Using Alf for classifier behaviors

An *active Class* is one that has a *classifier behavior* associated with it, which defines the asynchronous behavior of instances of the class. When such a classifier behavior is an Activity or an Opaque Behavior, you can specify it using Alf code.

If an active Class is instantiated using an Alf instance creation expression, then the classifier behavior for the Class is explicitly started on the newly created instance *after* the completion of initialization of the instance. However, the default Cameo Simulation Toolkit option is to **Aut ostart Active Objects**, in which case the classifier behavior for an active object will be automatically started as soon as the object is created. While CST will ensure that any Properties with default values are properly initialized before the classifier behavior is started, if you have initialization code in a constructor for an active Class, then this will *not* execute before the classifier behavior starts.

Therefore, if you intend to instantiate an active Class from Alf code using a constructor that carries out initialization on which the classifier behavior depends, make sure that the classifier behavior waits for the constructor execution to complete. You can do this by having the classifier behavior wait for a special *Start* Signal before carrying out any other behavior, and having the constructor invoke *this.Start()* when it is done with its initialization behavior. (If the classifier behavior is an activity, then use an Accept Event Action to wait for the Start signal. If the classifier behavior is a State Machine, have the State Machine transition from its initial Pseudostate to a *Waiting* State that is exited by a Transition triggered by the *Start* Signal.)

Alternatively, you can turn off the **Autostart Active Objects** option:

1. Select **Options > Environment** from the main menu.
2. Choose **Simulation** (the last option group on the left).
3. Uncheck the **Autostart Active Objects** option under **Simulation Frameworks** (so it is *false*).

However, doing this will turn off the option globally for all projects, which may cause some other simulation projects to not work. You can turn this option off for just a specific project if you use a simulation configuration, by setting the **Autostart Active Objects** property to *false* in your configuration.

To create a classifier behavior using Alf

1. Open the Specification window for the Class and check the **Is Active** property.

   If the **Is Active** property does not appear in the Specification window, select **All** from the **Properties** drop-down at the top right of the window.

2. Close the Specification window.
3. Select the Class and open the Alf editor window (select **Windows > Alf**), if it isn't already open.
4. If the Class does not have a classifier behavior yet, then no code will appear in the Alf editor window, but the **Create** button will be active. Press **Create** to create a new classifier behavior for the Class.
5. Enter the Alf code for the classifier behavior and press **Save** to compile and save the code.

To edit an existing Alf body for a classifier behavior

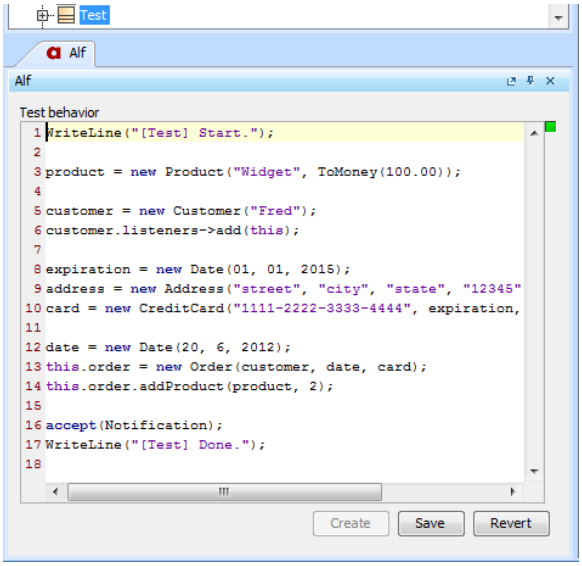
- Select the Class in the Model Browser or on a Class diagram and open the Alf editor window (select **Windows > Alf**), if it isn't already open. The existing Alf code will appear in the window.

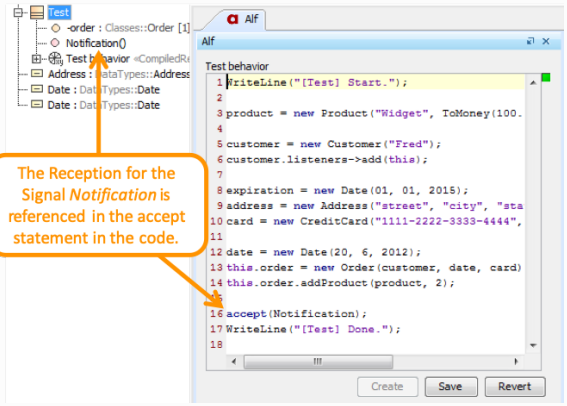
  The Alf code for an Operation is actually stored as the Alf body of the method Behavior of the Operation (which is an Activity or Opaque Behavior, as created above). It is therefore possible to directly the edit the code on the method Behavior (as described in Using Alf to define Behaviors), but it is usually easier to act directly on the Operation.

Related pages

- The Alf editor
- The Alf compiler



While a classifier behavior may be edited like any other Behavior with an Alf body, as an asynchronous Behavior it is allowed to have Alf *accept* statements, which can handle Signals sent to instances of its active Class and are not allowed in synchronously called Behaviors. However, Alf requires that, for any Signal appearing in an *accept* statement, the containing Class must have a Reception (as shown below).



Referencing a Reception in an Alf accept statement