Grammar defines a set of sentences
Sentence is sequence of symbols or events

Defined by “productions”
Each production maps non-terminal to some expression

Terminal is some literal that cannot be broken down
and stays constant

Terminals typically lower case
Non-terminals start with capital letter

Production Expressions
Sequence
A ::= B C
An A is a B followed by a C

Iteration
A ::= B*
An A is zero or more Bs
A ::= B+
Shorthand for A ::= B B*

Choice
A ::= B | C
An A is a B or a C

Option
A ::= B?
An A is a B or is empty

A Grammar for Email

Email ::= Word “@” Domain
Domain ::= LowLevel+ TopLevel
LowLevel ::= Word “.”
TopLevel ::= com | edu | org
Word ::= Char+
Char ::= alpha | digit
Removing Extra Whitespace

```java
nextByte = in.read();
while(nextByte != -1) {
    while((nextByte != -1) && (nextByte != ' ')) {
        out.write(nextByte);
        nextByte = in.read();
    }
    out.write(nextByte);
    while((nextByte != -1) && (nextByte == ' ')) {
        nextByte = in.read();
    }
}
```

Grammar
Sentence ::= Blob*
Blob ::= Word Gap
Word ::= (non whitespace character)+
Gap ::= (whitespace character)+

Operations
(1) nextChar = in.read()
(2) out.write(nextChar)

Run Length Encoding

```java
nextByte = in.read();
while(nextByte != -1) {
    runByte = nextByte;
    count = 1;
    nextByte = in.read();
    while((nextByte != -1) &&
        (nextByte == runByte) &&
        (count < 255)) {
        count++;
        nextByte = in.read();
    }
    out.write(runByte);
    out.write(count);
}
```

Grammar
Runs ::= Run*
Run ::= First-byte Remaining-bytes
First-byte ::= byte
Remaining-bytes ::= byte*

Operations
(1) nextByte=in.read()
(2) count++
(3) count=1
(4) out.write(nextByte)
(5) out.write(count)
(6) runByte=nextByte