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:

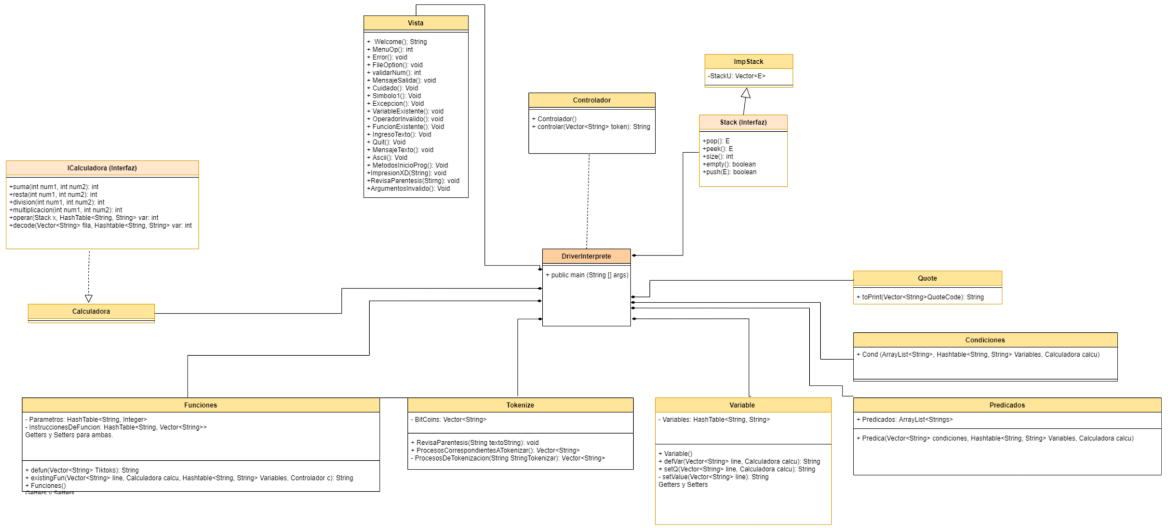


Diagrama de casos:

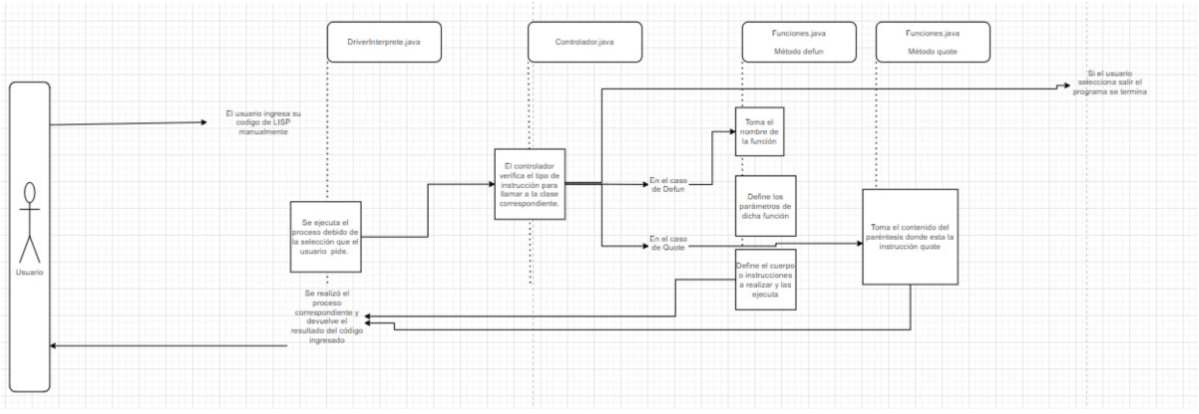
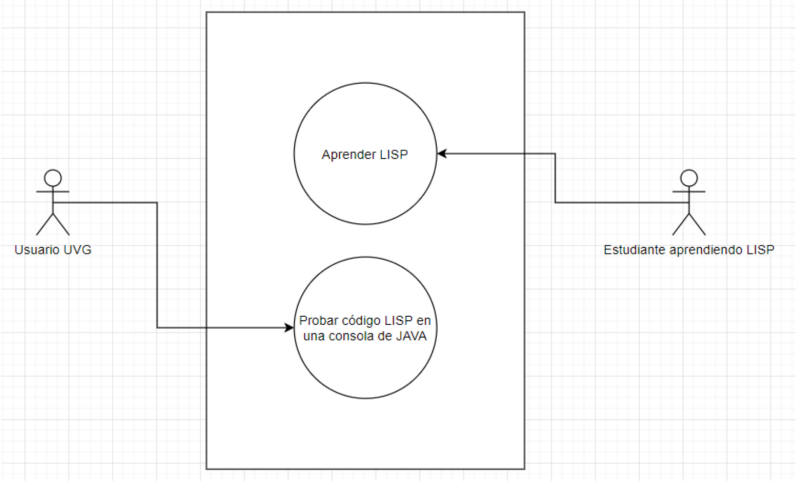


Diagrama de uso:



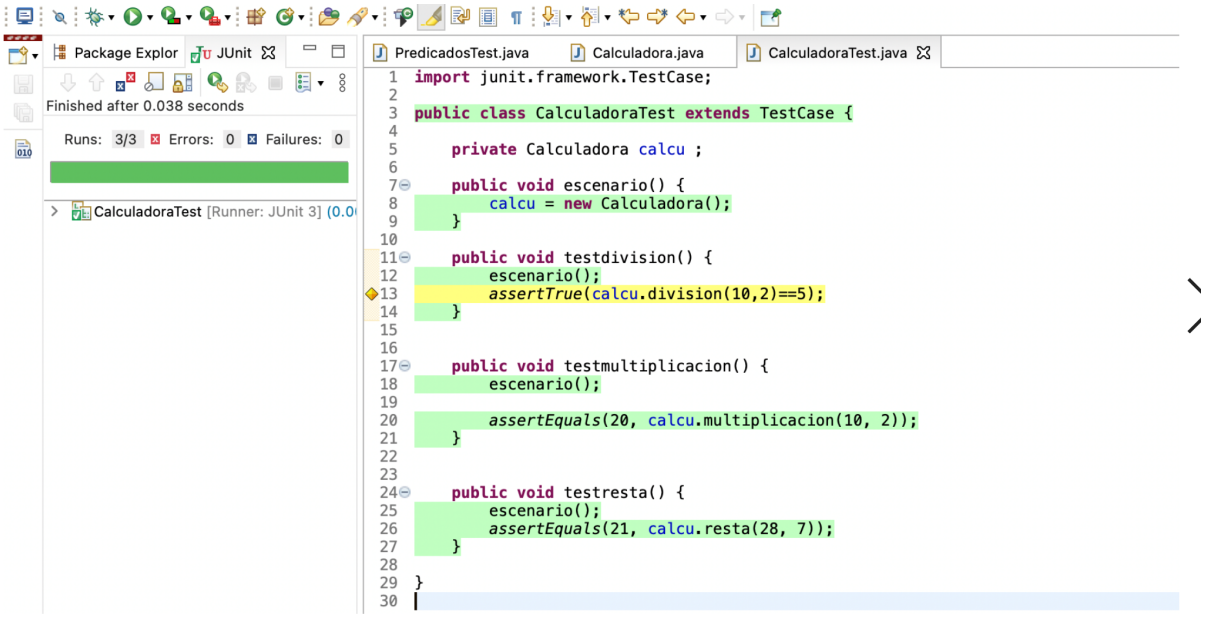
b. Control de versiones de todos los documentos y código que seleccione el grupo.

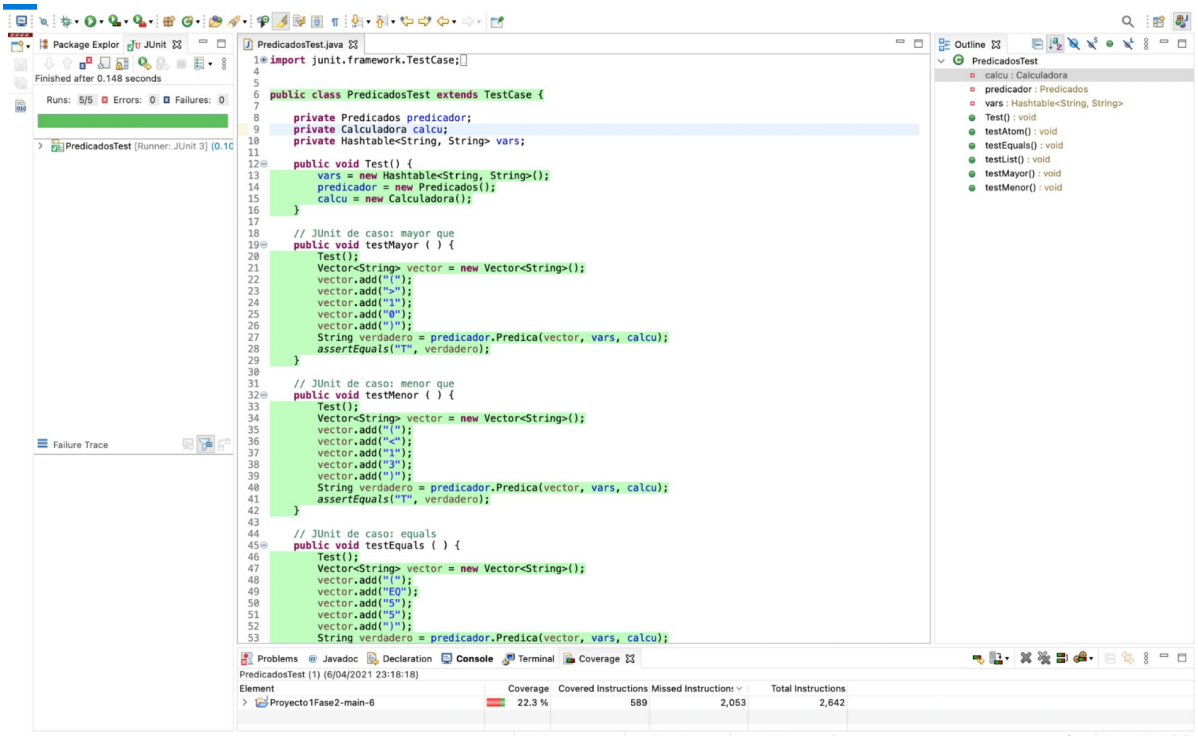
Para la realización de esta entrega, se optó por utilizar la plataforma GitHub para subir las distintas versiones realizadas.

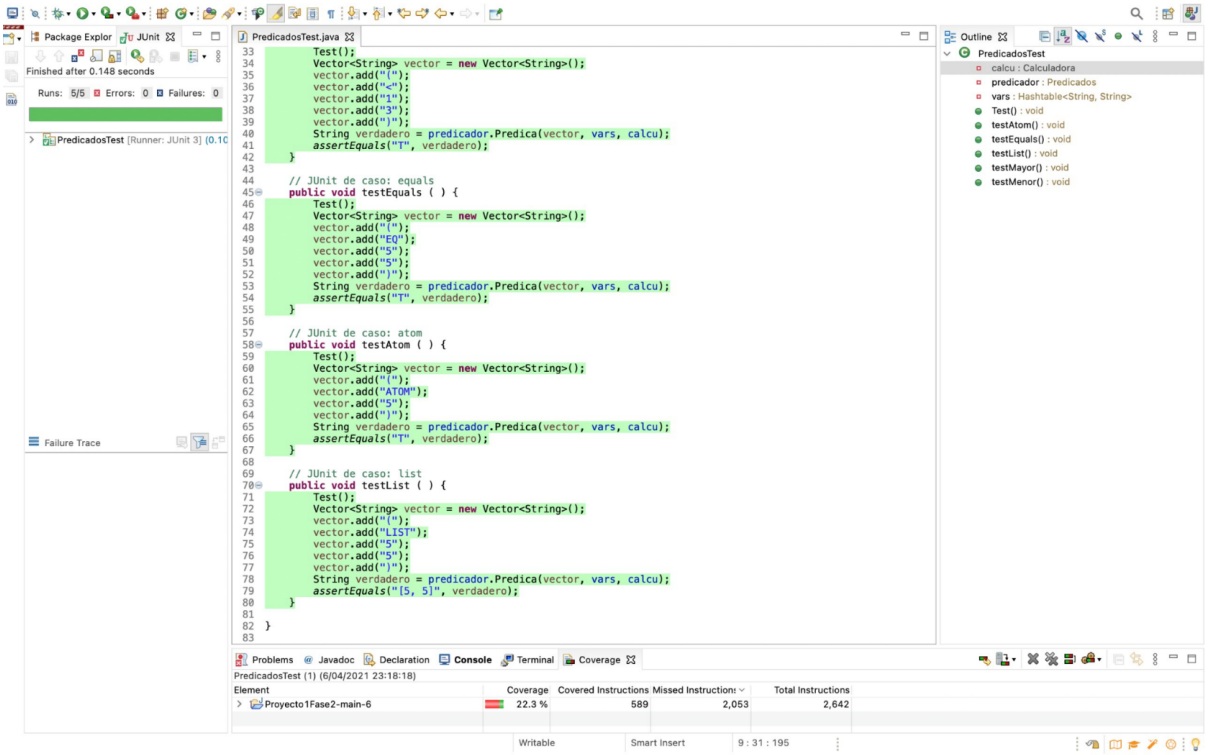
https://github.com/aleg001/Proyecto1Fase2

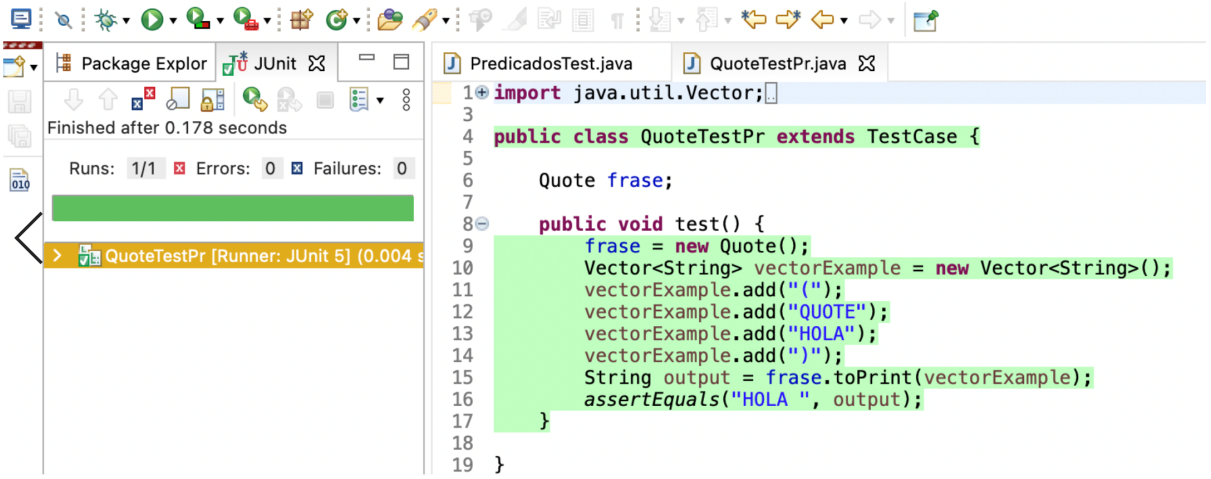
c. Un esquema de pruebas unitarias de las principales estructuras y partes críticas del intérprete. Deben estar también bajo el control de versiones.

JUnits:




Package Explor
JUnit
Finished after 0.148 seconds
Runs: 5/5 Errors: 0 Failures: 0
PredicadosTest [Runner: JUnit 3] (0.1
Failure Trace
PredicadosTest.java
import junit.framework.TestCase;
public class PredicadosTest extends TestCase {
private Predicados predicador;
private Calculadora calcu;
private Hashtable<String, String> vars;
public void Test() {
vars = new Hashtable<String, String>();
predicador = new Predicados();
calcu = new Calculadora();
}
// JUnit de caso: mayor que
public void testMayor ( ) {
Test();
Vector<String> vector = new Vector<String>();
vector.add("(");
vector.add(">");
vector.add("1");
vector.add("0");
vector.add(")");
String verdadero = predicador.Predica(vector, vars, calcu);
assertEquals("T", verdadero);
}
// JUnit de caso: menor que
public void testMenor ( ) {
Test();
Vector<String> vector = new Vector<String>();
vector.add("(");
vector.add("<");
vector.add("1");
vector.add("3");
vector.add(")");
String verdadero = predicador.Predica(vector, vars, calcu);
assertEquals("T", verdadero);
}
// JUnit de caso: equals
public void testEquals ( ) {
Test();
Vector<String> vector = new Vector<String>();
vector.add("(");
vector.add("EQ");
vector.add("5");
vector.add("5");
vector.add(")");
String verdadero = predicador.Predica(vector, vars, calcu);
Outline
PredicadosTest
calcu : Calculadora
predicador : Predicados
vars : Hashtable<String, String>
Test() : void
testAtom() : void
testEquals() : void
testList() : void
testMayor() : void
testMenor() : void
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 %
589
2,053
2,642
Writable
Smart Insert
9 : 31 : 195


Package Explor
JUnit
Finished after 0.148 seconds
Runs: 5/5 Errors: 0 Failures: 0
PredicadosTest [Runner: JUnit 3] (0.1
Failure Trace
PredicadosTest.java
Test();
Vector<String> vector = new Vector<String>();
vector.add("(");
vector.add("<");
vector.add("1");
vector.add("3");
vector.add(")");
String verdadero = predicador.Predica(vector, vars, calcu);
assertEquals("T", verdadero);
}
// JUnit de caso: equals
public void testEquals ( ) {
Test();
Vector<String> vector = new Vector<String>();
vector.add("(");
vector.add("EQ");
vector.add("5");
vector.add("5");
vector.add(")");
String verdadero = predicador.Predica(vector, vars, calcu);
assertEquals("T", verdadero);
}
// JUnit de caso: atom
public void testAtom ( ) {
Test();
Vector<String> vector = new Vector<String>();
vector.add("(");
vector.add("ATOM");
vector.add("5");
vector.add(")");
String verdadero = predicador.Predica(vector, vars, calcu);
assertEquals("T", verdadero);
}
// JUnit de caso: list
public void testList ( ) {
Test();
Vector<String> vector = new Vector<String>();
vector.add("(");
vector.add("LIST");
vector.add("5");
vector.add("5");
vector.add(")");
String verdadero = predicador.Predica(vector, vars, calcu);
assertEquals("[5, 5]", verdadero);
}
}
Outline
PredicadosTest
calcu : Calculadora
predicador : Predicados
vars : Hashtable<String, String>
Test() : void
testAtom() : void
testEquals() : void
testList() : void
testMayor() : void
testMenor() : void
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 %
589
2,053
2,642
Writable
Smart Insert
9 : 31 : 195



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

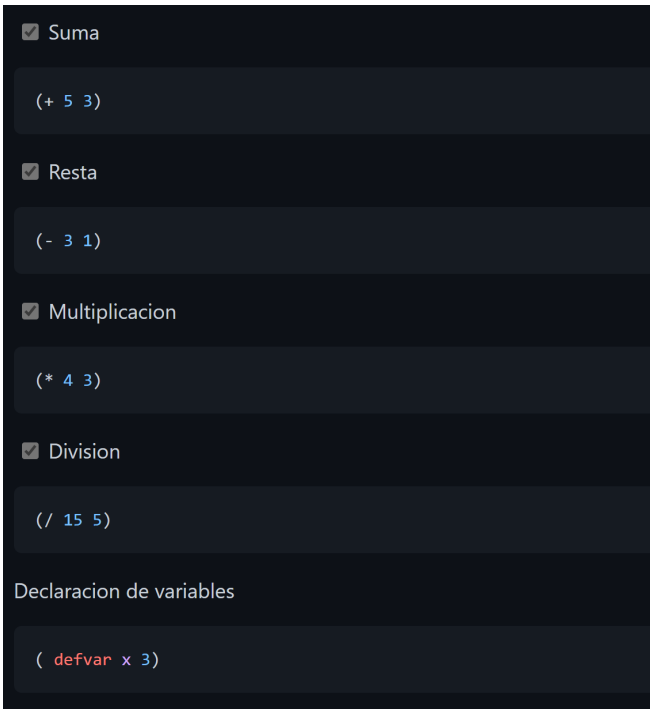
En la carpeta del proyecto se incluyen bastantes ejemplos de corrida del intérprete de LISP en Java.

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

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

Casos aritméticos:



Operaciones con variables

- [x] Operaciones con números de 1 dígito.

```
( + x 2 )
```

Declaracion de funciones

- [x] Declarar y utilizar funciones

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

Predicados

- [x] ATOM

```
( atom 2 )
```

- [x] LIST

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

- [x] >

```
(> 3 2)
```

- [x] <

```
(< 4 2)
```

- [x] EQUALS

```
(eq 0 1)
```

- [x] COND

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