```java
package xyz.amtstl.soup;

import java.io.BufferedReader;

/**
 * Soup
 * @author Alexander Christian Migala
 */
public class Soup {
    public static int lineNumber = 1;
    private static boolean isOneLine = false;

    // controllers
    private static LogicController logic = new LogicController();
    static LanguageDictionary lang = new LanguageDictionary();

    /**
     * Main thread marshal
     * @param args args from user
     * @throws Exception for forced exit
     */
    public static void main(String args[]) throws Exception {
        FileReader reader = null;
        BufferedReader buff = null;

        if (args[0].contains(".soup")) {

            try {
                reader = new FileReader(System.getProperty("user.dir") + "/" + args[0].toString());

                // pass flag
                try {
                    FlagController.passFlag(args[1].toString().toLowerCase());
                }
                catch (Exception e) {

                }
            }
            catch (Exception ex) {
                System.out.println("File not found! Are you sure it is in this folder?");
                System.exit(0);
            }
            buff = new BufferedReader(reader);
            }
            else {
                FlagController.execSoup(args[0]);

                isOneLine = true;

                try {
                    FlagController.passFlag(args[1].toString().toLowerCase());
                }
                catch (Exception e) {

                }
            }
```

```
65
66          /*
67           * ALWAYS USE BREAKS WHEN ADDING NEW TOKENS AND FUNCTSystem.outNS
68           *
69           */
70          while (true && isOneLine == false) {
71              final String cache = buff.readLine();
72
73              try {
74                  for (int i = 0; i < cache.length(); i++) {
75                      char c = cache.charAt(i);
76                      switch (c) {
77                      case '+' : // add two numbers
78                          logic.soupAdd(i, cache);
79                          i = logic.getIndex();
80                          break;
81                      case '_' : // subtract two numbers
82                          logic.soupSubtract(i, cache);
83                          i = logic.getIndex();
84                          break;
85                      case '*' : // multiply two numbers
86                          logic.soupMultiply(i, cache);
87                          i = logic.getIndex();
88                          break;
89                      case '%' : // divide two numbers
90                          logic.soupDivide(i, cache);
91                          i = logic.getIndex();
92                          break;
93                      case '^' : // pow one number
94                          logic.soupPow(i, cache);
95                          i = logic.getIndex();
96                          break;
97                      case '#' : // base 10 logarithm
98                          logic.soupLog(i, cache);
99                          i = logic.getIndex();
100                         break;
101                     case '@' : // break soup
102                         System.out.println("Soup exiting with code 2 (requested per
    program)");
103                         System.exit(0);
104                         break;
105                     case 'A' : // area
106                         logic.soupArea(i, cache);
107                         i = logic.getIndex();
108                         break;
109                     case '=' : // basic if statement
110                         logic.soupIf(i, cache);
111                         i = logic.getIndex();
112                         break;
113                     case 'P' : // print line
114                         logic.soupPrint(i, cache);
115                         i = logic.getIndex();
116                         break;
117                     case ';' : // extension of if
118                         logic.soupIfDo(i, cache);
119                         i = logic.getIndex();
120                         break;
```

```
121                     case ':' : // stores last result
122                         logic.soupStoreVar(i, cache);
123                         i = logic.getIndex();
124                         break;
125                     case 'V': // gets a variable
126                         logic.soupRetrieveVar(i, cache);
127                         i = logic.getIndex();
128                         break;
129                     case 'I': // gets var from user and stores it
130                         logic.soupStoreUserIn(i, cache);
131                         i = logic.getIndex();
132                         break;
133                     case '$' : // trigonometric functions
134                         logic.soupTrig(i, cache);
135                         i = logic.getIndex();
136                         break;
137                     case '|' : // absolute value
138                         logic.soupAbs(i, cache);
139                         i = logic.getIndex();
140                         break;
141                     case '?' : // round number to int
142                         logic.soupRound(i, cache);
143                         i = logic.getIndex();
144                         break;
145                     case '&' : // square root
146                         logic.soupSquareRoot(i, cache);
147                         i = logic.getIndex();
148                         break;
149                     case 'R' : // random number generator
150                         logic.soupRandomNum(i, cache);
151                         i = logic.getIndex();
152                         break;
153                     case 'H' : // html generator
154                         logic.soupHTMLHandler(i, cache);
155                         i = logic.getIndex();
156                         break;
157                     case '~': // stores a single variable
158                         logic.soupStoreSingle(i, cache);
159                         i = logic.getIndex();
160                         break;
161                     case '/': // comments
162                         logic.soupComment(i, cache);
163                         i = logic.getIndex();
164                         break;
165                     case '[' : // loop
166                         logic.soupForLoop(i, cache);
167                         i = logic.getIndex();
168                         break;
169                     case ']' :
170                         break;
171                     case 'W' : // while loop
172                         logic.soupWhileLoop(i, cache);
173                         i = logic.getIndex();
174                         break;
175                     case 'D' : // for decrement
176                         logic.soupForLoopDecre(i, cache);
177                         i = logic.getIndex();
```

```
178                     break;
179                 case '<' : // less than if
180                     logic.soupIfLessThan(i, cache);
181                     i = logic.getIndex();
182                     break;
183                 case '>' : // greater than if
184                     logic.soupIfGreaterThan(i, cache);
185                     i = logic.getIndex();
186                     break;
187                 case 'X' : // breaks loop
188                     logic.soupBreakLoop();
189                     i = logic.getIndex();
190                     break;
191                 case 'N' : // while not
192                     logic.soupWhileNotLoop(i, cache);
193                     i = logic.getIndex();
194                     break;
195                 case 'S' : // store a function
196                     logic.soupStoreFunction(i, cache);
197                     i  = logic.getIndex();
198                     break;
199                 case 'F' :
200                     logic.soupGetFunction(i, cache);
201                     i = logic.getIndex();
202                     break;
203                 case '.' : // like a semicolon
204                     break;
205                 case ' ': // space nullifier
206                     break;
207                 case ')' :
208                     break;
209                 case '-' :
210                     break;
211                 default :
212                     throw new SoupSyntaxException(cache.charAt(i), i+1, lineNumber);
213                 }
214 
215                 if (FlagController.getPrintIndex()) {
216                     System.out.println("Current Index: " + String.valueOf(i));
217                 }
218             }
219         } catch (NullPointerException ex) {
220             System.exit(0);
221         }
222         lineNumber++;
223         logic.setIndex(0);
224     }
225 }
226 
227 /**
228  * Did the corresponding function per character
229  * @param c
230  * @param i
231  * @param cache
232  * @throws NumberFormatException
233  * @throws SoupVariableException
234  * @throws SoupSyntaxException
```

```
235      * @throws SoupFunctionNotDeclaredException
236      */
237     public static void parseFunc(char c, int i, String cache) throws NumberFormatException,
    SoupVariableException, SoupSyntaxException, SoupFunctionNotDeclaredException {
238         switch (c) {
239         case '+' : // add two numbers
240             logic.soupAdd(i, cache);
241             break;
242         case '_' : // subtract two numbers
243             logic.soupSubtract(i, cache);
244             break;
245         case '@' : // break soup
246             System.out.println("Soup exiting with code 2 (requested per program)");
247             System.exit(0);
248             break;
249         case '*' : // multiply two numbers
250             logic.soupMultiply(i, cache);
251             break;
252         case '%' : // divide two numbers
253             logic.soupDivide(i, cache);
254             break;
255         case '^' : // pow one number
256             logic.soupPow(i, cache);
257             break;
258         case '#' : // basic logarithm
259             logic.soupLog(i, cache);
260             break;
261         case 'A' : // area
262             logic.soupArea(i, cache);
263             break;
264         case '=' : // basic if statement
265             logic.soupIf(i, cache);
266             break;
267         case ';' : // extension of if
268             logic.soupIfDo(i, cache);
269             break;
270         case ':' : // stores last result
271             logic.soupStoreVar(i, cache);
272             break;
273         case 'V': // gets a variable
274             logic.soupRetrieveVar(i, cache);
275             break;
276         case 'I': // gets var from user and stores it
277             logic.soupStoreUserIn(i, cache);
278             break;
279         case '$' : // trig
280             logic.soupTrig(i, cache);
281             break;
282         case '|' : // absolute value
283             logic.soupAbs(i, cache);
284             break;
285         case '?' : // round number to int
286             logic.soupRound(i, cache);
287             break;
288         case '&' : // square root
289             logic.soupSquareRoot(i, cache);
290             break;
```

```
291         case 'R' : // random number generator
292             logic.soupRandomNum(i, cache);
293             break;
294         case 'H' : // html generator
295             logic.soupHTMLHandler(i, cache);
296             break;
297         case '~': // stores a single variable
298             logic.soupStoreSingle(i, cache);
299             break;
300         case '/': // comments
301             logic.soupComment(i, cache);
302             break;
303         case '[' : // loop
304             logic.soupForLoop(i, cache);
305             break;
306         case '.' : // like a semicolon
307             break;
308         case 'D' : // for decrement
309             logic.soupForLoopDecre(i, cache);
310             break;
311         case ']' :
312             break;
313         case ' ': // space nullifier
314             break;
315         case 'P':
316             logic.soupPrint(i, cache);
317             break;
318         case '<' : // less than if
319             logic.soupIfLessThan(i, cache);
320             break;
321         case '>' : // gretaer than if
322             logic.soupIfGreaterThan(i, cache);
323             break;
324         case 'W' :
325             logic.soupWhileLoop(i, cache);
326             break;
327         case 'N' : // while not
328             logic.soupWhileNotLoop(i, cache);
329             break;
330         case 'X' : // breaks loop
331             logic.soupBreakLoop();
332             break;
333         case ')' :
334             break;
335         case '-' :
336             break;
337         case 'S' : // store a function
338             logic.soupStoreFunction(i, cache);
339             break;
340         case 'F' :
341             logic.soupGetFunction(i, cache);
342             break;
343         default :
344             throw new SoupSyntaxException(cache.charAt(i), i+1, lineNumber);
345         }
346     }
347
```

```
348     /**
349      * Checks tokens
350      * @param i
351      * @param cache
352      * @param c
353      * @throws NumberFormatException
354      * @throws SoupVariableException
355      * @throws SoupSyntaxException
356      * @throws SoupFunctionNotDeclaredException
357      */
358     public static void checkToken(int i, String cache, char c) throws NumberFormatException,
    SoupVariableException, SoupSyntaxException, SoupFunctionNotDeclaredException {
359         for (int e = 0; e < lang.LanguageTokens.size(); e++) {
360             if (cache.charAt(i) == lang.LanguageTokens.get(e)) {
361                 parseFunc(c, i, cache);
362             }
363         }
364     }
365 
366     public static LogicController getMainLogic() {
367         return logic;
368     }
369 }
```