Fehlerbehandlung

- Panic Mode
  - Abbruch beim ersten Fehler
  - **Übung 3**
- Allgemeine Fangsymbole
  - Synchronisation der restlichen Eingabe mit der Grammatik
  - Parser kennt an jeder Stelle alle gültigen Nachfolge-Symbole
  - Aufwendig
- Spezielle Fangsymbole
  - Synchronisation nur an besonders "sicheren" Stellen.
  - Beispiele: Schlüsselwörter, Strichpunkte, …
  - **Übung 4**

Beispiel: Deklarationen


```
DeclPart    = { ForwardDecl } "{" Body "}" .
ForwardDecl = "void" ident "(" ")" ";" .
Body        = ... .
```

Welche Deklarationen kann man damit erzeugen?

```
    void p1();
    void p2();
    void p3();
    ...
    {
        ...
    }
```

Beispiel: Deklarationen


```
DeclPart    = { ForwardDecl } "{" Body "}" .
ForwardDecl = "void" ident "(" ")" ";" .
Body        = ... .
```

```
void DeclPart () {
  while (sym == void_) {
    ForwardDecl();
  }
  check(lbrace); Body(); check(rbrace);
}
```

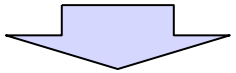
Bsp: Fehler in ForwardDecl


```
void p [];
{ … }
```

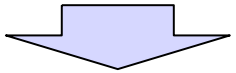
```
                    Erkenne DeclPart
next()  → void_         Erkenne ForwardDecl
                          void_ erkannt
next()  →  ident          ident erkannt
next()  →  lbrack         ERROR: "( expected"
                                 ERROR: ") expected"
                                 ERROR: "; expected"
                          ERROR: "{ expected"
                          ...
                          ERROR: "} expected"
```

Bsp: First/Follow-Sets


```
DeclPart    = { ForwardDecl } "{" Body "}" .
ForwardDecl = "void" ident "(" ")" ";" .
Body        = ... .
```



```
First(ForwardDecl)  = void_
Follow(ForwardDecl) = First(ForwardDecl) + lbrace
                    = void_, lbrace
```



```
private EnumSet<Token.Kind> followFwdDecl =
    EnumSet.of(void_, lbrace, eof);
```

Beispiel: Deklarationen


```
DeclPart    = { ForwardDecl } "{" Body "}" .
ForwardDecl = "void" ident "(" ")" ";" .
Body        = ... .
```

```
void DeclPart () {
  for (;;) {
    if (sym == void_) { ForwardDecl(); }
    else if (sym == lbrace) { break; }
    else { recoverFwdDecl(); }
  }
  check(lbrace); Body(); check(rbrace);
}


void recoverFwdDecl() {
  error("invalid forward declaration");
  do {
    scan();
  } while (!followFwdDecl.contains(sym));
}
```

Bsp: Fehler in ForwardDecl (2)


```
void p [];
{ … }
```

```
                        Erkenne DeclPart
next()   →   void_          Erkenne ForwardDecl
                              void_ erkannt
next()   →   ident             ident erkannt
next()   →   lbrack            ERROR: "( expected"
                                     ERROR: ") expected"
                                     ERROR: "; expected"
                               ERROR: "invalid forward decl."
next()   →   rpar
next()   →   semicolon
next()   →   lbrace        lbrace erkannt
next()   →   ...                    Erkenne Body
...                                    ...
next()   →   rbrace        rbrace erkannt
```

LL(1)-Bedingung

- Alternativen haben verschiedene terminale Anfänge
- Linksrekursionen verboten!

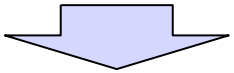
⇨ Bei Top-Down-Analyse
mit einem Vorgriffssymbol entscheiden,
welche Alternative ausgewählt werden muss.

- Abhilfen:
  - gleiche Anfänge ⇨ Faktorisieren
  - Linksrekursionen ⇨ Umwandlung in Iteration

Regel Statement


```
Statement
= Assignment
| ProcedureCall
| Increment | Decrement
| … .
```

gut lesbar, aber nicht LL(1), alle Alternativen beginnen mit **ident**



Abhilfe: Faktorisieren

```
Statement
= Designator
  ( AssignOp Expr          // Assignment
  | ActPars                // ProcedureCall
  | "++" | "--"            // Increment | Decrement
  ) ";"
| … .
```

Beispiel: LL(1)


```
S = x B B B | y C.
B = y B | x C.
C = S S | z.
```

$(S = S_1 \mid S_2. , S_1 = xBBB. , S_2 = yC.)$
$(B = B_1 \mid B_2. , B_1 = yB. , B_2 = xC.)$
$(C = C_1 \mid C_2. , C_1 = SS. , C_2 = z.)$

$$first(S1) \cap first(S2) = \{x\} \cap \{y\} = \{\}$$
$$first(B1) \cap first(B2) = \{y\} \cap \{x\} = \{\}$$
$$first(C1) \cap first(C2) = first(S) \cap \{z\} = \{x, y\} \cap \{z\} = \{\}$$

Beispiel: LL(1)


```
S = x B B B | y C.
B = y B | x C u.
C = [ S S | z ].
```

$(S = S_1 \mid S_2.\ S_1 = xBBB.\ S_2 = yC.)$
$(B = B_1 \mid B_2.\ B_1 = yB.\ B_2 = xCu.)$
$(C = C_1 \mid C_2 \mid C_3.\ C_1 = SS.\ C_2 = z.\ C_3 = \varepsilon.)$

$FC1 = first(C_1) = first(S) = \{x, y\}$
$FC2 = first(C_2) = \{z\}$
$FC3 = first(C_3) = follow(C) =$
$= \{u\} \cup follow(S) =$
$= \{u\} \cup first(S) \cup follow(C) =$
$= \{u\} \cup \{x, y\} =$
$= \{x, y, u\}$

$FC1 \cap FC2 = \{\}$
$FC2 \cap FC3 = \{\}$
$FC1 \cap FC3 = \{x, y\}$

Beispiel: LL(1)-Konflikt


```
S = x B B B | y C.
B = y B | x C u.
C = [ S S | z ].
```

Beispielsatz: x x y y u x u x u

```
S = x B             B B
B =   x C         u
C =     S       S
S =     y C
C =       S …
```