

第六章

Simtalk编程

方法对象



- 使用对象**方法**，可以用程序控制其他对象启动，并在模拟运行期间执行工厂模拟。可以在方法中使用内置方法、关键字、任务和控制结构的组合来构建自己的程序，编写自己的模板供自己使用。这种结构结合了生产力和灵活性。通过编写用户定义的方法，我们可以修改对象的行为，使其完全符合您的建模需求。

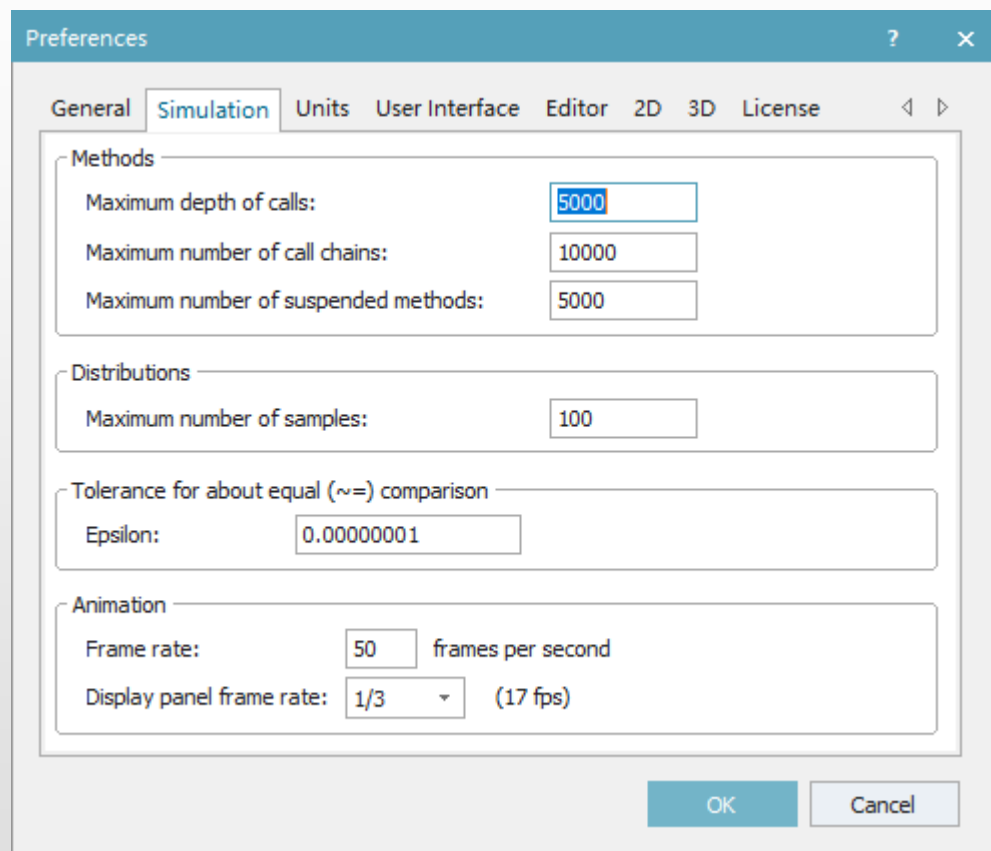
```
var n:integer:=station1.statNumIn
if n<=100
  @.move(station2)
elseif n<=300
  @.move(station3)
else
  @.move(station4)
end
```

方法对象

- 一个Method可以做到：
- ①在模拟过程中， 对一个准确的事件做出反应。
- ②获取条件和设置条件
- ③执行语句
- ④修改和扩展对象的行为
- ⑤为新的使用者提供一个默认的对话方块

设置Method编辑器首选项

- 编辑首选项 File→Preferences :
- **Maximum depth of calls (最大调用深度)**
 - — 多少methods可能由另一个方法调用.
- **Maximum number of call chains (最大调用数量)**
 - — 同一时间可以调用方法的最大数量。
- **Maximum number of suspended methods (最多挂起Method数量)**
 - — 同一时间可能会暂停并等待事件发生的方法数量。
- **Display line numbers 选择框**
 - -- 选它，你可以快速更改包含错误的源代码的行为



编辑界面

- 我们点开一个Method，进入方法的**编辑界面**。在这里我们可以查找源代码中的文本，自动完成语句（Ctrl+Space），做注释（Ctrl+Shift+Q），**使用语法模板**，激活/关闭源代码继承，更改应用源代码等



```
if...end
if...else...end
if...elseif...end
switch...case...end
switch...case...else...end
while...end
repeat...until
for...next
```



工具界面

- 在方法的**工具界面**里，我们可以运行方法（F5），打开调试器逐步调试（F11），打开/关闭断点（F9），**1.0和2.0语法的切换**，显示/隐藏行号等操作。



注释

- 注释可以提升method的可读性，说明读者理解函数。两个横杆--或两条斜杠//可用作一行的注释。
- 输入（/*...*/）用来编写多行注释，以/*开始，以*/结束，中间部分的全部内容都是注释。
- 如图所示，绿色字体即为该程序的注释。

```
repeat
  switch step      --重复执行下列步骤
  case 1 --步骤1

    portal.movehook(1) --龙门起重机吊钩长度初始值为1m

  case 2
    var motionOK:integer:=portal.moveToObject(singleproc1)    --龙门起重机移动到singleproc1的位置

  case 3
    portal.moveHookabs(1) --吊钩伸长1m
  case 4
    singleproc1.cont.move(portal.hook) --singleproc1的物料进到吊钩上
    waituntil portal.hook.full --当吊钩满时,执行下一步
  case 5
    portal.moveHook(1) --吊钩长度回到初始值1m
  case 6
    if singleproc3.empty then --如果singleproc3为空时
      motionOK:=portal.moveToPosition(SingleProc3.xPos,SingleProc3.YPos) --起重机移动到singleproc3的x,y的位置
    elseif singleproc4.empty then --否则如果singleproc4为空时
      motionOK:=portal.moveToPosition(SingleProc4.xPos,SingleProc4.YPos) --起重机移动到singleproc4的x,y的位置
    end
end
```

Method结构

- Method包含以下结构，并非每一部分都需要
- `param` [Arguments] --参数
- 可以通过添加参数来扩展函数，即可以在多个方法中调用参数来执行操作。触发者必须要有和声明同样数量的参数才能正常触发函数。传递的数据类型必须与预期数据类型相同。
- 例如传递二个整数参数并返回布尔 值: `param v1,v2 : integer -> boolean`
- `var` [Local Variables]--局部变量
- 局部变量只可以在这个method中被访问，必须先声明后使用。例如: `var i : integer :=0`
- [Other Statements]--源代码
- 输入原始代码，如标准的method、控制策略、触发method、循环语句等。例如, `print "Hello world"`
- [Result]--返回值
- 输入返回值的数据类型，是参数的一部分。必须分配给关键字`result`。例如 `result := v1<v2`

```
param v1,v2:integer->boolean
v1:=1
v2:=2
result :=v1/=v2

param v1,v2:string->string
v1:="plant "
v2:="simulation"
result :=v1+v2
```

Method结构

- 声明局部变量需要始于var关键字，例如：
- 每个局部变量一行：
- **var i : integer**
- **var j : real**
- •每种对象类型一行：
- **var i,j, k : integer**
- **var x : string**
- 声明变量时赋初始值：
- **var i,j, k : integer :=1**
- **var x : string := "Hi"**

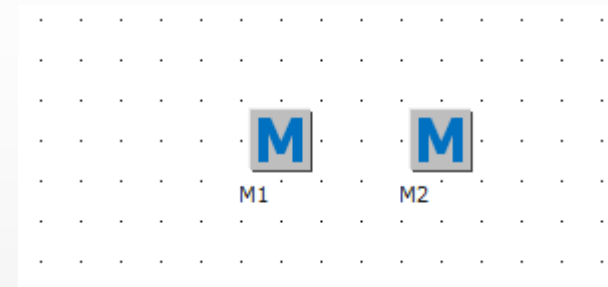
Method结构

- 参数的使用：
- 假定Method M1包含：
- `param number1 : real :=1, number2 : real:= 10`
- `var number3 := number1 + number2`
- `print number3`
- 如果Method M2中包含： --情况1
- `M2(3,5)`
- Then number3 is 8.
- 如果M2中包含： --情况2
- `M2(3)`
- Then number3 is 13.

Method结构

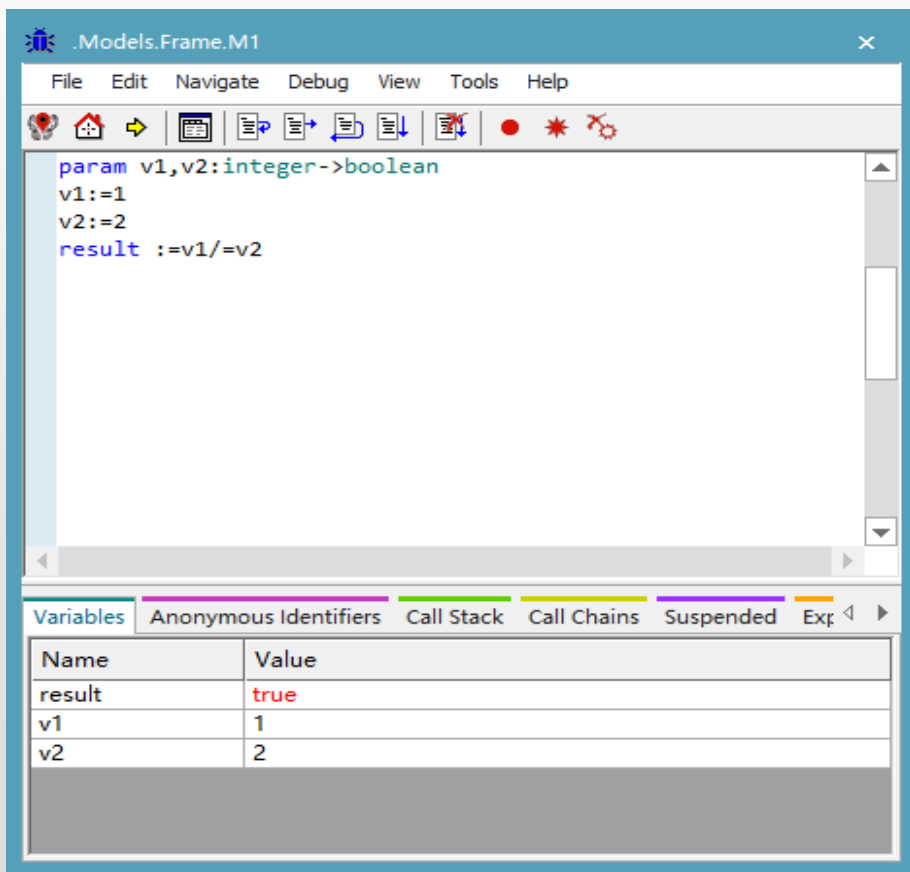
- 如果M2中包含代码： --情况3
- M2
- Then number3 is 11.

```
param number1 : real :=1, number2 : real:= 10  
var number3 := number1 + number2  
print number3
```



Method结构

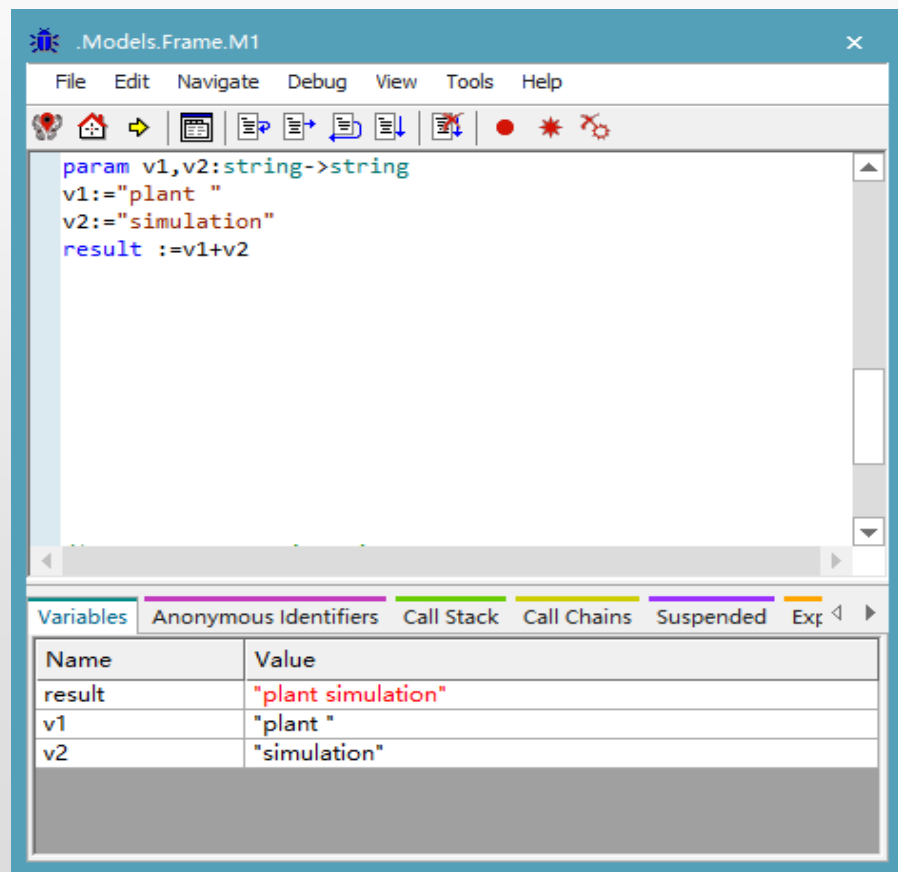
- 返回值的使用： -> 变量类型
- 分别执行下列代码，观察结果



The screenshot shows a code editor window titled ".Models.Frame.M1" with a menu bar (File, Edit, Navigate, Debug, View, Tools, Help) and a toolbar. The code defines a method with two integer parameters, v1 and v2, and a boolean return type. The method body assigns v1 to 1, v2 to 2, and calculates result as v1 divided by v2. The Variables panel at the bottom shows the state of the variables.

```
param v1,v2:integer->boolean
v1:=1
v2:=2
result :=v1/=v2
```

Name	Value
result	true
v1	1
v2	2



The screenshot shows a code editor window titled ".Models.Frame.M1" with a menu bar (File, Edit, Navigate, Debug, View, Tools, Help) and a toolbar. The code defines a method with two string parameters, v1 and v2, and a string return type. The method body assigns v1 to "plant " and v2 to "simulation", and calculates result as the concatenation of v1 and v2. The Variables panel at the bottom shows the state of the variables.

```
param v1,v2:string->string
v1:="plant "
v2:="simulation"
result :=v1+v2
```

Name	Value
result	"plant simulation"
v1	"plant "
v2	"simulation"

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：<https://d.book118.com/985130344033011042>