



Building Java Programs

Chapter 11
Sets and Maps

reading: 11.2 - 11.3



Road Map - Quarter

CS Concepts

- Client/Implementer
- Efficiency
- Recursion
- Regular Expressions
- Grammars
- Sorting
- Backtracking
- Hashing
- Huffman Compression

Data Structures

- Lists
- Stacks
- Queues
- Sets
- Maps
- Priority Queues

Java Language

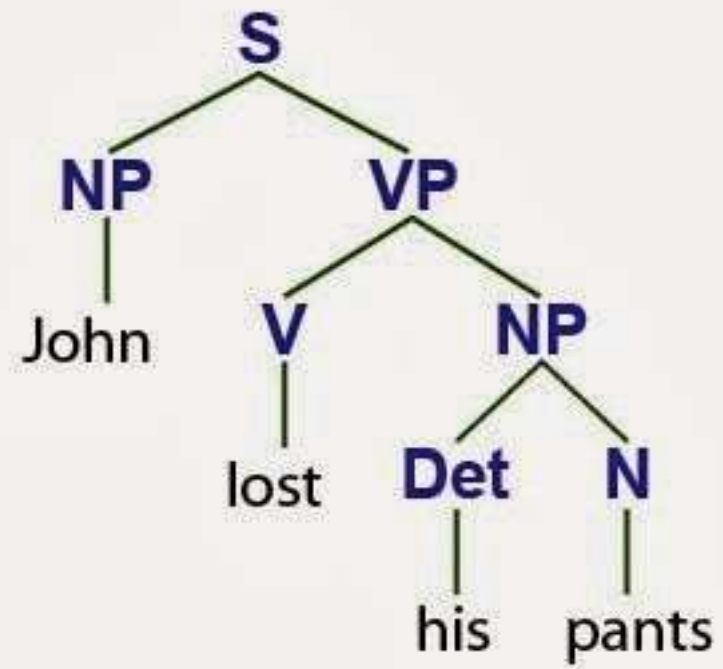
- Exceptions
- Interfaces
- References
- Comparable
- Generics
- Inheritance/Polymorphism
- Abstract Classes

Java Collections

- Arrays
- ArrayList ✖
- LinkedList ✖
- Stack
- TreeSet / TreeMap
- HashSet / HashMap
- PriorityQueue

Note Card Drawing(s)





Languages and grammars

- (formal) **language**: A set of words or symbols.
- **grammar**: A description of a language that describes which sequences of symbols are allowed in that language.
 - describes language *syntax* (rules) but not *semantics* (meaning)
 - can be used to generate strings from a language, or to determine whether a given string belongs to a given language

Backus-Naur (BNF)

- **Backus-Naur Form (BNF)**: A syntax for describing language grammars in terms of transformation *rules*, of the form:

<symbol> ::= <expression> | <expression> ... | <expression>

- **terminal**: A fundamental symbol of the language.
- **non-terminal**: A high-level symbol describing language syntax, which can be transformed into other non-terminal or terminal symbol(s) based on the rules of the grammar.
- developed by two Turing-award-winning computer scientists in 1960 to describe their new ALGOL programming language

An example BNF grammar

`<s> ::= <n> <v>`

`<n> ::= Marty | Victoria | Stuart | Jessica`

`<v> ::= cried | slept | belched`

- Some sentences that could be generated from this grammar:

Marty slept

Jessica belched

Stuart cried

BNF grammar version 2

```
<s> ::= <np> <v>  
<np> ::= <pn> | <dp> <n>  
<pn> ::= Marty | Victoria | Stuart | Jessica  
<dp> ::= a | the  
<n> ::= ball | hamster | carrot | computer  
<v> ::= cried | slept | belched
```

- Some sentences that could be generated from this grammar:

```
the carrot cried  
Jessica belched  
a computer slept
```

BNF grammar version 3

```
<s>::=<np> <v>
<np>::=<pn> | <dp> <adj> <n>
<pn>::=Marty | Victoria | Stuart | Jessica
<dp>::=a | the
<adj>::=silly | invisible | loud | romantic
<n>::=ball | hamster | carrot | computer
<v>::=cried | slept | belched
```

- Some sentences that could be generated from this grammar:

```
the invisible carrot cried
Jessica belched
a computer slept
a romantic ball belched
```


Grammars and recursion

```
<s> ::= <np> <v>
<np> ::= <pn> | <dp> <adjp> <n>
<pn> ::= Marty | Victoria | Stuart | Jessica
<dp> ::= a | the
<adjp> ::= <adj> <adjp> | <adj>
<adj> ::= silly | invisible | loud | romantic
<n> ::= ball | hamster | carrot | computer
<v> ::= cried | slept | belched
```

- Grammar rules can be defined *recursively*, so that the expansion of a symbol can contain that same symbol.
 - There must also be expressions that expand the symbol into something non-recursive, so that the recursion eventually ends.

Grammar, final version

```
<s> ::= <np> <vp>
<np> ::= <dp> <adjp> <n> | <pn>
<dp> ::= the | a
<adjp> ::= <adj> | <adj> <adjp>
<adj> ::= big | fat | green | wonderful | faulty | subliminal
<n> ::= dog | cat | man | university | father | mother | child
<pn> ::= John | Jane | Sally | Spot | Fred | Elmo
<vp> ::= <tv> <np> | <iv>
<tv> ::= hit | honored | kissed | helped
<iv> ::= died | collapsed | laughed | wept
```

- Could this grammar generate the following sentences?

Fred honored the green wonderful child

big Jane wept the fat man fat

- Generate a random sentence using this grammar.

Sentence generation

