**Homework #06 Written**

Noah Dunn

* + Java
    1. “Zero or one” <identifier>

<zero or one> ::= <identifier>?

* + 1. “Zero or more comma separated <identifier>s

<zero or more commas helper> ::= <zero or more commas>?

<zero or more commas> ::= <identifier> | < zero or more commas> , <identifier>

* + 1. “One or more” space separated <identifier>s

According to the BNF write up, “This BNF definition does not address such pragmatic issues as comment conventions and the use of ‘white space’ to delimit tokens”. As such, I think this to be not possible in this particular BNF’s definition.

* + C#  
    Although I could not locate a rule definition call like the ::=, I will use an equals sign for simplicity.
    1. “Zero or one” <identifier>

zero\_or\_one = [identifier]

* + 1. “Zero or more comma separated <identifier>s

zero\_or\_more = {zero\_or\_more\_helper}

zero\_or\_more\_helper = identifier { “,” identifier}

* + 1. “One or more” space separated <identifier>s

one\_or\_more = identifier {“ ” identifier}

|  |  |  |
| --- | --- | --- |
| **C# Grammar Rule** | **Strings** | **Yes/No** |
| hexadecimal\_integer\_literal | 0x | No |
| 0x0 | Yes |
| 0x34aE | Yes |
| 0x0L | Yes |
| real\_literal | .55 | Yes |
| 55. | No |
| 0.0 | Yes |
| 0.0M | Yes |
| E-2 | No |
| 3E+2 | Yes |

1. **(10/8)** With respect to the Java grammar, answer the following questions:
   * What is the shortest possible <synchronized statement>?

<*synchronized statement*> ::= **synchronized (** <*expression*> **)** <*block*>

synchronized -> required

( -> required

<expression> -> <assignment expression> -> <conditional expression> ->

<conditional or expression> -> <conditional and expression> -> <inclusive or

expression> -> <exclusive or expression> -> <and expression> -> <equality

expression> -> <relational expression> -> <shift expression> -> <additive

expression> -> <multiplicative expression> -> <unary expression> -> <unary

expression not plus minus> -> <postfix expression> -> <primary> -> <primary no

new array> -> <literal> -> <integer literal> -> <decimal integer literal> -> <decimal

numeral> -> 0

) -> required

<block> -> {<block statements>?} -> “{“ and “}” required but not <block statement>

Shortest possible statement:  **synchronized(0){}**

* + Do import statements have to appear at the top of a Java program? Justify your answer using the grammar.  
    <*compilation unit*> ::= <*package declaration*>? <*import declarations*>? <*type declarations*>?

If a package declaration is specified, an import will follow that. If no package declarations are specified , import declarations will be at the top. These will always appear prior to type declaration with no option of imports occuring after type declarations.  
  
**In short, they will always be at the top, right after package declarations.**

1. **(10/8)** Use the virtual machine instructions given in Chapter 3. Write the operational semantic definition of Java’s **do-while**.

do {

LOOP\_BODY

}

while(expr1);

Translates to

loop:

LOOP\_BODY

if expr1 == 0 goto out

goto loop

out:

...

1. **(10/8)** Compute the weakest precondition for:

b = 2b + 1 - 3

2b + 1 - 3 < 0

b < 1

**{b < 1}**

a = 2 \* b + 1;  
b = a - 3;  
**{ b < 0 }**

**(10/8)** Compute the weakest precondition for:

if block

2x + 1 > 3

**{x > y and x > 1}**

y = 2 \* x + 1

{y > 3}

Else block

3x - 1 > 3

**{x <= y and x > 4/3}**

y = 3 \* x - 1

{y > 3}

{x > 1} does not match {x > 4/3}

Strengthen {x > 1} -> {x > 4/3}

**{x > 4/3}**

if (x > y)  
 y = 2 \* x + 1;  
else  
 y = 3 \* x - 1;  
**{ y > 3 }**

**(10/8)** Write the denotational semantics for Mif . You may assume that the definitions used in class are available to you (e.g., Me, Msl, Ma, Mbe, Mdec).

**<ifstmt> → if <be> then <sl.if> else <sl.else>**

**<sl.if> → <sl>**

**<sl.else> → <sl>**

**Mif(if <be> then <sl.if> else <sl.else>) Δ=**

**if Mbe(<be>, s) == undef**

**ERROR**

**else if Mbe(<be>, s) == true**

**Msl(<sl.if>, s)**

**else**

**Msl(<sl.else>, s)**