Creating Lightweight instances

1. **How to create CrossCuttingConcern**

To create a package like this “**Package** **Persistence;**” you have to create this following source code:

1. **How to create Aspect**

To create an aspect like this “**public aspect connectionComposition{}**” you have to create this following source code:

If your aspect is inserted in a package you have to bound it the corresponding package, like this:

If you want to add some of its properties, you have to create a “*TaggedValue”* as you can see in the source code below:

Notice that when you create a “*TaggedValue”* you have to bound it to the corresponding element. In this case you have to bound the “*isPrivileged”* Tag to the “*ClassUnit myAspect”*.

1. **How to create Pointcut with one Joinpoint**

To create a pointcut like this “**public** **pointcut** **openConnection():** **execution** **(main());**”you have to create this following source code:

Notice that you have to create first an instance of MemberUnit element, and then apply the stereotype. In this kind of pointcut with just one jointpoint you have to set the name in the memberUnit that represents your joinpoint.

The second observation is that to set the “*operation”* of the pointcut you have to create a “*TaggedValue”*, apply the “*TagDefinition* *operation”* and then set the value. In order to complete the creation of this instance you have to bound the “*TaggedValue”* to the “*MemberUnit”* created, see the last source code line.

1. **How to create Pointcut with more than one Joinpoint**

To create a pointcut like this:

“**public pointcut openConnection(): execution (main()) || call (main());**” You have to create this following source code:

First you have to create a “*MemberUnit”* and then apply the “*compositePointCutUnit”* Stereotype. You have to set the property “*compositionType”* too, so you just have to create a “*TaggedValue*” and then you have to bound this “*TaggedValue”* to the “*MemberUnit”* stereotyped with “*CompositePointCutUnit”*.

Remember to bound the “*TaggedValue”* to the respective “*MemberUnit”*.

Notice that you have to set the name ONLY in the “*MemberUnit”* stereotyped with “*CompositePointCutUnit”*.

The next step is to create the joinpoints and for this you have to create a “*MemberUnit”* for each joinpoint specified on the pointcut. In this example we have two joinpoints, so the following source code represents these two joinpoints.

The first “*MemberUnit”* is stereotyped with “*callPointCutUnit”* and it has two “*TaggedValues”* bounded that represents the properties “*Operation*” and “*Composite*”. The second “*MemberUnit”* is stereotyped with “*callPointCutUnit”* and it has two “*TaggedValues”* bounded that represents the properties “*Operation*” and “*Composite*”, like in the first one.

1. **How to create Advice**

To create an advice like this: “**around() : move() {// Some JAVA souce code}**” You have to create this following source code:

The main properties that you have to set while creating an Advice are “*pointCut”* (which specifies the pointcut that it belongs) and the “*adviceExecutionType”* (that specifies the behavior of the advice).

You can set this properties by creating “*TaggedValues”* and applying the “*TagDefinitions” “pointCut”* and “*adviceExecutionType”* and then bound them into the “*ControlElement”* that represents the Advice.

1. **How to create Inter-type declaration**

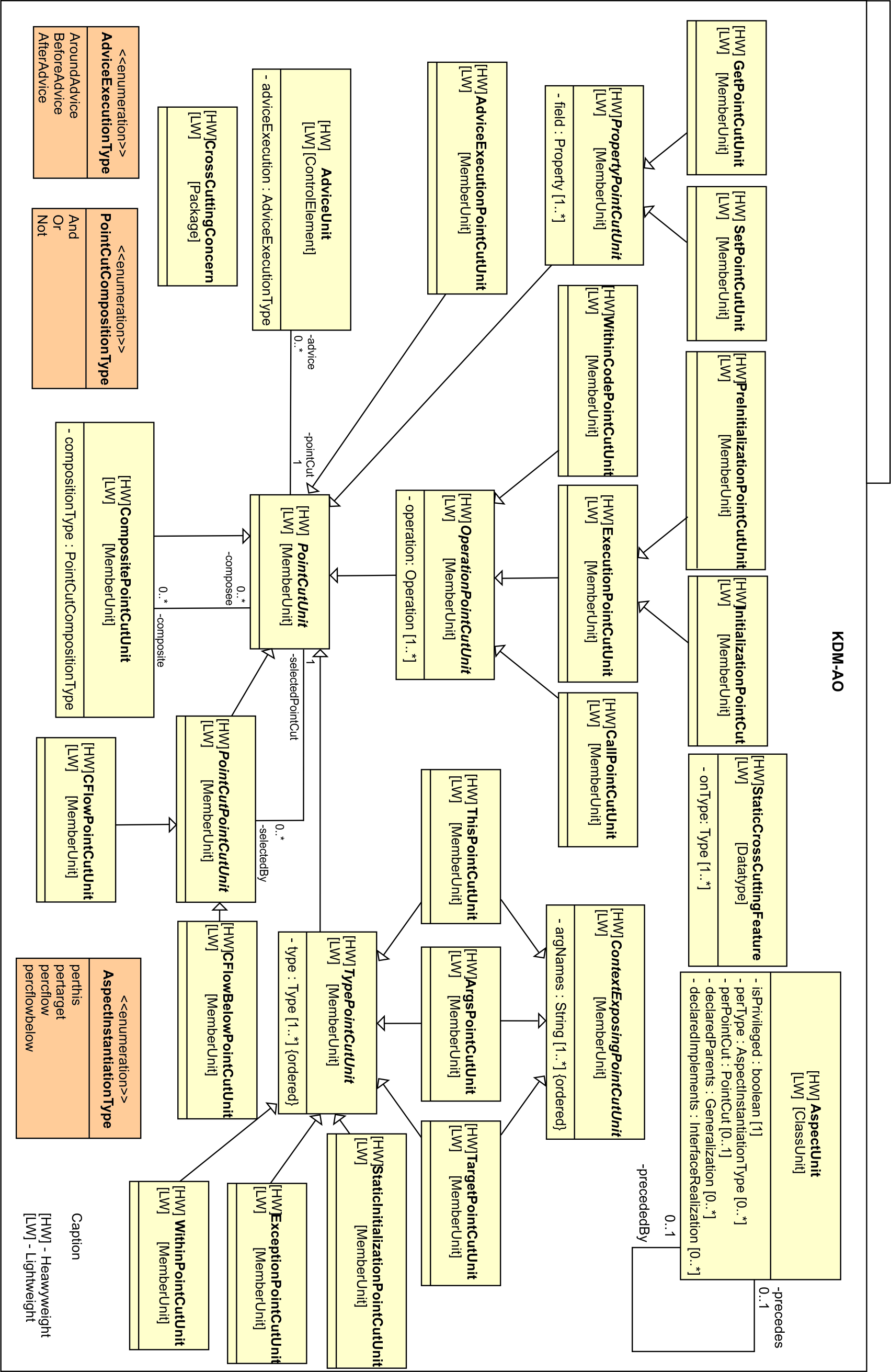
To create an advice like this:

“**public PersistentRoot.name;**

**public void PersistentRoot.move();**” You have to create this following source code:

Notice that you have to create a “*Datatype”* element and apply the stereotype “staticCrossCuttingFeature”. You can set the “*OnType*” property by creating a “*TaggedValue”*. Don’t forget to bound the TaggedValue to the Datatype element.

The attribute and the method have to be bounded to the “*Datatype”* too.



|  |  |  |
| --- | --- | --- |
| **AspectJ Profile Elements** | **KDM Elements** | **Stereotype** |
| Aspect | ClassUnit | aspectUnit |
| PointCut | MemberUnit | pointCutUnit |
| CompositePointCut | compositePointCutUnit |
| OperationPointCut | operationPointCutUnit |
| WithinCodePointCut | withinCodePointCutUnit |
| ExecutionPointCut | executionPointCutUnit |
| CallPointCut | callPointCutUnit |
| PreInitializationPointCut | preInitializationPointCutUnit |
| InitializationPointCut | initializationPointCutUnit |
| PropertyPointCut | propertyPointCutUnit |
| GetPointCut | getPointCutUnit |
| SetPointCut | setPointCutUnit |
| AdviceExecutionPointCut | adviceExecutionPointCutUnit |
| PointCutPointCut | pointCutPointCutUnit |
| CFlowPointCut | cFlowPointCutUnit |
| CFlowBelowPointCut | cFlowBelowPointCutUnit |
| TypePointCut | typePointCutUnit |
| WithinPointCut | withinPointCutUnit |
| ExceptionPointCut | exceptionPointCutUnit |
| StaticInitializationPointCut | staticInitializationPointCutUnit |
| TargetPointCut | targetPointCutUnit |
| ArgsPointCut | argsPointCutUnit |
| ThisPointCut | thisPointCutUnit |
| ContextExposingPointCuit | contextExposingPointCuitUnit |
| CrossCuttingConcern | Package | crossCuttingConcern |
| AdviceExecutionType | TaggedValue | setTag(Profiles.adviceExecutionType) |
| PointCutCompositionType | TaggedValue | setTag(Profiles.pointCutCompositionType) |
| AspectInstantiationType | TaggedValue | setTag(Profiles.aspectInstantiationType) |
| Intertype Declaration | Datatype | staticCrossCuttingFeature |
| Advice | ControlElement | adviceUnit |