Parsing the Grammar

The Grammar

statement → { **statement*** }

statement → if(**expression**) then
    statement [else **statement**]

booleanOp → && | ||

expression → **expPart** [**binop expPart**]

expPart → (**expression**)

- Production Rules
- Recursive Definitions
- Circular Definitions
- Terminal Definitions

Example (tokens)

```
int main ( )
    int a; int b;
{
    b= 3;
    print b + (a = b);
}
```

- Types of Tokens:
  - Integer
  - Float
  - Word
  - String
  - Character
  - Operator
  - Comment
  - EOF

Using the Tokenizer:

- Methods:
- peekAtKind()
- Gets (subset):
  - getInt
  - getWord
  - getCharachter
  - getOp
- match (exception)
- check (munches)
- test (doesn’t munch)
- pushBack()/canPushBack()

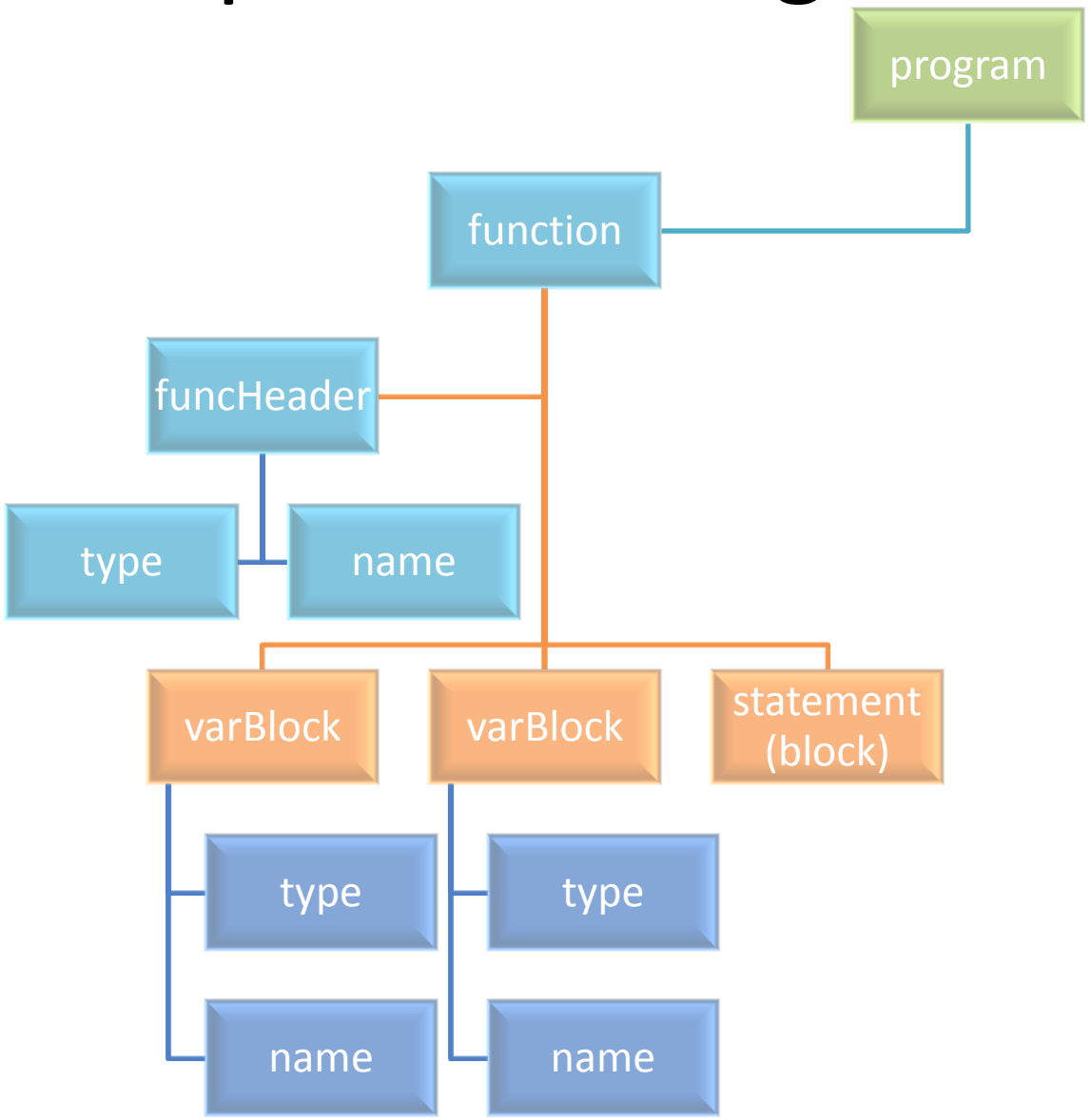
Using the Tokenizer 2:

```
public static boolean checkToken(Tokenizer tokenizer, char c)
{
      if(tokenizer.peekAtKind() != Tokenizer.TokenType.OPERATOR)
            return false;
    return tokenizer.check(c);
}
```

```
////////////////// Check for Character ////////////////////
if(tokenizer.peekAtKind() == Tokenizer.TokenType.CHARACTER)
{
    obj.constantType = Type.Character;
    obj.value = tokenizer.getCharacter();
    return obj;
}
```

Short Example of Parsing

```
int main ( )
    int a; int b;
{
    b= 3;
    print b + (a = b);
}
```



Short Example of Parsing (Cont…)

```
int main ( )
    int a; int b;
{
    b= 3;
    print b + (a = b);
}
```