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 Framewrok

```
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.funcionExist(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

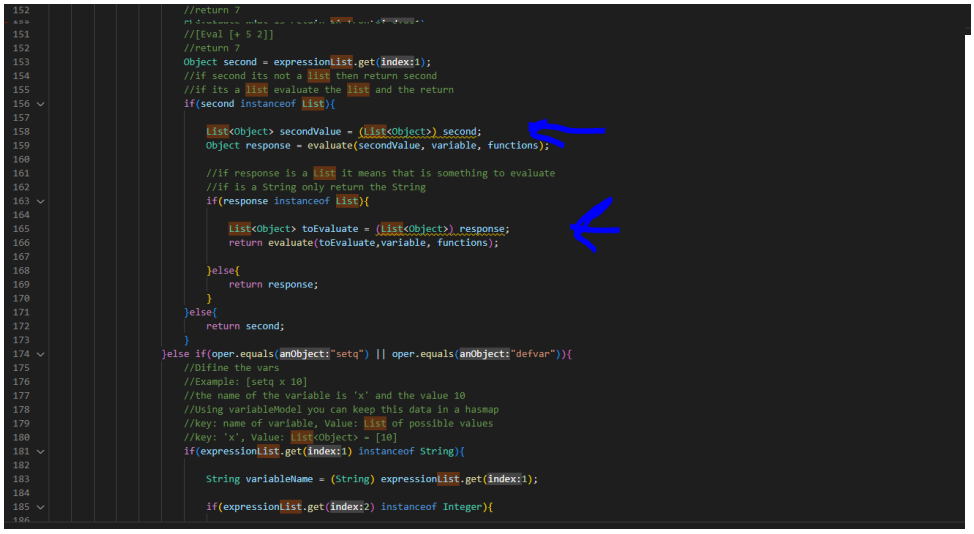
```
Project\parser\ParserExpression.java:
  4  import java.io.FileReader;
  5: import java.util.ArrayList;
  6  import java.util.List;

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

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

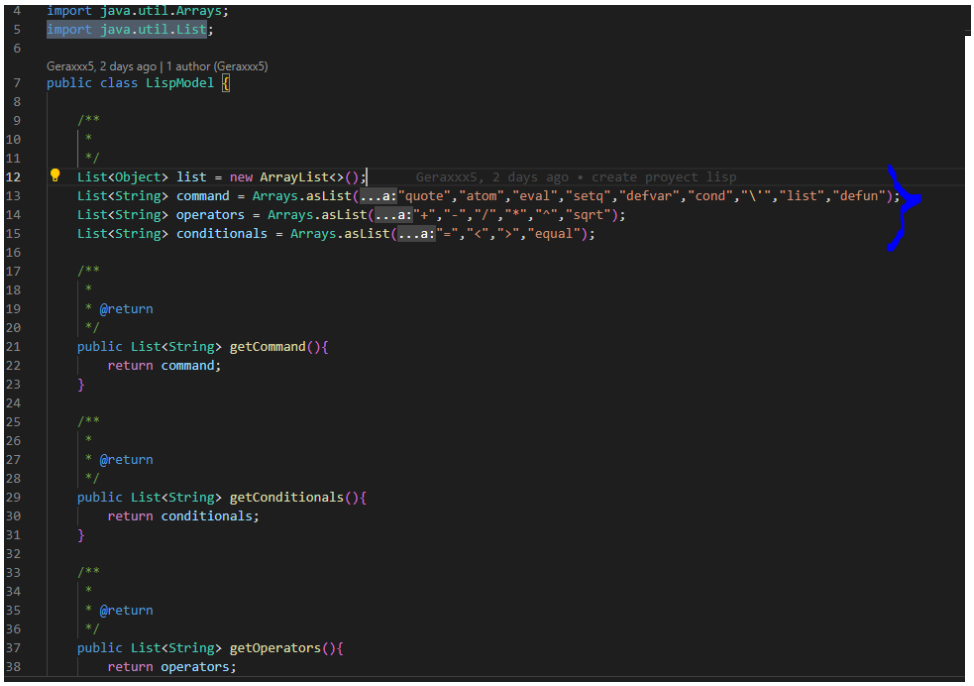
Para usar la interfaz List y su conjunto de métodos para manejar listas de elementos en un programa, importe la clase List desde java.util, se puede definir métodos para agregar, eliminar, acceder y manipular datos en una lista como add(), get(), set(), size y get().

Se empleo:

- Clase EvaluateExpression.java



- Clase LispModel.java



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

Geraxxx5, 2 days ago | 1 author (Geraxxx5)
public class LispModel {

    /**
     *
     */
    List<Object> list = new ArrayList<>();      Geraxxx5, 2 days ago • create proyect lisp
    List<String> command = Arrays.asList(...a:"quote","atom","eval","setq","defvar","cond","\'","list","defun");
    List<String> operators = Arrays.asList(...a:"+","-","/","*","^","sqrt");
    List<String> conditionals = Arrays.asList(...a:"=","<",">","equal");

    /**
     *
     * @return
     */
    public List<String> getCommand(){
        return command;
    }

    /**
     *
     * @return
     */
    public List<String> getConditionals(){
        return conditionals;
    }

    /**
     *
     * @return
     */
    public List<String> getOperators(){
        return operators;
```

- Clase: FunctionModel.java

```java
public class FunctionModel {
    HashMap<String, List<Object>> functions = new HashMap<>();

    /**
     *
     * @param name
     * @param Value
     */
    public void createNewFunction(String name, List<Object> Value){
        functions.put(name, Value);
    }

    /**
     *
     * @param key
     * @return boolean
     */
    public boolean funcionExist(String key){
        return functions.containsKey(key);
    }

    /**
     *
     * @return map Function
     */
    public HashMap<String, List<Object>> getFunctions(){
        return functions;
    }
```

```java
    }
```

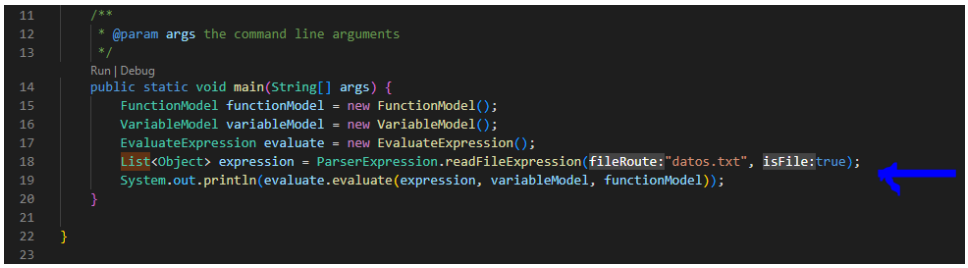
- Clase: VariableModel.java

```java
import java.util.List;

Geraxxx5, 2 days ago | 1 author (Geraxxx5)
public class VariableModel {
    HashMap<String, List<Object>> variable = new HashMap<>();

    public void createNewVariable(String name, List<Object> Value){
        variable.put(name, Value);
    }

    public void addTempValue(String name,String value){
        variable.get(name).add(value);
    }

    public boolean varibaleExist(String name){
        return variable.containsKey(name);
    }

    public Object lastValue(String key){
        return variable.get(key).get(variable.get(key).size()-1);
    }

    public void removeTempValue(String name){
        variable.get(name).remove(variable.size() -1);
    }

    public Object peekVariable(String key){
        return variable.get(key).get(variable.get(key).size());
    }

    public HashMap<String, List<Object>> getVariables(){
        return variable;
    }

}
```

- ParseExpresion.java

```java
    public static List<Object> parse(String input) {
        List<Object> result = new ArrayList<>();
        int i = 0;
        while (i < input.length()) {
            char c = input.charAt(i);
            if (c == '(') {
                int end = findMatchingParen(input, i);
                result.add(parse(input.substring(i + 1, end)));
                i = end + 1;
            } else if (c == ')') {
                throw new IllegalArgumentException(s:"Unexpected ')'");
            } else if (Character.isWhitespace(c)) {
                i++;
            } else {
                int end = findEndOfAtom(input, i);
                result.add(parseAtom(input.substring(i, end)));
                i = end;
            }
        }
        return result;
    }
```

- Clase: ProyectoAlgoritmoLisp.java



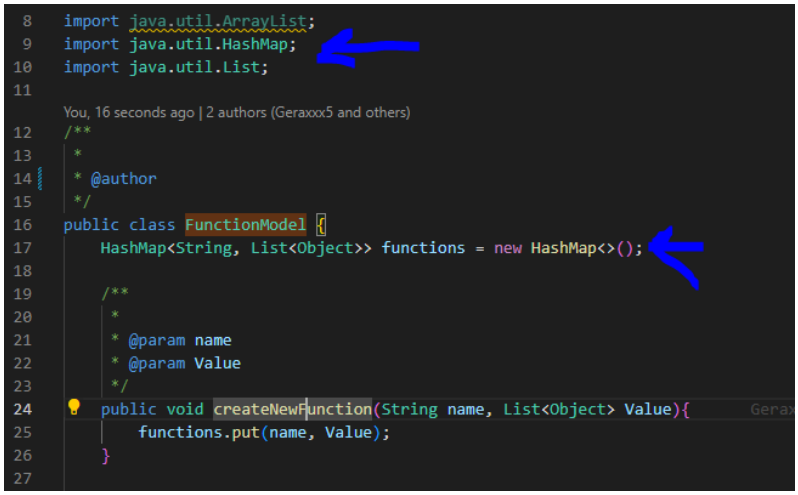
```
    /**
     * @param args the command line arguments
     */
    Run | Debug
    public static void main(String[] args) {
        FunctionModel functionModel = new FunctionModel();
        VariableModel variableModel = new VariableModel();
        EvaluateExpression evaluate = new EvaluateExpression();
        List<Object> expression = ParserExpression.readFileExpression(fileRoute:"datos.txt", isFile:true);
        System.out.println(evaluate.evaluate(expression, variableModel, functionModel));
    }

}
```

```
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.

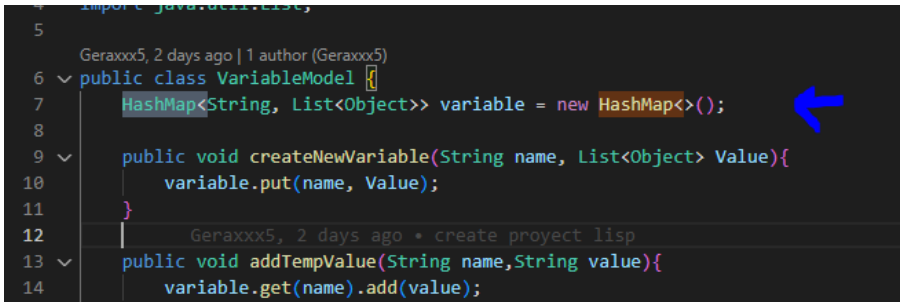
Se empleo en

- Clase FunctionModel.java

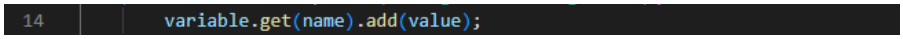


```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

You, 16 seconds ago | 2 authors (Geraxxx5 and others)
/**
 *
 * @author
 */
public class FunctionModel {
    HashMap<String, List<Object>> functions = new HashMap<>();

    /**
     *
     * @param name
     * @param Value
     */
    public void createNewFunction(String name, List<Object> Value){
        functions.put(name, Value);
    }

```

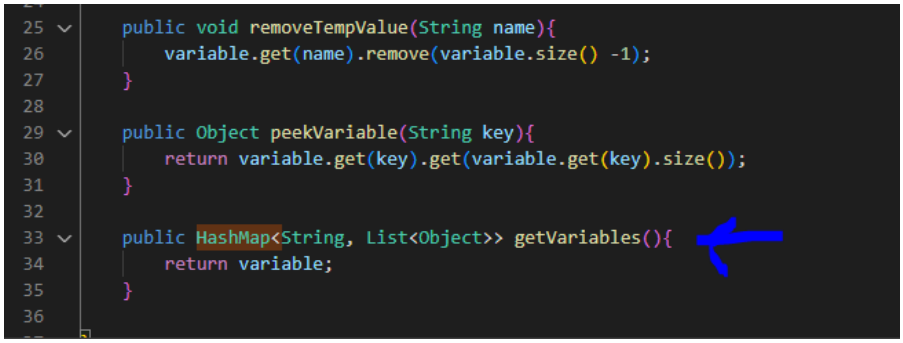
- Clase VariableModel.java



```
Geraxxx5, 2 days ago | 1 author (Geraxxx5)
public class VariableModel {
    HashMap<String, List<Object>> variable = new HashMap<>();

    public void createNewVariable(String name, List<Object> Value){
        variable.put(name, Value);
    }
        Geraxxx5, 2 days ago • create proyect lisp
    public void addTempValue(String name,String value){
        variable.get(name).add(value);
```



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


public void removeTempValue(String name){
variable.get(name).remove(variable.size() -1);
}
public Object peekVariable(String key){
return variable.get(key).get(variable.get(key).size());
}
public HashMap<String, List<Object>> getVariables(){
return variable;
}