```
// src/Calculator.java

import java.util.Scanner;
import com.github.sahasatvik.math.*;

/**
 * Calculator is a simple java application which parses mathematical expressions and evaluates
 * the result. Calculator can parse arithmetic operators as well as some functions, store variables,
 * and carry out some pre-defined commands.
 *
 * A complete manual on how to use this application can be found in the README files, or by entering
 * <code>/help</code> during runtime.
 *
 * A continuously updated version of this project can be found online at my
 * <a href="http://github.com/sahasatvik/Calculator">Github repository</a>.
 *
 *      @author         Satvik Saha
 *      @version        1.0, 17/10/2016
 *      @see            <a href="http://github.com/sahasatvik/Calculator">
 *                              http://github.com/sahasatvik/Calculator
 *                      </a>
 *      @see            com.github.sahasatvik.math.ExpressionParser
 *      @since          1.0
 */

public class Calculator {

        /**
         * Regex which matches a command. This is simply a string of characters following a forward
         * slash (<code>/</code>).
         */
        public static String commandRegex = "(\\s+)?(/)(.*)";

        /**
         * Stores the previously calculated answer. This can be retrieved during runtime as a variable.
         */
        public static String previousAns;

        /**
         * ExpressionParser object which contains methods for parsing arithmetic expressions.
         */
        public static ExpressionParser expParser;

        /**
         * Main method of Calculator.
```

```
 *
 *      @param  args                    the command-line arguments supplied to Calculator
 *      @since  1.0
 */
public static void main (String[] args) {

        /* Store the expression entered by the user */
        String expression;
        /* If the expression is a command, store it here */
        String command;
        /* Initialize the previous answer cache */
        previousAns = "";

        /* Startup message */
        System.out.print("\nCalculator by Satvik Saha");
        System.out.print("\n-------------------------");
        System.out.print("\n   An up-to-date version of Calculator can be found at : ");
        System.out.print("\n      https://github.com/sahasatvik/Calculator");
        System.out.print("\n");
        System.out.print("\n   Type  /help  to read a guide on how to use this program.");
        System.out.print("\n");

        /* Setup the input system */
        Scanner inp = new Scanner(System.in);

        /* Setup the ExpressionParser, which will parse the input */
        expParser = new ExpressionParser(32);
        /* Add some commonly used mathematical constants */
        expParser.addVariable("e", ("" + Math.E));
        expParser.addVariable("pi", ("" + Math.PI));
        expParser.addVariable("phi", ("" + (Math.sqrt(5.0) + 1.0) / 2.0));

        /* Start the input loop */
        while (true) {
                /* Display a simple prompt */
                System.out.print("\n?> ");

                /* Accept a line of input */
                expression = inp.nextLine().trim();

                /* Check whether the input is a command */
                if (expression.matches(commandRegex)) {
                        /* Extract the content of the command */
                        command = expression.substring(expression.indexOf("/") + 1).trim();

                        try {
```

```
                /* Parse the command */
                parseCommand(command);
        } catch (CommandNotFoundException e) {
                /* Display an error message if the command is not recognized */
                System.out.print("!> Command " + e.getCommand() + " not found !");
                System.out.print("\n   Try  /list  for a complete list of available commands.");
        }
        /* Go back to the start of the loop and get a new line of input */
        continue;
}

/* Enclose the expression processing section within a 'try' block */
try {
        /* Evaluate the expression and store it in the cache */
        previousAns = evaluate(expression);
        /* Display the result */
        System.out.print("=> " + previousAns);
} catch (NullExpressionException e) {
        /* Catch empty input */
        System.out.print("!> Null Expression !");
} catch (MissingOperandException e) {
        /* Catch input missing an operand */
        System.out.print("!> Missing operand to " + e.getOperator() + " !");
} catch (VariableNotFoundException e) {
        /* Catch input containing undefined variables */
        System.out.print("!> Variable " + e.getVar() + " not found !");
        System.out.print("\n   Try  /list vars  for a complete list of available variables.");
} catch (FunctionNotFoundException e) {
        /* Catch input containing unrecognized functions */
        System.out.print("!> Function " + e.getFunc() + "[] not found !");
        System.out.print("\n   Try  /list funcs  for a complete list of available functions.");
} catch (UnmatchedBracketsException e) {
        /* Catch input with unclosed brackets */
        System.out.print("!> Unmatched brackets in expression !");

        /* Display the expression entered */
        System.out.print("\n   " + e.getFaultyExpression());
        System.out.print("\n   ");

        /* Display a character pointing to where the unmatched bracket is */
        for (int i = 0; i < e.getIndexOfBracket(); i++) {
                System.out.print(" ");
        }
        System.out.print("^");
} catch (ExpressionParserException e) {
        /* Catch any other Exception encountered while parsing */
```

```
                        System.out.print("!> Invalid Expression !");
                }
        }
}

/**
 * Evaluate a mathematical expression and return the result.
 *
 *      @param  exp                     the expression to be parsed
 *      @return                         the evaluated result
 *      @throws com.github.sahasatvik.math.ExpressionParserException
 *                                      thrown when an exception is encountered while parsing
 *                                      the expression
 *      @see    com.github.sahasatvik.math.ExpressionParser#evaluate(String)
 *      @since  1.0
 */

public static String evaluate (String exp) throws ExpressionParserException {
        /* Substitute all indstances of '<ans>' with the previously evaluated answer in the cache */
        exp = exp.replaceAll("<(\\s+)?ans(\\s+)?>", previousAns);
        /* Return the expression as evaluated by the ExpressionParser library */
        return expParser.evaluate(exp);
}

/**
 * Parses a command intended for the Calculator shell.
 *
 *      @param  command                 the command to be parsed
 *      @throws CommandNotFoundException
 *                                      thrown when an unrecognized command is passed here
 *      @see    CommandNotFoundException
 *      @since  1.0
 */

public static void parseCommand (String command) throws CommandNotFoundException {
        if (command.equals("exit")) {
                /* If the commmand is '/exit', display an exit message exit the runtime */
                System.out.print("$> Exiting !");
                System.exit(0);
        } else if (command.equals("help")) {
                /* Display some general helptext */
                System.out.print("$> Calculator Helptext");
                System.out.print("\n   ~~~~~~~~~~~~~~~~~~");
                System.out.print("\n       Welcome to  'Calculator', a simple java application written to");
                System.out.print("\n    evaluate mathematical expressions.");
                System.out.print("\n       This program displays a prompt (?>), after which you can enter");
```

```
        System.out.print("\n    a mathematical expression. 'Calculator' will display the result,");
        System.out.print("\n    or point out errors in the expression.");
        System.out.print("\n");
        System.out.print("\n        'Calculator' can evaluate simple arithmetic expressions, using the");
        System.out.print("\n    operators (+, -, *, /, ^(power)), as well as parenthesis ('(', ')').");
        System.out.print("\n    'Calculator' follows the BODMAS rule.");
        System.out.print("\n");
        System.out.print("\n        Following are some valid expressions : ");
        System.out.print("\n            1 + 1                   =>              2.0");
        System.out.print("\n            1 * (2 + 3)             =>              5.0");
        System.out.print("\n            10 * (64 ^ -0.5)        =>             1.25");
        System.out.print("\n");
        System.out.print("\n        For help on more advanced topics, try entering the following : ");
        System.out.print("\n            /help vars              >       help on Variables");
        System.out.print("\n            /help funcs             >       help on Functions");
        System.out.print("\n            /help cmds              >       help on Commands");
        System.out.print("\n");
        System.out.print("\n        Enter '/list' for a complete list of valid commands.");
} else if (command.equals("help vars")) {
        /* Display help on 'variables' */
        System.out.print("\n$> Variables");
        System.out.print("\n   `````````");
        System.out.print("\n        'Calculator' can also store user-defined variables.");
        System.out.print("\n    The syntax for assigning and using variables is as follows : ");
        System.out.print("\n            var = value             >       assign 'value' to 'var'");
        System.out.print("\n            <var>                   >       <var> will be replaced");
        System.out.print("\n                                            its value.");
        System.out.print("\n");
        System.out.print("\n        Following are some valid uses of variables : ");
        System.out.print("\n            x = 3                   =>              3.0");
        System.out.print("\n            y = <x> + 1             =>              4.0");
        System.out.print("\n            (<x>^2 + <y>^2)^0.5     =>              5.0 ");
        System.out.print("\n");
        System.out.print("\n        Nesting of assignments is also supported, as follows : ");
        System.out.print("\n            x = 1 + (y = 1)         =>              2.0");
        System.out.print("\n            <x>                     =>              2.0");
        System.out.print("\n            <y>                     =>              1.0");
        System.out.print("\n");
        System.out.print("\n        A special variable <ans> stores the previous expression.");
        System.out.print("\n    Thus, the following is valid : ");
        System.out.print("\n            1 * 2 * 3 * 4           =>             24.0");
        System.out.print("\n            <ans> * 5               =>            120.0");
        System.out.print("\n");
        System.out.print("\n        Enter '/list vars' for a list of stored variables.");
} else if (command.equals("help funcs")) {
        /* Display help on 'funcitons' */
```

```
        System.out.print("\n$> Functions");
        System.out.print("\n   `````````");
        System.out.print("\n        'Calculator' supports the use of some basic functions.");
        System.out.print("\n    They can be used with the following syntax : ");
        System.out.print("\n            fnc[ value ]           >       evaluate 'fnc' of 'value'");
        System.out.print("\n");
        System.out.print("\n        Following are some valid uses of functions : ");
        System.out.print("\n            sin[<pi> / 2]          =>              1.0");
        System.out.print("\n            1 + abs[2 - 3]         =>              2.0");
        System.out.print("\n            log[<e> ^ 3]           =>              3.0");
        System.out.print("\n");
        System.out.print("\n        Enter '/list funcs' for a list of valid functions.");
} else if (command.equals("help cmds")) {
        /* Display help on 'commands' */
        System.out.print("\n$> Commands");
        System.out.print("\n   `````````");
        System.out.print("\n        'Calculator' interprets expressions starting with '/' as");
        System.out.print("\n    'commands'. These are special expressions which are not parsed as");
        System.out.print("\n    mathematical expressions, but as instructions to the 'Calculator'.");
        System.out.print("\n");
        System.out.print("\n        Enter '/list' for a complete list of valid commands.");
} else if (command.equals("list") || command.equals("list cmds")) {
        /* Display a list of valid, acceptable commands */
        System.out.print("$> Commands : \n");
        System.out.print("\n    /help                          >       general help");
        System.out.print("\n    /help vars                     >       help on Variables");
        System.out.print("\n    /help funcs                    >       help on Functions");
        System.out.print("\n    /help cmds                     >       help on Commands");
        System.out.print("\n    /list vars                     >       list variables");
        System.out.print("\n    /list funcs                    >       list functions");
        System.out.print("\n    /list cmds  or  /list          >       list commands");
        System.out.print("\n    /exit                          >       exit Calculator");
} else if (command.equals("list vars")) {
        /* Display a list of defined variables, currently stored in the ExpressionParser */
        System.out.print("$> Variables : \n");
        /* Loop through the variables in the array belonging to the ExpressionParser */
        for (int i = 0; i < expParser.numberOfVars; i++) {
                /* Pretty-print the variables */
                System.out.printf("%n\t%-16s=%30s", expParser.variables[i][0]
                                                 , expParser.variables[i][1]);
        }
        /* Display the previously evaluated answer as a special variable : 'ans' */
        System.out.printf("%n\t%-16s=%30s", "ans", previousAns);
} else if (command.equals("list funcs")) {
        /* Display a list of valid functions */
        System.out.print("$ Functions : \n");
```

```
                System.out.print("\n    abs[ x ]        >            absolute value of <x>");
                System.out.print("\n    exp[ x ]        >            exponent of <x> (<e> ^ <x>)");
                System.out.print("\n    log[ x ]        >            logarithm of <x> (base <e>)");
                System.out.print("\n    fct[ x ] or x!  >            factorial of <x>");
                System.out.print("\n    deg[ x ]        >            convert <x> to degrees from radians");
                System.out.print("\n    rad[ x ]        >            convert <x> to radians from degrees");
                System.out.print("\n                    _           ");
                System.out.print("\n    sin[ x ]         |          ");
                System.out.print("\n    cos[ x ]         |          ");
                System.out.print("\n    tan[ x ]          >         trigonometric functions");
                System.out.print("\n    csc[ x ]         |             ( <x> in radians )");
                System.out.print("\n    sec[ x ]         |          ");
                System.out.print("\n    ctn[ x ]         |          ");
                System.out.print("\n                    ~           ");
            } else {
                /* Throw an Exception if the command does not match any of the above */
                throw new CommandNotFoundException(command);
            }
        }
}
```