METHOD

```
METHOD method-name
  OF [INTERFACE] interface-name
  IS subprogram-name
END-METHOD
```

Dieses Kapitel behandelt folgende Themen:

- Funktion
- Syntax-Beschreibung
- Beispiel

Eine Erläuterung der in dem Syntax-Diagramm verwendeten Symbole entnehmen Sie dem Abschnitt *Syntax-Symbole*.

Verwandte Statements: CREATE OBJECT | DEFINE CLASS | INTERFACE | PROPERTY | SEND METHOD

Gehört zur Funktionsgruppe: *Komponentenbasierte Programmierung*

Funktion

Das `METHOD`-Statement weist ein Subprogramm als Implementierung zu einer Methode zu, und zwar außerhalb einer Interface-Definition. Es wird verwendet, wenn die betreffende Interface-Definition aus einem Copycode übernommen wird und auf eine klassenspezifische Weise implementiert werden soll.

Das `METHOD`-Statement kann nur innerhalb eines `DEFINE CLASS`-Statements und im Anschluss an die Interface-Definition verwendet werden. Die angegebenen Interface- und Methoden-Namen müssen innerhalb der Interface-Definitionen des `DEFINE CLASS`-Statements festgelegt werden.

Syntax-Beschreibung

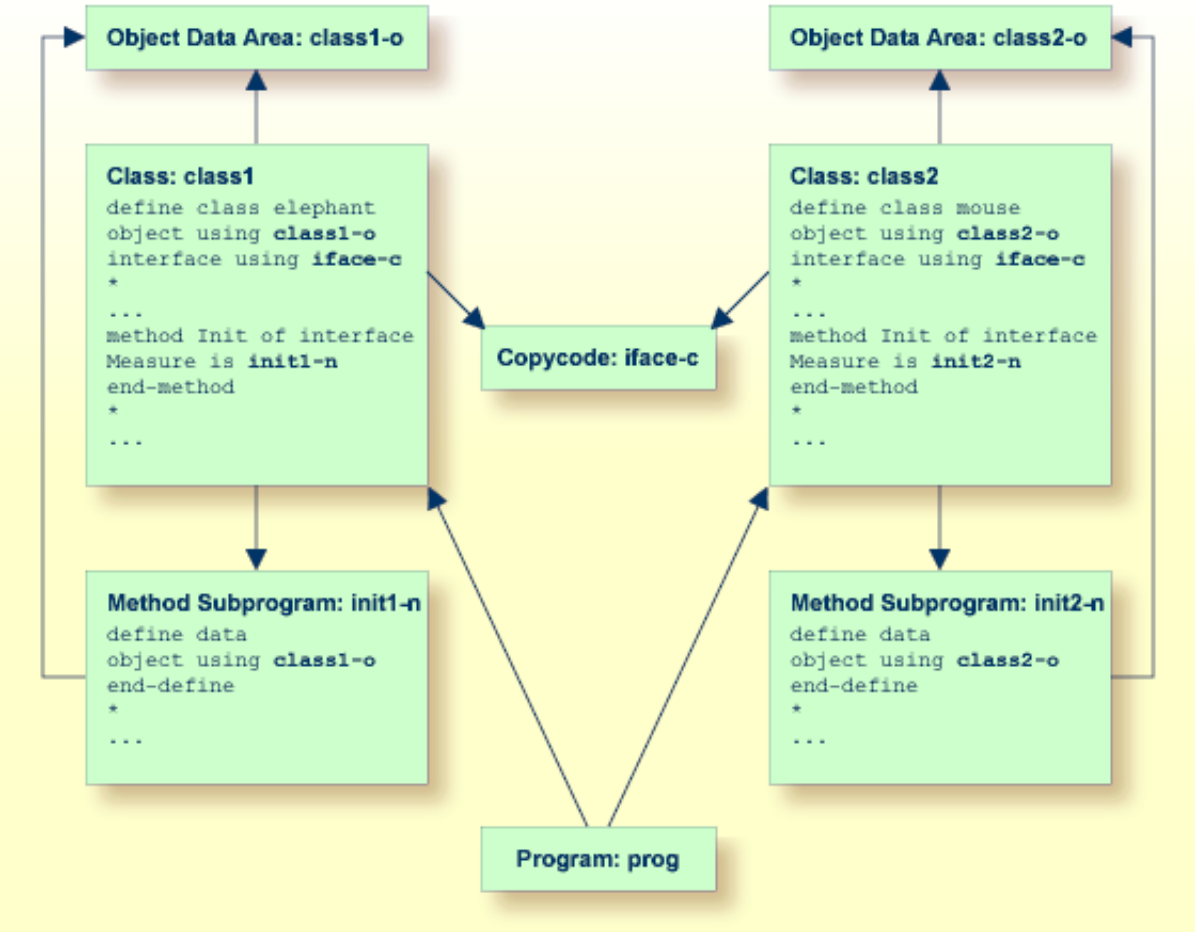
method-name	**Method-Name:** Dies ist der der Methode zugewiesene Name.
OF *interface-name*	**Interface-Name:** Dies ist der dem Interface zugewiesene Name.
IS *subprogram-name*	**Name des Subprogramms:** Dies ist der Name des Subprogramms, das die Methode implementiert. Der Name des Subprogramms besteht aus bis zu 8 Zeichen. Die Voreinstellung ist *method-name* (wenn die `IS`-Klausel nicht angegeben wird).
END-METHOD	Das für Natural reservierte Wort `END–METHOD` muss zum Beenden des `METHOD`-Statements benutzt werden.

Beispiel

Das folgende Beispiel zeigt, wie dasselbe Interface in zwei Klassen unterschiedlich implementiert wird und wie das `PROPERTY`-Statement und das `METHOD`-Statement zu diesem Zweck verwendet werden.

Das Interface `Measure` wird im Copycode `iface-c` definiert. Die Klassen `Elephant` und `Mouse` implementieren beide das Interface `Measure`. Deshalb beinhalten sie beide den Copycode `iface-c`. Die Klassen implementieren aber die Property `Height` mittels verschiedener Variablen von ihren betreffenden Object Data Areas, und sie implementieren die Methode `Init` mit unterschiedlichen Subprogrammen. Sie verwenden das Statement `PROPERTY`, um die ausgewählte Data Area-Variable der Property zuzuweisen, und das Statement `METHOD`, um das ausgewählte Subprogramm der Methode zuzuweisen.

Jetzt kann das Programm `prog` Objekte beider Klassen erstellen und sie mittels derselben Methode `Init` initialisieren, wobei die Schritte der Initialisierung der betreffenden Klassen-Implementierung überlassen werden.



Im folgenden finden Sie den vollständigen Inhalt der im vorstehenden Beispiel verwendeten Natural-Module:

Copycode: iface-c

```
interface Measure
*
property Height(p5.2)
end-property
*
property Weight(i4)
end-property
*
method Init
end-method
*
end-interface
```

Class: class1

```
define class elephant
object using class1-o
interface using iface-c
*
property Height of interface Measure is height
end-property
*
```

```
property Weight of interface Measure is weight
end-property
*
method Init of interface Measure is init1-n
end-method
*
end-class
end
```

LDA Object Data: class1-o

```
*   *** Top of Data Area ***
  1 HEIGHT                  P 5.2
  1 WEIGHT                  I 2
*   *** End of Data Area ***
```

Method Subprogram: init1-n

```
define data
object using class1-o
end-define
*
height := 17.3
weight := 120
*
end
```

Class: class2

```
define class mouse
object using class2-o
interface using iface-c
*
property Height of interface Measure is size
end-property
*
property Weight of interface Measure is weight
end-property
*
method Init of interface Measure is init2-n
end-method
*
end-class
end
```

LDA Object Data: class2-o

```
*   *** Top of Data Area ***
  1 SIZE                    P 3.2
  1 WEIGHT                  I 1
*   *** End of Data Area ***
```

Method Subprogram: init2-n

```
define data
object using class2-o
end-define
*
```

```
size := 1.24
weight := 2
*
end
```

Program: prog

```
define data local
1 #o handle of object
1 #height(p5.2)
1 #weight(i4)
end-define
*
create object #o of class 'Elephant'
send "Init" to #o
#height := #o.Height
#weight := #o.Weight
write #height #weight
*
create object #o of class 'Mouse'
send "Init" to #o
#height := #o.Height
#weight := #o.Weight
write #height #weight
*
end
```