MicroJava Quick Reference (thanks to Sacha Baebler)

Sample Program

```
program P
  final int size = 10;
  class Table {
    int[] pos;
    int[] neg;
  }
  Table val;
{
void main()
  int x, i;
  { //---------- initialize val ----------
    val = new Table;
    val.pos = new int[size];
    val.neg = new int[size];
    i = 0;
    while (i < size) {
      val.pos[i] = 0; val.neg[i] = 0; i = i + 1;
    }
    //---------- read values ----------
    read(x);
    while (x != 0) {
      if (x > 0)
        val.pos[x] = val.pos[x] + 1;
      else if (x < 0)
        val.neg[-x] = val.neg[-x] + 1;
      read(x);
    }
  }
}
```

Syntax

```
Program = "program" ident {ConstDecl | VarDecl | ClassDecl}
        "{" {MethodDecl} "}".
ConstDecl = "final" Type ident "=" (number | charConst) ";".
VarDecl = Type ident {"," ident } ";".
ClassDecl = "class" ident "{" {VarDecl} "}".
MethodDecl = (Type | "void") ident "(" [FormPars] ")" {VarDecl} Block.
FormPars = Type ident {"," Type ident}.
Type = ident ["[" "]"].
Block = "{" {Statement} "}".
Statement = Designator ("=" Expr | ActPars) ";"
        | "if" "(" Condition ")" Statement
          ["else" Statement]
        | "while" "(" Condition ")" Statement
        | "return" [Expr] ";"
        | "read" "(" Designator ")" ";"
        | "print" "(" Expr ["," number] ")" ";"
        | Block
        | ";".
ActPars = "(" [ Expr {"," Expr} ] ")".
Condition = Expr Relop Expr.
Relop = "==" | "!=" | ">" | ">=" | "<" | "<=".
Expr = ["-"] Term {Addop Term}.
Term = Factor {Mulop Factor}.
Factor = Designator [ActPars]
        | number
        | charConst
        | "new" ident ["[" Expr "]"]
        | "(" Expr ")".
Designator = ident {"." ident | "[" Expr "]"}.
Addop = "+" | "-".
Mulop = "*" | "/" | "%".
```

Lexical structure

Character classes:

```
letter = 'a'..'z' | 'A'..'Z'.
digit = '0'..'9'.
whiteSpace = ' ' | '\t' | '\r' |
'\n'.
```

Terminal classes:

```
ident = letter {letter |
digit}.
number = digit {digit}.
charConst = "'" char "'". //
including '\r', '\t', '\n'
```

Keywords:

```
program class
if else while read print return
void final new
```

Operators:

```
+ - * / %
== != > >= < <=
( ) [ ] { }
= ; , .
```

Comments: `// to the end of line`

```
class Token {
    int kind;       // token code
    int line;       // token line (for error messages)
    int col;        // token column (for error messages)
    int val;        // token value (for number and charCon)
    String str;     // token string
}
```

```
class Struct {
    static final int            // type kinds
        None = 0, Int = 1, Char = 2, Arr = 3, Class = 4;
    int kind;                   // None, Int, Char, Arr, Class
    Struct elemType;            // Arr: element type
    int nFields;                // Class: number of fields
    Obj fields;                 // Class: list of fields
}
```

```
class Obj {
    static final int
        Con = 0, Var = 1, Type = 2, Meth = 3;
    int kind;           // Con, Var, Type, Meth
    String name;
    Struct type;
    Obj next;
    int val;            // Con: value
    int adr;            // Var, Meth: address
    int level;          // Var: 0 = global, 1 = local
    int nPars;          // Meth: number of parameters
    Obj locals;         // Meth: parameters and local objects
}
```

```
class Operand {
    static final int
        Con = 0, Local = 1, Static = 2, Stack = 3,
        Fld = 4, Elem = 5, Meth = 6;
    int kind;           // Con, Local, Static, ...
    Struct type;        // type of the operand
    int val;            // Con: constant value
    int adr;            // Local, Static, Fld, Meth: address
    Obj obj;            // Meth: method object
}
```