Invasive Software Composition with Reuseware

Reuseware is a framework that can be used to define and use invasive software composition systems. In this task we will use and extend a composition system that can be used to compose UML state machine diagrams. To create and view UML diagrams we will use the TOPCASED open-source UML editor. Both Reuseware and TOPCASED can be installed as plugins in Eclipse.

Prerequisites:

Download:

- An installation of *Eclipse Modeling Tools Package* (Ganymede)
- EPF RichText feature (required by TOPCASED
  - *Windows:* http://www.eclipse.org/downloads/download.php?file=/technology/epf/composer/release/epfc-richtext-1.5.0.2-win32.zip
  - *Linux/Mac*: http://www.eclipse.org/downloads/download.php?file=/technology/epf/composer/release/epfc-richtext-1.5.0.2-linux.zip

Via the Eclipse Update Manager:

- TOPCASED: **http://topcased-mm.gforge.enseeiht.fr/release/update-site3.4**
  - You only need: *Topcased Toolkit > Topcased UML Editor*
- Reuseware: **http://reuseware.org/update**
  - Install everything from the Reuseware Composition Framework category

Import the extracted exercise project (*cbse.exercise*) into your Eclipse Workspace. Ensure that the folder "store" is declared as fragment store. A folder can be declared as fragment store by clicking the blue Reuseware button in the toolbar.

Notes about using TOPCASED with Reuseware:

- New diagrams can be created via *New > Other... > Topcased > Topcased Diagrams > UML Diagram*. Select in the *Template* menu which kind of UML diagram you want to create.
- If you want to use primitive types (String, Integer, Boolean) for attributes in the UML Editor, you need to right-click the model in the outline view of the editor and select *Import Primitive Types*.
- Unfortunately, the composition of layout of TOPCASED diagrams is buggy. Sometimes, the result of a composition contains red error boxes although the composition was correct. The correct model can always be explored in the outline view. Delete all elements from the diagram and drag and drop them from the outline view into the diagram again to obtain an improved view.

Task 1: Using a composition system

Familiarize yourself with Reuseware. Information can be found at **http://reuseware.org**. (Note: A good starting point is *http://reuseware.org/index.php/starter_guide.*)
Additionally you might look at the *TicketOrder* example that is included in the material.

1a)
In the provided composition system for UML state machines, *composite states* can be reused. Instead of define a new composite state, one can define a normal state and name it using the pattern `Hook_*` (e.g,, *Hook_TicketOrder*). Such a state marks a hook that can be replaced by a composite state during composition.
Define a new state machine for a pizza shop (use your imagination) that includes two hooks: one for a *pizza order composite state* and one for a *pizza delivery composite state*. Inspect your state machine fragment in the fragment repository view. The defined hooks should be displayed when expanding the fragment.
Hint: Inspect the *TicketSaleStateMachine.uml*

1b)
Define a composition program that composes your state machine with two appropriate composite states from the library (“CBSE/library/StateMachines”).

Hint: Inspect the *TicketSaleStateMachineComposition.fcdi*

Task 2: Defining a composition system

2a)
Introduce a hook concept for classes into the composition system by extending the “/CBSE/library/compositionSystem/UMLCompositionSystem.rex”. Use the convention that classes with the naming pattern `Hook_*` identify hooks. All other classes should be accessible as prototypes. Observe how the composition interface of a class diagram of the pizza shop (“CBSE/exercise/ClassDiagrams/PizzaShop.uml”) changes.

Hint: Learn more about how to define composition systems through reuse extensions at *http://reuseware.org/index.php/reuse_extension_language.*

2b)
In the extended composition system, the fragment “CBSE/exercise/ClassDiagrams/PizzaShop.uml” defines a hook for the superclass for all kinds of pizzas (`Pizza_Hook`). Model a new UML fragment with a single class called `Pizza` with some properties that should be common for all pizzas. Define a composition program that binds your `Pizza` to `Pizza_Hook`.