

Algoritmos y Estructura de Datos

Proyecto 1: Intérprete de LISP

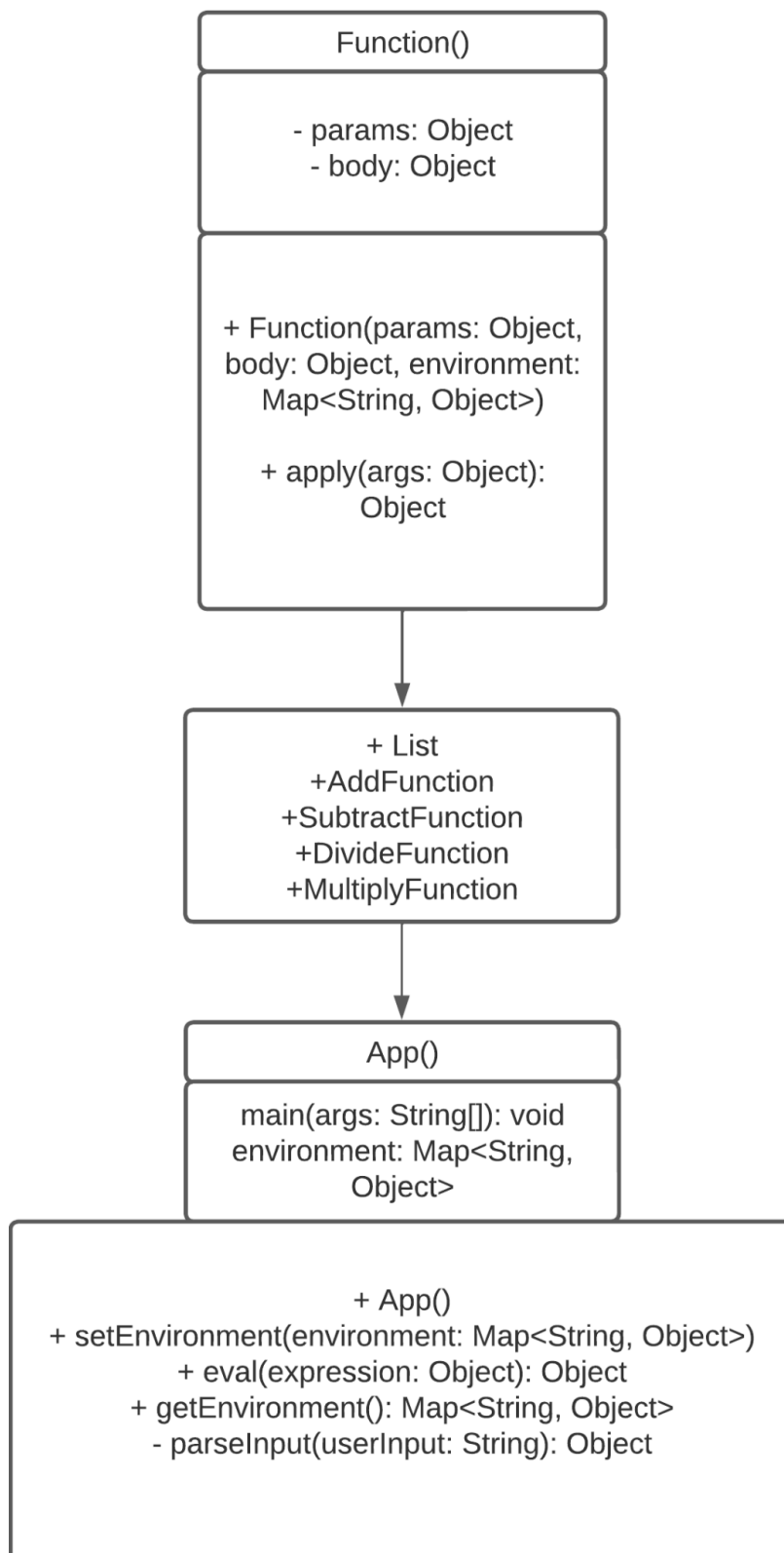
Objetivos:

- a. Utilizar el Java Collections Framework para desarrollo de aplicaciones.
- b. Saber escoger estructuras de datos, con su complejidad en tiempo y espacio para la implementación de aplicaciones.
- c. Identificar los principales elementos y usos de la programación Funcional (Functional Programming).

“Desarrollo de un intérprete LISP para un subconjunto sencillo de instrucciones de alguno de los dos dialectos principales (Common LISP y Scheme). Deben implementarse como mínimo:

- a) operaciones aritméticas.*
- b) instrucción QUOTE o ‘ (single quote, para interrumpir el proceso de evaluación de expresiones).*
- c) Definición de funciones (DEFUN).*
- d) SETQ.*
- e) predicados (ATOM, LIST, EQUAL, <, >).*
- f) condicionales (COND), nota: no es necesario implementar IF.*
- g) paso de parámetros. (un parámetro puede ser incluso una función).”*

a. Diagrama UML diseño final de intérprete LISP:



b. Control de versiones (GitHub):

<https://github.com/adimnn/Proyecto-1-Fase-2-LISP.git>

c. Prueba unitaria JUnit:

The screenshot shows an IDE with a project named 'pry1fase2'. The Explorer panel on the left shows the project structure, including a 'test' directory with a package 'uvg.edu.gt' containing the 'AppTest.java' file. The main editor displays the code for 'AppTest.java'.

```
src > test > java > uvg > edu > gt > J AppTest.java > ...
1 package uvg.edu.gt;
2
3 import static org.junit.Assert.assertEquals;
4 import org.junit.Test;
5 import java.util.ArrayList;
6 import java.util.List;
7
8 /**
9  * Unit test for simple App.
10  */
11 public class AppTest
12 {
13     /**
14      * Rigorous Test :-)
15      */
16     @Test
17     public void testArithmeticEquation() {
18         App interpreter = new App();
19         List<Object> expression = new ArrayList<>();
20         expression.add(e:"*");
21         expression.add(e:5);
22         expression.add(e:3);
23         Object result = interpreter.eval(expression);
24         assertEquals(8, result); // Expected result is 8
25     }
26 }
```

The bottom panel shows the 'TEST RESULTS' tab, which displays the following output:

```

%TESTC 1 v2
%TSTTREE1,uvg.edu.gt.AppTest,true,1,false,-1,uvg.edu.gt.AppTest,,
%TSTTREE2,testArithmeticEquation(uvg.edu.gt.AppTest),false,1,false,-1,testArithmeticEquation(uv
%TESTS 2,testArithmeticEquation(uvg.edu.gt.AppTest)

%TESTE 2,testArithmeticEquation(uvg.edu.gt.AppTest)

%RUNTIME40
```

On the right side of the bottom panel, there is a list of test results with icons indicating success (green circle) or failure (red circle). The list shows multiple successful test runs.