

Universidad del Valle de Guatemala

Facultad de Ingeniería

Algoritmos y Estructuras de datos

Sección 20



FASE 2 PROYECTO 1

Gerardo Pineda 22880

Brandon Reyes 22992

Jose Alejandro 2210417

Pedro Guzmán 22111

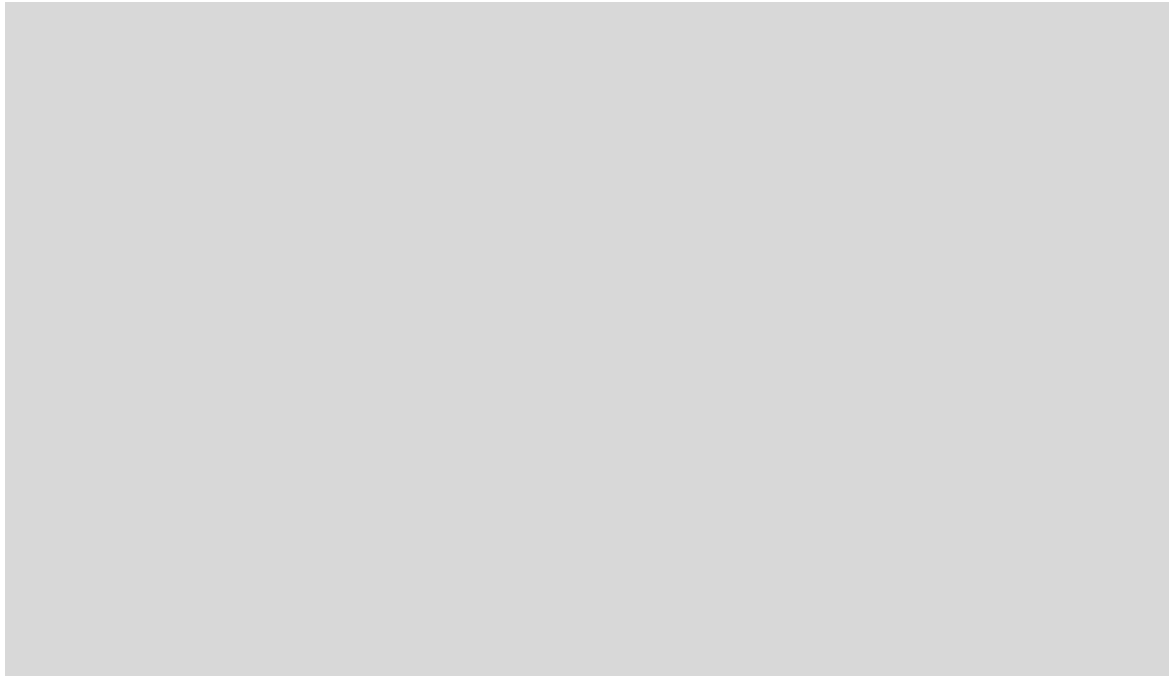
17 de Marzo del 2023, Guatemala de la Asunción

Link Repositorio – GitHub

<https://github.com/Geraxxx5/LispProject.git>

Link Video

Fase 2 Proyecto 1 - Grupo #6



Estructuras de Java Collection Framework

`import java.util.ArrayList`

Se utiliza para crear una lista dinámica (es decir, que puede crecer y reducir su tamaño durante la ejecución del programa) de elementos. Al utilizar `import java.util.ArrayList`, el programa puede crear objetos de la clase `ArrayList` y utilizar sus métodos para agregar, eliminar y manipular elementos en la lista. En `LispModel` se empleó para crear listas ya predefinidas.

- `EvaluateExpression.java`

```
2
3: import java.util.ArrayList;
4 import java.util.List;

186
187:         List<Object> varibaleValue = new ArrayList<>();
188         varibaleValue.add(String.valueOf(expressionList.get(2)));

296
297:         List<Object> list = new ArrayList<>();
298

321         if(!functions.functionExist(nameOfFunction)){
322:             List<Object> attributes = new ArrayList<>();
323             attributes.add(expressionList.get(2));

443         String var = (String) params.get(index);
444:         List<Object> value = new ArrayList<>();
445         Object expressionToEvaluate = expressionList.get(1+index);
```

- FunctionModel.java

```
Project\modelo\FunctionModel.java:
7
8: import java.util.ArrayList;
9 import java.util.HashMap;
```

- LisModel.java

```
Project\modelo\LispModel.java:
2
3: import java.util.ArrayList;
4 import java.util.Arrays;

11 */
12: List<Object> list = new ArrayList<>();
13 List<String> command = Arrays.asList("quote","atom","eval","setq","defvar","cond","'", "list","defun");
```

- ParseExpression.java

```
Project\parser\ParserExpression.java:
4 import java.io.FileReader;
4 import java.io.FileReader;
5: import java.util.ArrayList;
6 import java.util.List;

47 public static List<Object> parse(String input) {
48: List<Object> result = new ArrayList<>();
49 int i = 0;
```

```
import java.util.List;
```

Para usar la interfaz List y su conjunto de métodos para manejar listas de elementos en un programa, importe la clase List desde java.util, se puede definir métodos para agregar, eliminar, acceder y manipular datos en una lista como add(), get(), set(), size y get().

Se empleo:

- Clase EvaluateExpression.java

```
152 //return 7
153 //[[Eval [+ 5 2]]]
154 //return 7
155 Object second = expressionList.get(index:1);
156 //If second its not a List then return second
157 //If its a List evaluate the list and the return
158 if(second instanceof List){
159     List<Object> secondValue = (List<Object>) second;
160     Object response = evaluate(secondValue, variable, functions);
161
162     //If response is a List it means that is something to evaluate
163     //If is a String only return the String
164     if(response instanceof List){
165         List<Object> toEvaluate = (List<Object>) response;
166         return evaluate(toEvaluate, variable, functions);
167     }
168     return response;
169 }
170 }
171 }
172 return second;
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
```

- Clase LispModel.java

```

4  import java.util.Arrays;
5  import java.util.List;
6
7  Geraxxx5, 2 days ago | 1 author (Geraxxx5)
8  public class LispModel {
9
10     /**
11     */
12     List<Object> list = new ArrayList<>();
13     List<String> command = Arrays.asList(...a:"quote","atom","eval","setq","defvar","cond","\',"list","defun");
14     List<String> operators = Arrays.asList(...a:"+","-","/","**","^","sqrt");
15     List<String> conditionals = Arrays.asList(...a:"=", "<",">","equal");
16
17     /**
18     */
19     * @return
20     */
21     public List<String> getCommand(){
22         return command;
23     }
24
25     /**
26     */
27     * @return
28     */
29     public List<String> getConditionals(){
30         return conditionals;
31     }
32
33     /**
34     */
35     * @return
36     */
37     public List<String> getOperators(){
38         return operators;

```

- Clase: FunctionModel.java

```

15  public class FunctionModel {
16      HashMap<String, List<Object>> functions = new HashMap<>();
17
18      /**
19      */
20      * @param name
21      * @param Value
22      */
23      public void createNewFunction(String name, List<Object> Value){
24          functions.put(name, Value);
25      }
26
27      /**
28      */
29      * @param key
30      * @return boolean
31      */
32      public boolean funcionExist(String key){
33          return functions.containsKey(key);
34      }
35
36      /**
37      */
38      * @return map Function
39      */
40      public HashMap<String, List<Object>> getFunctions(){
41          return functions;
42      }

```

```

42  }

```

- Clase: VariableModel.java

```
4 import java.util.List;
5
6 Geraxoo5, 2 days ago | 1 author (Geraxoo5)
7 public class VariableModel {
8     HashMap<String, List<Object>> variable = new HashMap<>();
9
10    public void createNewVariable(String name, List<Object> Value){
11        variable.put(name, Value);
12    }
13
14    public void addTempValue(String name, String value){
15        variable.get(name).add(value);
16    }
17
18    public boolean varibaleExist(String name){
19        return variable.containsKey(name);
20    }
21
22    public Object lastValue(String key){
23        return variable.get(key).get(variable.get(key).size()-1);
24    }
25
26    public void removeTempValue(String name){
27        variable.get(name).remove(variable.size() -1);
28    }
29
30    public Object peekVariable(String key){
31        return variable.get(key).get(variable.get(key).size());
32    }
33
34    public HashMap<String, List<Object>> getVariables(){
35        return variable;
36    }
37 }
```

- ParseExpresion.java

```
47 public static List<Object> parse(String input) {
48     List<Object> result = new ArrayList<>();
49     int i = 0;
50     while (i < input.length()) {
51         char c = input.charAt(i);
52         if (c == '(') {
53             int end = findMatchingParen(input, i);
54             result.add(parse(input.substring(i + 1, end)));
55             i = end + 1;
56         } else if (c == ')') {
57             throw new IllegalArgumentException(s:"Unexpected ')'");
58         } else if (Character.isWhitespace(c)) {
59             i++;
60         } else {
61             int end = findEndOfAtom(input, i);
62             result.add(parseAtom(input.substring(i, end)));
63             i = end;
64         }
65     }
66     return result;
67 }
```

- Clase: ProyectoAlgoritmoLisp.java

```

11  /**
12   * @param args the command line arguments
13   */
14  Run | Debug
15  public static void main(String[] args) {
16      FunctionModel functionModel = new FunctionModel();
17      VariableModel variableModel = new VariableModel();
18      EvaluateExpression evaluate = new EvaluateExpression();
19      List<Object> expression = ParserExpression.readFileExpression(fileRoute:"datos.txt", isFile:true);
20      System.out.println(evaluate.evaluate(expression, variableModel, functionModel));
21  }
22  }
23

```

```
import java.util.HashMap;
```

Con la librería Java Util HashMap para utilizar el metodo Hashmap los datos se almacenan en una tabla Hash, donde calcula el índice de la tabla para cada elemento basado en su clave, luego el dato se almacena en la tabla. Cuando se necesita recuperar un elemento, se busca el índice en la tabla, pasando índice por índice hasta encontrar el índice y retornar el valor o value.

Se empleo en

- Clase FunctionModel.java

```

8  import java.util.ArrayList;
9  import java.util.HashMap;
10 import java.util.List;
11
12 /**
13  *
14  * @author
15  */
16 public class FunctionModel {
17     HashMap<String, List<Object>> functions = new HashMap<>();
18
19     /**
20     *
21     * @param name
22     * @param Value
23     */
24     public void createNewFunction(String name, List<Object> Value){
25         functions.put(name, Value);
26     }
27

```

- Clase VariableModel.java

```

4  import java.util.List;
5
6  Geraxxx5, 2 days ago | 1 author (Geraxxx5)
7  public class VariableModel {
8      HashMap<String, List<Object>> variable = new HashMap<>();
9
10     public void createNewVariable(String name, List<Object> Value){
11         variable.put(name, Value);
12     }
13
14     Geraxxx5, 2 days ago * create project lisp
15     public void addTempValue(String name,String value){
16         variable.get(name).add(value);
17     }
18

```

```
14     variable.get(name).add(value);
```

```
25 ✓ public void removeTempValue(String name){
26     variable.get(name).remove(variable.size() -1);
27 }
28
29 ✓ public Object peekVariable(String key){
30     return variable.get(key).get(variable.get(key).size());
31 }
32
33 ✓ public HashMap<String, List<Object>> getVariables(){ ←
34     return variable;
35 }
36
37
```