Abstract Syntax Trees

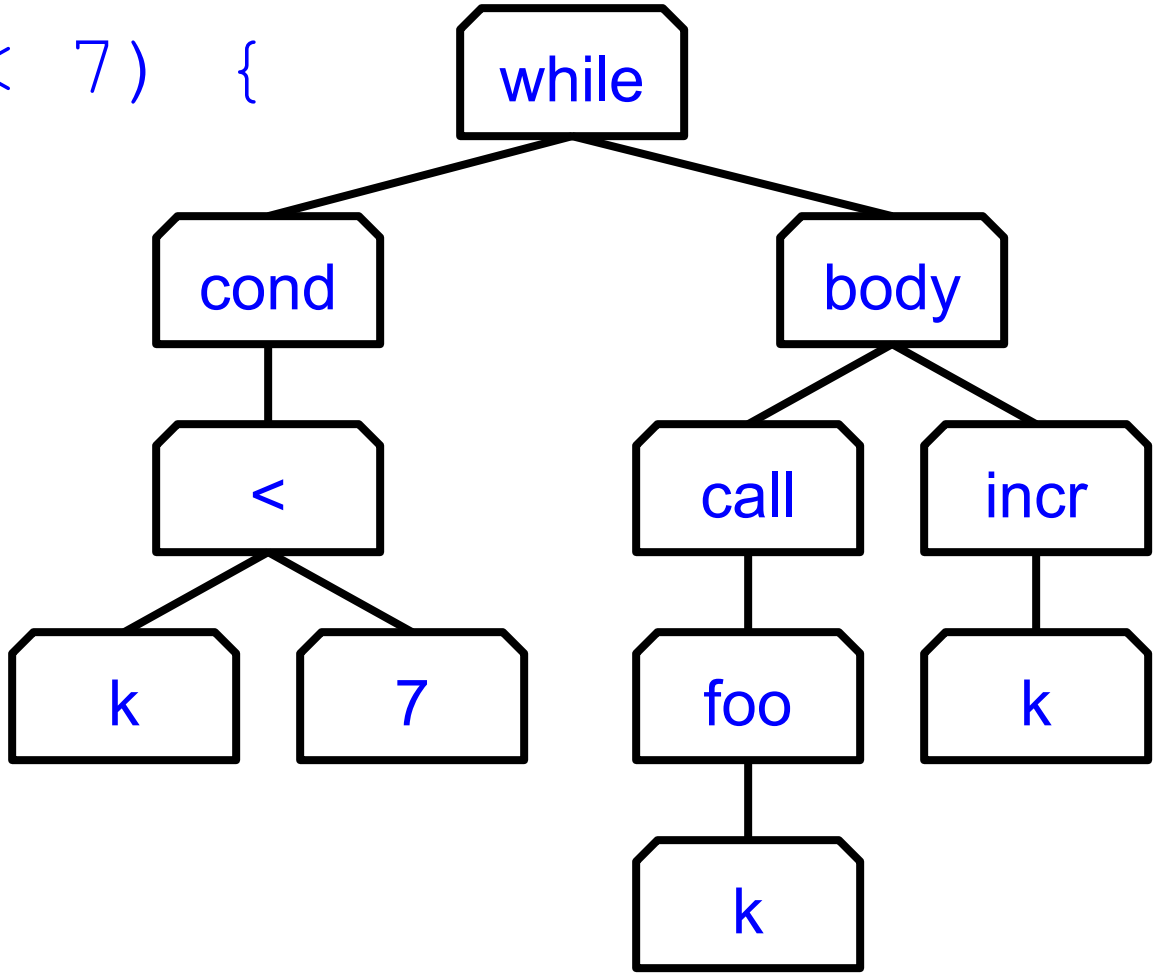



Abstract Syntax Tree

- An ***abstract syntax tree (AST)*** is a ***`tree`*** model of an entire program or a certain "program structure" (e.g., a statement or an expression in a Java program)
- An AST is "abstract" in the sense that some of the actual characters used in the "concrete" program text do not appear in the AST

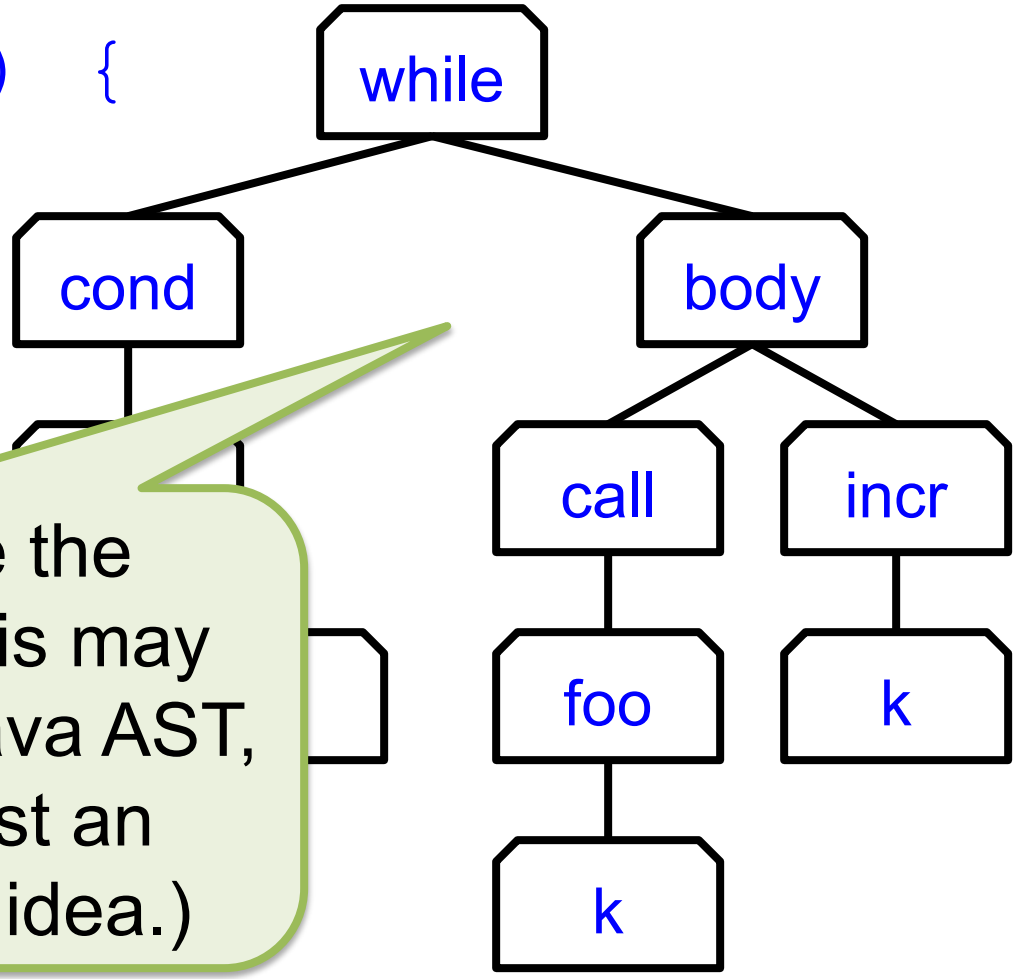
Example: A Java Statement

```
while (k < 7) {
   foo(k);
   k++;
}
```



Example: A Java Statement

```
while (k < 7) {
   foo(k);
   k++;
}
```



You should see the connections! (This may not be an *actual* Java AST, however; it is just an illustration of the idea.)

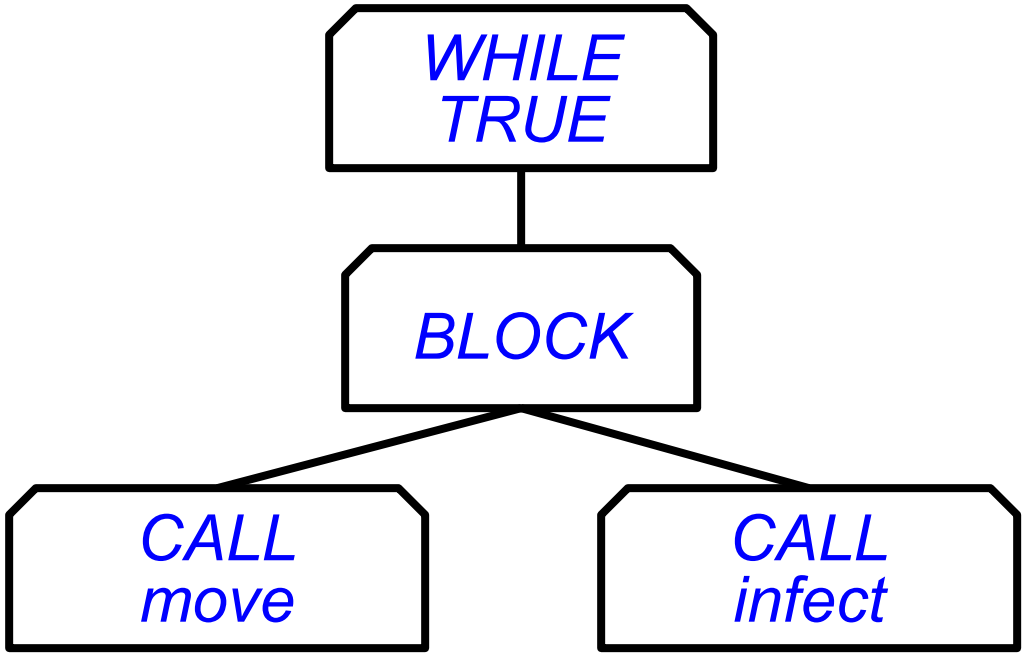
Example: A BL Statement

WHILE true DO

move

infect

END WHILE



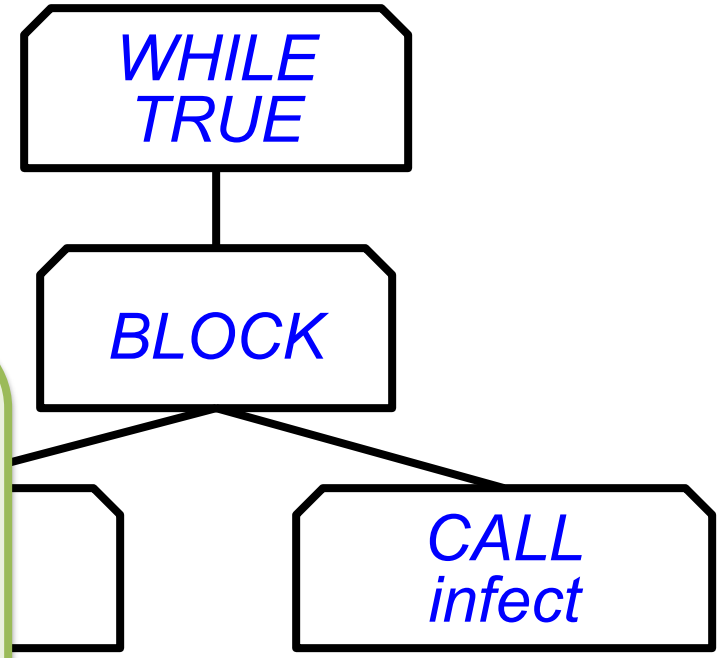
Example: A BL Statement

WHILE true DO

move

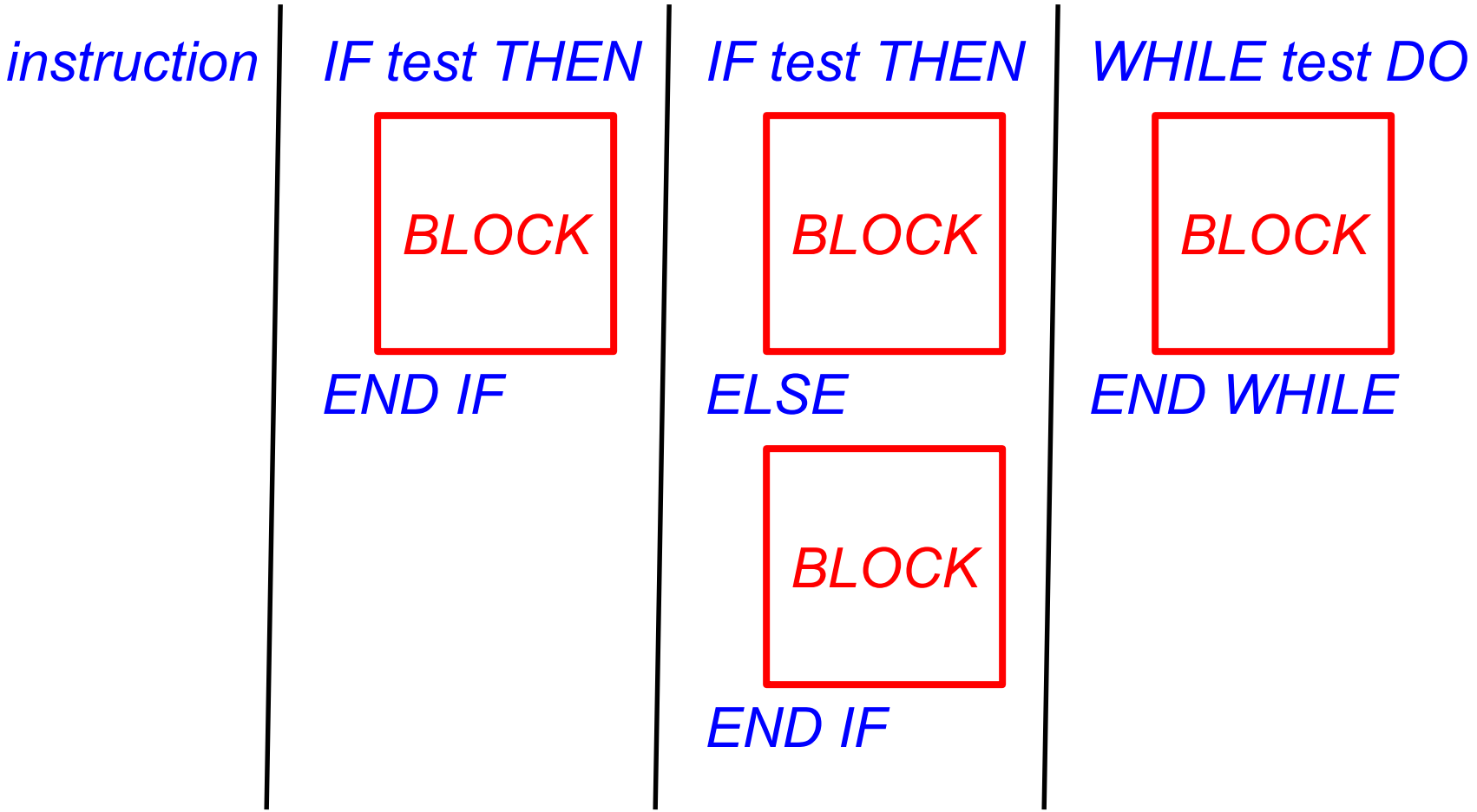
infect

END WHILE

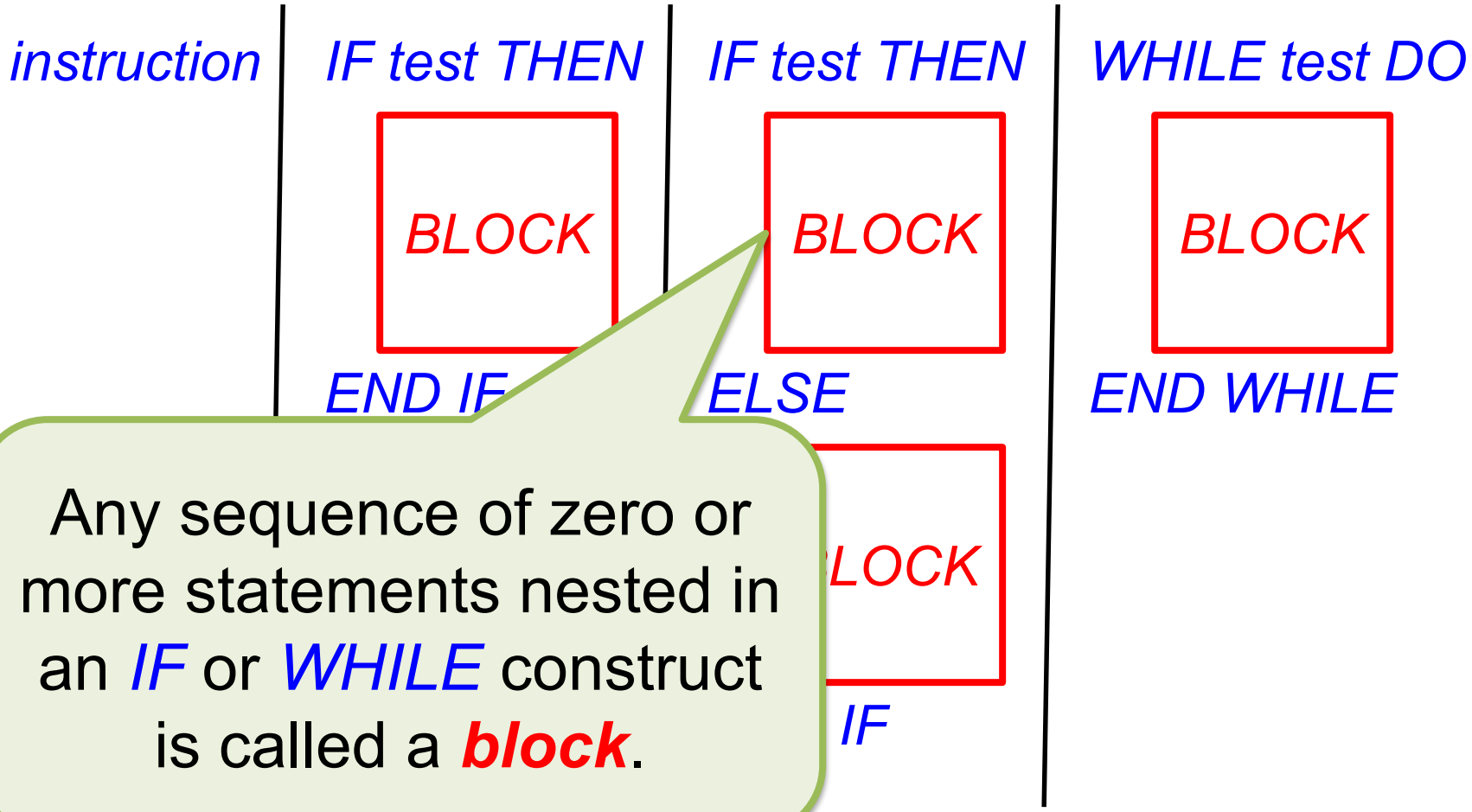


You should see the connections! (This is an actual AST for BL; notice it uses a different "design".)

BL Statement Kinds



BL Statement Kinds

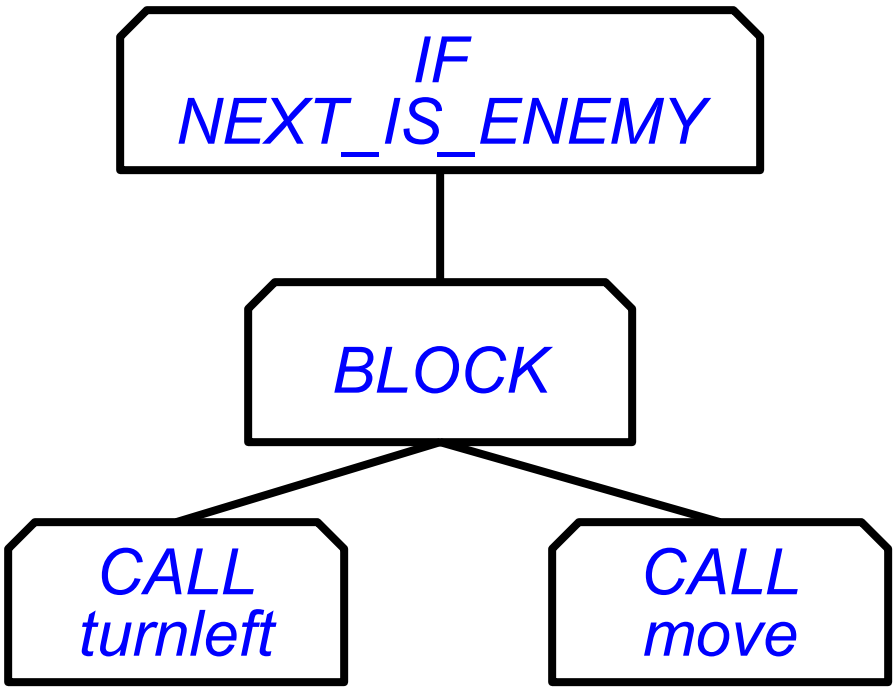


CALL Statement

BL Example	***AST***
turnleft	

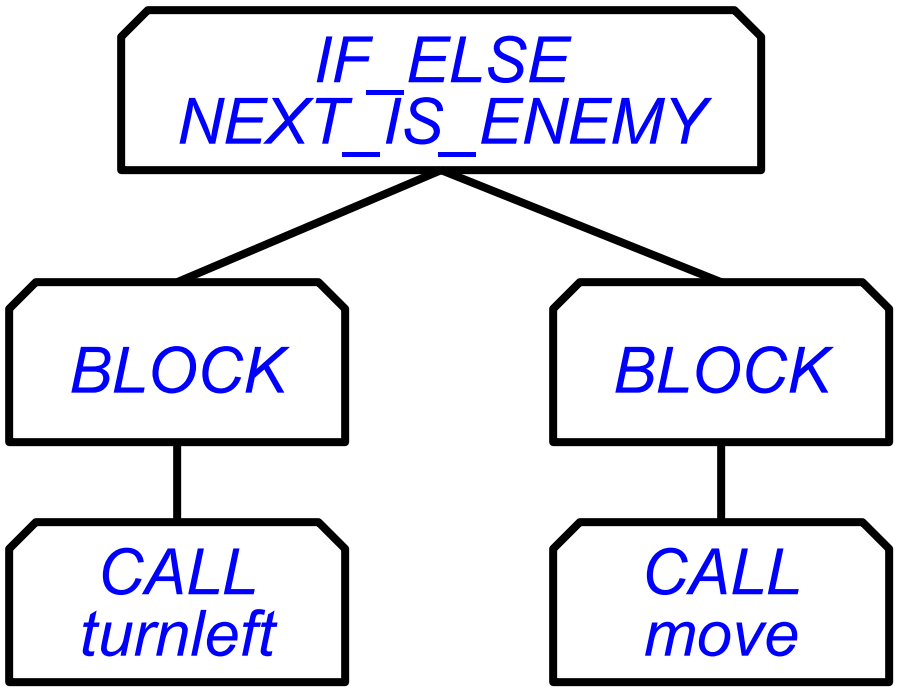
IF Statement

BL Example	***AST***

IF next-is-enemy THEN
turnleft
move
END IF
	

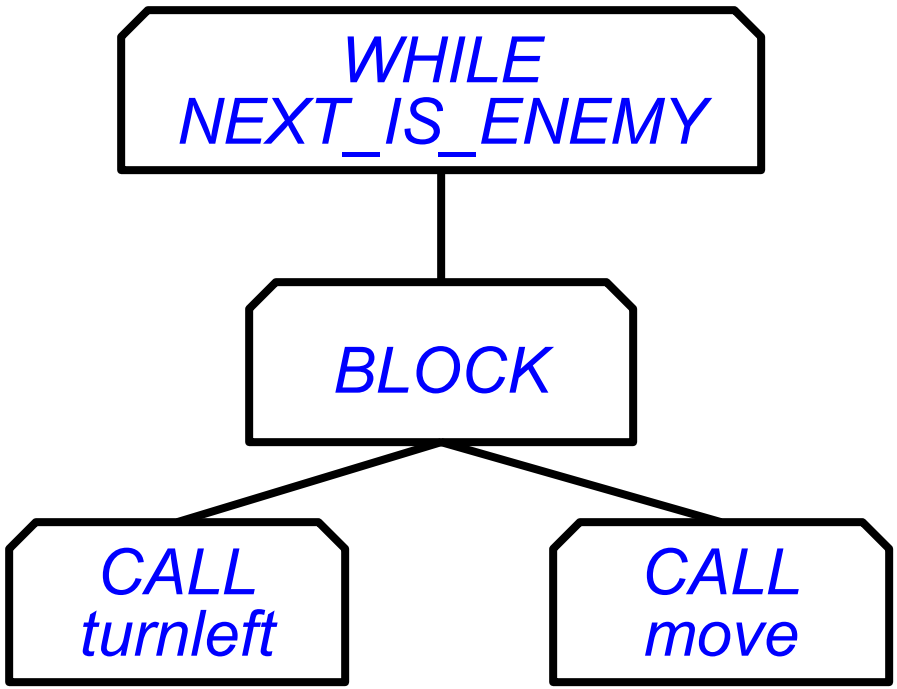
IF_ELSE Statement

BL Example	**AST**

IF next-is-enemy THEN
turnleft
ELSE
move
END IF
	

WHILE Statement

BL Example	***AST***

WHILE next-is-enemy DO
turnleft
move
END WHILE
	

Why *BLOCK*?

- Draw the AST for this BL code with and without the intermediate notion of *BLOCK*:

```
IF next-is-empty THEN
    move
    turnright
ELSE
   infect
END IF
```

Why *BLOCK*?

- Draw the AST for t
  without the interme

```
IF next-is-empty THEN
    move
    turnright
ELSE
   infect
END IF
```

If it's not clear, draw the AST for this code with and without *BLOCK*:

```
IF next-is-empty THEN
    move
ELSE
    turnright
    infect
END IF
```

AST Node Labels

- An AST for BL is a ***`tree of`*** ... what?
- Each node has *some* of the following:
  - The ***kind*** of statement (e.g., *BLOCK*, *WHILE*)
  - The ***test*** condition (e.g., *NEXT_IS_EMPTY*, *TRUE*)
  - The ***call*** of an instruction (e.g., *infect*, *move*), realizing that this may be an instruction defined elsewhere in the program (e.g., *FindObstacle* in an earlier BL example)

This mathematical 3-tuple of information (of which either ***test*** or ***call*** might be relevant, depending on the ***kind***) will be called a *STATEMENT_LABEL*.

t?

g:

- The ***kind*** of statement (e.g., *BLOCK*, *WHILE*)
- The ***test*** condition (e.g., *NEXT_IS_EMPTY*, *TRUE*)
- The ***call*** of an instruction (e.g., *infect*, *move*), realizing that this may be an instruction defined elsewhere in the program (e.g., *FindObstacle* in an earlier BL example)

The mathematical model of an AST for a BL statement is therefore a ***tree of*** *STATEMENT_LABEL*.

t?

g:

- The ***kind*** of statement (e.g., *BLOCK*, *WHILE*)
- The ***test*** condition (e.g., *NEXT_IS_EMPTY*, *TRUE*)
- The ***call*** of an instruction (e.g., *infect*, *move*), realizing that this may be an instruction defined elsewhere in the program (e.g., *FindObstacle* in an earlier BL example)

Resources

- Wikipedia: Abstract Syntax Tree
  - http://en.wikipedia.org/wiki/Abstract_syntax_tree