

## Lexikalische Fehler ("Zeichen/ Wortwahl")

```
Verwendung von Zeichen, die nicht zum Alphabet gehören
```

error: illegal character: '\u00a7' - int x, y, §;

error: illegal character: '#' - int # = rand.nextInt(90);

Verletzung von Namenskonventionen

error: <identifier> expected - int x, y, 2z; Zahlbereichsüberschreitungen bei Konstanten

error: integer number too large: 111111111111111111 - final int r =

### 

Rechtschreibfehler wie

error: cannot find symbol - a[1] = new Integer(numberField2.GETText());

error: cannot find symbol - integer n = r.length-1;

werden erst bei der syntaktischen Analyse erkannt (bei der lexikalischen werden sie zwar nicht als Schlüsselwörter ausgemacht, aber nicht als fehlerhaft eingestuft!)

## Syntaktische Fehler ("Grammatik")

```
Fehler in der Zeichen-/ Anweisungsfolge
```

```
error: ']' expected - public static int[] MinSort (int[ r) {
```

error: '(' expected - while i < n { (runde Klammern fehlen)

error: not a statement – timer1.stop; (Methodenklammern fehlen)

error: illegal start of expression - j+;

error: ';' expected - int x, y, z(Semikolon am Zeilenende fehlt)

error: not a statement - j == i+1;

error: cannot find symbol – numberField2.set(""+a); (setText!)

error: cannot find symbol – timer1start(); (Punkt fehlt)

## Semantische Fehler ("Datentyp")

Variable nicht/falsch deklariert

error: incompatible types: int cannot be converted to boolean - **boolean i = 0**;

error: cannot find symbol - min = i;

error: cannot find symbol - **M**in = j; (falsch geschrieben)

## inkompatible Typen

error: incompatible types: String cannot be converted to int - z = "dreißig";

error: incompatible types: int cannot be converted to boolean - if (x = 160) {

error: no suitable method found for setText(int) - numberField3.setText(a);

### keine/falsche Parameterübergabe

error: method MinSort in class MinSort cannot be applied to given types; - MinSort();

error: no suitable method found for setLocation(int,int,int) – button1.setLocation(x,y,z);

# Compiler-Phasen s:≈a\*2+p\*2

## **Analyse-Phase (Front-End)**

Quelltext

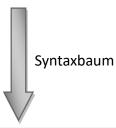


# Lexikalische Analyse (Scanner)



Tokenfolge, 1. Symboltabelle

## Syntaktische Analyse (Parser)



**Semantische Analyse** ("Typechecker")



verfeinerte Tokenfolge und Symboltabelle



Index	Tokenwert Tokentyp		
1	:=	zuweisung	
2	*	punktoperator	
3	+	strichoperator	
4	S	bezeichner	
5	a	bezeichner	
6	2	zahl	

- zerlegt den Quellcode in eine Folge sprachlicher Einheiten – Wörter werden in Kategorien gepackt (Token), z.B. Schlüsselwort, Bezeichner, Zahl, ... und in einer entsprechenden Symboltabelle gespeichert (vgl. oben)
- entfernt überflüssige Zeichen (Leer-, Tabulatorzeichen, Kommentare, ...)
- prüft, ob die Tokenfolge aus der der Sprache zugrunde liegenden Grammatik ableitbar ist
- gibt Position und Art gefundener Fehler an
- führt im Wesentlichen Typanalysen durch
- außerdem Fragen wie
  - o verwendete Variable deklariert?
  - Variable initialisiert vor Aufruf?
  - richtige Anzahl von Parametern beim Prozeduraufruf?

## Synthese-Phase (Back-End)

Optimierung

Umwandlung in Assemblercode

Maschinencode



Index	Tokenwert	Tokentyp	Adresse
1	:=	zuweisung	
2	*	punktoperator	
3	+	strichoperator	
4	S	variablenname	xxx
5	а	variablenname	xxx
6	2	zahl	
7	b	konstantenname	xxx