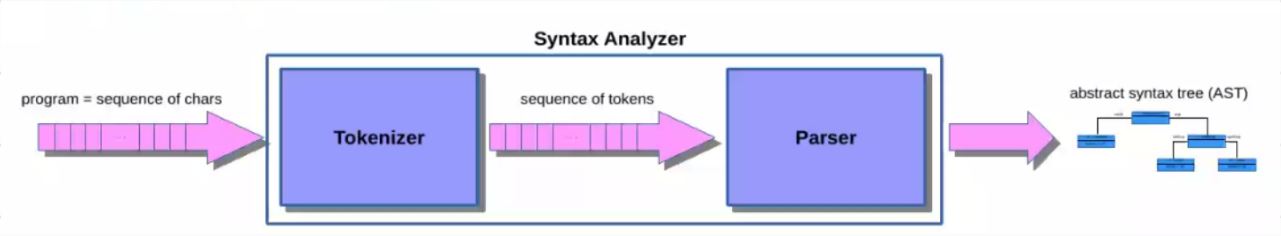
Parser / 08-04

E' l'analizzatore del programma in input che riconosce una sequenza di token. In caso di successo costruisce un albero di sintassi (AST)



Example 1 with C/Java/C++/C# syntax

Input string: `"x2 042=;"`

Recognized tokens:

- `IDENTIFIER` with syntactic data: the name `"x2"`
- `INT_NUMBER` with semantic data: the value thirty-four
- `ASSIGN_OP` with no further data
- `STATEMENT_TERMINATOR` with no further data

failure: sequence not recognized

Il parser può essere generato automaticamente da una grammatica (e tokenizer)

Approcci, basati su come il parser si sviluppa:

→ top-down: più semplice, si parte a partire dalla radice → utilizza ricorsione

→ bottom-up: si parte a partire dalle foglie

I parser più semplici si focalizzano su un token successivo per volta (detto one (look-ahead token)

Nota: un parser con numeri fissi di look-ahead non può riconoscere e scegliere una produzione rispetto a un'altra in base ai contesti se ad esempio ho un'addizione alla fine di una serie di moltiplicazioni (che hanno priorità): non lo sa a priori

Si estende dunque la notazione BNF in EBNF

es. Exp ::= Mul ('+' Mul)*

↓ zero o più ripetizioni

Così facendo ho solo una produzione

Exp ::= Mul (Mul '+' Mul → Exp ::= Mul ('+' Mul)*

Nota: * e '*' sono diversi e ( ) formano la precedenza
→ * EBNF operator (≠ '(' e ')')
→ '*' simbolo terminale

Enum

I token possono essere facilmente rappresentati mediante costanti delle enum.

```
public enum Season {
    Winter, Spring, Summer, Fall
}
```

NON si possono costruire più oggetti di quei quattro
(non mi è permesso fare "new ...")

accedo con Season.Winter
(o in uno switch (s) case Winter, abbreviato).
↓ season

Di base gli enum in Java sono public static final

E' permesso fare == (confronti) con enum, è sicuro perché non posso avere reference (oggetti) diversi per una stessa costante.

Non posso estendere le enum, ma possono implementare interfacce

Ogni tipo T di enum estende ed eredita da Enum<T>

enum Season eredita da Enum<Season>

metodi predefiniti:

- ordinal() se itero, restituisce la posizione della enum corrente
- values() (di classe) restituisce un array con le costanti di enum in ordine per come è definito
- name() restituisce il nome relativo al valore della costante (0 -> Winter)
- valueOf (di classe) restituisce il valore relativo al nome della costante (Winter -> 0)