

Constructors

Example File (Builder project Basic Modeling/Data):



<your example path>\Basic Modeling\Data\uml\constructors.xml

On this Page:

- [Declaring Constructors](#)
 - [Example: Creating Objects](#)
- [Calling Constructors](#)
- [Calling Constructors of Super Classes](#)

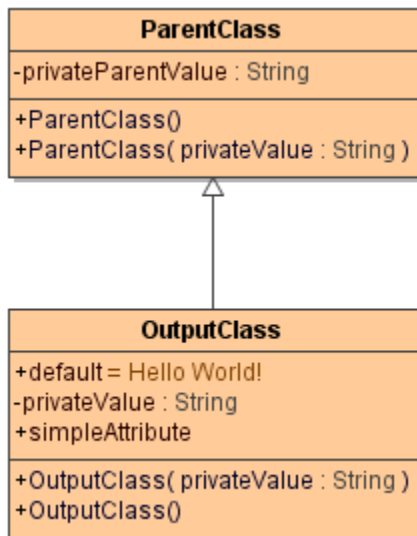
Declaring Constructors

A constructor "initializes" an object. Each time an object gets created, its constructor is called. The constructor then executes its duties like setting initial values or even processing complex operations.

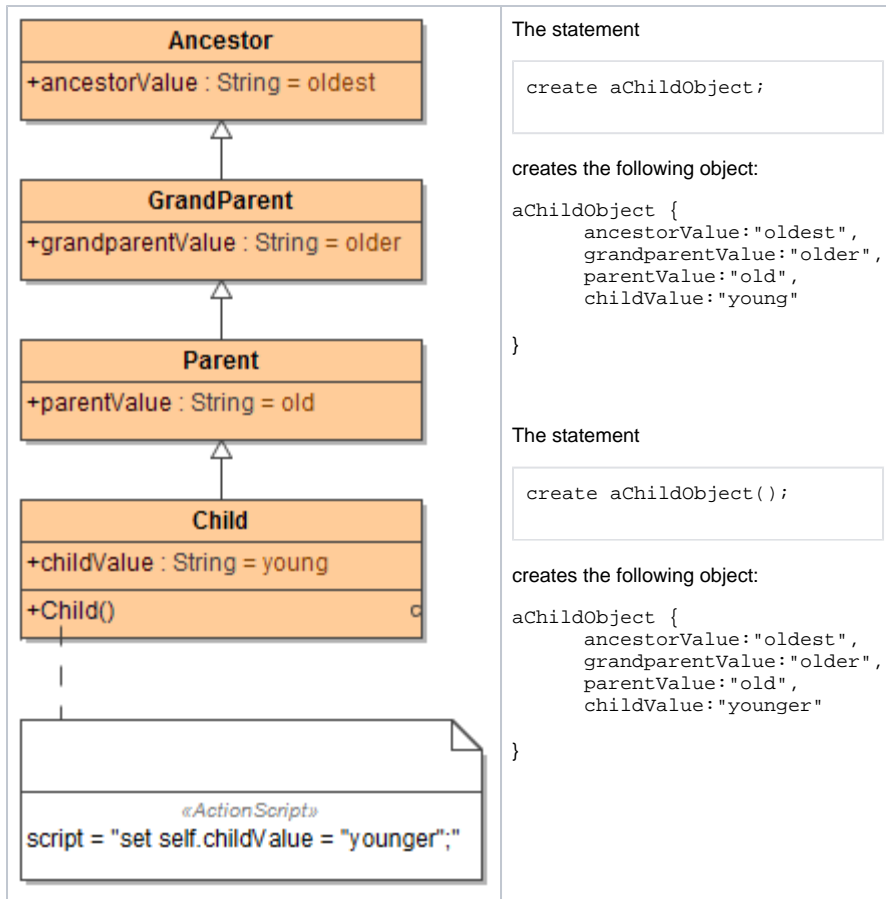
A `create` statement creates an object by invoking its constructor. Object fields are guaranteed to be initialized to their default values when the constructor is called. If no user-defined default constructor exists, an empty constructor will be implicitly created. The implicit default constructor does nothing. The default constructor is a constructor that can be called without parameters (it has no parameters or all its parameters have default values).

Each constructor initializes its part of the final class. Each super class's constructor is guaranteed to be executed first, before the constructor of the child class gets executed. By default, a default constructor of the parent is implicitly run. One can however influence this and replace the implicit call by an explicit call as described in [Calling Constructors of Super Classes](#).

User-defined constructors are operations having the same name as the owning class. For example, the classes **ParentClass** and **OutputClass** contain four explicitly defined constructors: **ParentClass()**, **ParentClass(privateValue:String)** respectively **OutputClass()**, **OutputClass(privateValue:String)**:



Example: Creating Objects



Calling Constructors

Constructors are called if the name of the created object is followed by parentheses:

| | | |
|-----------------------|---|--|
| Syntax | <pre>create <objectName>(<constructor parameters>); create local <objectName>(<constructor parameters>) using ...;</pre> | |
| Semantics | Calling a constructor. | |
| Substitutables | objectName | Any given name. |
| | constructorParameters | Parameters of the constructor. The compiler will invoke a constructor of the class of objectName having the same parameters. |
| Examples | <pre>create o1; create o2(); // difference to o1: looking for a constructor without any arguments create o3(input.aString); create local l1 using typeOf(o1); create local l2() using OutputUsage; create local l3("local value") using typeOf(o3);</pre> | |

Calling Constructors of Super Classes

If a constructor has to call the constructor of its super class explicitly, the keyword **super** can be used. The `super()` statement invokes the constructor that is matching parameters of the `super()` call itself. This call has to happen as the first statement in the first action.

| | | |
|------------------|--|--|
| Syntax | <pre>super(<constructor parameters>);</pre> | |
| Semantics | Calling a constructor of the super (parent) class. | |
| | constructorParameters | Parameters of the constructor. The compiler will then invoke a constructor of the class of <code>objectName</code> having the same parameters. |
| Examples | <pre>super(); super(privateValue);</pre> | |