```
lastVisited = inserted:
                                                           if (rightSibling == null && lastVisited == node.getFirstChild()) {
                                                              rightSibling = node.getSecondChild();
                                                           if (inserted == node.getSecondChild()) { // Parenthesis
                                                              isSecondApplicationChild = true;
                 InsertionVisitor
-inserted: LambdaTerm
                                                            if (view.hasNodeView(node))
-view: LambdaTermView
                                                              insertChild(view.getActor(node));
-rightSibling: LambdaTerm = null
                                                            } else {
-lastVisited: LambdaTerm
                                                              lastVisited = node;
-isSecondApplicationChild: boolean = false
                                                              node.getParent().accept(this);
+InsertionVisitor(inserted: LambdaTerm, (
  view: LambdaTermView)
+visit(node: LambdaApplication)
+visit(node: LambdaAbstraction)
+visit(node: LambdaVariable)
                                                            insertChild(view.getActor(node));
-insertChild(parent: LambdaNodeView)
                                                            assert(false);
                                                            parent.insertChild(NodeViewCreator.create(inserted, parent, isSecondApplicationChild, view), rightSibling);
                                                            inserted.accept(new InsertionRecursionVisitor(view));
               NodeViewCreator
-result: LambdaNodeView = null
-parent: LambdaNodeView
-view: LambdaTermView
-canCreateApplicationNodeView: boolean
                                                             if (canCreateApplicationNodeView) {
+NodeViewCreator(parent: LambdaNodeView.
                                                               result = new LambdaParenthesisView(node, parent, view);
  canCreateApplicationNodeView: boolean,
  view: LambdaTermView)
+visit(node: LambdaApplication) (
                                                             NodeViewCreator visitor = new NodeViewCreator(parent, canCreateApplicationNodeView, view);
+visit(node: LambdaAbstraction)
                                                             term.accept(visitor):
+visit(node: LambdaVariable)
                                                             return visitor.result;
+create(term: LambdaTerm,
  parent: LamdbaNodeView, (
  canCreateApplicationNodeView: boolean,
  view: LambdaTermView): LambdaNodeView)
             InsertionRecursionVisitor
                                                            if (node.getFirstChild() != null) {
-view: LambdaTermView
                                                              node.accept(new InsertionVisitor(node.getFirstChild(), view));
+InsertionRecursionVisitor(view: LambdaTermView)
+visit(node: LambdaApplication) (
                                                            if (node.getSecondChild() != null) {
+visit(node: LambdaAbstraction)
                                                              node.accept(new InsertionVisitor(node.getSecondChild(), view)):
+visit(node: LambdaVariable)
```