

**Assignment 1**  
**First & Follow; Recursive Descent Parser Proof**

**First & Follow**

Production Rule	Non-Terminal	FIRST	FOLLOW
<u>Rule1</u> $\langle \text{jClass} \rangle ::= \langle \text{className} \rangle B \langle \text{varlist} \rangle \{ \langle \text{method} \rangle \} E$	$\langle \text{JClass} \rangle$	$\text{FIRST}(\langle \text{className} \rangle) = \{C, D\}$	Start Symbol Rule ends with implicit '\$' = {'\$'}
<u>Rule2</u> $\langle \text{className} \rangle ::= C   D$	$\langle \text{className} \rangle$	$\{C, D\}$	(from Rule1) {B}
<u>Rule3</u> $\langle \text{varlist} \rangle ::= \langle \text{vardef} \rangle \{ , \langle \text{vardef} \rangle \};$	$\langle \text{varlist} \rangle$	$\text{FIRST}(\langle \text{vardef} \rangle) = \text{FIRST}(\langle \text{type} \rangle) = \{I, S\}$	(from Rule1) $\text{FIRST}(\langle \text{method} \rangle) U$ (from Rule1 if no method) $\{E\} = \{P, E\}$
<u>Rule4</u> $\langle \text{vardef} \rangle ::= \langle \text{type} \rangle \langle \text{var} \rangle$	$\langle \text{vardef} \rangle$	$\text{FIRST}(\langle \text{type} \rangle) = \{I, S\}$	(from Rule3 first occurrence of vardef) $\text{FIRST}(\{ , \langle \text{vardef} \rangle \}) U$ (from Rule3 second occurrence of vardef) $\{ ; \}$ $U$ (from Rule7 second occurrence of vardef $\{ \}$ ). Remember first occurrence is similar to first occurrence in Rule3 $= \{ ' , ' , ' ; ' \}$
<u>Rule5</u> $\langle \text{type} \rangle ::= I   S$	$\langle \text{type} \rangle$	$\{I, S\}$	(from Rule4) $\text{FIRST}(\langle \text{var} \rangle) U$ (from Rule7) $\text{FIRST}(\langle \text{mname} \rangle) = \{V, Z, M, N\}$
<u>Rule6</u> $\langle \text{var} \rangle ::= V   Z$	$\langle \text{var} \rangle$	$\{V, Z\}$	(from Rule4 follow(vardef) ) (from Rule11 and Rule13) $\{ ' = ' , ' == ' , ' , ' , ' ; ' \}$
<u>Rule7</u> $\langle \text{method} \rangle ::= P \langle \text{type} \rangle \langle \text{mname} \rangle ( \langle \text{vardef} \rangle \{ , \langle \text{vardef} \rangle \} ) B \langle \text{stmtnt} \rangle \langle \text{returnstmtnt} \rangle E$	$\langle \text{method} \rangle$	$\{P\}$	(Rule1 method is followed by method in {method} AND IF NO METHOD IS followed by E) $\text{FIRST}(\langle \text{method} \rangle) U \{E\} = \{P, E\}$
<u>Rule8</u> $\langle \text{mname} \rangle ::= M   N$	$\langle \text{mname} \rangle$	$\{M, N\}$	(Rule7) $\{ ' ( ' \}$
<u>Rule9</u> $\langle \text{stmtnt} \rangle ::= \langle \text{ifstmtnt} \rangle   \langle \text{assignstmtnt} \rangle  $	$\langle \text{stmtnt} \rangle$	$\text{FIRST}(\langle \text{ifstmtnt} \rangle) U \text{FIRST}(\langle \text{assignstmtnt} \rangle)$	(Rule7) $\text{FIRST}(\langle \text{returnstmtnt} \rangle) U$

<u>&lt;whilestmt&gt;</u>		U FIRST(<whilestmt>) = {F,V,Z,W}	(Rule10 and 12: stmt followed by stmt in {stmt}) {FIRST(<stmt>)} U {E} = {R,F,V,Z,W,E}
<u>Rule10</u> <u>&lt;ifstmt&gt; ::= F &lt;cond&gt; T</u> <u>B {&lt;stmt&gt;} E</u> <u>[L B { &lt;stmt&gt; } E]</u>	<ifstmt>	{F}	Same as Follow(stmt) since ifstmt has empty follow in rule9. FOLLOW(<stmt>) = {R,F,V,Z,W,E}
<u>Rule11</u> <u>&lt;assignstmt&gt; ::=</u> <u>&lt;var&gt; = &lt;digit&gt;;</u>	<assignstmt>	{V,Z}	Same as Follow(stmt) since assignstmt has empty follow in rule9.  FOLLOW(<stmt>) = {R,F,V,Z,W,E}
<u>Rule12</u> <u>&lt;whilestmt&gt; ::= W</u> <u>&lt;cond&gt; T B &lt;stmt&gt;</u> <u>{&lt;stmt&gt;} E</u>	<whilestmt>	{W}	Same as Follow(stmt) since whilestmt has empty follow in rule9.  FOLLOW(<stmt>) = {R,F,V,Z,W,E}
<u>Rule13</u> <u>&lt;cond&gt; ::=</u> <u>( &lt;var&gt; == &lt;digit&gt; )</u>	<cond>	{'('}	(Rule12) {T}
<u>Rule14</u> <u>&lt;returnstmt&gt; ::=</u> <u>R &lt;var&gt;;</u>	<digit>	{0,1,2,3,4,5,6,7,8,9}	(Rule7) {E}
<u>Rule15</u> <u>&lt;digit&gt; ::=</u> <u>0 1 2 3 4 5 6 7 8 9</u>	<returnstmt>	{R}	(Rule11 and 13) {';','}'}

See Proof for usability of Recursive Descent Parser for this Grammar in next page.

**Proof to verify if the Recursive Descent Parsing technique can be used for above grammar.**

**Rule 1: Consider the Production Rules 5,6,8,9,16 which have the right hand side using '|' to separate alternate rules.**

**Applying the constraint rule on Page 7 of notes (05.Syntactic analysis and Predictive Parsing):**

For  $\langle \text{stmtnt} \rangle ::= \langle \text{ifstmtnt} \rangle | \langle \text{assignstmtnt} \rangle | \langle \text{whilestmtnt} \rangle$ :  $\text{FIRST}(\langle \text{ifstmtnt} \rangle) \cap \text{FIRST}(\langle \text{assignstmtnt} \rangle) \cap \text{FIRST}(\langle \text{whilestmtnt} \rangle) = \{F\} \cap \{V, Z\} \cap \{W\} = \{\emptyset\}$

For  $\langle \text{digit} \rangle ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$ , Clearly  $\cap$  of separate parts =  $\{\emptyset\}$

Same with Production Rules 5,6,8 which is obvious.

**Rule 2:**

No such rules apply here

**Optional Part of an RHS (constraint rules on Page 8 of notes):**

$\langle \text{ifstmtnt} \rangle ::= F \langle \text{cond} \rangle T B \{ \langle \text{stmtnt} \rangle \} E [L B \{ \langle \text{stmtnt} \rangle \} E]$

Because  $[L B \{ \langle \text{stmtnt} \rangle \} E]$  ends rule, it implies that it may end as  $\lambda$ , therefore:  
 $\text{FIRST}([L B \{ \langle \text{stmtnt} \rangle \} E]) \cap \text{FOLLOW}(\langle \text{ifstmtnt} \rangle) = \{\lambda\} \cap \{R, F, V, Z, W, E\} = \{\emptyset\}$

**Indefinite Repeats**

**For all cases object following all repeats do not generate  $\lambda$ , therefore (Constraint rule on Page 8)**

$\langle \text{jClass} \rangle ::= \langle \text{className} \rangle B \langle \text{varlist} \rangle \{ \langle \text{method} \rangle \} E$   
 $\text{FIRST}(\langle \text{method} \rangle) \cap \text{FIRST}(E) = \{P\} \cap \{E\} = \{\emptyset\}$

$\langle \text{varlist} \rangle ::= \langle \text{vardef} \rangle \{, \langle \text{vardef} \rangle\}$   
 $\text{FIRST}\{, \langle \text{vardef} \rangle\} \cap \text{FIRST}(',') = \{', '\} \cap \{', '\} = \{\emptyset\}$

$\langle \text{method} \rangle ::= P \langle \text{type} \rangle \langle \text{mname} \rangle (\langle \text{vardef} \rangle \{, \langle \text{vardef} \rangle\}) B \langle \text{stmtnt} \rangle \langle \text{returnstmtnt} \rangle E$   
 $\text{FIRST}\{, \langle \text{vardef} \rangle\} \cap \text{FIRST}(')') = \{', '\} \cap \{', '\} = \{\emptyset\}$

$\langle \text{ifstmtnt} \rangle ::= F \langle \text{cond} \rangle T B \{ \langle \text{stmtnt} \rangle \} E [L B \{ \langle \text{stmtnt} \rangle \} E]$

$\text{FIRST}(\langle \text{stmtnt} \rangle) \cap \text{FIRST}(E) = \{F, V, Z, W\} \cap \{E\} = \{\emptyset\}$

This applies for both instances of  $\{ \langle \text{stmtnt} \rangle \}$ .

$\langle \text{whilestmtnt} \rangle ::= W \langle \text{cond} \rangle T B \langle \text{stmtnt} \rangle \{ \langle \text{stmtnt} \rangle \} E$

$\text{FIRST}(\langle \text{stmtnt} \rangle) \cap \text{FIRST}(E) = \{F, V, Z, W\} \cap \{E\} = \{\emptyset\}$

**Because Rules 1, 2, optional part of RHS and indefinite repeats were followed we can use predictive parsing.**