```java
package xyz.amtstl.soup;

import java.io.BufferedReader;

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

    // controllers
    public static LogicController logic = new LogicController();
    public static LanguageDictionary lang = new LanguageDictionary();
    public static FlagController flags = new FlagController();

    /**
     * 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) {

                }
            }

        /*
         * ALWAYS USE BREAKS WHEN ADDING NEW TOKENS AND FUNCTIONS
         *
         */
        while (true && isOneLine == false) {
            final String cache = buff.readLine();

            try {
                for (int i = 0; i < cache.length(); i++) {
                    char c = cache.charAt(i);
                    switch (c) {
                    case '+' : // add two numbers
                        LogicController.soupAdd(i, cache);
                        i = LogicController.index;
                        break;
                    case '_' : // subtract two numbers
                        LogicController.soupSubtract(i, cache);
                        i = LogicController.index;
                        break;
                    case '*' : // multiply two numbers
                        LogicController.soupMultiply(i, cache);
                        i = LogicController.index;
                        break;
                    case '%' : // divide two numbers
                        LogicController.soupDivide(i, cache);
                        i = LogicController.index;
                        break;
                    case '^' : // pow one number
                        LogicController.soupPow(i, cache);
                        i = LogicController.index;
                        break;
                    case '#' : // base 10 logarithm
                        LogicController.soupLog(i, cache);
                        i = LogicController.index;
                        break;
                    case '@' : // break soup
                        System.out.println("Soup exiting with code 2 (requested per
program)");
                        System.exit(0);
                        break;
                    case 'A' : // area
                        LogicController.soupArea(i, cache);
                        i = LogicController.index;
                        break;
                    case '=' : // basic if statement
                        LogicController.soupIf(i, cache);
                        i = LogicController.index;
                        break;
                    case 'P' : // print line
                        LogicController.soupPrint(i, cache);
                        i = LogicController.index;
                        break;
                    case ';' : // extension of if
                        LogicController.soupIfDo(i, cache);
                        i = LogicController.index;
                        break;
                    case ':' : // stores last result
                        LogicController.soupStoreVar(i, cache);
                        i = LogicController.index;
                        break;
                    case 'V': // gets a variable
                        LogicController.soupRetrieveVar(i, cache);
                        i = LogicController.index;
                        break;
                    case 'I': // gets var from user and stores it
                        LogicController.soupStoreUserIn(i, cache);
                        i = LogicController.index;
                        break;
                    case '$' : // trigonometric functions
                        LogicController.soupTrig(i, cache);
                        i = LogicController.index;
                        break;
                    case '|' : // absolute value
                        LogicController.soupAbs(i, cache);
                        i = LogicController.index;
                        break;
                    case '?' : // round number to int
                        LogicController.soupRound(i, cache);
                        i = LogicController.index;
                        break;
                    case '&' : // square root
                        LogicController.soupSquareRoot(i, cache);
                        i = LogicController.index;
                        break;
                    case 'R' : // random number generator
                        LogicController.soupRandomNum(i, cache);
                        i = LogicController.index;
                        break;
                    case 'H' : // html generator
                        LogicController.soupHTMLHandler(i, cache);
                        i = LogicController.index;
                        break;
                    case '~': // stores a single variable
                        LogicController.soupStoreSingle(i, cache);
                        i = LogicController.index;
                        break;
                    case '/': // comments
                        LogicController.soupComment(i, cache);
                        i = LogicController.index;
                        break;
                    case '[' : // loop
                        LogicController.soupForLoop(i, cache);
                        i = LogicController.index;
                        break;
                    case ']' :
                        break;
                    case 'W' : // while loop
                        LogicController.soupWhileLoop(i, cache);
                        i = LogicController.index;
                        break;
                    case 'D' : // for decrement
                        LogicController.soupForLoopDecre(i, cache);
```

```
                        i = LogicController.index;
                        break;
                    case '<' : // less than if
                        LogicController.soupIfLessThan(i, cache);
                        i = LogicController.index;
                        break;
                    case '>' : // greater than if
                        LogicController.soupIfGreaterThan(i, cache);
                        i = LogicController.index;
                        break;
                    case 'X' : // breaks loop
                        LogicController.soupBreakLoop();
                        i = LogicController.index;
                        break;
                    case 'N' : // while not
                        LogicController.soupWhileNotLoop(i, cache);
                        i = LogicController.index;
                        break;
                    case 'S' : // store a function
                        LogicController.soupStoreFunction(i, cache);
                        i  = LogicController.index;
                        break;
                    case 'F' :
                        LogicController.soupGetFunction(i, cache);
                        i = LogicController.index;
                        break;
                    case '.' : // like a semicolon
                        break;
                    case ' ': // space nullifier
                        break;
                    case ')' :
                        break;
                    case '-' :
                        break;
                    default :
                        throw new SoupSyntaxException(cache.charAt(i), i+1, lineNumber);
                    }

                    if (FlagController.printIndex) {
                        System.out.println("Current Index: " + String.valueOf(i));
                    }
                }
            } catch (NullPointerException ex) {
                System.exit(0);
            }
            lineNumber++;
            LogicController.index = 0;
        }
    }

    /**
     * Did the corresponding function per character
     * @param c
     * @param i
     * @param cache
     * @throws NumberFormatException
     * @throws SoupVariableException
     * @throws SoupSyntaxException
     * @throws SoupFunctionNotDeclaredException
     */
    public static void parseFunc(char c, int i, String cache) throws NumberFormatException,
SoupVariableException, SoupSyntaxException, SoupFunctionNotDeclaredException {
        switch (c) {
        case '+' : // add two numbers
            LogicController.soupAdd(i, cache);
            break;
        case '_' : // subtract two numbers
            LogicController.soupSubtract(i, cache);
            break;
        case '@' : // break soup
            System.out.println("Soup exiting with code 2 (requested per program)");
            System.exit(0);
            break;
        case '*' : // multiply two numbers
            LogicController.soupMultiply(i, cache);
            break;
        case '%' : // divide two numbers
            LogicController.soupDivide(i, cache);
            break;
        case '^' : // pow one number
            LogicController.soupPow(i, cache);
            break;
        case '#' : // basic logarithm
            LogicController.soupLog(i, cache);
            break;
        case 'A' : // area
            LogicController.soupArea(i, cache);
            break;
        case '=' : // basic if statement
            LogicController.soupIf(i, cache);
            break;
        case ';' : // extension of if
            LogicController.soupIfDo(i, cache);
            break;
        case ':' : // stores last result
            LogicController.soupStoreVar(i, cache);
            break;
        case 'V': // gets a variable
            LogicController.soupRetrieveVar(i, cache);
            break;
        case 'I': // gets var from user and stores it
            LogicController.soupStoreUserIn(i, cache);
            break;
        case '$' : // trig
            LogicController.soupTrig(i, cache);
            break;
        case '|' : // absolute value
            LogicController.soupAbs(i, cache);
            break;
        case '?' : // round number to int
            LogicController.soupRound(i, cache);
            break;
        case '&' : // square root
            LogicController.soupSquareRoot(i, cache);
```

```java
                break;
            case 'R' : // random number generator
                LogicController.soupRandomNum(i, cache);
                break;
            case 'H' : // html generator
                LogicController.soupHTMLHandler(i, cache);
                break;
            case '~': // stores a single variable
                LogicController.soupStoreSingle(i, cache);
                break;
            case '/': // comments
                LogicController.soupComment(i, cache);
                break;
            case '[' : // loop
                LogicController.soupForLoop(i, cache);
                break;
            case '.' : // like a semicolon
                break;
            case 'D' : // for decrement
                LogicController.soupForLoopDecre(i, cache);
                break;
            case ']' :
                break;
            case ' ': // space nullifier
                break;
            case 'P':
                LogicController.soupPrint(i, cache);
                break;
            case '<' : // less than if
                LogicController.soupIfLessThan(i, cache);
                break;
            case '>' : // gretaer than if
                LogicController.soupIfGreaterThan(i, cache);
                break;
            case 'W' :
                LogicController.soupWhileLoop(i, cache);
                break;
            case 'N' : // while not
                LogicController.soupWhileNotLoop(i, cache);
                break;
            case 'X' : // breaks loop
                LogicController.soupBreakLoop();
                break;
            case ')' :
                break;
            case '-' :
                break;
            case 'S' : // store a function
                LogicController.soupStoreFunction(i, cache);
                break;
            case 'F' :
                LogicController.soupGetFunction(i, cache);
                break;
            default :
                throw new SoupSyntaxException(cache.charAt(i), i+1, lineNumber);
            }
        }
```

```

    /**
     * Checks tokens
     * @param i
     * @param cache
     * @param c
     * @throws NumberFormatException
     * @throws SoupVariableException
     * @throws SoupSyntaxException
     * @throws SoupFunctionNotDeclaredException
     */
    public static void checkToken(int i, String cache, char c) throws NumberFormatException, SoupVariableException, SoupSyntaxException, SoupFunctionNotDeclaredException {
        for (int e = 0; e < LanguageDictionary.languageTokens.size(); e++) {
            if (cache.charAt(i) == LanguageDictionary.languageTokens.get(e)) {
                parseFunc(c, i, cache);
            }
        }
    }
}
```