

# Looper.java

```

1 package xyz.amtstl.soup.logic;
2
3 import xyz.amtstl.soup.Soup;
4
5
6
7
8 public class Looper {
9     private static int groundState = 0;
10
11     /**
12      * Executes a new loop sequence
13      * @param minBound starting point
14      * @param maxBound loop will continue until
15      * @param cache line of code from program
16      * @throws NumberFormatException
17      * @throws SoupVariableException
18      * @throws SoupSyntaxException
19      * @throws SoupFunctionNotDeclaredException
20      */
21     public static void execNewForLoop(int minBound, int maxBound, String cache, String
direction) throws NumberFormatException, SoupVariableException, SoupSyntaxException,
SoupFunctionNotDeclaredException {
22         groundState = Soup.getMainLogic().getIndex();
23         for (int e = minBound; e < maxBound; e++) {
24
25             for (int i = groundState; i < cache.length(); i++) {
26                 if (cache.charAt(i) == ';') {
27                     Soup.checkToken(i, cache, cache.charAt(i));
28                     i = Soup.getMainLogic().getIndex();
29                 }
30                 else {
31                     Soup.checkToken(i, cache, cache.charAt(i));
32                 }
33             }
34             Soup.getMainLogic().setIndex(groundState);
35             Soup.getMainLogic();
36             LogicController.v.insertVar((float) Float.valueOf(e), 1000);
37
38             if (LogicController.isBreak) {
39                 LogicController.setBreak(false);
40                 break;
41             }
42         }
43     }
44
45     /**
46      * Function for doing a decrementing for loop
47      * @param maxBound max bound
48      * @param minBound min bound
49      * @param cache
50      * @param direction depreciated
51      * @throws NumberFormatException
52      * @throws SoupVariableException
53      * @throws SoupSyntaxException
54      * @throws SoupFunctionNotDeclaredException
55      */
56     public static void execNewForLoopDecre(int maxBound, int minBound, String cache, String
direction) throws NumberFormatException, SoupVariableException, SoupSyntaxException,
SoupFunctionNotDeclaredException {

```

# Looper.java

```

57     groundState = Soup.getMainLogic().getIndex();
58     for (int e = maxBound; e > minBound; e--) {
59
60         for (int i = groundState; i < cache.length(); i++) {
61             if (cache.charAt(i) == ';') {
62                 Soup.checkToken(i, cache, cache.charAt(i));
63                 i = Soup.getMainLogic().getIndex();
64             }
65             else {
66                 Soup.checkToken(i, cache, cache.charAt(i));
67             }
68         }
69         Soup.getMainLogic().setIndex(groundState);
70         LogicController.v.insertVar((float) Float.valueOf(e), 1000);
71         if (LogicController.isBreak) {
72             LogicController.isBreak = false;
73             break;
74         }
75     }
76 }
77
78 /**
79  * Executes a new while loop
80  * @param cache
81  * @throws NumberFormatException
82  * @throws SoupVariableException
83  * @throws SoupSyntaxException
84  * @throws SoupFunctionNotDeclaredException
85  */
86 public static void execNewWhileLoop(String cache) throws NumberFormatException,
    SoupVariableException, SoupSyntaxException, SoupFunctionNotDeclaredException {
87     groundState = Soup.getMainLogic().getIndex();
88
89     int firstCondition = (int)Integer.valueOf((int)
    Float.parseFloat(LogicController.ns.get(0)));
90     int secondCondition = (int)Integer.valueOf((int)
    Float.parseFloat(LogicController.ns.get(1)));
91
92     while ((float)firstCondition == (float)secondCondition) {
93         // parse the line
94         for (int i = groundState; i < cache.length(); i++) {
95             if (cache.charAt(i) == ';') {
96                 Soup.checkToken(i, cache, cache.charAt(i));
97                 i = Soup.getMainLogic().getIndex();
98             }
99             else {
100                 Soup.checkToken(i, cache, cache.charAt(i));
101             }
102         }
103
104         Soup.getMainLogic().setIndex(groundState);
105         LogicController.ns = LogicController.p.parse(0, cache);
106         firstCondition = (int)Integer.valueOf((int)
    Float.parseFloat(LogicController.ns.get(0)));
107         secondCondition = (int)Integer.valueOf((int)
    Float.parseFloat(LogicController.ns.get(1)));
108         if (Soup.getMainLogic().isBreak) {

```

Looper.java

```

109         Soup.getMainLogic().isBreak = false;
110         break;
111     }
112 }
113 }
114
115 /**
116  * Executes a new new while not loop
117  * @param cache
118  * @throws NumberFormatException
119  * @throws SoupVariableException
120  * @throws SoupSyntaxException
121  * @throws SoupFunctionNotDeclaredException
122  */
123 public static void execNewWhileNotLoop(String cache) throws NumberFormatException,
    SoupVariableException, SoupSyntaxException, SoupFunctionNotDeclaredException {
124     groundState = Soup.getMainLogic().getIndex();
125
126     int firstCondition = (int)Integer.valueOf((int)
    Float.parseFloat(LogicController.ns.get(0)));
127     int secondCondition = (int)Integer.valueOf((int)
    Float.parseFloat(LogicController.ns.get(1)));
128
129     while ((float)firstCondition != (float)secondCondition) {
130         // parse the line
131         for (int i = groundState; i < cache.length(); i++) {
132             if (cache.charAt(i) == ';') {
133                 Soup.checkToken(i, cache, cache.charAt(i));
134                 i = Soup.getMainLogic().getIndex();
135             }
136             else {
137                 Soup.checkToken(i, cache, cache.charAt(i));
138             }
139         }
140
141         Soup.getMainLogic().setIndex(groundState);
142         LogicController.ns = LogicController.p.parse(0, cache);
143         firstCondition = (int)Integer.valueOf((int)
    Float.parseFloat(LogicController.ns.get(0)));
144         secondCondition = (int)Integer.valueOf((int)
    Float.parseFloat(LogicController.ns.get(1)));
145         if (Soup.getMainLogic().isBreak) {
146             Soup.getMainLogic().isBreak = false;
147             break;
148         }
149     }
150 }
151 }

```