Type of Visibility	Character
`public`	+
`private`	-
`protected`	#

Table 2: Visibility table showing VDM++ visibility definitions and the corresponding PlantUML notation.

The element is declared with a name and a type. The `ownedAttribute` keyword is also used when defining custom types.

3. An Operation definition. It is declared with a visibility and a return type. The `ownedOperation` keyword is also used when defining functions, which are similar but more restricted in their use.
4. An attribute definition where the used element is defined in or is another class, the result of which is an association.
5. A generalization definition. This is a result of a class inheriting from another class. This occurs in VDM++ by using the keywords `is subclass of` followed by class name after the name of the subclass is defined, in the class header.

It should also be noted that the translator will not directly be searching for the presented XML passage, extracting the relevant and generating PlantUML from that. Rather, the translator will be accessing them at a higher level of abstraction by interacting with the UML file through the framework established by the Overture UML connection.

PlantUML also has the ability to define the visibility of elements. A diagram of which is illustrated below.

4.5 Transformation Example

Using the XML2PlantUML table, as well as the Visibility table, it is possible to construct an abstract example that showcases how a VDM model will be transformed into PlantUML. The example model is a simplification of the AlarmPP model used in the Overture tutorial, modified to include inheritance [8]. The classes of this model is now presented.