# 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* (www.eclipse.org/downloads)

Via the Eclipse Update Manager:

- Reuseware: http://reuseware.org/update
  - Install everything from the **Reuseware** and the **Reuseware Applications** category

To get access to the exercise material, select in Eclipse:

• New > Project... > Reuseware > Reuseware Examples > CBSE Exercise (UML)

## 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*.
- To see that changes in a composed model/diagram in the Topcased editor, you need to close and re-open the diagram.

## Task 1: Using a composition system

Familiarize yourself with Reuseware. Information can be found at **http://reuseware.org**. 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").

<u>Hint</u>: Inspect the *TicketSaleStateMachineComposition.ucldi* 

## 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.

<u>Hint</u>: Learn more about how to define composition systems through reuse extensions at: http://reuseware.org/index.php/Reuseware Documentation

## **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.