Part 1, Code Understanding

a) How is class type represented in the intermediate representation? Show code snippets from the relevant file.

In order to identify the methods used for class type identification, we look for methods implementing the MJType class in the IR and IRbuilder classes.

```
public MJType visitType(MiniJavaParser.TypeContext ctx) { return (MJType)visitChildren(ctx); }

public MJType visitTypeBasic(TypeBasicContext ctx) {
    return (MJType)visitChildren(ctx);
}

public MJType visitTypeBoolean(MiniJavaParser.TypeBooleanContext ctx) {
    return MJType.getBooleanType();
}

public MJType visitTypeInt(MiniJavaParser.TypeIntContext ctx) {
    return MJType.getIntType();
}

public MJType visitTypeClass(MiniJavaParser.TypeClassContext ctx) {
    return MJType.getClassType(ctx.className.getText());
}

public IR visitTypeArray(TypeArrayContext ctx) {
    return MJType.getArrayType( (MJType)visitTypeBasic(ctx.typeBasic()));
}
```

b) How does the assignment check, whether the assigned value has the correct type to be stored in the variable on the left hand side?

```
public boolean isSame(MJType t) {

    if (t.getType() == this.getType()) {
        if (t.getType() == TypeEnum.CLASS) {
            if (t.getName().equals(this.getName())) {
                return true;
            }
            return false;
        }
        if (t.getType() == TypeEnum.ARRAY) {
            if (this.getBaseType().isSame(t.getBaseType())) {
                return true;
            }
            return false;
        }
        return true;
    }
    return false;
}
```

It checks if it is the same type, but comparing the type of the current class, and the class which is passed down in the parameter of the function.

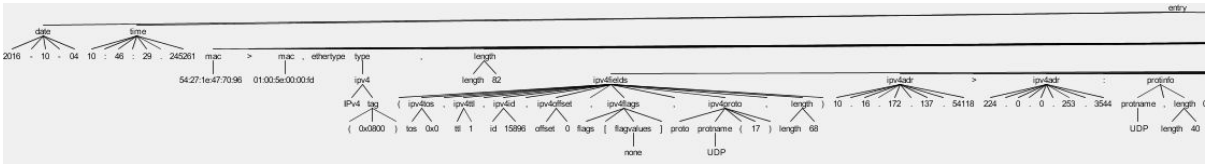
Part 2, network analyser

a) Download the file network.pdf from CN. It contains the description of a grammar for network traffic. Read the file carefully.

b) Write the grammar in ANTLR.

The grammar file can be found in the attached file *network_part2.g4*

c) Try out the parser on the files from last week, and on files you have created yourself. Add screendumps of the generated parse tree.



d) Take a break.

We had a wonderful break :) We went deer hunting, it was very fun.

The following assignments(e & f), can be found in the zipped eclipse project.

e) Write the necessary IR classes to store the information

The written IR classes is located in our eclipse project under “irclasses”.

f) Write a pretty printer for your IR. Use the visitor approach seen in the class and in the MiniJava compiler.

The pretty printer is located in our eclipse project under “ir”.

Rectores tenebrarum
Tobias(S144826), Kristofer(s153646), Morten(s144827), Magnus(S153086),
Alexander(s153391) og Mohammad(s154088)