Drew Pulliam – DTP180003

Assignment 1

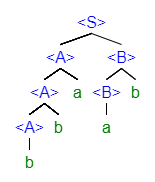
CS 4337.0U2

1. Java is not a simple language, it has multiple ways to do operations and operator overloading is possible, this contributes to worse readability. Java is an orthogonal language, a simple example is you can create and array to hold ints, floats, doubles, etc. Java does have adequate data types which increases readability. Java syntax is decent, it uses if/else keywords and others that help with readability, but open/close brackets “{}” decrease readability as they are less obvious.

Java is an extremely writeable language. It is already fairly readable, which contributes to being writable as well. It also supports abstraction extremely well because it is built on OOP (object oriented programming). For the same reason that Java is not extremely readable it becomes very writeable because there are more than one way to do operations. (x++ is easier to write than x=x+1)

Because Java is very writable that contributes to being more reliable as there is less awkward programming that may react unexpectedly. Java also supports type checking and exception handling very well and has a robust access specification system for variables. This allows the programmer to specify what parts of the program have access to any specific variable.

1. The grammar is not ambiguous because there will always be only one unique parse tree for any combination. It will always repeat the same pattern as this tree (it can only be longer or shorter, never different).



1. Java Do-While loop denotational semantics (do expression L, then check if B is true and keep doing L while B is true)

Do L

Ml (while B do L, s) Δ = if Mb (B, s) == undef

Then error

Else if Mb­ (B,s) == false

Then s

Else if Msl (L, s) == error

Then error

Else Ml (while B do L, Msl (L, s) )

1. A. {x < 1/3 – y/2} x = 3 \* (2 \* x + y) {x < 2}

{x < 2} Y = 4 \* x – 1 {y < 7}

**{x < 1/3 - y/2}**

B. if (x == y) y = 3 \* x + 1

Else y = 3 \* x {y > 6}

{x > 2} for the else block

{x > 5/3} for the if block

**{x > 2} overall**

1. <doBlock> -> do ‘{‘ <assign> ‘}’

<assign> -> id = <expr>

<expr> -> <term> { (+ | -) <term> }

<term> -> id | int\_constant

<whileLoop> -> while ‘(‘ <cond> ’)’

<cond> -> <expr> (> | < | >= | <= | ==) <expr>

1. S => XyZ => Xyz => Yyyz => Xzyyz => xYYzyyz => xYzzyyz => xXzzzyyz