

Universidad del Valle de Guatemala  
Algoritmos y Estructuras de Datos  
Sección no. 30  
Docente: Moises Alonso

Proyecto No.1  
Fase 2

Alejandro Gomez 20347  
Paola de León 20361  
Marco Jurado 20308

# Entrega FASE 2:

Como parte del curso de Algoritmos y Estructura de Datos, se presenta este proyecto. Correspondiente a un intérprete de LISP en el lenguaje de Java. El trabajo realizado cumple con los requerimientos solicitados por el catedrático. En cuanto a la realización del intérprete, se optó por hacer que el usuario ingrese todo el código en una misma línea de texto, facilitando así, el ingreso y uso del mismo.

## a. Diseño final del interprete Lisp:

Tomando como referencia el primer UML realizado para la primera entrega, se optó por realizar cambios en cómo está estructurado el UML y sus clases. En este caso, con la creación de clases extras, las cuales se observan en los siguientes diagramas.

Diagrama de clases:

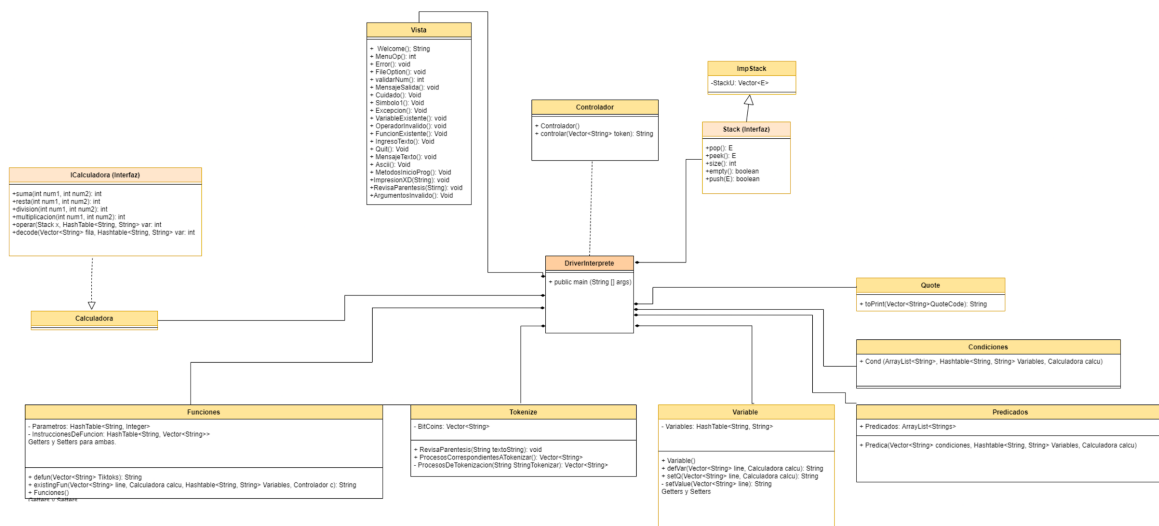


Diagrama de casos:

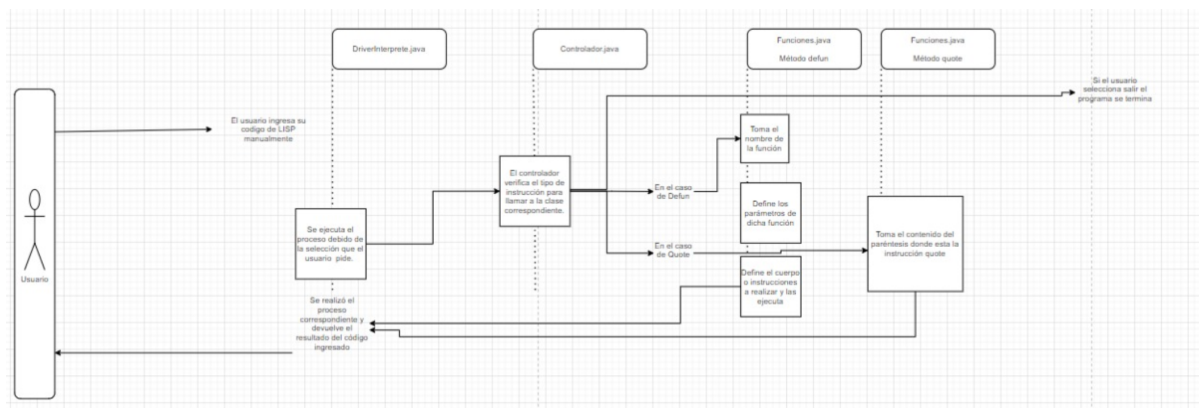
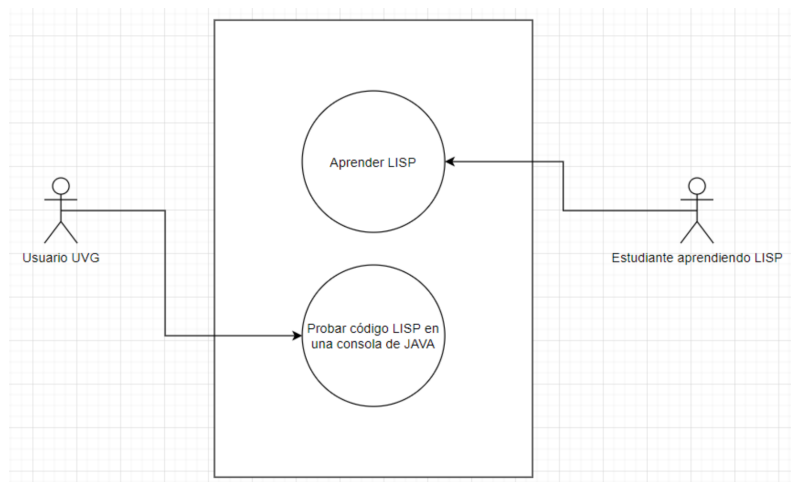


Diagrama de uso:



**b. Control de versiones de todos los documentos y código que seleccione el grupo.**

Para la realización de esta entrega, se optó por utilizar la plataforma GitHub para subir las distintas versiones realizadas.

<https://github.com/aleg001/Proyecto1Fase2>

**c. Un esquema de pruebas unitarias de las principales estructuras y partes críticas del intérprete. Deben estar también bajo el control de versiones.**

JUnits:

```
1 import junit.framework.TestCase;
2
3 public class CalculadoraTest extends TestCase {
4
5     private Calculadora calcul;
6
7     public void escenario() {
8         calcul = new Calculadora();
9     }
10
11     public void testdivision() {
12         escenario();
13         assertTrue(calcul.division(10,2)==5);
14     }
15
16     public void testmultiplicacion() {
17         escenario();
18
19         assertEquals(20, calcul.multiplicacion(10, 2));
20     }
21
22     public void testresta() {
23         escenario();
24         assertEquals(21, calcul.resta(28, 7));
25     }
26
27 }
28
29
30
```

Package Explorer JUnit 3

Finished after 0.148 seconds

Runs: 5/5 Errors: 0 Failures: 0

PredicadosTest [Runner: JUnit 3] [0.10]

```

1 import junit.framework.TestCase;
2
3 public class PredicadosTest extends TestCase {
4
5     private Predicados predicador;
6     private Calculadora calcul;
7     private Hashtable<String, String> vars;
8
9     public void Test() {
10         vars = new Hashtable<String, String>();
11         predicador = new Predicados();
12         calcul = new Calculadora();
13     }
14
15     // JUnit de caso: mayor que
16     public void testMayor () {
17         Test();
18         Vector<String> vector = new Vector<String>();
19         vector.add("1");
20         vector.add("2");
21         vector.add("3");
22         vector.add("0");
23         vector.add("4");
24         String verdadero = predicador.Predica(vector, vars, calcul);
25         assertEquals("T", verdadero);
26     }
27
28     // JUnit de caso: menor que
29     public void testMenor () {
30         Test();
31         Vector<String> vector = new Vector<String>();
32         vector.add("1");
33         vector.add("<");
34         vector.add("1");
35         vector.add("3");
36         vector.add("4");
37         String verdadero = predicador.Predica(vector, vars, calcul);
38         assertEquals("T", verdadero);
39     }
40
41     // JUnit de caso: equals
42     public void testEquals () {
43         Test();
44         Vector<String> vector = new Vector<String>();
45         vector.add("1");
46         vector.add("E");
47         vector.add("5");
48         vector.add("5");
49         vector.add("1");
50         String verdadero = predicador.Predica(vector, vars, calcul);
51         assertEquals("T", verdadero);
52     }
53
54 }

```

Problems Javadoc Declaration Console Terminal Coverage

PredicadosTest (1) (6/04/2021 23:18:18)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
> Proyecto1Fase2-main-6	22.3 %	689	2,053	2,642

Writable Smart Insert 9:31:195

Outline

- calcul: Calculadora
- predicador: Predicados
- vars: Hashtable<String, String>
- Test(): void
- testAtom(): void
- testEquals(): void
- testList(): void
- testMayor(): void
- testMenor(): void

Package Explorer JUnit 3

Finished after 0.148 seconds

Runs: 5/5 Errors: 0 Failures: 0

PredicadosTest [Runner: JUnit 3] [0.10]

```

33 Test();
34 Vector<String> vector = new Vector<String>();
35 vector.add("1");
36 vector.add("<");
37 vector.add("1");
38 vector.add("3");
39 vector.add("4");
40 String verdadero = predicador.Predica(vector, vars, calcul);
41 assertEquals("T", verdadero);
42 }
43
44 // JUnit de caso: equals
45 public void testEquals () {
46     Test();
47     Vector<String> vector = new Vector<String>();
48     vector.add("1");
49     vector.add("E");
50     vector.add("5");
51     vector.add("5");
52     vector.add("1");
53     String verdadero = predicador.Predica(vector, vars, calcul);
54     assertEquals("T", verdadero);
55 }
56
57 // JUnit de caso: atom
58 public void testAtom () {
59     Test();
60     Vector<String> vector = new Vector<String>();
61     vector.add("1");
62     vector.add("ATOM");
63     vector.add("5");
64     vector.add("1");
65     String verdadero = predicador.Predica(vector, vars, calcul);
66     assertEquals("T", verdadero);
67 }
68
69 // JUnit de caso: list
70 public void testList () {
71     Test();
72     Vector<String> vector = new Vector<String>();
73     vector.add("1");
74     vector.add("LIST");
75     vector.add("5");
76     vector.add("5");
77     vector.add("1");
78     String verdadero = predicador.Predica(vector, vars, calcul);
79     assertEquals("5, 5", verdadero);
80 }
81
82 }
83

```

Problems Javadoc Declaration Console Terminal Coverage

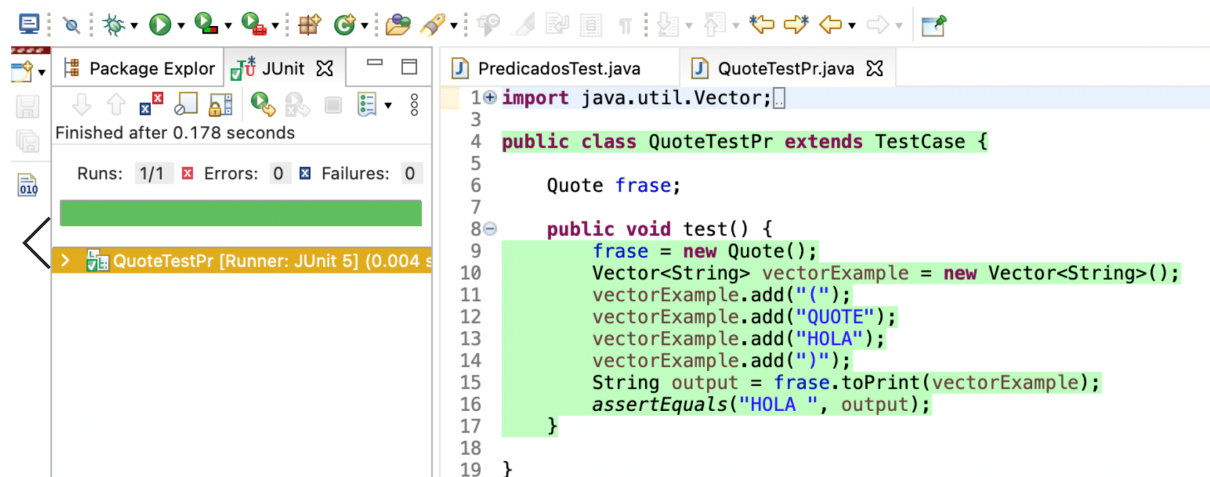
PredicadosTest (1) (6/04/2021 23:18:18)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
> Proyecto1Fase2-main-6	22.3 %	689	2,053	2,642

Writable Smart Insert 9:31:195

Outline

- calcul: Calculadora
- predicador: Predicados
- vars: Hashtable<String, String>
- Test(): void
- testAtom(): void
- testEquals(): void
- testList(): void
- testMayor(): void
- testMenor(): void



e. Un pequeño programa ejecutándose en el ambiente Lisp implementado por el grupo. Debe contener solamente funcionalidad básica del lenguaje. Subirlo a Canvas.

En la carpeta del proyecto se incluyen bastantes ejemplos de corrida del intérprete de LISP en Java.

f. Videos demostrativos de todo el proceso de desarrollo y del intérprete funcionando.

[https://www.youtube.com/watch?v=vc9NqGMBwck&ab\\_channel=AleG%C3%B3mez](https://www.youtube.com/watch?v=vc9NqGMBwck&ab_channel=AleG%C3%B3mez)

g. Ejemplos de código para el intérprete de LISP en Java

Casos aritméticos:



## Operaciones con variables

- ☒ Operaciones con números de 1 dígito.

```
( + x 2 )
```

## Declaracion de funciones

- ☒ Declarar y utilizar funciones

```
( defun resta ( a b ) ( - a b ) )  
( resta ( 1 2 ) )
```

## Predicados

- ☒ ATOM

```
( atom 2 )
```

- ☒ LIST

```
(list 1 2 3)
```

- ☒ >

```
(> 3 2)
```

- ☒ <

```
(< 4 2)
```

- ☒ EQUALS

```
(eq 0 1)
```

- ☒ COND

```
( cond ( eq 1 2 ) ( + 1 2 ) ( - 1 2 ) )
```