JJTREE VISITOR SUPPORT

- Assuming that
  - the jjtree file is called HL.jjt
  - In HL.jjt, `VISITOR=true;`
- SimpleNode and all the AST classes will then include the following method:

```
/** Accept the visitor. **/
public Object jjtAccept(HLVisitor visitor, Object data) {
  return visitor.visit(this, data);
}
```

- A new interface HLVisitor.java will be created:

```
public interface HLVisitor
{
  public Object visit(SimpleNode node, Object data);
  public Object visit(ASTEOFReached node, Object data);
  public Object visit(ASTbody node, Object data);
  public Object visit(ASTclause node, Object data);
  // etc…
}
```

- A new default visitor class HLDefaultVisitor.java will be created:

```
public class HLDefaultVisitor implements HLVisitor {
  public Object defaultVisit(SimpleNode node, Object data){
    node.childrenAccept(this, data);
    return data;
  }
  public Object visit(SimpleNode node, Object data){
    return defaultVisit(node, data);
  }
  // etc…
}
```

WORKING WITH JJTREE VISITORS

- To write a visitor, define a new class that implements HLVisitor & write code for each of the methods defined in the interface. To do so, you can work either from the interface HLVisitor.java which has all the method signatures, of from the HLDefaultVisitor.java that has code stubs for all the methods.

In all all cases your file should start with something like.

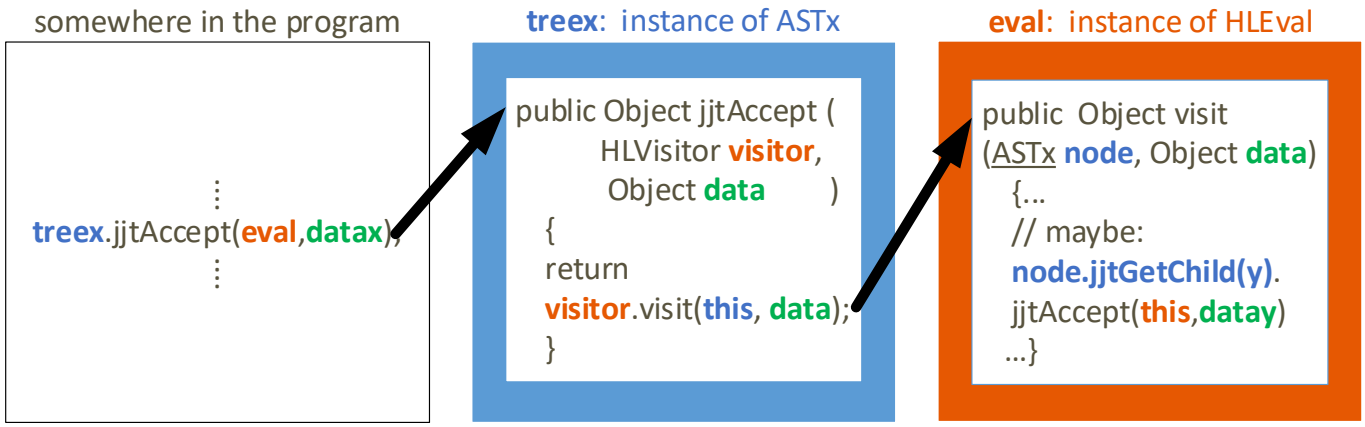
```
public class HLEval implements HLVisitor
```

- The visitors will need to access the node fields of the AST classes. Methods to support this are defined in the interface Node.java and the class SimpleNode.java

- To use a visitor, instantiate the class and ask the AST to accept the instantiation's visit.

```
private static HL parser;
SimpleNode tree;
HLEval eval = new HLEval();
tree = parser.start();
System.out.println(tree.jjtAccept(eval,null));
```

- Here is the sequence of function calls that make visitors work:



  - The method jjtAccept of the AST node "accepts" the visitor object with an additional object, called data, which may be useful during that visit.
  - The AST node instance invokes the visit method of the visitor with itself and the data object as parameters.
  - The visit method of visitors is overloaded, with one method for each type of ASTnode, so the correct method will be activated based on the type of AST node which is passed as the first parameter.

- Note that as a result of this chain of function calls the data is passed to the visit method. In many cases this data is simply a null pointer, but it can be useful if a parent wants to send data to a child to be used during the visit to the child.