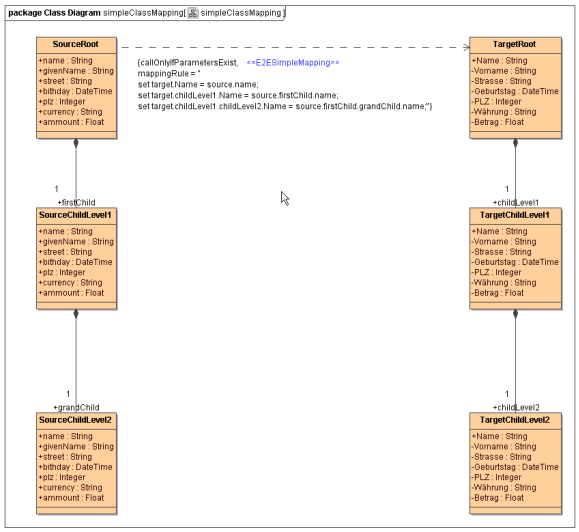
Simple Mapping of Classes

Example File (Builder projectAdvanced Modeling/Mapping):

<your example path>\Advanced Modeling\Mapping\uml\mappingSimple.xml

For mappings between complex type objects it may is easier to draw the mappings only on class level and define the mapping between the attributes by set statements within the mapping rule.



Within the mapping rule it is possible to use the action script language. To refer source and target objects attributes, use the keyword source or target within the action script (see example in class diagram **simple ClassMapping** above).

- Associations with multiplicity = 1 can directly be referenced, following the structure, delimited by a dot, e.g.

```
set target.childLevel1.childLevel2.Name = source.firstChild.
grandChild.name;
```

- Associations with multiplicity > 1 cannot directly be referenced.The xUML Compiler will throw an error.

The necessary elements on the target side will be created automatically.

For further examples for using the simple mapping action <<Mapping>> consult the simple Mapping example.

Caveat

In many cases it makes no sense to apply an operation to optional input values if they are NULL. Therefore, if an operation has input parameters only that are optional at the source class, the operation is applied only, if at least one of the input values is not NULL. This is technically implemented by generating an implicit guard. This behavior can be overridden by setting the tagged value **callOnlyIfParametersExist** on the <<E2ESimpleMapping>> stereotype to false.

Per default, **callOnlyIfParametersExist** is true, so defined **mapping rules** are only applied to existing parameters. The compiler will generate guard statements to **all** variables used in the action script in this case.

You need to be aware of this behavior when writing your mapping script. If you use nested if clauses that check for the existence of source parameters like e.g.

On this Page:

Related Pages:

```
set target.field =
        if          source.fieldA.exists()         and
                  source.fieldA.normalizeSpaces().stringLength() > 0
        then source.fieldA.normalizeSpaces()
        else if   source.fieldB.exists()         and
                     source.fieldB.normalizeSpaces().stringLength() > 0
             then source.fieldB.normalizeSpaces()
[...]
```

this will result in the mapping only being performed if **all** source fields are set. Also, guard statements will be applied to local variables used the mapping script.

You can override this behavior setting **callOnlyIfParametersExist** to **false** on the mapping relation. Then, no guard statements will be generated but you need to take care for non-existing values by yourself.