Ngoc Pham – Hoat Vu – Sangwook Park
CSC 135
Assignment #1

Proof

	First	Follow
javaclass	{C, D}	{$}
classname	{C, D}	{X} U {B} U FIRST(varref) = {X, B, J, K, (}
varlist	{I, S, C, D}	{;} U {) } = { ; ,) }
vardef	{I, S, C, D}	{ , } U FOLLOW(varlist} = {',' , ; ,) }
type	{I, S}	FIRST(varname) U FIRST(methodname) = {Y, Z, M, N}
varname	{Y, Z}	FOLLOW(vardef) U {=} U FOLLOW(oprnd) = { ',' , ; ,) , = , *, + , <, >, ! }
letter	{Y, Z}	FIRST(char) U FOLLOW(char) U FOLLOW(varname) = {Y, Z, 0, 1, 2, 3, ',' , ; ,) , = , *, +, <, >, !}
char	{Y, Z, 0, 1, 2, 3}	FOLLOW(varname) U FIRST(char) = {Y, Z, 0, 1, 2, 3, ',' , ; ,) , = , *, +, <, >, !}
digit	{0, 1, 2, 3}	FOLLOW(char) U FIRST(digit) U FOLLOW(integer) = {Y, Z, 0, 1, 2, 3, ',' , ; ,) , = , *, +, <, >, !}
integer	{0, 1, 2, 3}	FOLLOW(oprnd) = { *, +, ; ,), <, =, >, ! }
varref	{J, K}	FOLLOW(vardef) U {=} U {.} = { = , . , ',' , ; ,) }
method	{P, V}	{E} U FIRST(method) = {E, P, V}
accessor	{P, V}	FIRST(type) = {I, S}
methodname	{M, N}	{ (}
statemt	{F, Y, Z, J, K, W}	FIRST(returnstatemt) U FIRST(statemt) U FIRST(ifstatemt) U FIRST(assignstatemt) U FIRST(whilestatemt) U FIRST(methodcall) U {E} = {F, Y, Z, J, K, W, R, E}
ifstatemt	{F}	FOLLOW(statemt) = {F, Y, Z, J, K, W, R, E}
assignstatemt	{Y, Z, J, K}	{ ; }
mathexpr	{0,1, 2, 3, Y, Z, (, J, K}	FOLLOW(assignstatemt) U {)} = { ; ,) }
factor	{0,1, 2, 3, Y, Z, (, J,	{+} U FOLLOW(mathexpr) = {+, ; ,)}

	K}	
oprnd	{0,1, 2, 3, Y, Z, (, J, K}	{*} U FOLLOW(factor) U FIRST(operator) U {) } = { *, +, ; ,), <, =, >, ! }
getvarref	{O, J, K}	FOLLOW(assignstatemt) = { ; }
whilestatemt	{W}	FOLLOW(statemt) = {F, Y, Z, J, K, W, R, E}
cond	{ (}	{ T }
operator	{<, =, >, !}	FIRST(oprnd) = {0,1, 2, 3, Y, Z, (, J, K}
returnstatemt	{R}	{E}
methodcall	{J, K}	FOLLOW(statemt) U FOLLOW(oprnd) U FOLLOW(getvarref) = {F, Y, Z, J, K, W, R, E, *, +, ; ,), <, =, >, !}

Proof:<statemt> ::= <ifstatemt> | <assignstatemt>;|<whilestatemt>|<methodcall>

FIRST(ifstatemt), FIRST(assignstatemt), FIRST(whilestatemt), FIRST(methodcall)

- ⇨ We have FIRST(assignstatemt) ∩ FIRST(methodcall) = {Y, Z, J, K} ∩ {J, K} are not pairwise disjoint. Therefore, we cannot use a Recursive Descent Parser with this grammar