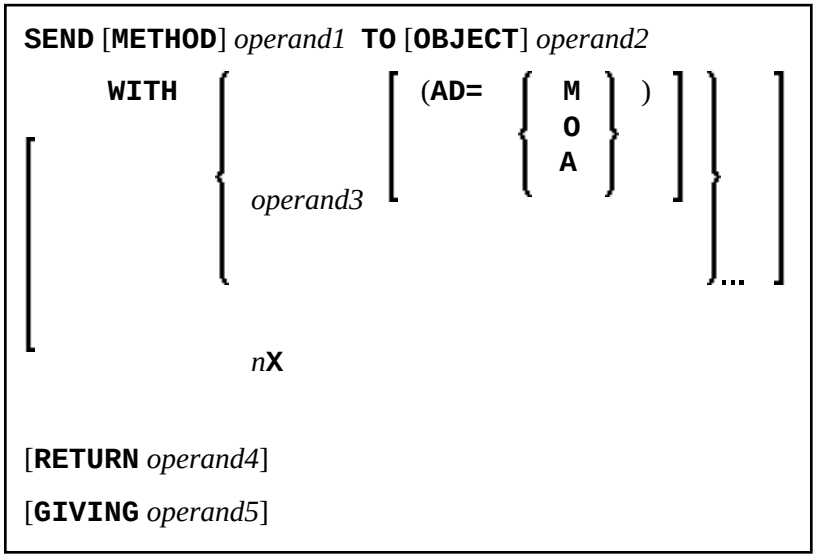
SEND METHOD



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` | `METHOD` | `PROPERTY`

Gehört zur Funktionsgruppe: *Komponentenbasierte Programmierung*

Funktion

Das `SEND METHOD`-Statement dient dazu, eine bestimmte Methode eines Objekts aufzurufen. Informationen zur komponentenbasierten Programmierung, siehe *NaturalX* im *Leitfaden zur Programmierung*.

Syntax-Beschreibung

Operanden-Definitionstabelle:

Operand	Mögliche Struktur					Mögliche Formate													Referenzierung erlaubt	Dynam. Definition
operand1	C	S				A													ja	nein
operand2		S																O	nein	nein
operand3	C	S	A	G		A	U	N	P	I	F	B	D	T	L	C	G	O	ja	nein
operand4		S	A			A	U	N	P	I	F	B	D	T	L	C	G	O	ja	nein
operand5		S			N					I									ja	nein

Format C und G kann nur an Methoden lokaler Klassen übergeben werden. Weitere Informationen siehe Abschnitt *Local Classes*.

Syntax-Element-Beschreibung:

operand1	**Method-Name:**
	
	operand1 ist der Name einer Methode, die vom in *operand2* angegebenen Objekt unterstützt wird.
	
	Da die Methoden-Namen in unterschiedlichen Interfaces einer Klasse identisch sein können, kann der Methoden-Name in *operand1* auch mit dem Interface-Namen versehen werden, um Mehrdeutigkeiten zu vermeiden.
	
	Im folgenden Beispiel hat das Objekt `#O3` ein Interface `Iterate` mit der Methode `Start`. Es gelten die folgenden Statements:
	
	`* Specifying only the method name.`
	`SEND 'Start' TO #O3`
	`* Qualifying the method name with the interface name.`
	`SEND 'Iterate.Start' TO #O3`
	
	Wenn kein Interface-Name angegeben wird, sucht Natural den Methoden-Namen in allen Interfaces der Klasse. Wenn der Methoden-Name in mehr als einem Interface gefunden wurde, tritt ein Laufzeitfehler auf.

operand2	**Object-Handle:** Die Handle des Objekts, an das der Aufruf der Methode gesendet werden soll. *operand2* muss als Objekt-Handle (`HANDLE OF OBJECT`) definiert werden. Das Objekt muss bereits vorhanden sein. Um eine Methode des aktuellen Objekts innerhalb einer Methode aufzurufen, verwenden Sie die Systemvariable `*THIS-OBJECT`.

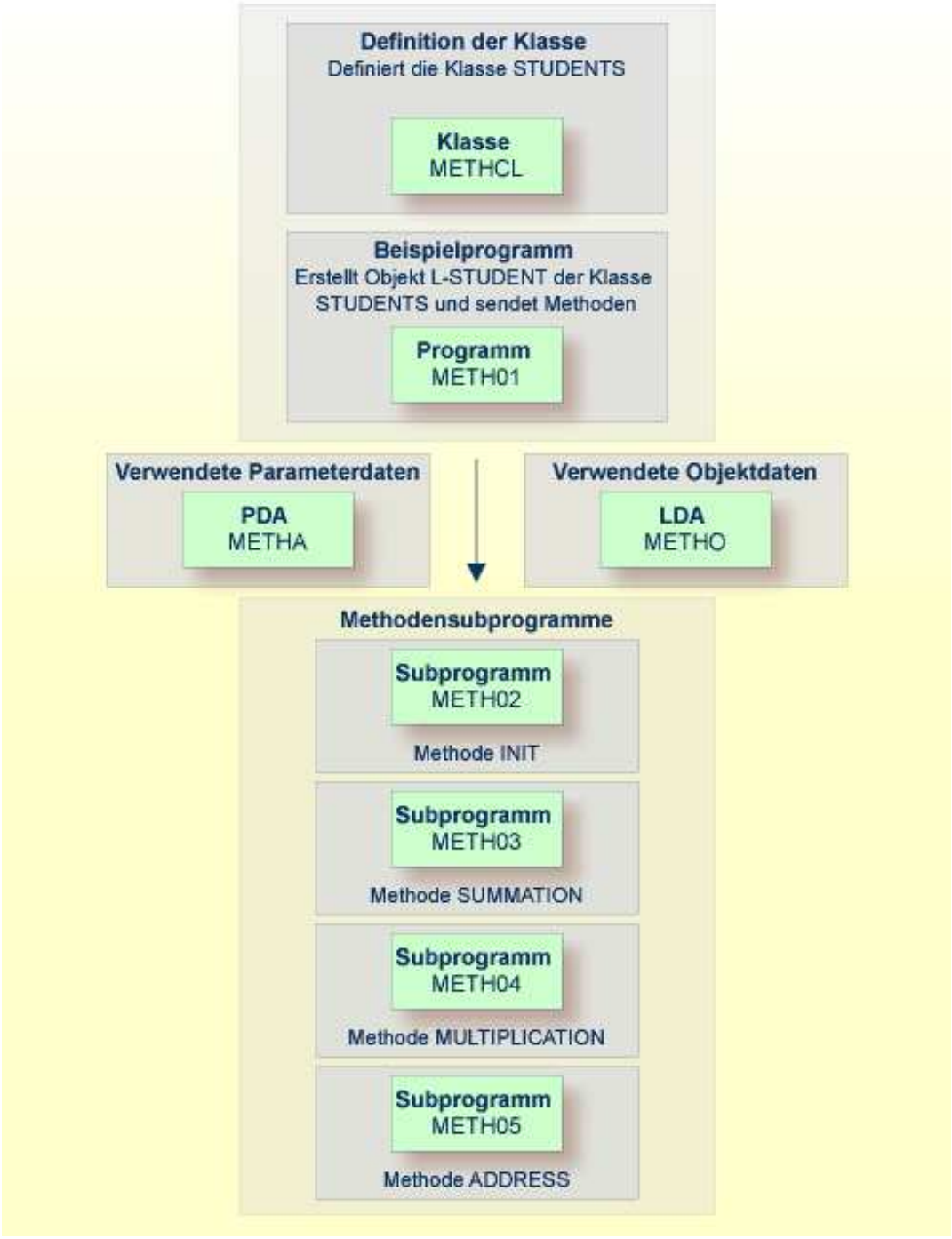
operand3	**Methodenspezifische Parameter:**
	
	Als *operand3* können Sie Parameter angeben, die methodenspezifisch sind.
	
	Im folgenden Beispiel hat das Objekt `#O3` die Methode `PositionTo` mit dem Parameter `Pos`. Die Methode wird wie folgt aufgerufen:
	
	`SEND 'PositionTo' TO #O3 WITH Pos`
	
	Methoden können optionale Parameter haben. Optionale Parameter brauchen nicht angegeben zu werden, wenn die Methode aufgerufen wird. Um einen optionalen Parameter wegzulassen, verwenden Sie den Platzhalter `1X`. Um *n* optionale Parameter wegzulassen, verwenden Sie den Platzhalter *n*X.
	
	Im folgenden Beispiel hat die Methode `SetAddress` des Objekts `#O4` die Parameter `FirstName` (Vorname), `MiddleInitial` (Mittlere Initiale), `LastName` (Nachname), `Street` (Straße) und `City` (Stadt), wobei `MiddleInitial`, `Street` und `City` optional sind. Es gelten die folgenden Statements:
	
	`* Specifying all parameters.`
	`SEND 'SetAddress' TO #O4 WITH FirstName MiddleInitial LastName Street City`
	`* Omitting one optional parameter.`
	`SEND 'SetAddress' TO #O4 WITH FirstName 1X LastName Street City`
	`* Omitting all optional parameters.`
	`SEND 'SetAddress' TO #O4 WITH FirstName 1X LastName 2X`
	
	Wenn ein Pflichtparameter weggelassen wird, führt dies zu einem Laufzeitfehler.

AD=	**Attribut-Definition:** Wenn *operand3* eine Variable ist, können Sie sie wie folgt kennzeichnen:
	AD=O — Nicht modifizierbar, siehe Session-Parameter `AD=O`.
	AD=M — Modifizierbar, siehe Session-Parameter `AD=M`. Dies ist die Voreinstellung.
	AD=A — Nur für Eingabe, siehe Session-Parameter `AD=A`.
	Wenn *operand3* eine Konstante ist, kann `AD` nicht explizit angegeben werden. Für Konstanten gilt immer `AD=O`.
***n*X**	**Zu überspringende Parameter:** Mit der Notation *n*`X` können Sie angeben, dass die nächsten *n* Parameter übersprungen werden sollen (zum Beispiel `1X`, um den nächsten Parameter zu überspringen, oder `3X`, um die nächsten drei Parameter zu überspringen). Dies bedeutet, dass für die nächsten *n* Parameter an die Methode keine Werte übergeben werden. Bei einer in Natural implementierten Methode muss ein zu überspringender Parameter mit dem Schlüsselwort `OPTIONAL` im `DEFINE DATA PARAMETER`-Statement des Subprogramms der Methode definiert sein. `OPTIONAL` bedeutet, dass ein Wert vom aufrufenden Objekt an einen solchen Parameter übergeben werden kann, aber nicht unbedingt muss.

RETURN ***operand4***	**RETURN-Klausel:** Wenn die `RETURN`-Klausel weggelassen wird und die Methode einen Rückgabewert hat, wird der Rückgabewert nicht berücksichtigt. Wenn die `RETURN`-Klausel angegeben wird, enthält *operand4* den Rückgabewert der Methode. Wenn die Ausführung der Methode ohne Erfolg abgebrochen wird, wird *operand4* auf seinen ursprünglichen Wert zurückgesetzt. **Anmerkung:** Bei in Natural geschriebenen Klassen wird der Rückgabewert einer Methode durch Eingabe eines zusätzlichen Parameters in der Parameter Data Area der Methode und durch Kennzeichnung mit `BY VALUE RESULT` definiert. Weitere Informationen siehe Abschnitt *PARAMETER-Klausel*. Deshalb enthält die Parameter Data Area einer Methode, die in Natural geschrieben ist, und die einen Rückgabewert hat, neben den Methoden-Parametern immer ein zusätzliches Feld. Dies ist zu berücksichtigen, wenn Sie eine Methode einer in Natural geschriebenen Klasse aufrufen und die Parameter Data Area der Methode im `SEND`-Statement verwenden möchten.
GIVING ***operand5***	**GIVING-Klausel:** Wenn die `GIVING`-Klausel nicht angegeben wird, wird die Natural-Laufzeitfehlerverarbeitung angestoßen, wenn ein Fehler auftritt. Wenn die `GIVING`-Klausel angegeben wird, enthält *operand5* die Natural-Meldungsnummer, wenn ein Fehler aufgetreten ist, oder Null (0), wenn kein Fehler aufgetreten ist.

Beispiel

Das folgende Diagramm gibt eine Übersicht über die Programmierobjekte, die in diesem Beispiel benutzt werden. Der entsprechende Sourcecode und die Programm-Ausgabe sind im Folgenden veranschaulicht



Programm METH01: CTREATE OBJECT und SEND METHOD mit einer Klasse und mehreren Methoden:

```
** Example 'METH01':  CREATE OBJECT and SEND METHOD
**                    using a class and several methods (see METH*)
************************************************************************
DEFINE DATA
LOCAL
  USING METHA
LOCAL
1 L-STUDENT HANDLE OF OBJECT
1 #NAME     (A20)
```

```
1 #STREET   (A20)
1 #CITY     (A20)
1 #SUM      (I4)
1 #MULTI    (I4)
END-DEFINE
*
CREATE OBJECT L-STUDENT OF CLASS 'STUDENTS' /* see METHCL for class
*
L-STUDENT.<> := 'John Smith'
*
SEND METHOD 'INIT' TO L-STUDENT             /* see METHCL
     WITH #VAR1 #VAR2 #VAR3 #VAR4
*
SEND METHOD 'SUMMATION' TO L-STUDENT        /* see METHCL
     WITH #VAR1 #VAR2 #VAR3 #VAR4
*
SEND METHOD 'MULTIPLICATION' TO L-STUDENT   /* see METHCL
     WITH #VAR1 #VAR2 #VAR3 #VAR4
*
#NAME  := L-STUDENT.<>
#SUM   := L-STUDENT.<>
#MULTI := L-STUDENT.<>
*
SEND METHOD 'ADDRESS' TO L-STUDENT          /* see METHCL
*
#STREET := L-STUDENT.<>
#CITY   := L-STUDENT.<>
*
*
WRITE 'Name  :' #NAME
WRITE 'Street:' #STREET
WRITE 'City  :' #CITY
WRITE ' '
WRITE 'The summation of     ' #VAR1 #VAR2 #VAR3 #VAR4
WRITE 'is' #SUM
WRITE 'The multiplication of' #VAR1 #VAR2 #VAR3 #VAR4
WRITE 'is' #MULTI
*
END
```

Vom Programm METH01 benutzte Klassen-Definition METHCL:

```
** Example 'METHCL': DEFINE CLASS (used by METH01)
************************************************************************
* Defining class STUDENTS for METH01
*
DEFINE CLASS STUDENTS
  OBJECT
    USING METHO                 /* Object data for class STUDENTS
  /*
  INTERFACE STUDENT-ARITHMETICS
    PROPERTY FULL-NAME
      IS NAME
    END-PROPERTY
    PROPERTY SUM
    END-PROPERTY
    PROPERTY MULTI
    END-PROPERTY
*
    METHOD INIT
      IS METH02
      PARAMETER USING METHA
```

```
    END-METHOD
    METHOD SUMMATION
      IS METH03
      PARAMETER USING METHA
    END-METHOD
    METHOD MULTIPLICATION
      IS METH04
      PARAMETER USING METHA
    END-METHOD
  END-INTERFACE
*
  INTERFACE STUDENT-ADDRESS
    PROPERTY STUDENT-NAME
      IS NAME
    END-PROPERTY
    PROPERTY STREET
    END-PROPERTY
    PROPERTY CITY
    END-PROPERTY
*
    METHOD ADDRESS
      IS METH05
    END-METHOD
  END-INTERFACE
END-CLASS
END
```

Local Data Area METHO (Objektdaten), die von der Klasse METHCL und den Subprogrammen METH02, METH03, METH04 und METH05 benutzt wird:

```
Local     METHO     Library SYSEXSYN                      DBID    10 FNR    32
Command                                                               > +
I T L  Name                            F Length     Miscellaneous
All -- --------------------------------- ---------- ------------------------>
     1 NAME                            A         20
     1 STREET                          A         30
     1 CITY                            A         20
     1 SUM                             I          4
     1 MULTI                           I          4
```

Parameter Data Area METHA, die vom Programm METH01, der Klasse METHCL und den Subprogrammen METH02, METH03 und METH04 benutzt wird:

```
Parameter METHA     Library SYSEXSYN                      DBID    10 FNR    32
Command                                                               > +
I T L  Name                            F Length     Miscellaneous
All -- --------------------------------- ---------- ------------------------>
     1 #VAR1                           I          4
     1 #VAR2                           I          4
     1 #VAR3                           I          4
     1 #VAR4                           I          4
```

Subprogramm METH02 - vom Programm METH01 verwendete Methode INIT:

```
** Example 'METH02': Method INIT (used by METH01)
************************************************************************
DEFINE DATA
PARAMETER
  USING METHA
OBJECT
  USING METHO
```

```
END-DEFINE
*
* Method INIT of class STUDENTS
*
#VAR1 := 1
#VAR2 := 2
#VAR3 := 3
#VAR4 := 4
*
END
```

Subprogramm METH03 - vom Programm METH01 verwendete Methode SUMMATION:

```
** Example 'METH03': Method SUMMATION (used by METH01)
************************************************************************
DEFINE DATA
PARAMETER
  USING METHA
OBJECT
  USING METHO
END-DEFINE
*
* Method SUMMATION of class STUDENTS
*
COMPUTE SUM = #VAR1 + #VAR2 + #VAR3 + #VAR4
END
```

Subprogramm METH04 - vom Programm METH01 verwendete Methode MULTIPLICATION:

```
** Example 'METH04': Method MULTIPLICATION (used by METH01)
************************************************************************
DEFINE DATA
PARAMETER
  USING METHA
OBJECT
  USING METHO
END-DEFINE
*
* Method MULTIPLICATION of class STUDENTS
*
COMPUTE MULTI = #VAR1 * #VAR2 * #VAR3 * #VAR4
END
```

Subprogramm METH05 - vom Programm METH01 verwendete Methode ADDRESS:

```
** Example 'METH05': Method ADDRESS (used by METH01)
************************************************************************
DEFINE DATA
  OBJECT USING METHO
END-DEFINE
*
* Method ADDRESS of class STUDENTS
*
IF NAME = 'John Smith'
  STREET := 'Oxford street'
  CITY   := 'London'
END-IF
END
```

Ausgabe des Programms METH01:

```
Page      1                                          05-01-17  15:59:04

Name  : John Smith
Street: Oxford street
City  : London

The summation of               1          2          3          4
is          10
The multiplication of          1          2          3          4
is          24
```