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

import java_cup.runtime.ComplexSymbolFactory.Location;

public class KAssign extends KStatement {
    private String name;
    private KExpr expr;

    public KAssign(String name, KExpr expr, Location startPos, Location endPos)
{
        super(startPos, endPos);
        this.name = name;
                this.expr = expr;
    }

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

        public String getVarName() {
                return name;
        }

        public KExpr getExpr() {
                return expr;
        }

}
```

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

import java_cup.runtime.ComplexSymbolFactory.Location;

public abstract class KASTNode {
        Location startPos;  // objects with getLine and getColumn
        Location endPos;


        /* package */ KASTNode(Location startPos, Location endPos) {
                this.startPos = startPos;
                this.endPos = endPos;
        }

        public Location getStartPos() {
                return startPos;
        }

        public Location getEndPos() {
                return endPos;
        }

        public int getEndColumn() {
                return endPos.getColumn();
        }

        public abstract void accept(KASTVisitor visitor);

}
```

```
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);
}
```

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

import java.util.List;

import java_cup.runtime.ComplexSymbolFactory.Location;

public class KCall extends KExpr {
    private KExpr fun;
    private List<KExpr> arguments;

    public KCall(KExpr fun, List<KExpr> arguments, Location startPos, Location e
ndPos) {
        super(startPos, endPos);
                this.fun = fun;
                this.arguments = arguments;
    }

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

    public KExpr getFun() {
        return fun;
    }

    public List<KExpr> getArguments() {
        return arguments;
    }

}
```

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

import java.util.List;

import java_cup.runtime.ComplexSymbolFactory.Location;

public class KClosure extends KExpr {
    private List<String> params;
    private KStatement body;

    public KClosure(List<String> params, KStatement body, Location startPos, Loc
ation endPos) {
        super(startPos, endPos);
        this.params = params;
        this.body = body;
    }

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

    public List<String> getParams() {
        return params;
    }

    public KStatement getBody() {
        return body;
    }
}
```

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

import java_cup.runtime.ComplexSymbolFactory.Location;

public class KEVar extends KExpr {
        private String name;

        public KEVar(String name, Location startPos, Location endPos) {
                super(startPos, endPos);
                this.name = name;
        }

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

    public String getName() {
        return name;
    }

}
```

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

import java_cup.runtime.ComplexSymbolFactory.Location;

public abstract class KExpr extends KASTNode {

        /* package */ KExpr(Location startPos, Location endPos) {
                super(startPos, endPos);
        }


}
```

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

import java_cup.runtime.ComplexSymbolFactory.Location;

public class KFalse extends KExpr {

        public KFalse(Location startPos, Location endPos) {
                super(startPos, endPos);
        }

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

}
```

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

import java.util.List;

import java_cup.runtime.ComplexSymbolFactory.Location;

public class KIf extends KStatement {
    private KExpr cond;
    private KStatement kthen;
    private KStatement kelse;

    public KIf(KExpr cond, KStatement kthen, KStatement kelse, Location startPos
, Location endPos) {
        super(startPos, endPos);
        this.cond = cond;
        this.kthen = kthen;
        this.kelse = kelse;
    }

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

        public KExpr getCond() {
                return cond;
        }

        public KStatement getThen() {
                return kthen;
        }

        public KStatement getElse() {
                return kelse;
        }
}
```

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

import java_cup.runtime.ComplexSymbolFactory.Location;

public class KInt extends KExpr {
        private int value;

        public KInt(int value, Location startPos, Location endPos) {
                super(startPos, endPos);
                this.value = value;
        }

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

    public int getValue() {
        return value;
    }

}
```

```java
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");
```

```java
                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(KExpr 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("]");
        }

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

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

import java_cup.runtime.ComplexSymbolFactory.Location;

public class KProg extends KASTNode {
        private String filename;
        private KStatement body;

        public KProg(String filename, KStatement body, Location startPos, Locati
on endPos) {
                super(startPos, endPos);
                this.filename = filename;
                this.body = body;
        }

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

        public KStatement getBody() {
                return body;
        }

        @Override
        public String toString() {
                KPrettyPrint pretty = new KPrettyPrint();
                accept(pretty);
                return pretty.toString();
        }

}
```

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

import java_cup.runtime.ComplexSymbolFactory.Location;

public class KReturn extends KStatement {
    private KExpr expr;

    public KReturn(KExpr expr, Location startPos, Location endPos) {
        super(startPos, endPos);
                this.expr = expr;
    }

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

        public KExpr getExpr() {
                return expr;
        }

}
```

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

import java.util.List;

import java_cup.runtime.ComplexSymbolFactory.Location;

public class KSeq extends KStatement {
        private List<KStatement> stmts;

        public KSeq(List<KStatement> stmts, Location startPos, Location endPos)
{
                super(startPos, endPos);
                this.stmts = stmts;
        }

        public static KStatement buildKSeq(List<KStatement> stmts, Location defa
ultStartPos, Location defaultEndPos) {
                if(stmts.size() == 1) {
                        // not a *real* sequence
                        return stmts.get(0);
                }

                if(!stmts.isEmpty()) {
                        defaultStartPos = stmts.get(0).getStartPos();
                        defaultEndPos = stmts.get(stmts.size()-1).getEndPos();
                }
                return new KSeq(stmts, defaultStartPos, defaultEndPos);
        }

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

        public List<KStatement> getStatements() {
                return stmts;
        };

}
```

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

import java_cup.runtime.ComplexSymbolFactory.Location;

public abstract class KStatement extends KASTNode {

        /* package */ KStatement(Location startPos, Location endPos) {
                super(startPos, endPos);
        }

}
```

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

import java_cup.runtime.ComplexSymbolFactory.Location;

public class KTrue extends KExpr {

        public KTrue(Location startPos, Location endPos) {
                super(startPos, endPos);
        }

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

}
```

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

import java_cup.runtime.ComplexSymbolFactory.Location;

public class KVar extends KStatement {
    private String name;
    private KExpr expr;

    public KVar(String name, KExpr expr, Location startPos, Location endPos) {
        super(startPos, endPos);
        this.name = name;
                this.expr = expr;
    }

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

        public String getName() {
                return name;
        }

        public KExpr getExpr() {
                return expr;
        }
}
```

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

import java_cup.runtime.ComplexSymbolFactory.Location;

public class KVoidExpr extends KStatement {
        private KExpr expr;

        public KVoidExpr(KExpr expr, Location startPos, Location endPos) {
                super(startPos, endPos);
                this.expr = expr;
        }

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

        public KExpr getExpr() {
                return expr;
        };

}
```