

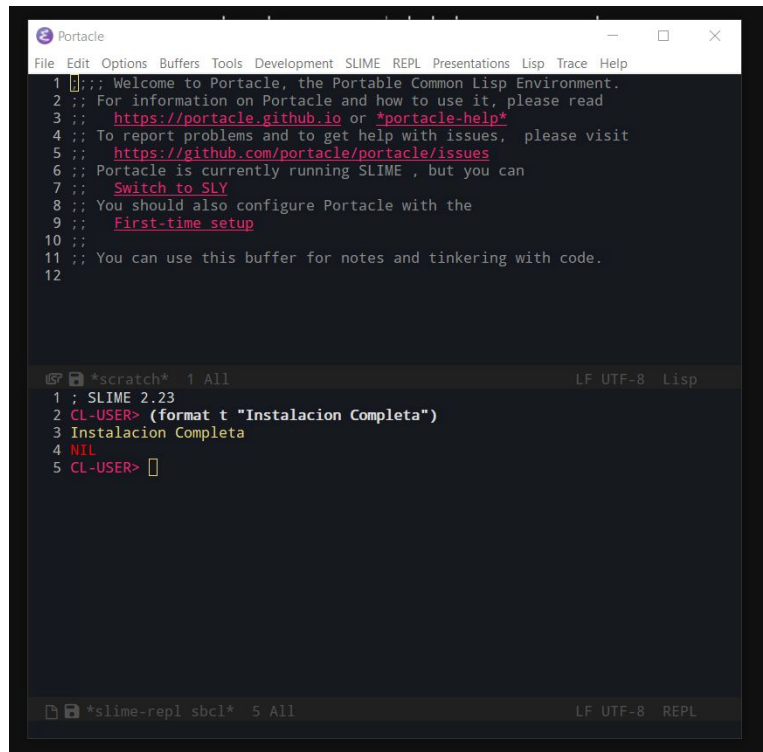
**Universidad del Valle de Guatemala**  
**Algoritmos y Estructuras de Datos**

**Proyecto 1 Fase 1**  
**17/02/2019**

**Integrantes:**

**Fernando Jose Garavito 18071**  
**Kristen Amanda Brandt 171482**  
**Javier Ramirez Cospin 18099**

**Instalación de una versión de LISP**



The screenshot shows the Portacle application window. The top menu bar includes File, Edit, Options, Buffers, Tools, Development, SLIME, REPL, Presentations, Lisp, Trace, and Help. The main text area displays a welcome message for Portacle, the Portable Common Lisp Environment, with instructions on how to use it and links to the Portacle GitHub repository and issue tracker. Below the welcome message, there are two buffers. The first buffer, named '\*scratch\*', contains the following text: 1 ; SLIME 2.23, 2 CL-USER> (format t "Instalacion Completa"), 3 Instalacion Completa, 4 NIL, 5 CL-USER> . The second buffer, named '\*slime-repl sbcl\*', contains the text: 1 ; SLIME 2.23, 2 CL-USER> (format t "Instalacion Completa"), 3 Instalacion Completa, 4 NIL, 5 CL-USER> .

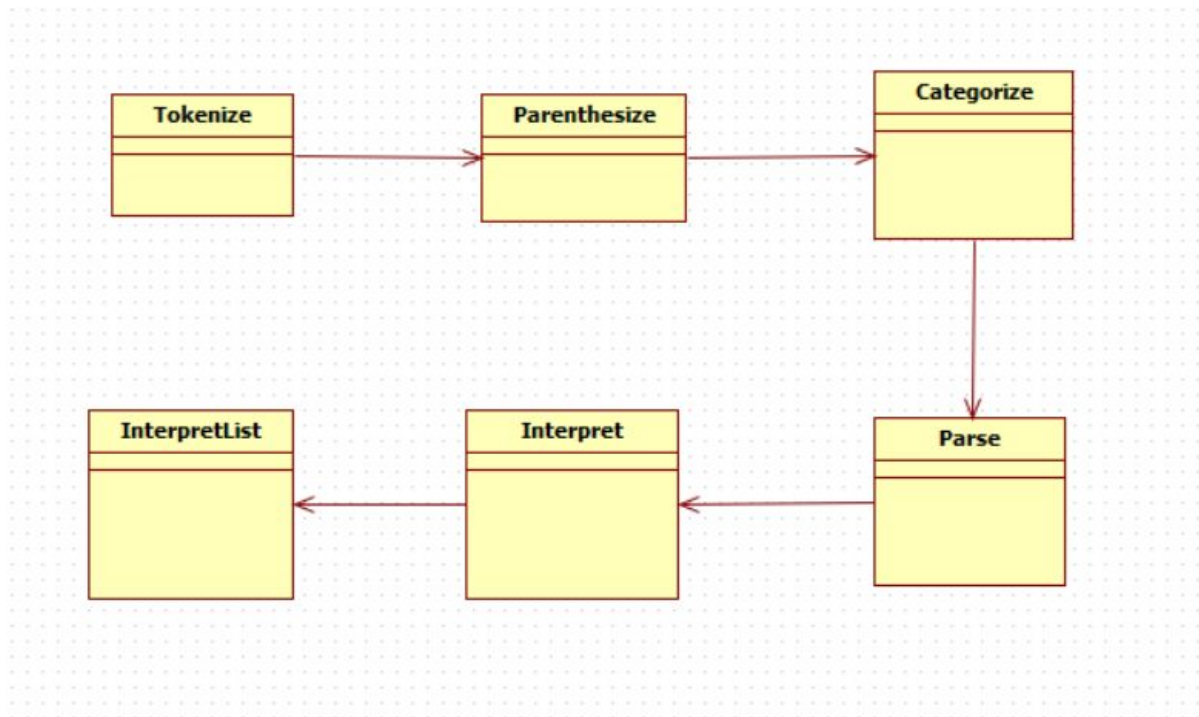
```
Portacle
File Edit Options Buffers Tools Development SLIME REPL Presentations Lisp Trace Help

1 ;; Welcome to Portacle, the Portable Common Lisp Environment.
2 ;; For information on Portacle and how to use it, please read
3 ;; https://portacle.github.io or \*portacle-help\*
4 ;; To report problems and to get help with issues, please visit
5 ;; https://github.com/portacle/portacle/issues
6 ;; Portacle is currently running SLIME, but you can
7 ;; Switch to SLY
8 ;; You should also configure Portacle with the
9 ;; First-time setup
10 ;;
11 ;; You can use this buffer for notes and tinkering with code.
12

*scratch* 1 All LF UTF-8 Lisp
1 ; SLIME 2.23
2 CL-USER> (format t "Instalacion Completa")
3 Instalacion Completa
4 NIL
5 CL-USER>

*slime-repl sbcl* 5 All LF UTF-8 REPL
1 ; SLIME 2.23
2 CL-USER> (format t "Instalacion Completa")
3 Instalacion Completa
4 NIL
5 CL-USER>
```

## UML de Intérprete



**Tokenize():** Se encarga de leer el archivo como string y añade espacios entre paréntesis y separa cuando haya espacios vacíos.

**Parenthesize():** Agarra los tokens que realizó el Tokenize() y crea un Nested Array que imita la estructura de un código LISP. Si el paréntesis es abierto, Parenthesize empieza a hacer una lista a partir de ese punto.

**Categorize():** Clase encargada de categorizar caracteres o lambdas en tipos.

**Parse():** Clase encargada de leer el programa y aplicar las primeras tres funciones (Tokenize(), Parenthesize() y Parse()).

**Interpret():** Se encarga de ejecutar el programa. Clase en donde las variables y sus valores se guardan.

**InterpretList():** Se encarga de llamar a la clase Interpret() para cada elemento.

## Link Repositorio Github

<https://github.com/KristenBrandt/Proyecto1>