Assessment 1, Part 2 (Lexical & Syntax Analysis)

Organisation: This coursework assessment may be completed individually, or in pairs, and there is no penalty for working in a pair. You will hand work in as individuals but the handin procedure will allow you to say who you collaborated with (and you must do so!). *Pairs* only – not threesomes!

Hand in: The deadline and the electronic handin procedure, and exactly what you should hand-in, are documented online. Obviously you should change file and directory permissions while you are working so that your work is not visible to others – remember plagiarism carries severe penalties. Finally, guard your work, don't risk plagiarism charges by leaving a USB key with your work around, or "sharing answers" etc.

Weighting: This component is worth 60% of assessment 1 (or if you prefer, 7.2% of the entire module).

The task: MiniJava modification

Copy the MiniJava implementation directory to where you want to work, using:
`cp -R /soi/sw/courses/daveb/IN2009/minijava/chap4` (or download it from Cityspace)

Files `README` in the various directories give a brief description of the structure of the implementation. The MiniJava BNF and reference manual are online.

1. Add a Java-like `do`-statement to the MiniJava implementation

$$<\textit{Statement}> \; -> \; \textit{do} \; <\textit{Statement}> \; \textit{while} \; (<\textit{Exp}>) \; ;$$

2. In a similar way, add the Java-like try and throw statements defined below to the implementation:

$$<\textit{Statement}> \; -> \; \textit{try} \; \{ <\textit{Statement}> \} \; <\textit{Catch}>^* \; \textit{finally} \; \{ <\textit{Statement}> \}$$

$$<\textit{Statement}> \; -> \; \textit{throw} \; <\textit{Exp}> \; ;$$

$$<\textit{Catch}> \; -> \; \textit{catch} \; (<\textit{Type}> <\textit{id}>) \; \{ <\textit{Statement}> \}$$

   Note that the $<\textit{Catch}>$s sequence may be represented as a list in your abstract syntax (and hence programmed like the other lists).

You will need to add the new statements to the JavaCC specification (including appropriately adjusting the token regular expressions), so that it builds the correct abstract syntax trees, and write new appropriate abstract syntax tree classes. The pretty-printer will also need to be updated to appropriately print the new abstract syntaxes you have introduced.

Note: remember that these exercises deal only with lexical and syntax analysis and producing appropriate abstract syntax trees: you do not yet have to worry about how MiniJava programs are type-checked, execute or have code produced for them!