

```
if (x < 3) { // comment
```

we see:

```
let y = 52;
}
```

Computer sees: "if (x < 3) { let ... }" 10/11BCD

Java program

```
"if (x < 3) { ..."
```

10/11A

tokenizing /
scanning / lexing /
lexical analysis

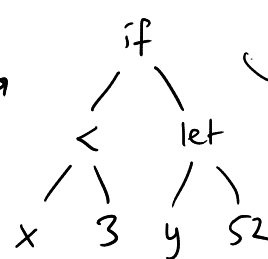
- get rid of spaces + comments, break into logical "tokens":

token stream

```
[if, (, x, <, 3, ...]
```

parsing.

Abstract Syntax tree (AST)



code generation

VM code.

(we can skip this step)

Tokenizing

Use a simple example language with:

- variables
- integers
- +, -, *, parens
- let statements
- comments

```
let x5 = y + 99; // blah
let z = (x5 + 3) - (2 * 7);
```

keyword / symbols / integer literals.
identifiers

kinds of tokens:

Keywords (let)

Symbols (-, +, i, =, etc.)

Identifiers (Names the programmer made up).

Integer literals (eg. 99, 3, etc.)

Tokenizer

- init: take input text, initialize
- has_more_tokens: are there more tokens?
- advance: go to the next token.