

```
package microjs.jcompiler.middleend.kast;

import java_cup.runtime.ComplexSymbolFactory.Location;

public class KEchange extends KStatement {
    private String name_g;
    private String name_d;

    public KEchange(String name_g, String name_d,
                    Location startPos, Location endPos) {
        super(startPos, endPos);
        this.name_g = name_g;
        this.name_d = name_d;
    }

    @Override
    public void accept(KASTVisitor visitor) {
        visitor.visit(this);
    }

    public String getVarNameG() {
        return name_g;
    }

    public String getVarNameD() {
        return name_d;
    }
}
```

```
package microjs.jcompiler.middleend.kast;

public interface KASTVisitor {
    /* program visitor */
    public void visit(KProg prog);
    /* statement visitors */
    public void visit(KVoidExpr stmt);
    public void visit(KVar stmt);
    public void visit(KIf stmt);
    public void visit(KSeq stmt);
    public void visit(KAssign stmt);
    public void visit(KReturn stmt);
    /* expression visitors */
    public void visit(KInt expr);
    public void visit(KTrue expr);
    public void visit(KFalse expr);
    public void visit(KEVar expr);
    public void visit(KCall expr);
    public void visit(KClosure expr);

    public void visit(KEchange stmt);
}
```

```

package microjs.jcompiler.middleend.kast;

public class KPrettyPrint implements KASTVisitor {
    private StringBuilder buf;
    private int indent_level = 0;
    public static int INDENT_FACTOR = 2;

    public KPrettyPrint() {
        reset();
    }

    private void reset() {
        indent_level = 0;
        this.buf = new StringBuilder();
    }

    private void indent() {
        for(int i=0;i<indent_level * INDENT_FACTOR;i++) {
            buf.append(' ');
        }
    }

    public void visit(KProg prog) {
        reset();
        prog.getBody().accept(this);
    }

    public void visit(KVoidExpr stmt) {
        indent();
        buf.append("KVoidExpr[\n");
        indent_level++;
        stmt.getExpr().accept(this);
        buf.append("\n");
        indent_level--;
        indent();
        buf.append("]");
    }

    public void visit(KVar var) {
        indent();
        buf.append("KVar(");
        buf.append(var.getName());
        buf.append(")[\n");
        indent_level++;
        indent();
        var.getExpr().accept(this);
        buf.append("\n");
        indent_level--;
        indent();
        buf.append("]");
    }

    public void visit(KIf stmt) {
        indent();
        buf.append("KIf[\n");
        indent_level++;
        stmt.getCond().accept(this);
        buf.append("\n");
        indent();
        buf.append(") <then>[\n");
        indent_level+=2;
        stmt.getThen().accept(this);
        buf.append("\n");
        indent_level-=2;
        indent();
        buf.append("] <else>[\n");

```

```

        indent_level+=2;
        stmt.getElse().accept(this);
        buf.append("\n");
        indent_level-=2;
        indent();
        buf.append("]\n");
        indent_level--;
        indent();
        buf.append("]");
    }

    public void visit(KSeq seq) {
        indent();
        buf.append("KSeq[\n");
        indent_level++;
        for(KStatement stmt : seq.getStatements()) {
            stmt.accept(this);
            buf.append("\n");
        }
        indent_level--;
        indent();
        buf.append("]");
    }

    public void visit(KAssign stmt) {
        indent();
        buf.append("KAssign(");
        buf.append(stmt.getVarName());
        buf.append(")[\n");
        indent_level++;
        stmt.getExpr().accept(this);
        buf.append("\n");
        indent_level--;
        indent();
        buf.append("]");
    }

    public void visit(KReturn stmt) {
        indent();
        buf.append("KReturn[\n");
        indent_level++;
        stmt.getExpr().accept(this);
        buf.append("\n");
        indent_level--;
        indent();
        buf.append("]");
    }

    public void visit(KInt expr) {
        indent();
        buf.append("KInt[");
        buf.append(expr.getValue());
        buf.append("]");
    }

    public void visit(KTrue expr) {
        indent();
        buf.append("KTrue");
    }

    public void visit(KFalse expr) {
        indent();
        buf.append("KFalse");
    }

```

```
public void visit(KEVar expr) {
    indent();
    buf.append("KEVar (");
    buf.append(expr.getName());
    buf.append(")");
}

public void visit(KCall expr) {
    indent();
    buf.append("KCall[\n");
    indent_level++;
    expr.getFun().accept(this);
    buf.append("\n");
    indent_level--;
    indent();
    buf.append("](\n");
    indent_level++;
    String sep = "";
    for(KEExpr arg : expr.getArguments()) {
        buf.append(sep);
        if(sep.equals("")) {
            sep = ",\n";
        }
        arg.accept(this);
    }
    buf.append("\n");
    indent_level--;
    buf.append(")");
}

public void visit(KClosure expr) {
    indent();
    buf.append("KClosure (");
    String sep = "";
    for(String param : expr.getParams()) {
        buf.append(sep);
        if(sep.equals("")) {
            sep = ", ";
        }
        buf.append(param);
    }
    buf.append(")(\n");
    indent_level++;
    expr.getBody().accept(this);
    buf.append("\n");
    indent_level--;
    indent();
    buf.append("]");
}

public void visit(KEchange stmt) {
    indent();
    buf.append("KEchange(\n");
    buf.append(stmt.getVarNameG());
    buf.append(", ");
    buf.append(stmt.getVarNameD());
    buf.append(")");
    buf.append("\n");
}

@Override
public String toString() {
    return buf.toString();
}
}
```