANGLE Image token: ISOSCELES Image token: PARALLELOGRAM Image token: REGULARPOLYGON Syntax Error on Line: 6 Expecting token NUMBER not SEMICOLON </pre>	PASS

	<p>Text Color (255, 0, 255) at (200, 50) HELLOWORLD; End.</p>		
8.	<p>This is to test the color changes in the polygons and the locations.</p> <p>The File is scene8.txt</p> <p>Scene Polygons (500, 500)</p> <p>RightTriangle Color (20, 20, 20) at (40, 30) Height 100 Width 300;</p> <p>Rectangle Color (0, 50, 255) at (50, 50) Height 200 Width 100;</p> <p><u>Isosceles</u> Color (150, 30, 50) at (100, 100) Height 100 Width 200;</p> <p>Parallelogram Color (40, 20, 255) at (340, 50) (440, 120) Offset 30;</p> <p>RegularPolygon Color (0, 0, 0) at (400, 200) Sides 6 Radius 80;</p>		PASS

	<p>Text Color (0, 0, 0) at (200, 50) HELLOWORLD; End.</p>		
9.	<p>This is to test repeated polygons and text. It should appear as 6 polygons, where 3 are right triangles and 3 are rectangles.</p> <p>The File is scene9.txt</p> <p>Scene Polygons (500, 500)</p> <p>RightTriangle Color (255, 0, 255) at (10, 50) Height 100 Width 300;</p> <p>RightTriangle Color (0, 0, 0) at (40, 70) Height 100 Width 300;</p> <p>RightTriangle Color (50, 50, 50) at (80, 10) Height 100 Width 300;</p> <p>Rectangle Color (0, 0, 0) at (120, 20) Height 200 Width 100;</p> <p>Rectangle Color (50, 0, 200) at</p>		PASS

	<p>(160, 60) Height 200 Width 100;</p> <p>Rectangle Color (10, 0, 100) at (200, 40) Height 200 Width 100;</p> <p>Text Color (0, 0, 0) at (200, 50) HELLOWORLD;</p> <p>End.</p>		
10	<p>This is testing repeated solid polygons. It also tests if I can change the sides of the polygon. This should appear in the repeated polygons, repeated isosceles, and one parallelogram.</p> <p>The File is scene10.txt</p> <p>Scene Polygons (500, 500)</p> <p><u>Isosceles</u> Color (255, 0, 0) at (200, 120) Height 100 Width 200;</p> <p><u>Isosceles</u> Color (255, 0, 0) at (300, 300) Height 100 Width 200;</p> <p>Parallelogram Color (0, 0, 255)</p>		PASS

	at (200, 100) (440, 200) Offset 30; RegularPolygon Color (255, 0, 255) at (250, 60) Sides 6 Radius 80; RegularPolygon Color (255, 0, 255) at (100, 60) Sides 5 Radius 60; End.		
--	--	--	--

Lessons Learned (brief paragraphs):

To achieve my project goals, I learned how to create a project, classes, immutable classes, generics, try/catch, throw exceptions, encapsulation, inheritance, interfaces, information hiding, polymorphism, methods, and functions for object-oriented programming for the Shapes Program. I learned that the Shapes Program allowed me to comprehend the significant principles of object-oriented programming and software development. A class defines all the attributes an object can have and methods that define the object's functionality. A subclass inherits the properties and behaviors of another class. Immutable classes in Java mean that once an object is created, we cannot change its content. So, the main lesson learned from the Project1 class is that it is responsible for creating the Shapes Program Selection and drawing the shapes based on the scene.txt file. Moreover, these immutable classes are HollowPolygon, SolidPolygon, Rectangle, RightTriangle, Isosceles, Parallelogram, Regular Polygon, and Text. Other classes given are Image, Lexer, Parser, Main, SyntaxError, Scene, and Polygon_. The HollowPolygon is what other classes inherit from if it is a hollow shape. The SolidPolygon is what other classes inherit

from if it is a solid shape. The Image is what the Text inherits if it is a text class to input text.

The main goals in this class's design include its constructor, which initializes the object, and its methods so that the objects can compare the shapes from the user input the drawings when called through the Parser class. These lessons helped me understand good modular design for developing object-oriented programming with Shapes and geometry. In real life, users can utilize this application to create shape drawings and understand Parser, Lexer(Lexemes), and Tokens.

The parser is about comprehending the classes, hierarchy, inheritance, encapsulation, interfaces, information hiding, polymorphism, Lexer(Lexemes), Syntax, Tokens, and subclasses. I learned that the two restrictions are that the grammar cannot have any left recursive productions and must not require more than one token look ahead. A recursive descent parser is a top-down parser that parses any given string using recursive functions or procedures (GeeksforGeeks, 2023, June 9). A top-down parser builds the parse tree from the top and then down which is the start non-terminal (GeeksforGeeks, 2023, June 9). This means that this parsing relates every non-terminal with the corresponding procedures and every procedure reads a sequence of given characters that can be produced by that non-terminal. However, there is a special case of recursive descent parser, where no backtracking is required and it is a predictive parser (GeeksforGeeks, 2023, June 9). Through this parser, it removes the left recursion and left factoring from it. The resulting grammar is a grammar that is parsed by a recursive descent parser. Some of the improvements that could be made are expanding more on the text class to allow it to produce different types of text such as bolding or italics. Another improvement is that I should have debugged more in the text section, to make sure everything is working fine.

Overall, I learned to apply it to Project 1 with the lessons about classes, subclasses, packages,

importing libraries, constructors, object-oriented programming, encapsulation, inheritance, interfaces, information hiding, polymorphism, and inheriting classes.

My design approach was to create all the required classes before implementing the Parser class. I started with a Bottom-Up Design when building the code, but then debugged the code through a Top-Down Design. I followed the instructions on what is asked for Project 1 and the other shapes classes. I utilized the lessons to apply them to Parser. Once it was finished, I went back into the Parser class to create the Shapes program according to the rubric, and then the user inputs would be passed and it should pass back through the Token class. The main class creates a Shape File Scene program. This allows a File Manager to pop up in Java GUI and the user must input the scene.txt file to draw the shapes based on the input on that file. If it is an invalid input, it will tell the user in the error section of the console. To debug Parser, I looked at the lessons, my old codes, and online concepts about this chapter. I then modified the classes. Then, I checked back to see if the output was correct through the Parser, Lexer, and Token classes.

References

GeeksforGeeks. (2023, June 9). *Recursive Descent Parser*. GeeksforGeeks.

<https://www.geeksforgeeks.org/recursive-descent-parser/>

Jarc, D. J. (2018). *CMSC 330: (Week 2) [PowerPoint slides]*.

University of Maryland Global Campus.

<https://leocontent.umgc.edu/content/dam/permalink/f1c30adc-2624-4d1d-90fe-bf7a35045fa2.html?ou=1277895>

Nievergelt, J. (n.d.). *Algorithms and data structures*. University of Maryland Global Campus.

https://leocontent.umgc.edu/content/dam/course-content/tus/cmsc/cmsc-330/document/Nievergelt_Algorithms%20and%20Data%20Structures08%20%281%29.pdf?ou=1277895

UMGC. (n.d.). *Module 1: Formal Syntax and Semantics*. UMGC Leo Content.

<https://leocontent.umgc.edu/content/umuc/tus/cmsc/cmsc330/2245/modules/m1-module-1/s3-commentary.html?ou=1277895>