Visitor

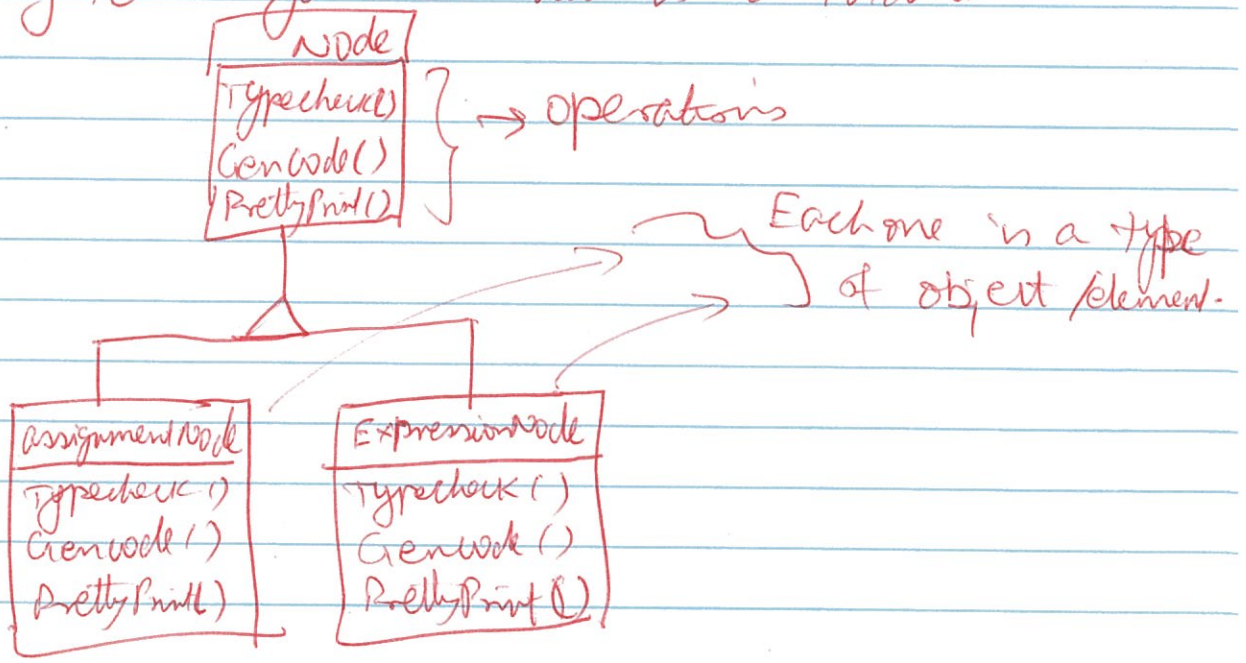
It represents an operation that is to be performed on the elements of an object structure.

The need for this is when the object structure remains the same and there are several operations to be done on the different objects (differently).

Eg. Source code has diff kinds of elements
- ↳ assignment
- ↳ expression
- ↳ variable declaration etc.

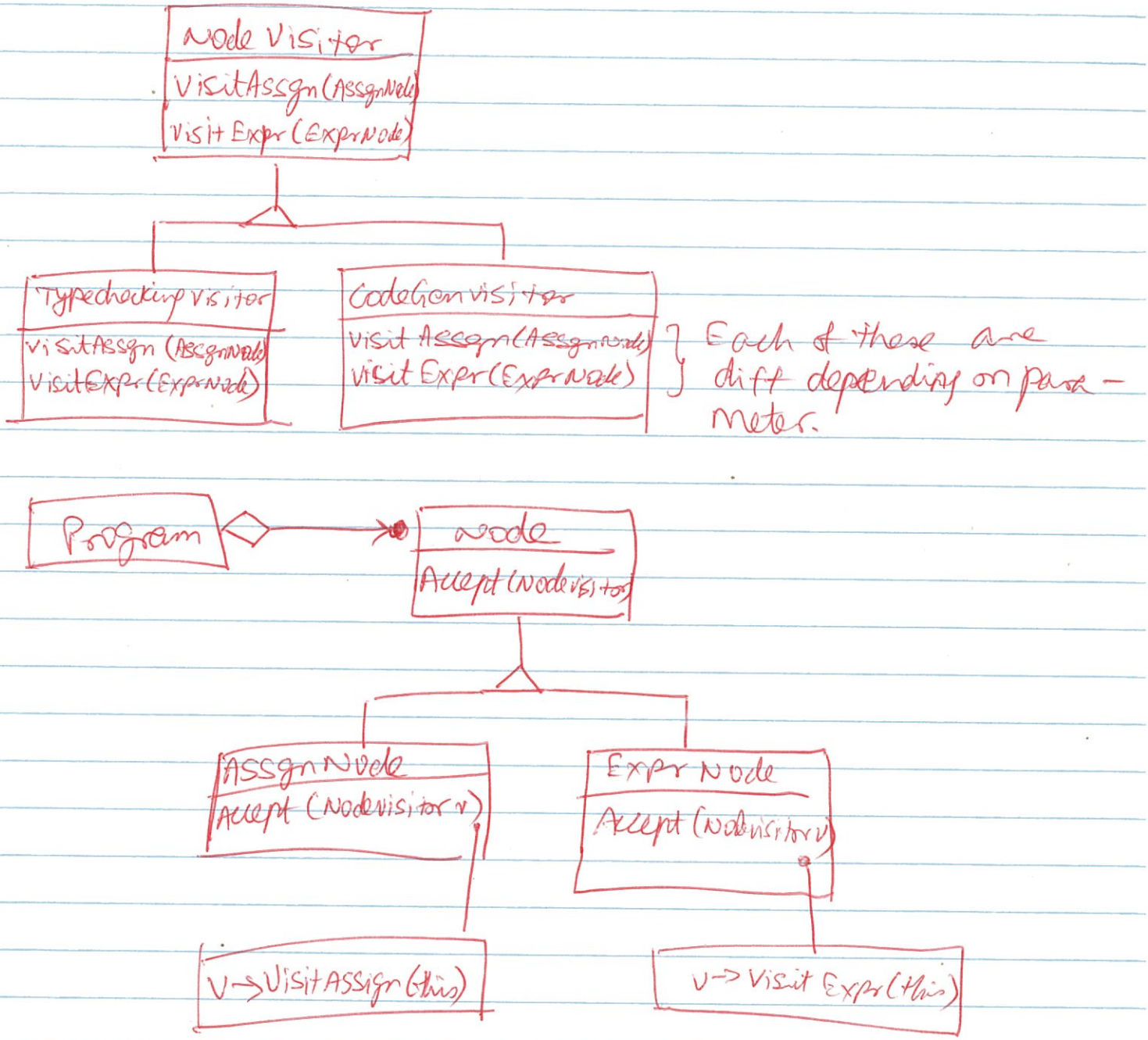
The operations on the code elements are
- ↳ Type checking
- ↳ generating code
- ↳ Pretty printing.

One way to organize this is a follows.



Problem with this is that when we add a new ~~pattern~~ operation, we need to edit every class.

The visitor pattern resolves it as follows.



Depending on what operation needs to be done, the appropriate concrete visitor class is called.

Participants

- Visitor (NodeVisitor)
- ConcreteVisitor (TypeCheckingVisitor)
- Element (Node)
- ConcreteElement (AssgnNode)
- ObjectStructure (Program)

Advr

① It makes adding an operation easy. Just add a new Subclass to visitor

② It gathers similar operations in one class & splits dissimilar ones into ~~separate~~ separate classes.

③ They can accumulate state info as the traversal proceeds. In our eppie, we can know which line is being operated on by what operation

Disadv

① If Object structure changes, the edits required are difficult. For eg, if we have a new code element, then changing the node Subclasses is difficult.

NFP's

① Modifiability
② Reliability + since it improves debugging (state info)
③ comprehensability.

Related patterns

→ Composite
→ Interpreter.