

# 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        = ... .
```

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

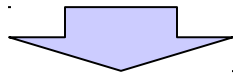
# Bsp: Fehler in *ForwardDecl*

```
void p [ );
{ ... }
```

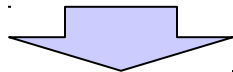
	Erkenne DeclPart
next() → void_	Erkenne <b>ForwardDecl</b>
	void_ erkannt
next() → ident	ident erkannt
next() → lbrack	ERROR: "( expected"
	ERROR: ") expected"
	ERROR: "; expected"
	ERROR: "{ expected"
	...
	ERROR: "} expected"

# 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        = ... .
```

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

private 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

# Beispiel: Expression

```

E = T { "+" T } .
T = F { "*" F } .
F = id | num | "(" E ")" .

```

```

private void parse() {
    scan();
    E();
    check(eof);
}

private void F() {
    if (sym == id) scan();
    else if (sym == num) scan();
    else if (sym == lpar) {
        scan();
        E();
        check(rpar);
    } else error("...");
}

```

```

private void E() {
    for (;;) {
        T();
        if (sym == plus) scan();
        else break;
    }
}

private void T() {
    for (;;) {
        F();
        if (sym == times) scan();
        else break;
    }
}

```



# Beispiel: Allgemeine Fangsymbole



```
E = T { "+" T } .  
T = F { "*" F } .  
F = id | num | "(" E ")" .
```

```
private void check(int expected, BitSet sux) {  
    if (sym == expected) scan();  
    else error(name[expected] + " expected", sux);  
}
```

```
private void error (String msg, BitSet sux) {  
    System.out.println("...");  
    while (!sux.get(sym)) scan();  
}
```

# Beispiel: Allgemeine Fangsymbole



```
E = T { "+" T } .  
T = F { "*" F } .  
F = id | num | "(" E ")" .
```

```
private void parse() {  
    scan();  
    E();  
    check(eof);  
}  
private void T(BitSet sux) {  
    for (;;) {  
        F({times} U sux);  
        if (sym == times) scan();  
        else if (sym ∈ sux) break;  
        else {  
            error("...", sux);  
            break;  
        }  
    }  
} } }
```

```
private void E(BitSet sux) {  
    for (;;) {  
        T({plus} U sux);  
        if (sym == plus) scan();  
        else if (sym ∈ sux) break;  
        else {  
            error("...", sux);  
            break;  
        }  
    }  
}
```

# Beispiel: Allgemeine Fangsymbole

```
E = T { "+" T } .
T = F { "*" F } .
F = id | num | "(" E ")" .
```

```
First(F) = { id, num, "(" }
```

```
private void F(BitSet sux) {
    if (sym ∉ First(F))
        error("...", First(F) ∪ sux);
    if (sym == id) scan();
    else if (sym == num) scan();
    else if (sym == lpar) {
        scan();
        E({rpar} ∪ sux);
        check(rpar, sux);
    } // no error case
}
```

# Beispiel: Allgemeine Fangsymbole



```
E = T { "+" T } .  
T = F { "*" F } .  
F = id | num | "(" E ")" .
```

```
private void parse() {  
    scan();  
    E({eof});  
    check(eof);  
}
```

```
private void T(BitSet sux) {  
    for (;;) {  
        F({times} ∪ sux);  
        if (sym == times) scan();  
        else if (sym ∈ sux) break;  
        else {  
            error("...", sux);  
            break;  
        }  
    }  
}
```

```
private void E(BitSet sux) {  
    for (;;) {  
        T({plus} ∪ sux);  
        if (sym == plus) scan();  
        else if (sym ∈ sux) break;  
        else {  
            error("...", sux);  
            break;  
        }  
    }  
}
```

```
private void F(BitSet sux) {  
    if (sym ∉ First(F))  
        error("...", First(F) ∪ sux);  
    if (sym == id) scan();  
    else if (sym == num) scan();  
    else if (sym == lpar) {  
        scan();  
        E({rpar} ∪ sux);  
        check(rpar, sux);  
    } // no error case  
}
```

# Beispiel: Spezielle Fangsymbole



```
E = T { "+" T } .  
T = F { "*" F } .  
F = id | num | "(" E ")" .
```

Wiederaufsatz nach E.

```
private void parse() {  
    scan();  
    E();  
    check(eof);  
}
```

```
private void F() {  
    if (sym == id) scan();  
    else if (sym == num) scan();  
    else if (sym == lpar) {  
        scan();  
        E();  
        check(rpar);  
    } else error("...");  
}
```

```
private void E() {  
    for (;;) {  
        T();  
        if (sym == plus) scan();  
        else break;  
    }  
}
```

```
private void T() {  
    for (;;) {  
        F();  
        if (sym == times) scan();  
        else break;  
    }  
}
```

# Beispiel: Spezielle Fangsymbole



```
E = T { "+" T } .  
T = F { "*" F } .  
F = id | num | "(" E ")" .
```

Wiederaufsatz nach E.

```
Follow(E) = { eof, ")" }
```

```
private void E() {  
    for (;;) {  
        T();  
        if (sym == plus) scan();  
        else if (sym ∈ Follow(E))  
            break;  
        else {  
            recoverE();  
            break;  
        }  
    }  
}
```

```
private void recoverE() {  
    error("...");  
    do {  
        scan();  
    } while (sym ∉ Follow(E));  
}
```

# Beispiel: Spezielle Fangsymbole



```
E = T { "+" T } .  
T = F { "*" F } .  
F = id | num | "(" E ")" .
```

```
private void E() {  
    for (;;) {  
        T();  
        if (sym == plus) scan();  
        else if (sym ∈ Follow(E))  
            break;  
        else {  
            recoverE();  
            break;  
        }  
    }  
}
```

```
private void recoverE() {  
    error("...");  
    do {  
        scan();  
    } while (sym ∉ Follow(E));  
}
```

```
private void parse() {  
    scan();  
    E();  
    check(eof);  
}
```

```
private void T() {  
    for (;;) {  
        F();  
        if (sym == times) scan();  
        else break;  
    }  
}
```

```
private void F() {  
    if (sym == id) scan();  
    else if (sym == num) scan();  
    else if (sym == lpar) {  
        scan();  
        E();  
        check(rpar);  
    } else error("...");  
}
```