

mapEqualNames

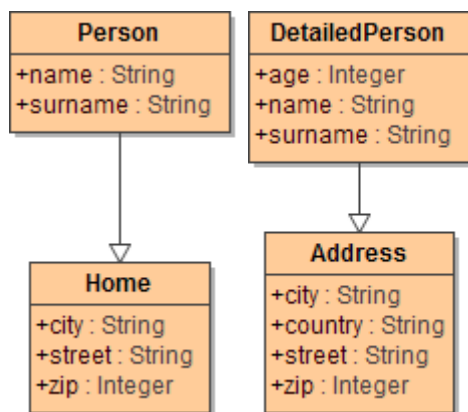
Syntax	<pre>set aTargetObject = anInputObject.mapEqualNames ([anotherInputObject] +); set aTargetObject = anInputObject.mapEqualNamesIfExists ([anotherInputObject] +); append mapEqualNames(anInputObject) to anArrayOfTargetObjects;</pre>	
Semantics	• mapEqualNames: The macro generates set statements for all equal named, equally typed attributes found in the target object and at least one of the input objects. This applies to all public attributes including all inherited attributes. • Added in Builder 6.0.0.9 mapEqualNamesIfExists: The macro works as mapEqualNames if the mapped source attribute exists. If it doesn't exist, the mapping does not take place. Thus, existing target attributes are not overwritten by NULL values of source attributes. Note, only the existence of the attribute values is checked. It is assumed that the input object exists. These macros do not work recursively and thus do not perform a deep copy of attributes of complex types.	
Substitutables	anInputObject, anotherInputObject	Can be any complex object having attributes of any type.
Examples	<pre>set person = person1.mapEqualNames(person2, person3); set person.address = mapEqualNames(address); set person.alternativeAddress = detailedPerson.address. mapEqualNames(); set person.address = mapEqualNames(detailedPerson.addresses [0]); append mapEqualNames(detailedPerson) to people;</pre>	

On this Page:

- [Detailed Examples](#)
- [Usage of mapEqualNames\(\) with append Statement](#)

Detailed Examples

Given are the two unrelated classes **Person** and **DetailedPerson**.



mapEqualNames()

<pre>anInputPerson { age: 45, name: "Jane", surname: "Doe", city: "Los Angeles", country: "USA", street: "22, Main Road", zip: 90077 }</pre>	The object anInputPerson is of type DetailedPerson.
<pre>set anOutputPerson = mapEqualNames (anInputPerson);</pre>	This statement assigns values to the matching attributes of object anOutputPerson which is of type Person: <pre>anOutputPerson { name: "Jane", surname: "Doe", city: "Los Angeles", street: "22, Main Road", zip: 90077 }</pre>

mapEqualNamesIfExists()	
<pre>aPerson { name: "John Ethan George", surname: "Smith", city: "New York", street: "65, Upper Eastside", zip: 0 }</pre>	The object aPerson of type Person already contains values. In our example, the zip code is unknown, so the attribute zip contains the value 0.
<pre>anotherInputPerson { age: 32, name: "John", surname: "Smith", city: "New York", country: "USA", street: NULL, zip: 12345 }</pre>	The object anotherInputPerson is of type DetailedPerson. In this example object, the attribute street is missing.
<pre>set aPerson = mapEqualNamesIfExists (anotherInputPerson);</pre>	This statement assigns values to the matching attributes of object aPerson: <pre>aPerson { name: "John", surname: "Smith", city: "New York", street: "65, Upper Eastside", zip: 12345 }</pre> The value of street remains unchanged, because in the source object it did not exist (was NULL).

Usage of mapEqualNames() with append Statement

You can use the mapEqualNames() macro along with the [append statement](#) to add a complex object with numerous attributes to an array of unrelated objects which needs only some of the information. The mapEqualNames() macro will create set statements for all equal named attributes found in the target object while the append statement will add the result to an array.

Given are the two unrelated classes **Person** and **DetailedPerson**.

Person	DetailedPerson
+name : String	+address : Address
+surname : String	+age : Integer
	+name : String
	+surname : String

<pre> aDetailedPerson { address: { city: "Los Angeles", country: "USA", street: "22, Main Road", zip: 90077 } age: 45, name: "Jane", surname: "Doe" } </pre>	aDetailedPerson is provided from an external source. For further processing, we are interested in name and surname only.
<pre> append mapEqualNames (aDetailedPerson) to people; </pre>	mapEqualNames(.)collects the data from aDetailedPerson and append adds it to the array people. people is an array of type Person and contains already the data of "John Smith" and "Erica Meyer". Result array: <pre> people [{name: "John", surname: "Smith";} {name: "Erica", surname: "Meyer";} {name: "Jane", surname: "Doe";}] </pre>