SIEMENS
Ingenuity for life

Tecnomatix Plant Simulation World Wide User Conference 2016

SimTalk 2.0

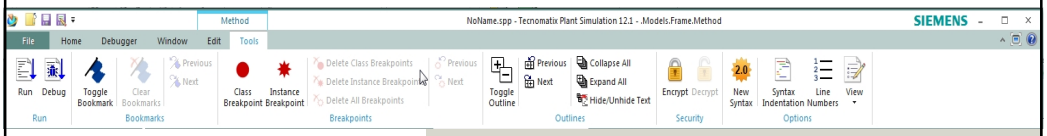
Moderate enhancements to increase user-friendliness

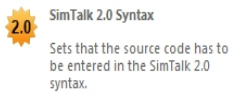
Michael Joos

 Realize innovation.

Working with Old Models

SIEMENS
Ingenuity for life

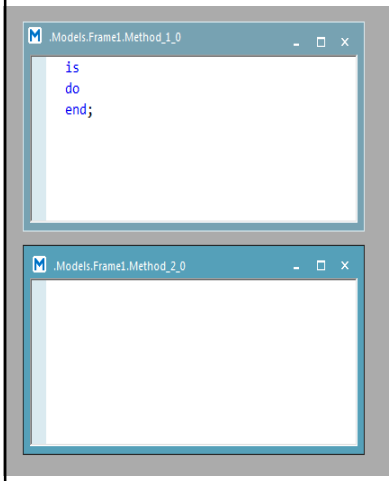




- When you load a model, which you saved in a previous version of Plant Simulation, it will run without changes.
- SimTalk 1.0 Methods and SimTalk 2.0 Methods can be used in the same model in parallel.
- The ribbon button Method > Tools > New Syntax independently sets for each Method if the source code has to be entered in SimTalk 1.0 or SimTalk 2.0 syntax.

Less is More

SIEMENS Ingenuity for life

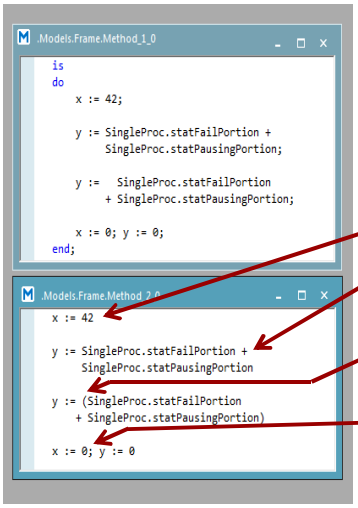


`is do end;` is obsolete in SimTalk 2.0.

A completely empty Method is a correct Method.

Semicolon is not required anymore

SIEMENS Ingenuity for life



- SimTalk 2.0 does not require a semicolon after a statement
- The line end defines the end of the statement
- If the statement is not yet complete (e.g. `y:=x+`) then it will automatically continue in the next line
- You can use parentheses to force the statement to continue in the next line
- The semicolon separates multiple statements in one line

If-Then-Else

SIEMENS Ingenuity for life

.Models.Frame1.Method_1_0

```
is
do
    if a = 5 then
        print 5;
    end;
end;
```

.Models.Frame1.Method_2_0

```
if a = 5
    print 5
end
```

The `if`-statement does not require the keyword `then` anymore

- Less source code to write
- `then` is still allowed if you prefer to use it

While-Loop

SIEMENS Ingenuity for life

.Models.Frame1.Method_1_0

```
is
do
    while a > 5 loop
        a := a - 1;
    end;
end;
```

.Models.Frame1.Method_2_0

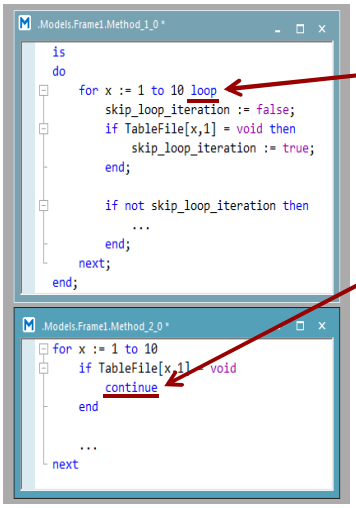
```
while a > 5
    a := a - 1
end
```

The `while`-loop does not require the keyword `loop` anymore

- Less source code to write
- `loop` is still allowed if you prefer to use it

For-Loop

SIEMENS Ingenuity for life



The `for`-loop does not require the keyword `loop` anymore

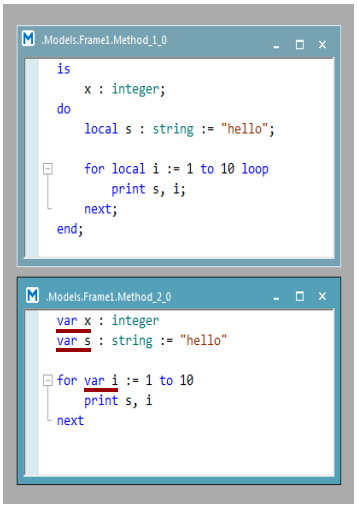
- Less source code to write
- `loop` is still allowed if you prefer to use it

The `continue`-statement skips the rest of the loop iteration and continues with the next iteration

The `exitloop`-statement (which is available in both SimTalk 1.0 and in SimTalk 2.0) exits the loop

Local Variable Declaration

SIEMENS Ingenuity for life



Local variables are declared with the keyword `var`

`var` is shorter than `local` and is used in Java and JavaScript too

Switch-Statement

SIEMENS Ingenuity for life

.Models.Frame1.Method_1_0

```
is
do
    inspect number
    when 1 then
        print "One";
    when 2, 3, 5, 7 then
        print "Prime number";
    else
        print "No special number";
    end;
end;
```

.Models.Frame1.Method_2_0

```
switch number
case 1
    print "One"
case 2, 3, 5, 7
    print "Prime number"
else
    print "No special number"
end
```

The `inspect`-`when`-statement was replaced by the `switch`-`case`-statement

`switch` is used in C++, Java and JavaScript too

Parameter and Return Value Declaration

SIEMENS Ingenuity for life

.Models.Frame1.Method_1_0

```
(obj: object;
 x, y: integer) : string
is
do
    obj.XPos := x;
    obj.YPos := y;
    return obj.Name;
end;
```

.Models.Frame1.Method_2_0

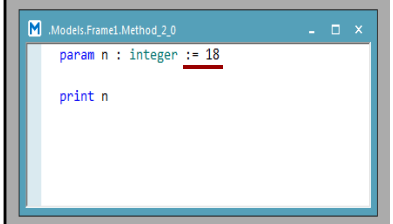
```
param obj: object,
      x, y: integer -> string

obj.XPos := x
obj.YPos := y
return obj.Name
```

- `param` declares method parameters
- Different parameter types (here *object* and *integer*) are separated with a comma instead of a semicolon. This reflects the call syntax better, for example `Method_2_0(SingleProc, 10, 20)`
- The return value of a method is now declared with `->` instead of `:`

Default Arguments (1/2)

SIEMENS Ingenuity for life



Parameters can be predefined with *default arguments* in the declaration (starting with Plant Simulation 12.2).

- If an argument is passed to the parameter, it will be used in the Method
- If no argument is passed, the *default value* will be used instead

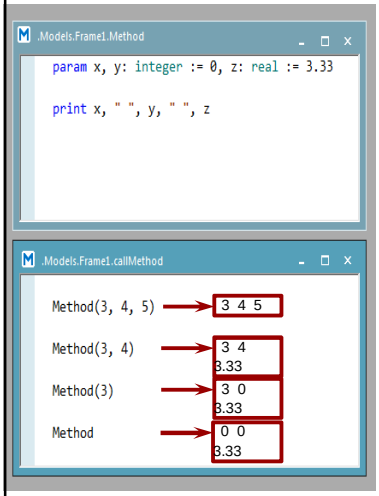
Example:

```
param n: integer := 18

Method_2_0      -- prints 18
Method_2_0(19)  -- prints 19
```


Default Arguments (2/2)

SIEMENS Ingenuity for life



A default argument is allowed for the data types `integer`, `real`, `length`, `speed`, `acceleration`, `weight`, `time`, `boolean`, `string`, `object`, `table`, `list`, `stack` and `queue`.

It is not allowed for the data types `date`, `datetime` and arrays.

For the data types *object*, *table*, *list*, *stack* and *queue*, the only allowed default value is `void`.

A default argument is only allowed for the *n last parameters*. It cannot be used in the middle of a declaration.

```
param x:real, s:string, z:integer:=42
```

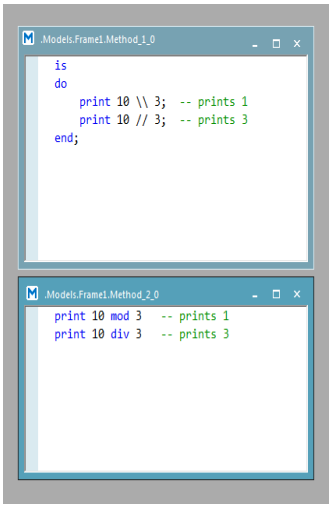
O.K.

```
param x:real, s:string:="Hello", z:integer:=42
```

O.K.

Mathematical Operators Modulo and Integer Division

SIEMENS Ingenuity for life

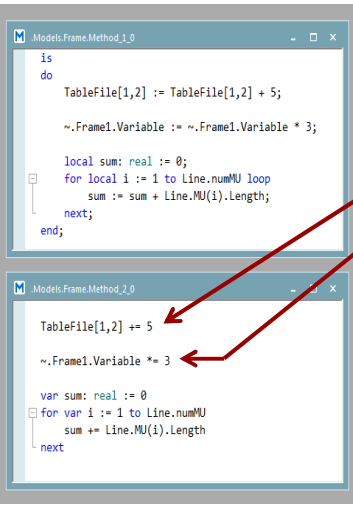


The mathematical operators *modulo* and *integer division* now use the more intuitive keywords `mod` and `div`.

- `\\` was replaced by `mod`
- `//` was replaced by `div`

Easier Change of Values

SIEMENS Ingenuity for life



New operators `+=` , `-=` , `*=`

Example:

`i:=i+1 -> i+=1`

The value in the table cell [1,2] can be directly increased by 5

The value of the *global Variable* can be directly multiplied by 3

`x+=y` is short for `x:=x+y`

`x-=y` is short for `x:=x-y`

`x*=y` is short for `x:=x*y` ⚠

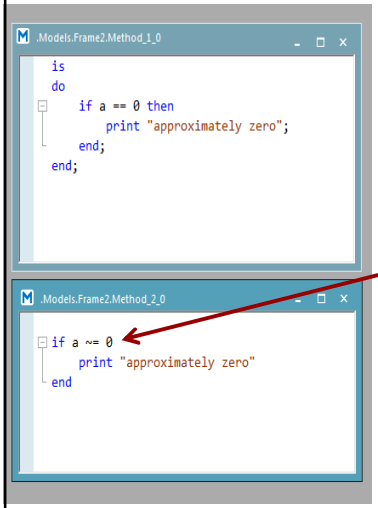
`x/=y` is "unequal" as in SimTalk 1.0

More is possible, for example `SingleProc.cont.Length -= 0.5`

The new operators also increase performance.

About-Equal-Operator

SIEMENS Ingenuity for life



The about-equal-operator has been changed

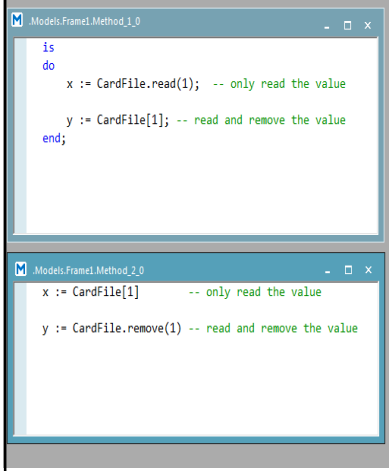
- Improved readability
- Avoids misinterpretation as C++ or Java syntax

You now have to use:

- For *about equal*: `~=` instead of `==`
- For *less than or about equal*: `<~=` instead of `<==`
- For *greater than or about equal*: `>~=` instead of `>==`

Reading CardFiles

SIEMENS Ingenuity for life




```
x := CardFile[1]
```

Formulas behave differently without warning!

SimTalk 1.0:

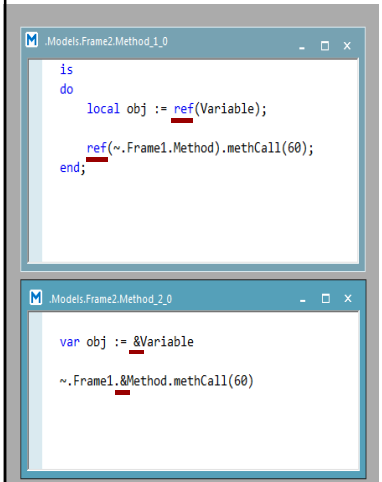
When reading the contents of a cell of a *CardFile* or *stack* with the bracket operator, Plant Simulation *reads and removes* the cell.
The remaining cells move up one position.

SimTalk 2.0:

When reading the contents of a cell of a *CardFile* or *stack* with the bracket operator, Plant Simulation *reads the cell and leaves it* in the *CardFile/stack*.

Ref-Operator Replaced By &

SIEMENS Ingenuity for life



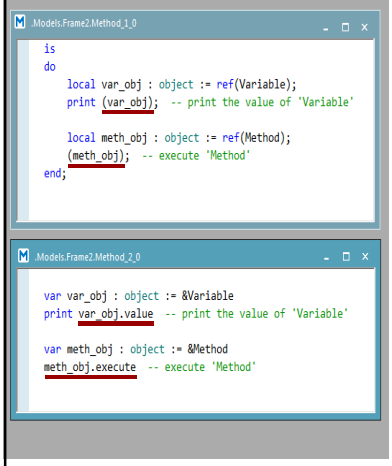
The `ref`-operator is replaced by `&`

- Leads to more readable code
- `ref()` raised questions as it looked like a function
- The parentheses are not needed anymore
- The `&` is positioned where it is expected (right in front of the *Variable/Method*)

`&` prevents the evaluation of a *Variable* or the execution of a *Method*

Dereferencing using Parentheses

SIEMENS Ingenuity for life



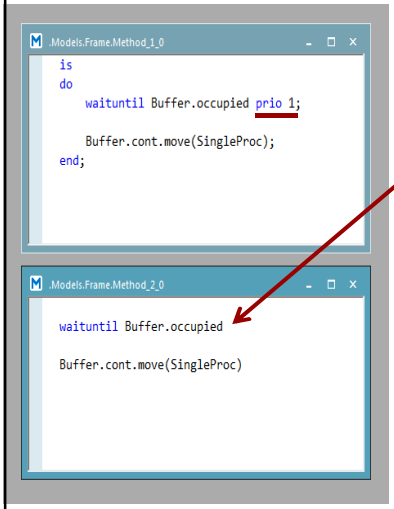
In SimTalk 1.0 parentheses could be used to dereference an object.

This was hard to comprehend.

In SimTalk 2.0 you need to use the built-in method *value* to dereference a *global Variable*, and *execute* to dereference a *Method*.

(This is also possible in SimTalk 1.0)

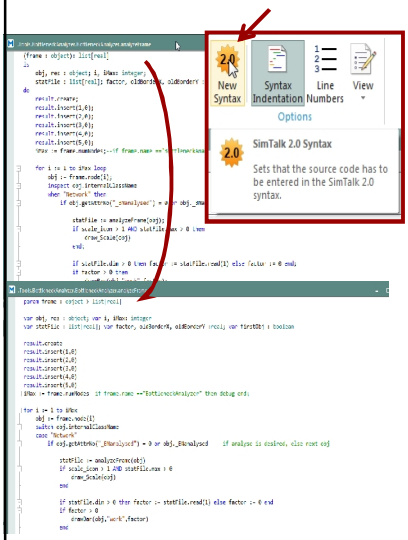
Waituntil/Stopuntil



The priority definition of the *waituntil*-statement and the *stopuntil*-statement is now optional.

If you do not define it explicitly, the priority is 1.

SimTalk 1.0 to 2.0 Converter (1/2)



For a single *Method*:

- Click the New Syntax button
- Press F7 or click the Apply Changes button to apply the converted source code

Bracket operators on *CardFiles* will be detected and automatically handled correctly:

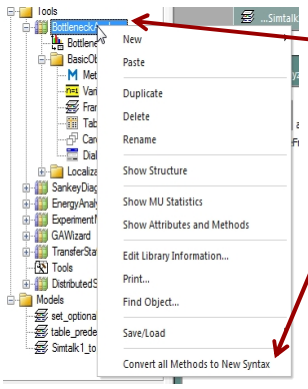
```
CardFile[1] := CardFile[2];
```

will be converted to

```
CardFile.insert(1, CardFile.remove(2))
```

SimTalk 1.0 to 2.0 Converter (2/2)

SIEMENS
Ingenuity for life



For all *Methods* inside a folder or all *Methods* in the entire model:

- Shift + right mouse click on a *Folder* or the *basis* in the class library
- Select Convert all Methods to New Syntax

When you convert all *Methods* of the model, locked libraries will *not* be converted.

Thank you!

SIEMENS
Ingenuity for life

Michael Joos
Tecnomatix Plant Simulation

Weissacher Str. 11
70499 Stuttgart

E-mail: michael.joos@siemens.com

siemens.com