Listening to related elements in hierarchy events

There is a possibility to listen for changes of the elements, which are somehow related with the given element. For example, we want to be notified, when the element's owner name is changed. The *com.nomagic.uml2.ext.jmi.smartlistener.SmartPropertyChangeListener* class is dedicated for the situations like this. The main idea of the solution is to provide the *com.nomagic.uml2.ext.jmi.smartlistener.SmartListenerConfig*, which will provide the chain of the property names to listen to.

The *SmartListenerConfig* class provides static methods for the default configuration of property chains. This is useful, when listening for common property change events.

Use the *SmartPropertyChangeListener* class *createSmartPropertyListener(...)* method to create and register such listener for the particular element, which will be notified with events provided by the *SmartListenerConfig*.

> ⚠ If the listener is not needed anymore, unregister it using the *removePropertyChangeListener(...)* method.

Example #1. Listen to the element owner's name

```
                  Element element = ...; // some element

        SmartListenerConfig config = new SmartListenerConfig();
        config.listenTo(PropertyNames.NAME); // listen to the element's
name
        config.listenToNested(PropertyNames.OWNER).listenTo(PropertyNames.
NAME); //listen to the element owner's name

        // create the listener
        SmartPropertyChangeListener.createSmartPropertyListener(element,
new PropertyChangeListener()
        {
            public void propertyChange(PropertyChangeEvent evt)
            {
               // the change event comes here
            }
        }, config);
```

Example #2. Listen recursively to the owner's name and the *"Is Abstract"* property

```
            Element element = ...; // some element

        SmartListenerConfig config = new SmartListenerConfig();
        config.listenTo(PropertyNames.NAME);
        config.listenTo(PropertyNames.IS_ABSTRACT);
        config.listenTo(PropertyNames.OWNER, config);

        SmartPropertyChangeListener.createSmartPropertyListener(element,
new PropertyChangeListener()
        {
            public void propertyChange(PropertyChangeEvent evt)
            {
                // the change event comes here
            }
        }, config);
```