Java

expression - any line or Part of code that can compute a value.

2•2=4→integer expression.

a >=4 → boolean (true or false) expression.

"Hi" +1 = "Hi1" → String expression.

a -> Variable as expression.

Basic data types

16-bit 32-bit 64-bit 8-bit 16-bit

integers → char, int, long, etc, byte, short

integers + floating point = floating point.

32-bit 64-bit

Floating → float, double

textual → String

Void -> No data

larger data type gives, usually, larger type in expression.

Statement - any line/part of code that performs an action. Statements have to do something

examples: if-stmt:

if (<boolean expression>) <statement> else <statement>

bracket-stmt

{<statement(s)}

loop-stmts: runs statement until expression is false

do <statement> while(<boolean expression>);

while(<boolexpr>) <statement>

declaration stmt:

<type-name> <variablename> = <expression that matches type-name>;

return stmt: exits the method back to where it was called.

return <expression of the method's type>;

if method has a void return type => return;

Switch-stmt:

```
switch(<integer or string expression>) {
    Case <integer or string expression constant>:
        <Statement(s)>;
        break;
    default:
        <statement(s)>
}
```

No duplicate cases
Can have many cases as needed.
default case optional

Flow-stmt:

'break' & 'continue'

in loops, break completely ends a loop.
continue skips an iteration, of the loop, and recheck loop boolean expression.

For-stmt: Part of loop family

```
For(<Variable declaration or Variable ~~statemt~~ statement;
    <boolean ~~condition~~ expression>;
    <Variable expression>) <statement>
```

example "For(int i=0; i<10; i++){"
        code
    }

= "int i=0;
   while(i<10){
       code;
       i++;
   }"

Variable expression statements

"a" represents any Variable name.

$\underbrace{a = \text{<Value that's same type as Variable>;}}$

if variable 'a' already exists

assignment statement

if a is an integer or Floating Point type
a=a+b; -> a+=b;

Operators binary ↙modulo

+, -, *, /, % Arithmetic

&, |, ^, ! Bitwise

&&, || Logical

<, >, <=, >=, ==, != Relational

a[integer expressio] array Index

a.b — field accessor

a.b(_)
a.b.C(_) } method calls

---

10 / 3 = 3

10 % 3 = 1 → Closest multiple of 3 to 10 ⇒ 9
10-9=1

13%3 = → 13-12=1

9%3 = 0

6%2=0

20%2=0

16%2=0

a%b=0 if a is a multiple of b.