Grammar for the Exp programming language

Version 5

program → main

main → (statement)+

statement → st_print | st_attrib | st_if | st_while | st_break | st_continue

st_print → **print (** expression (**,** expression)* **)**

st_attrib → **name =** expression

st_if → **if** comparison **{** (statement)+ **}**

st_while → **while** comparison **{** (statement)+ **}**

st_break → **break**

st_continue → **continue**

comparison → expression (**==** | **!=** | **<** | **<=** | **>** | **>=**) expression

expression → term ((**+** | **-**) term)*

term → factor ((***** | **/** | **%**) factor)*

factor → **number** | **(** expression **)** | **name** | **read_int ( )**

---

tokens: **+ - * / % ( ) = , { } == != < <= > >=**
print number name read_int if while break continue

bytecode instructions:

+1 **ldc** *value*
-1 **iadd**
-1 **isub**
-1 **imul**
-1 **idiv**
-1 **irem**
-1 **istore** *index*

+1 **iload** *index*
+1 **getstatic**
-2 **invokevirtual .../print(I)V**
-1 **invokevirtual .../println()V**
+1 **invokestatic .../readInt()I**
-2 **if_icmp??** *label* **??** → **eq** | **ne** | **lt** | **le** | **gt** | **ge**
0 **goto** *label*