SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Reinhard Wilhelm
Prof. Dr. Sebastian Hack
Dipl.-Inform. Daniel Grund
Michael Jacobs, M.Sc.

Compiler Construction Project WS11/12

Project task D. Pretty Printing

Up to now your compiler can already lex and parse input as well as construct a corresponding parse tree. In this project phase you are to add two kinds of pretty printers to your compiler, i.e., functionality to generate nicely formatted source code from the parse tree. For printing this newly generated source code adhere to the Java formatting conventions, in general. However, do ignore the line size limits of these conventions.
The first pretty printer must produce *minimally parenthesized* output, like in the following example:

```
class HelloWorld {
    public int bar(int a, int b) {
        return c = a + b;
    }
    public static void main(String[] args) {
        System.out.println(43110 + 0);
        boolean b = true && !false;
        if (23 + 19 == (42 + 0) * 1)
            b = 0 < 1;
        else if (!true) {
            int x = 0;
            x = x + 1;
        } else {
            new HelloWorld().bar(0, -1);
        }
    }
    public int c;
}
```

The second pretty printer must produce *fully parenthesized* output, like in the following example:

```
class HelloWorld {
    public int bar(int a, int b) {
        return c = (a + b);
    }
    public static void main(String[] args) {
        (System.out).println(43110 + 0);
        boolean b = true && (!false);
        if ((23 + 19) == ((42 + 0) * 1))
            b = (0 < 1);
        else if (!true) {
            int x = 0;
            x = (x + 1);
        } else {
            (new HelloWorld()).bar(0, -1);
        }
    }
    public int c;
}
```

For both pretty printers:

- Print all classes in alphabetical order. Within each class, first print all methods in alphabetical order and then print all fields in alphabetical order.
- For inner statements, except for `if`-statements after an `else` and blocks, start a new line and increment the indentation level.
- To realize indentation use one tabulator per indentation level; do not use spaces.
- The use of whitespace within expressions can be extrapolated from the examples.
- If something is unclear refer to the examples and the provided reference files. Compare your output to the reference files via `diff`.

For the second pretty printer:

- Every subexpression must be parenthesized, except for literals (`0`, `false`, ...), identifiers, `this`, and the outermost subexpression.
- Write test cases that allow one to detect mis-constructed parse trees if one inspects the fully parenthesized output of `--ast`.

Additional technical requirements and restrictions: For a syntactically correct input program,

- `mjavac --ast [file]` must print its fully parenthesized version.
- `mjavac --astmin [file]` must print its minimally parenthesized version.

Both new switches must *augment* the `mjavac --parse` functionality, i.e., its acceptance and rejection behavior as well as the respective return codes. Please check in your solution into your repository until 2011-11-24, 12:00, noon.