MvK Frontend: User Manual

1. Introduction

This document describes the explicitly modelled user interface for the Modelverse. In its current state, the frontend provides a minimal interface to the Modelverse, allowing models to be created, updated, and deleted. Model elements are visualized using a default concrete syntax.

2. Files

Three files are central to the frontend:

- **frontend.xml**: contains the model of the frontend, described in SCCDXML. It can be compiled to Python, using the SCCD compiler.
- **frontend.py**: is the compiled frontend code.
- **runner.py**: a wrapper to actually start the application.

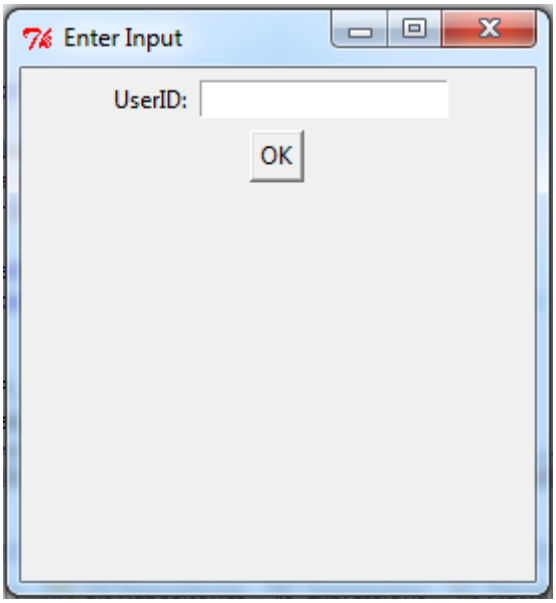
3. Compilation

To compile the frontend, run **python python_sccd_compiler/sccdc.py frontend.xml -o frontend.py -p gameloop**

4. Startup

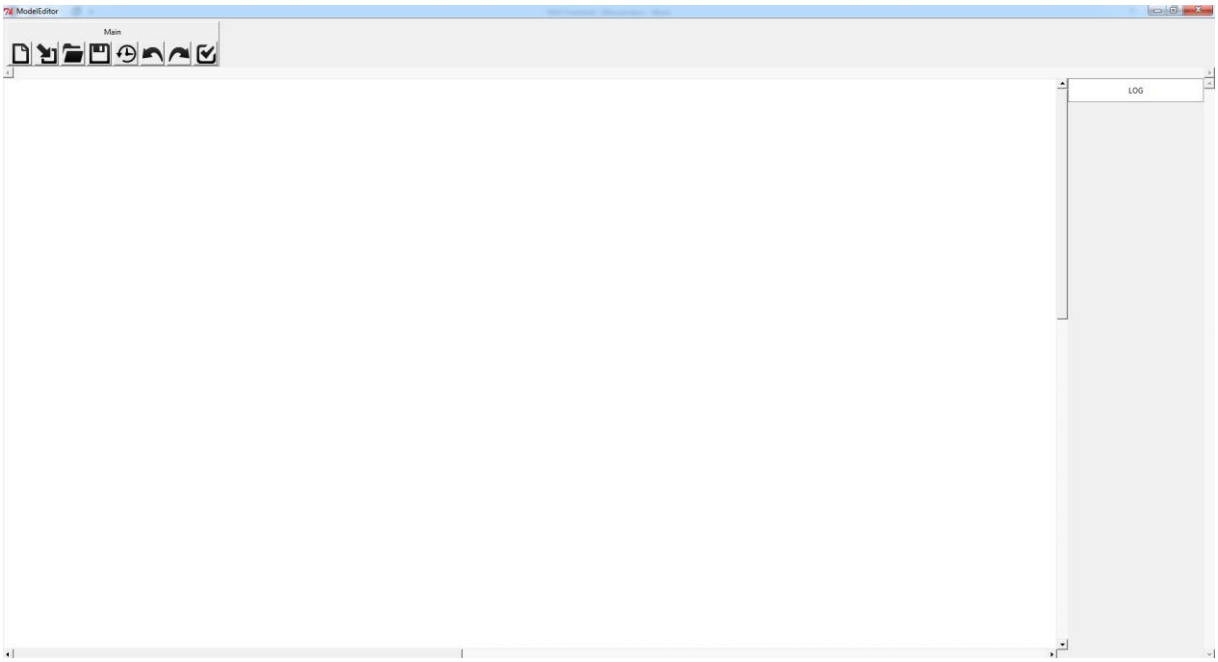
By default, the frontend connects to an MvK server running on localhost, on port 8000. To run a server, go to the **SCCDclientserver** folder, and execute the following command: **python simple_http_server 8000**.

To run the frontend, run **python runner.py**.



The first screen you see will ask for a user id. This is used when multiple clients are connecting to the same server, which is usually not the case. Just enter 0 here, and press OK.

5. Main Window

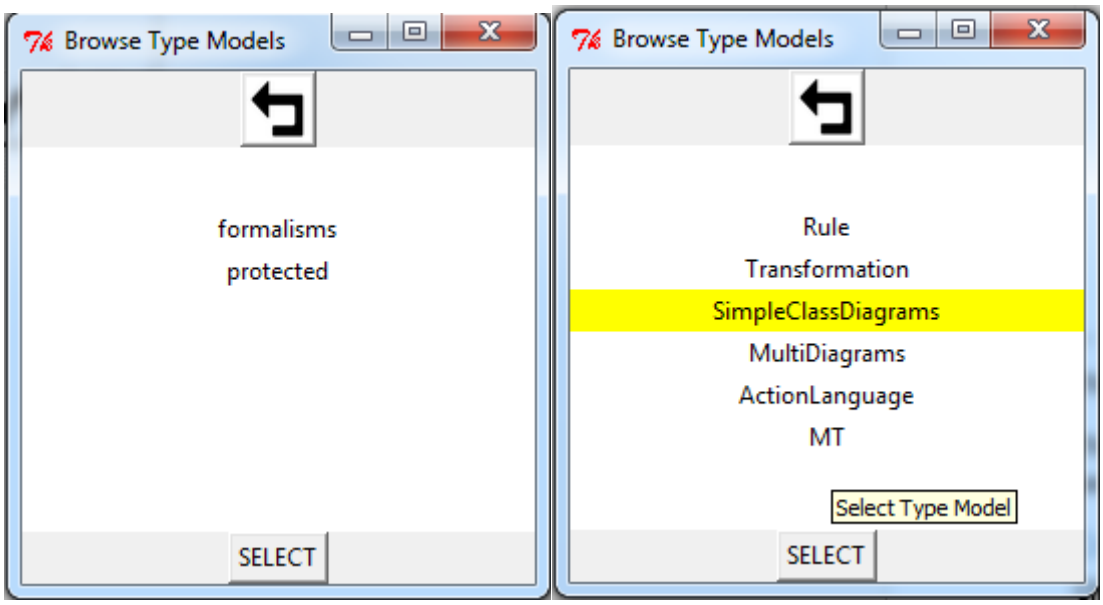


The main window is split into three parts: a **toolbar area**, a **log area**, and a **canvas**.

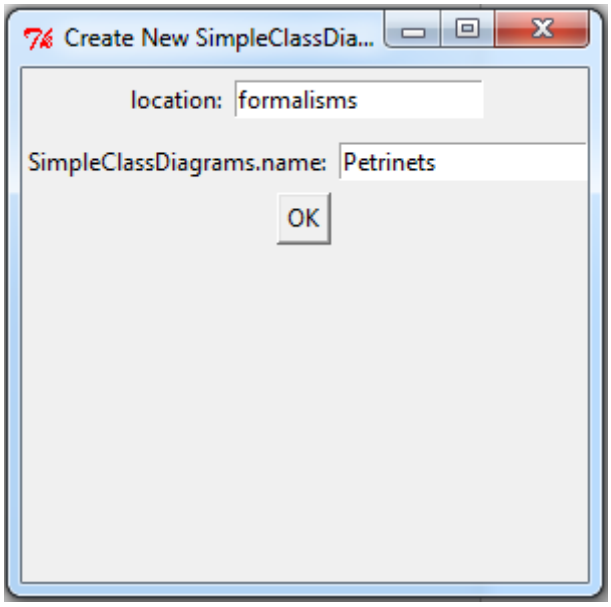
The **Main Toolbar** is loaded by default, and contains 8 buttons. Their function is explained in the next sections.

5.1. Create a Model

Clicking this button opens a type model browser. It allows you to browse through packages (with a double-click), and select a type model by clicking on it once, and the pressing the SELECT button.



Once a type model is selected, the attributes for the newly created model need to be filled in. In the case of SimpleClassDiagrams, this is **location** (where to create the model) and **SimpleClassDiagrams.name** (the name of the model).



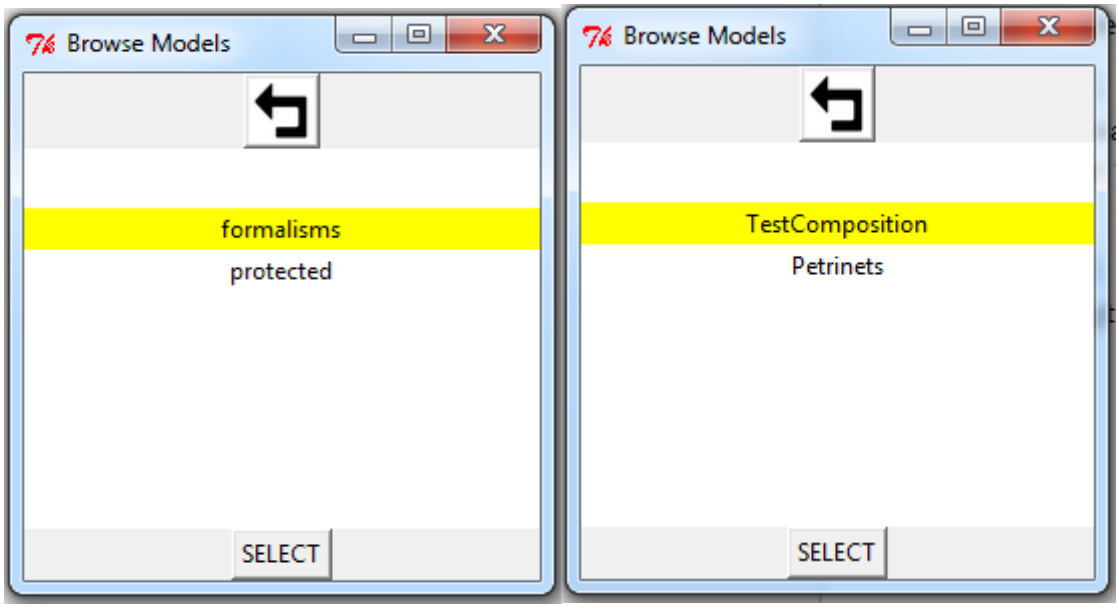
Once the details are entered, click **OK**. The formalism toolbar will be loaded.

5.2. Load a Formalism

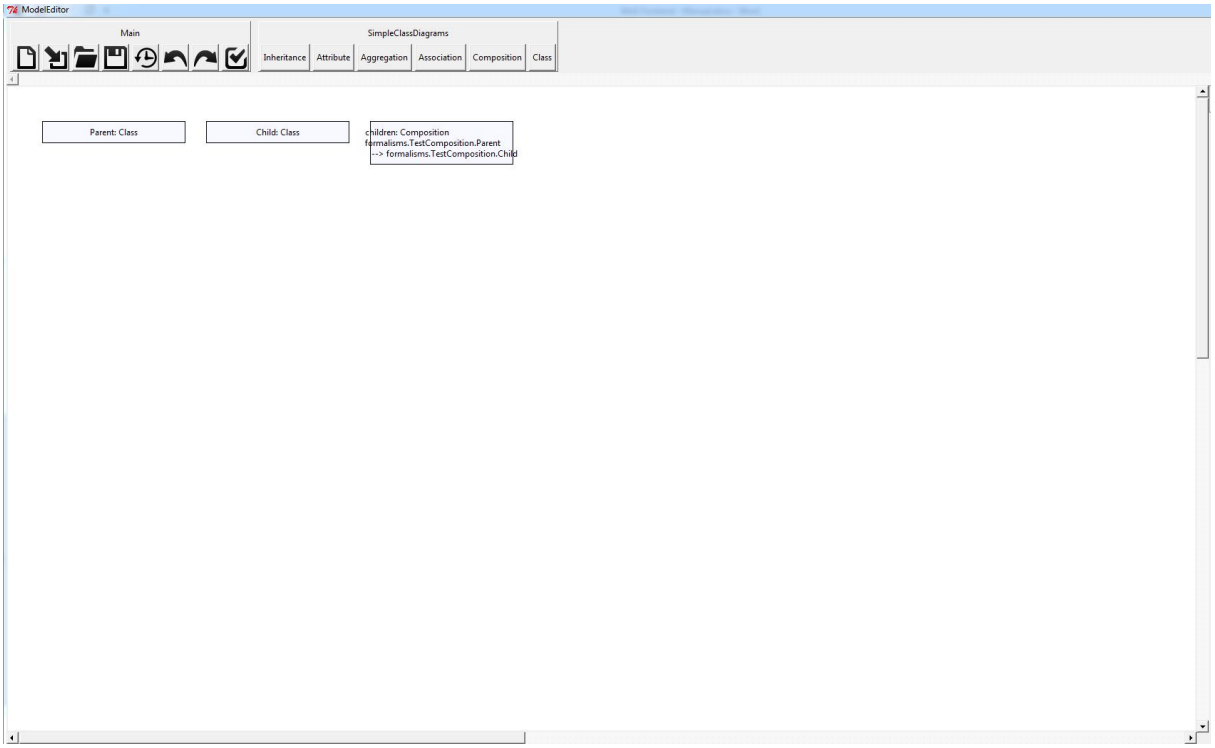
Additional formalisms can be loaded by clicking this button. Again, the type model browser will open, allowing you to browse the modelverse for a type model to load. An additional formalism toolbar will open.

5.3. Open a Model

Clicking this button opens a browse window, allowing you to browse to a model to open.



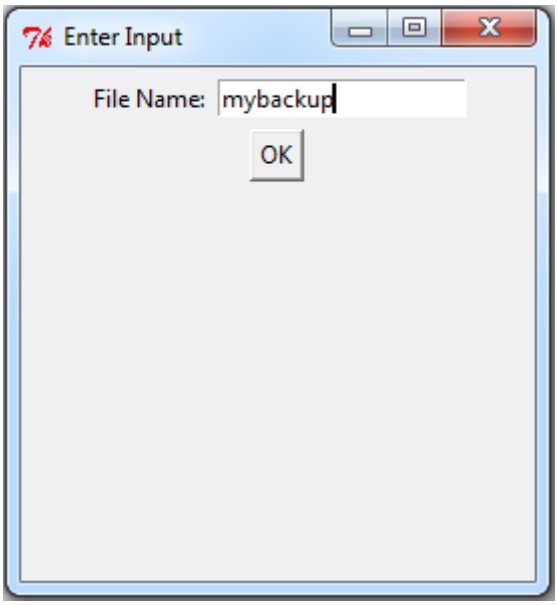
As anything in the Modelverse (except packages) are models, this example will open the TestComposition type model, as a SimpleClassDiagrams model.



Elements are placed in rows, in no particular order. No positions are saved when editing a model, as concrete visual syntax has not been implemented yet.

5.4. Save the Modelverse

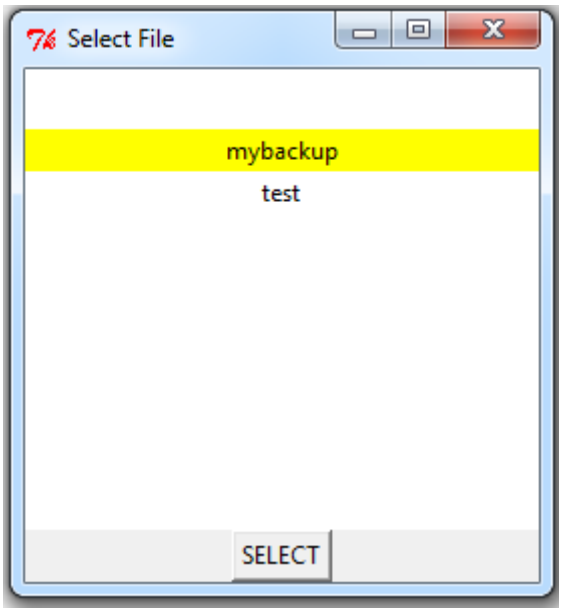
This allows you to backup the entire Modelverse to a file.



The file is stored on the server.

5.5. Restore the Modelverse

This allows you to restore the state of the Modelverse from a previously made backup.



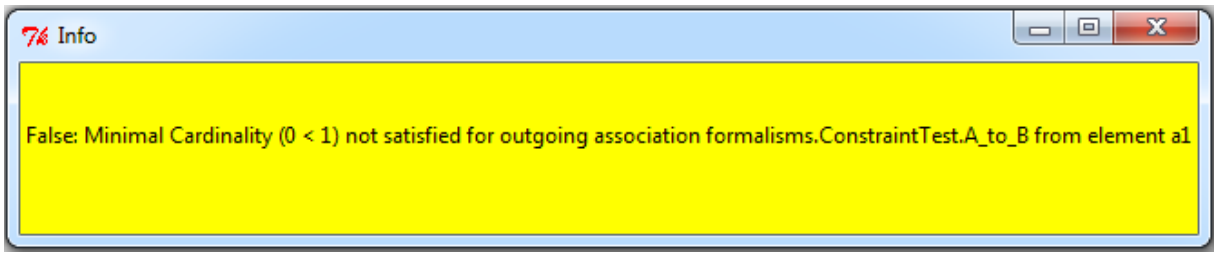
Any changes done after the backup was made will be lost.

5.6. Undo/Redo

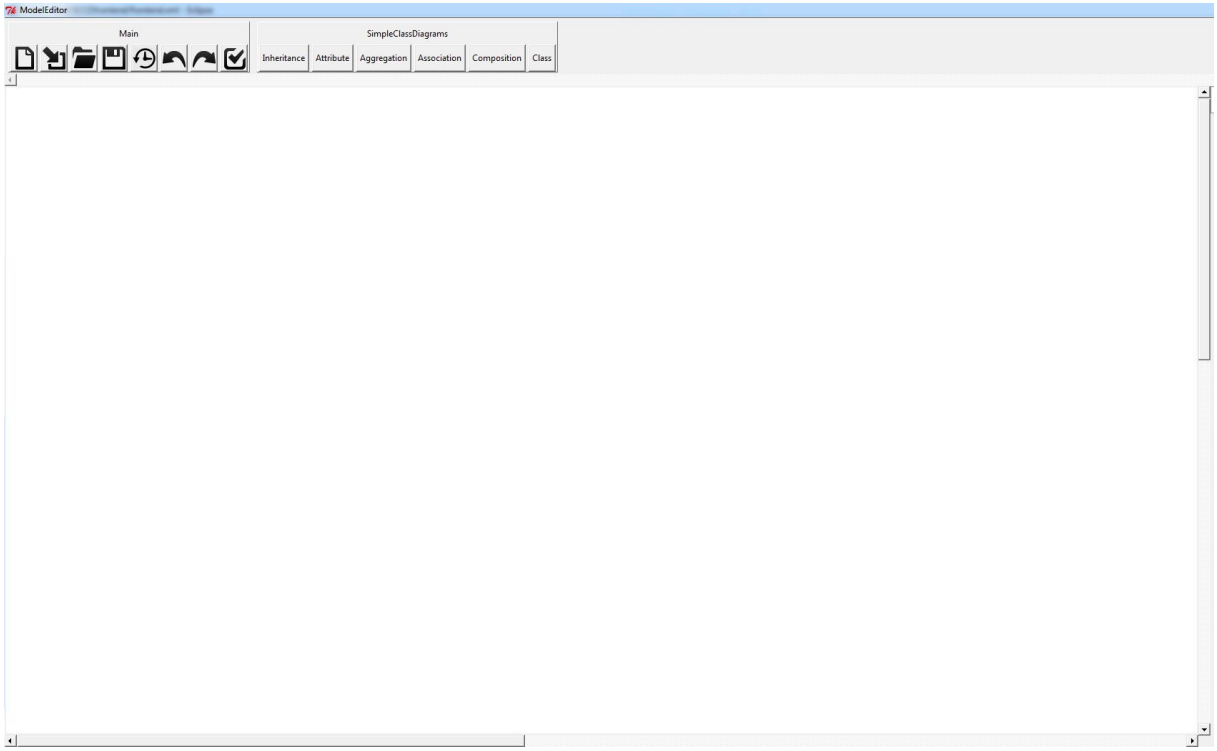
To be implemented.

5.7. Verify

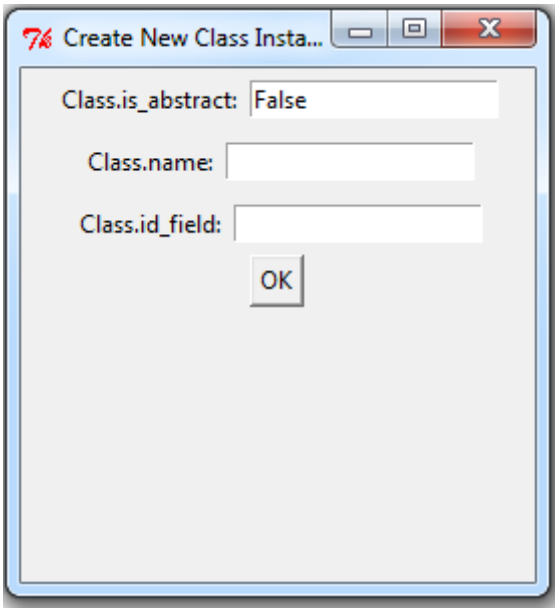
This button verifies whether the current model conforms to its type model. It opens a popup window with the result:



5.8. Creating Instances



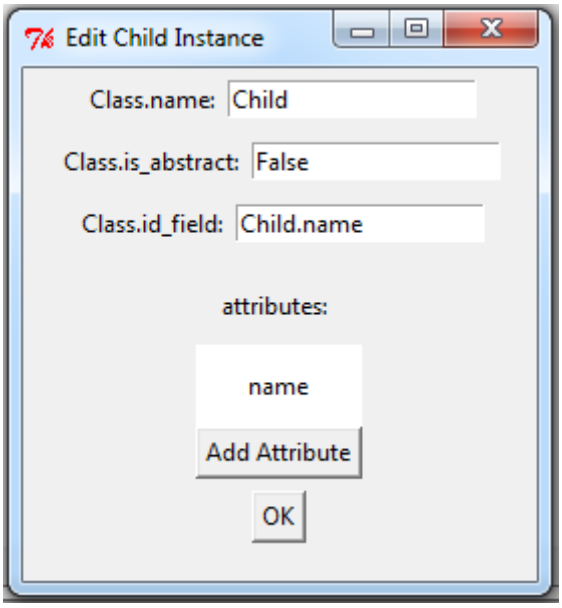
Clicking a button of the formalism toolbar will select a type to create. Right-clicking anywhere in the canvas will instantiate the type, opening a window where its attributes need to be given a value.



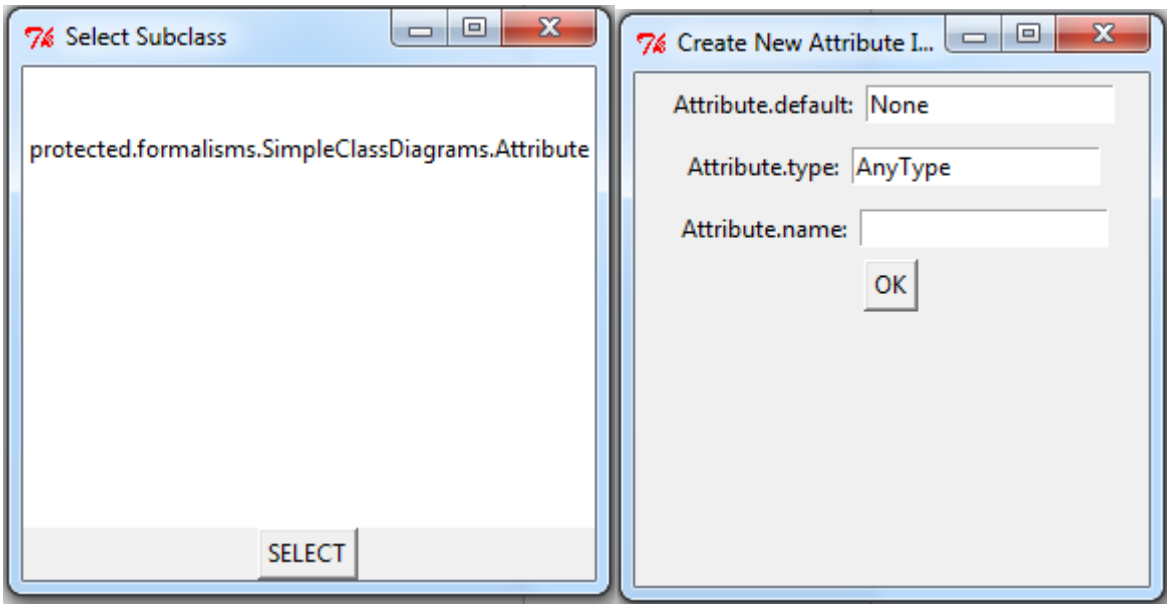
Default values are entered automatically.

5.9. Editing Instances

An instance is edited by middle-clicking or control-left-clicking its icon on the canvas. An edit window will pop up.



Here, attribute values can be changed. For each outgoing composition, children can be added to the instance, as compositions signify a parent-child relationship (or "insideness"). In this case, 'Child' is an instance of SimpleClassDiagrams.Class, which has a composition with SimpleClassDiagram.Attribute. Clicking the 'Add Attribute' button will open a new window, allowing you to choose a concrete subclass to instantiate. In this case, only Attribute can be chosen.



A window will then pop up, allowing to instantiate the Attribute. Once you click 'OK', the Attribute is instantiated, as well as the composition link connecting the Class with the Attribute. Note that in this case, the composition link is automatically instantiated by the physical mapper. This is only the case because SimpleClassDiagrams.Attribute is mapped onto the physical Attribute class. If this is not the case, and the type model has a 'normal' composition between two clabjects, an extra window will pop up, allowing you to fill in the attributes of the composition link.

5.10. Deleting Instances

A selected instance is deleted by pressing the ‘Delete’ key.