Aufgabe 02)

```
public class MyGrammatikVisitor extends GrammatikBaseVisitor <String>{
    private int leerzeichen = 0;
    public String visitStatement(GrammatikParser.StatementContext ctx) {
        if(ctx.BEZEICHNER() != null) {
            System.out.print(ctx.BEZEICHNER().getText() + " := ");
            visitAusdruck(ctx.ausdruck());
            System.out.println();
            return "";
        }
        return visitChildren(ctx);
    }

    public String visitStatements(GrammatikParser.StatementsContext ctx) {
        for(int j = 0; j < ctx.statement().size(); j++) {
            for (int i = 0; i < leerzeichen; i++) {
                System.out.print(" ");
            }
            visitStatement(ctx.statement(j));
        }
        return "";
    }
    public String visitIf_stats(GrammatikParser.If_statsContext ctx) {
        for(int j = 0; j < ctx.statement().size(); j++) {
            for (int i = 0; i < leerzeichen; i++) {
                System.out.print(" ");
            }
            visitStatement(ctx.statement(j));
        }
        return "";
    }
    public String visitElse_stats(GrammatikParser.Else_statsContext ctx) {
        for(int j = 0; j < ctx.statement().size(); j++) {
            for (int i = 0; i < leerzeichen; i++) {
                System.out.print(" ");
            }
            visitStatement(ctx.statement(j));
        }
        return "";
    }

    public String visitAusdruck(GrammatikParser.AusdruckContext ctx) {
        if(ctx.atom() != null){
            return visitAtom(ctx.atom());
        }
        visitAusdruck(ctx.ausdruck(0));
        System.out.print(" "+ctx.children.get(1).getText()+" ");
        visitAusdruck(ctx.ausdruck(1));
        return "";
    }

    public String visitWhile(GrammatikParser.WhileContext ctx) {
        System.out.print("while ");
        leerzeichen += 4;
        visitBedingung(ctx.bedingung());
        System.out.print(" do");
        System.out.println();
        visitStatements(ctx.statements());
        leerzeichen -= 4;
        System.out.println("end");
        return "";
    }

    public String visitIf(GrammatikParser.IfContext ctx) {
        System.out.print("if ");
        visitBedingung(ctx.bedingung());
```

```
        System.out.print(" do");
        System.out.println();
        leerzeichen += 4;
        visitIf_stats(ctx.if_stats());
        if(ctx.children.get(5).getText().equals("else")){
            System.out.println("else do");
            visitElse_stats(ctx.else_stats());
        }
        leerzeichen -= 4;
        System.out.print("end");
        return "";
    }


    public String visitBedingung(GrammatikParser.BedingungContext ctx) {
        if(ctx.INTEGER() != null){
            System.out.print(ctx.INTEGER().getText());
            return "";
        }
        visitAusdruck(ctx.ausdruck(0));
        System.out.print(" "+ctx.children.get(1).getText()+" ");
        visitAusdruck(ctx.ausdruck(1));
        return "";
    }

    public String visitAtom(GrammatikParser.AtomContext ctx) {
        System.out.print(ctx.getText());
        return visitChildren(ctx);
    }


}
```

Mein Visitor. Die visitStatement-Methode die ich zuerst hatte, musst ich in drei kleinere zerlegen, da ich unterscheiden musste, welche Statemanets zu if bzw. zu else gehören. Der Code doppelt sich hier leider, aber es funktioniert. Ansonsten war die Vorgehensweise: ich habe aus dem generierten Visitor die Methoden überschrieben. Ich habe meine Grammatik abgelaufen und die festen Werte wie while, do, end... gedruckt. Sobald ein Übergang wie Statement oder Ausdruck kam, habe ich die zugehörige Visit-Methode aufgerufen und hier wieder die festen Werte gedruckt oder weiter besucht. Dies wurde solange wiederholt, bis man auf ein Atom stößt, von dem nicht weiter besucht wird. Dieses wird auch ausgedruckt und es wird returned. Der Zähler ist für die Einrückung da, je nach Tiefe werden unterschiedlich viele Leerzeichen gedruckt.