Before you start!

In this exercise you will use Xtext for developing a Domain Specific Language. Xtext is *not* installed in the classroom-environment. Start Eclipse and install Xtext from the Eclipse Marketplace.
This Eclipse will be used to develop the DSL (DSL Development Installation). After the development, the DSL will be exported as a plugin. Later in this exercise we will start a second instance of Eclipse (DSL Modeling Installation). Thus you will have two roles: the DSL-developer and the DSL-User. Figure 1 gives this basic workflow.

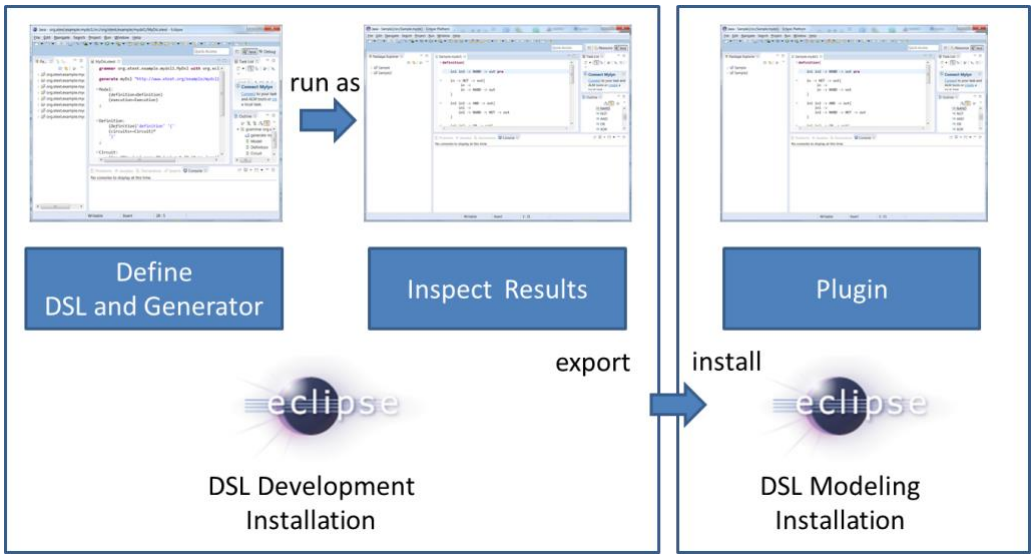

Figure 1: Basic Workflow

Important, if you are working at the University Environment:

- Overall Xtext seems to interact rather sensitive with other plugins and environment.
- Put the eclipse-Installations in the folder c:/java.
- Create your workspaces on the local C-drive – not on the P-drive, as Xtext cannot handle network-folders.
- Be aware, that this means, all your work will be erased if you shut down – so zip it and save it on the P-drive at the end.

1. Target of this exercise is to investigate the process of model driven development using a DSL and the Eclipse Modeling Tools.

   Please use a new, empty workspace for these two exercises!

   Run through the Xtext tutorial (see e-learning-system) to create a simple DSL for defining entities. Run through the complete tutorial including code-generation.

2. In the runtime-eclipsextext exchange the content of the file Sample.mydsl with the content from myCompany.mydsl in the Moodle-course (sample template for exercise 2), remove the enclosing package for the moment (you have to add this in task 2d again) and extend exercise 1 with the following features:

   a) The created class-files from exercise make no difference between attributes that are a list and attributes that are just single objects. Extend the Xtend file in a way, that there is different code generated: for single objects just use the corresponding data-type, for a list of objects use an **ArrayList**.

   Example. The DSL-code ...

```
entity Party {
        property name: String
        property address: Address[]
        }
```

   ... should lead to the following java-class ...

```
import java.util.ArrayList;

public class Party {

        private String name;

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        private ArrayList<Address> address;

        public ArrayList<Address> getAddress() {
                return address;
        }

        public void setAddress(ArrayList<Address> address) {
```

```
            this.address = address;
        }

}
```

Use the conditional statement **«IF condition»...«ELSE»...«ENDIF». property.many** evaluates to true if a property was defined with []. You can import the ArrayList statically.

b) Include the generation time in the javadoc-header of the classfile. To do that you have to write an extension in the xtend file:

```
def String timestamp() {
        new java.util.Date().toString();
}
```

After defining that extension it is possible to use that method **timestamp()** and modify the comment in the following way:

```
...
/*
 * Generated with Xtend
 * Generated at «timestamp()»
 */
...
```

c) After creating the files you see, that there is an error in the class **Clerk**. It uses the class **Date** for the attribute **birthdate**, but it should use **java.util.Date**. Write another extension in the xtend file that calculates the correct type for an attribute. If the type is **Date** it should return **java.util.Date**, otherwise the given type.

Advantage of the given solution (instead of including the correct import-statement): template keeps unchanged, easily extendable for other types.

d) Extend the DSL MyDsl.xtext in a way, that you can define a package for the class. One single package is sufficient; there is no hierarchy of packages necessary. There are two possibilities:

- Add a surrounding package that includes many types
- Add a possibility to define a package for entities.

Additionally extend the xtend file so that it creates the package-statement in the generated files.