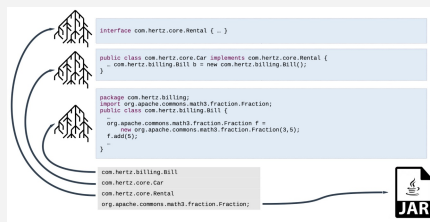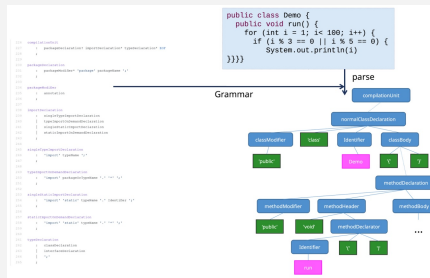## Source code to fully qualified name resolution

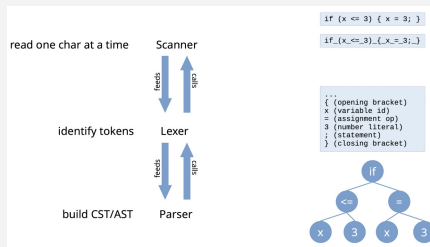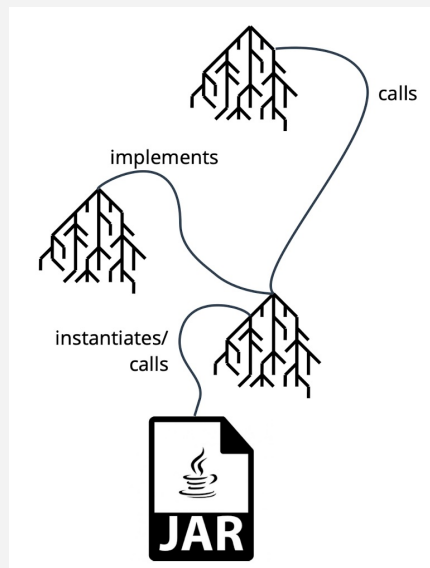

Concrete Syntax Tree (CST)

- 1:1 mapping of source code to a tree representation.
- Consists of simple tokens (i.e. strings, numbers).

From source code to lexing and parsing
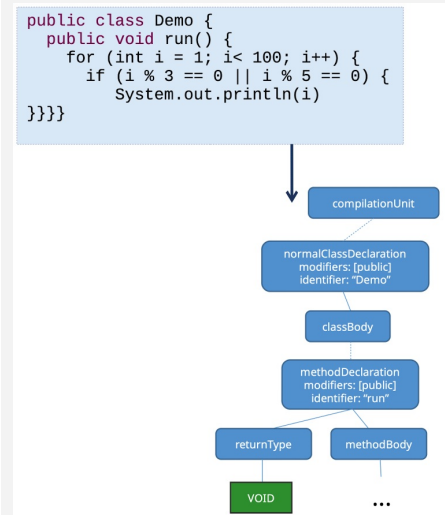

From source code to building an object model

Source code to writing output from the model

Bytecode / assembly / machine code.

- Generate machine or VM executable code.

Source code of a different language.

- Code synthesis: generating source code from the object model.

- Decompilation: generating the original source code.

Code metrics:

- Analyse the object model to compute metrics, i.e. number of classes, methods, complexity etc. > next week.

Metadata, e.g. lookup tables, API documentation.

- IDE needs info on classes / methods for code completion.

Source code to writing output from the model (cont)

- HTML API docs are hyperlinked.

Abstract Syntax Tree (AST)

```java
public class Demo {
  public void run() {
    for (int i = 1; i< 100; i++) {
      if (i % 3 == 0 || i % 5 == 0) {
        System.out.println(i)
}}}}
```


- Only necessary parts.

- Possible consisting of complex node objects.

- Possibly restructured for simplicity.

Source code to fully qualified name resolution

Internal representations:

E.g. JVM method descriptors:

- Method signature in Java source:

```
Object[] m(int i, double d,
Thread t) { ... }
```

- Method descriptor in JVM class file:

```
(IDLjava/lang/Thread;)[Ljava/l-
ang/Object
```