



ŠKODA AUTO University

# Computer Simulation of Logistics Processes

Programming in SimTalk

Jan Fábry

04/04/2023

# Programming in SimTalk

## Aim of the lecture

- To introduce the work with methods and conditional commands.



# Programming in SimTalk

## Structure of the lecture

- Method types.
- Method initialization.
- Value assignment:
  - Attributes.
- Conditional commands with IF.



# Programming in SimTalk

## Method types

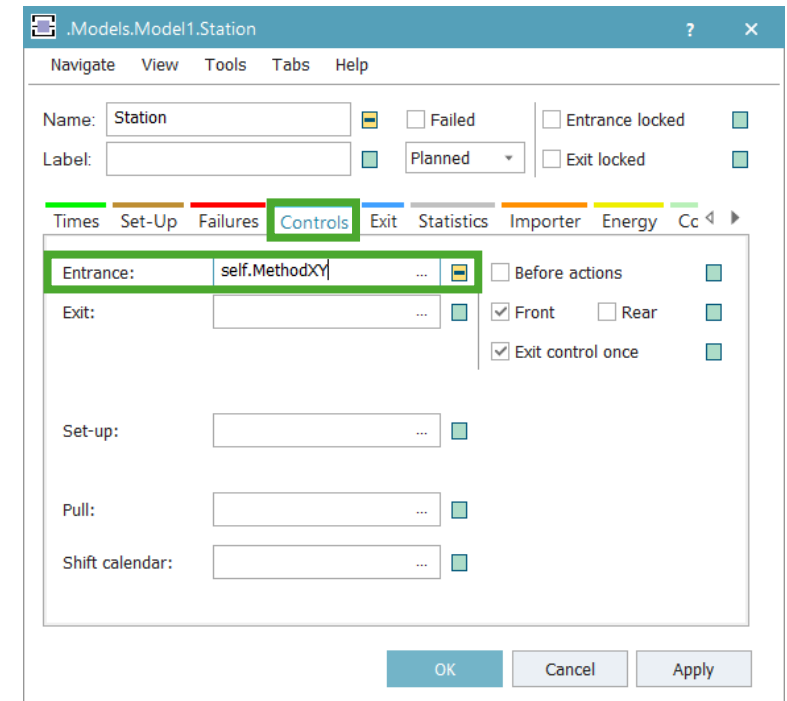
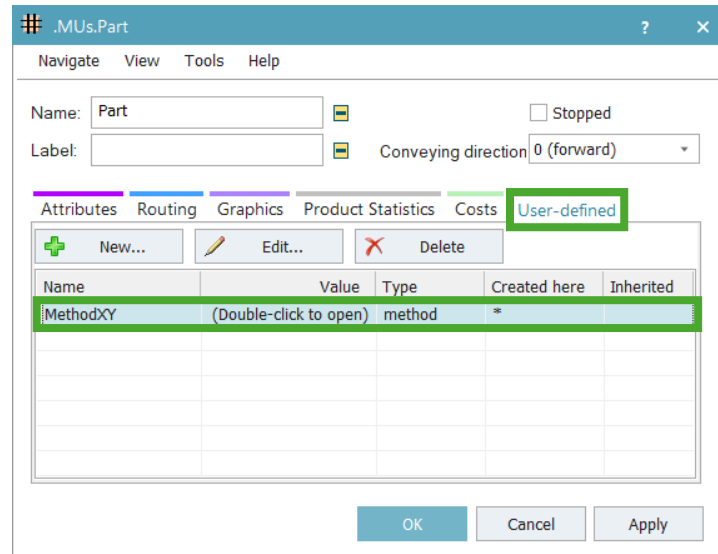
- Method with the name “**Method**” or another name defined by user.
  - It is triggered by the initiator or by another method.
- Method named “**Init**”:
  - It is performed automatically after clicking the button **Starts the simulation.**
- Method named “**EndSim**”:
  - It is performed automatically after the end of the simulation run.
- Method named “**Reset**”:
  - It is performed automatically after clicking the button **Resets the simulation.**
- **Own object’s method.**



# Programming in SimTalk

## Own object's method

- Possibility to create it in tab **User-defined Attributes**:
  - User-defined → New → Data Type → method.
- **Path** to own object's method:
  - **Absolute** “\*.<File>.<Frame>.<Object>.<method name>”.
  - **Relative** “self.<method name>”.



# Programming in SimTalk

## Inserting of the method into the object

- Using Drag&Drop (1).
- Selection from the list (2).

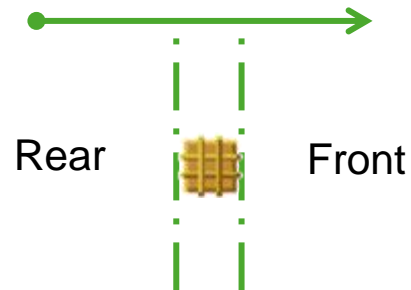
The image shows two windows from the SimTalk software. The main window, titled ".Models.Model1.Station", has a menu bar (Navigate, View, Tools, Tabs, Help) and a toolbar (Times, Set-Up, Failures, Controls, Exit, Statistics, Importer, Energy, Cc). The "Controls" tab is active, showing fields for Entrance, Exit, Set-up, Pull, and Shift calendar, along with checkboxes for "Before actions", "Front", "Rear", and "Exit control once". A green box labeled "1" points to a dropdown menu in the Entrance field, which is labeled "MethodX".

A second window, titled "Select Object", is open. It has a menu bar and a list of objects: BottleneckAnalyzer, MethodX, and TransferStation. A green box labeled "2" points to this list, with a callout box saying "List of all methods in the Frame". Below the list is a "Path:" field containing ".Models.Model1", with a callout box saying "Path". To the right of the list are buttons for "Back", "OK", and "Cancel". A green box labeled "Level up" points to the "Back" button.

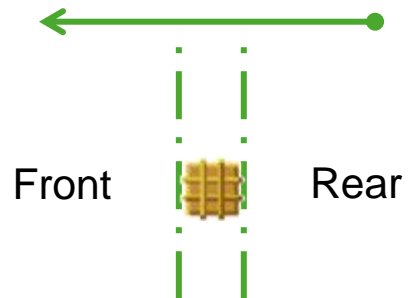
# Programming in SimTalk

## Method initialization

- Method with general **name is usually initialized** by moving unit on the calling object.
- Initialization possibilities of mobile unit (MU):
  - Object movement from **left to right** – forward movement.



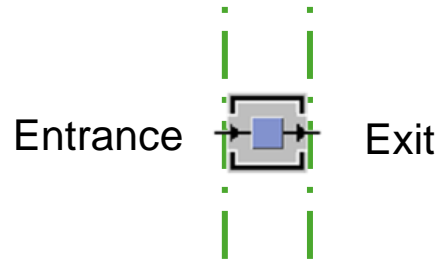
- Object movement from **right to left** – backward movement.



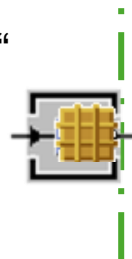
# Programming in SimTalk

## Method initialization

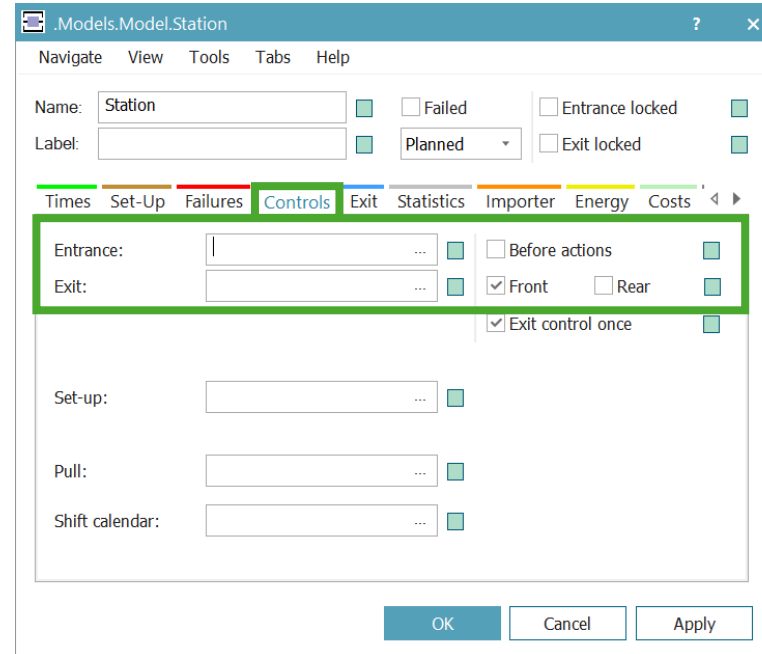
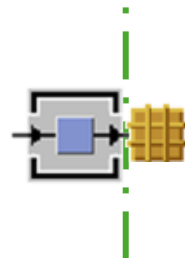
- Initialization on **the process-oriented object**.



- Initialization "Exit Front"



- Initialization "Exit Rear"

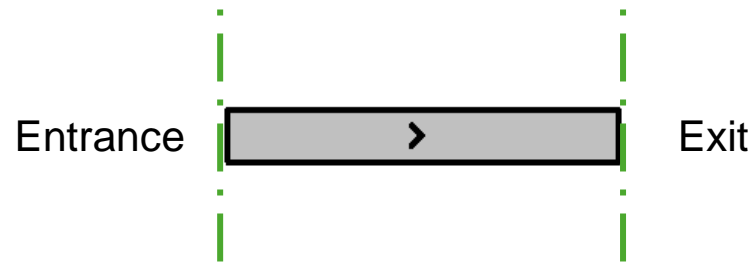




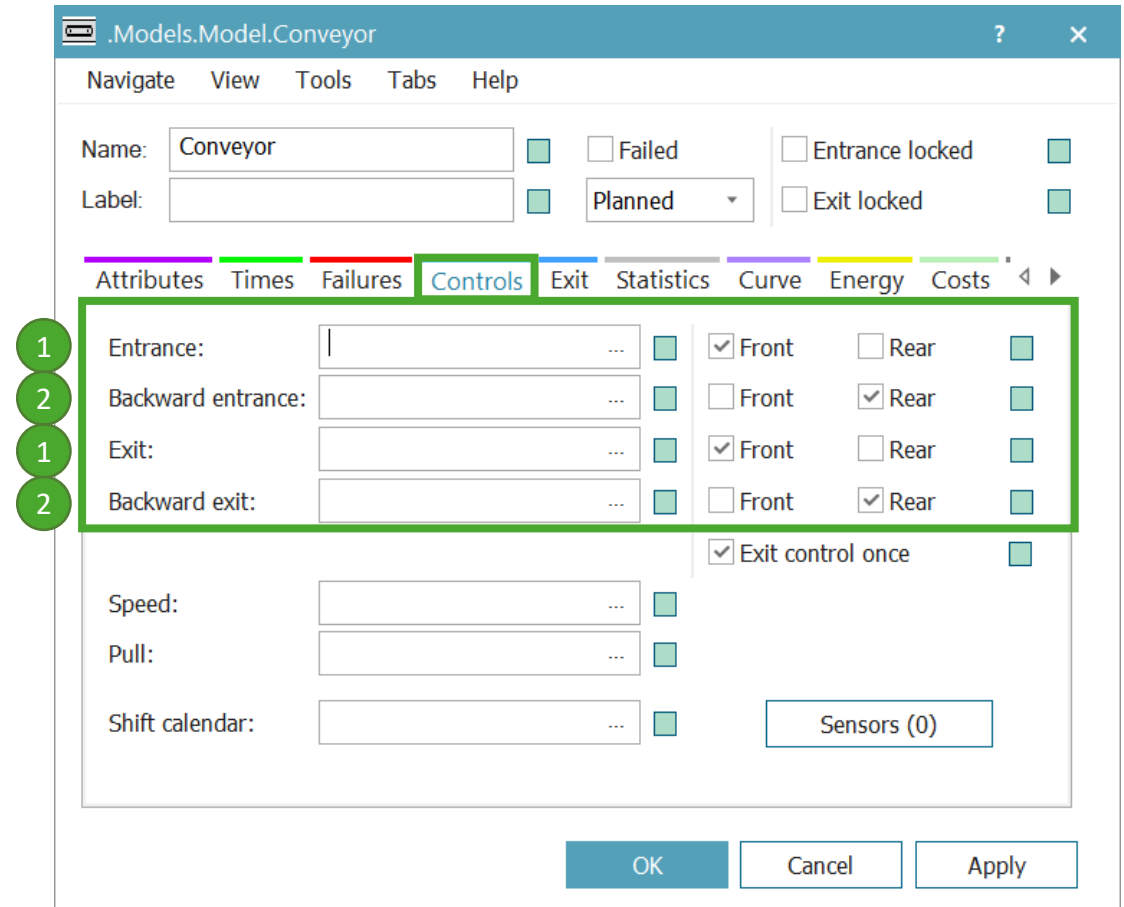
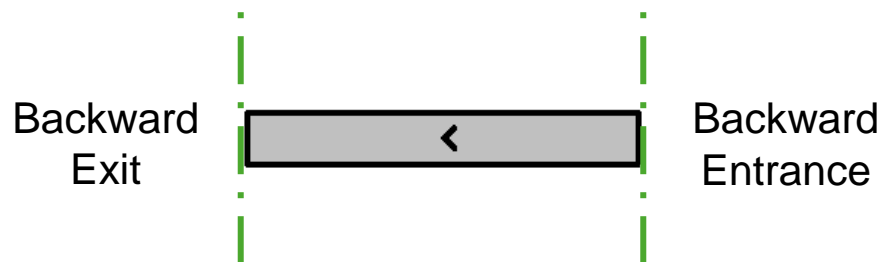
# Programming in SimTalk

## Method initialization

- Initialization on **the length-oriented object**.
  - Regular entrance/exit (1).**



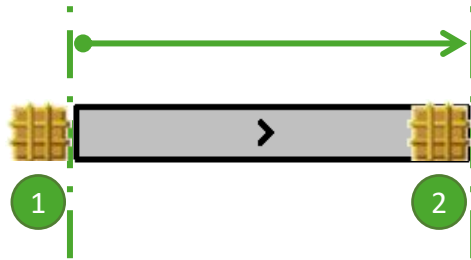
- Reversed entrance/exit (2).**



# Programming in SimTalk

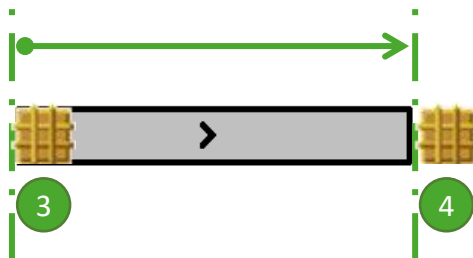
## Method initialization

- Initialization on **the length-oriented object**.
  - Initialization “Entrance Front” (1), “Exit Front” (2).



1	Entrance:	Method	...	<input checked="" type="checkbox"/> Front	<input type="checkbox"/> Rear	<input type="checkbox"/>
2	Exit:	Method	...	<input checked="" type="checkbox"/> Front	<input type="checkbox"/> Rear	<input type="checkbox"/>

- Initialization “Entrance Rear” (3), “Exit Rear” (4).

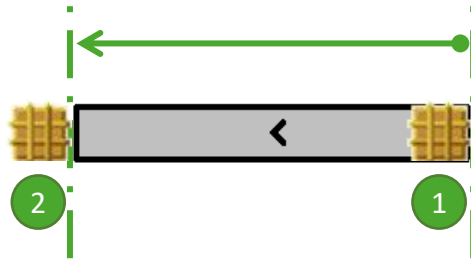


3	Entrance:	Method	...	<input type="checkbox"/> Front	<input checked="" type="checkbox"/> Rear	<input type="checkbox"/>
4	Exit:	Method	...	<input type="checkbox"/> Front	<input checked="" type="checkbox"/> Rear	<input type="checkbox"/>

# Programming in SimTalk

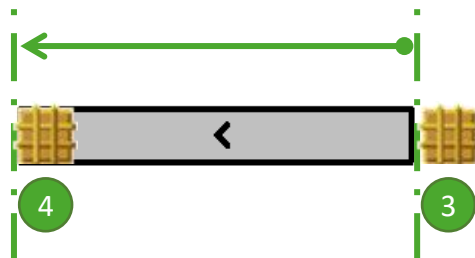
## Method initialization

- Initialization on **the length-oriented object**.
  - Initialization “Backward entrance Front” (1), “Backward exit Front” (2).



1	Backward entrance:	Method	...	<input checked="" type="checkbox"/> Front	<input type="checkbox"/> Rear
2	Backward exit:	Method	...	<input checked="" type="checkbox"/> Front	<input type="checkbox"/> Rear

- Initialization “Backward entrance Rear” (3), “Backward exit Rear” (4).

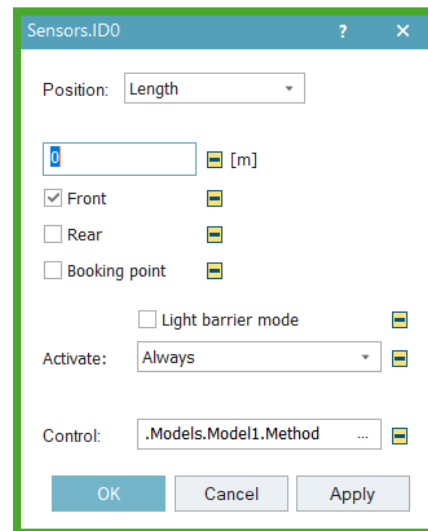
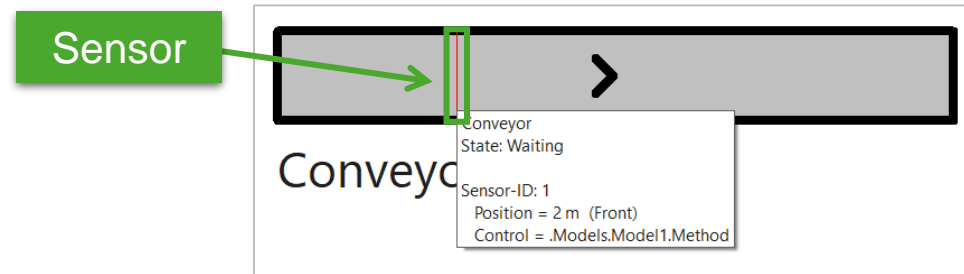


3	Backward entrance:	Method	...	<input type="checkbox"/> Front	<input checked="" type="checkbox"/> Rear
4	Backward exit:	Method	...	<input type="checkbox"/> Front	<input checked="" type="checkbox"/> Rear

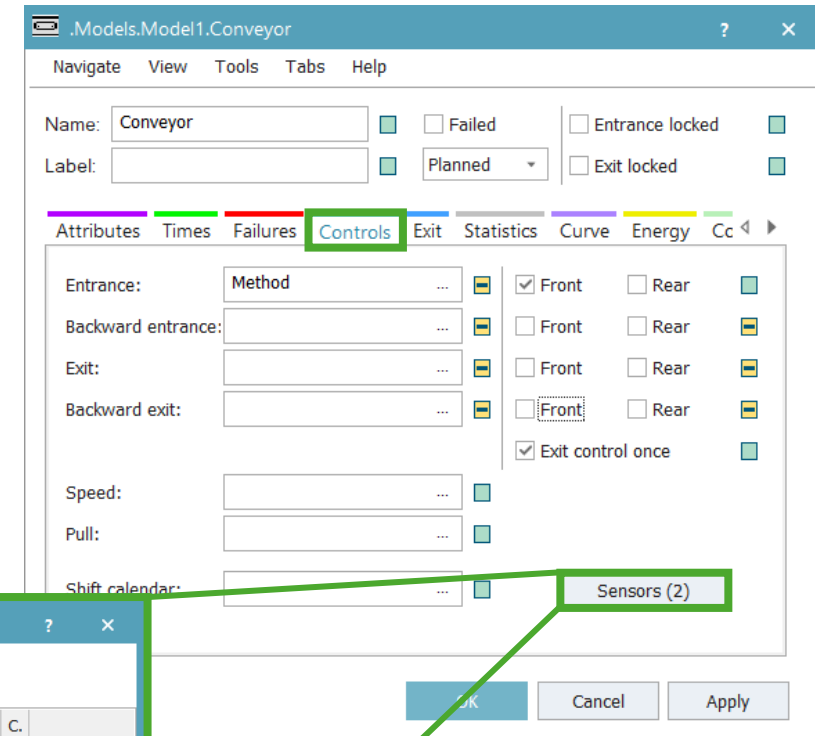
# Programming in SimTalk

## Method initialization

- Initialization on the sensor of length-oriented object.



ID	Position	Front	Rear	Boo...	L.	Path	C.
1	2m	x				.Models.Model1.Method	*
2	50%	x				.Models.Model1.Method	*



# Programming in SimTalk

## Method initialization

- Initialization by another method:
  - By typing the name of the called method as command of the edited method.

```
File Home Debugger Window Edit Tools Find a Command
Import Export Print Find Replace Previous Next Incremental Search Undo Redo Auto Complete Comment
RequiredMethod --name of the method we want to initiate
```

- Initialization of another method by calling object through own method.

```
File Home Debugger Window Edit Tools Find a Command
Import Export Print Find Replace Previous Next Incremental Search Undo Redo Auto Complete Comment
.Models.Model.CallingMethod *
param CallingObject: string --attribute data type assignment
if CallingObject= "Station" then --checking of the calling object
  debug --own command for calling object
end --end of own command
```

Dialog: .Models.Model.Station

Name: Station

Label:

Entrance:

Exit: self.MethodXY

Set-up:

Pull:

Shift calendar:

Buttons: OK, Cancel, Apply

```
File Home Debugger Window Edit Tools Find a Command
Import Export Print Find Replace Previous Next Incremental Search Undo Redo Auto Complete Comment
.Models.Model.MethodXY *
CallingMethod(?,Name) --name of the calling method and calling object
```



# Programming in SimTalk

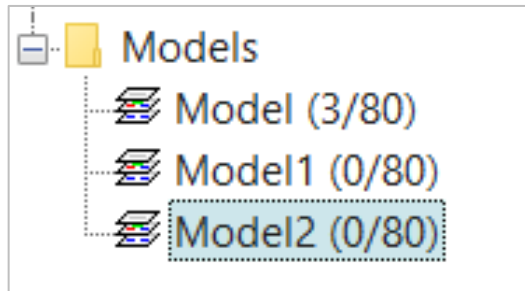
## Value assignment

- `<Object>.<attribute> := <new value>`
- `<Object>`:
  - The object to which we want to assign value.
  - Necessary to define the path to the object:
    - Absolute path (e.g. “.Models.Frame.Station.”).
    - Relative path (e.g. “~.Frame.SingleProc.” or “root.Station.”).
- Referring:
  - Tilde “~” refers on the path one level up in the FRAME hierarchy.
  - Name “root” refers on the top position in the FRAME hierarchy (usually, it is the FRAME with possibility to embed, i.e. with the object “EventController”).

# Programming in SimTalk

## Value assignment

- Example of **using links**



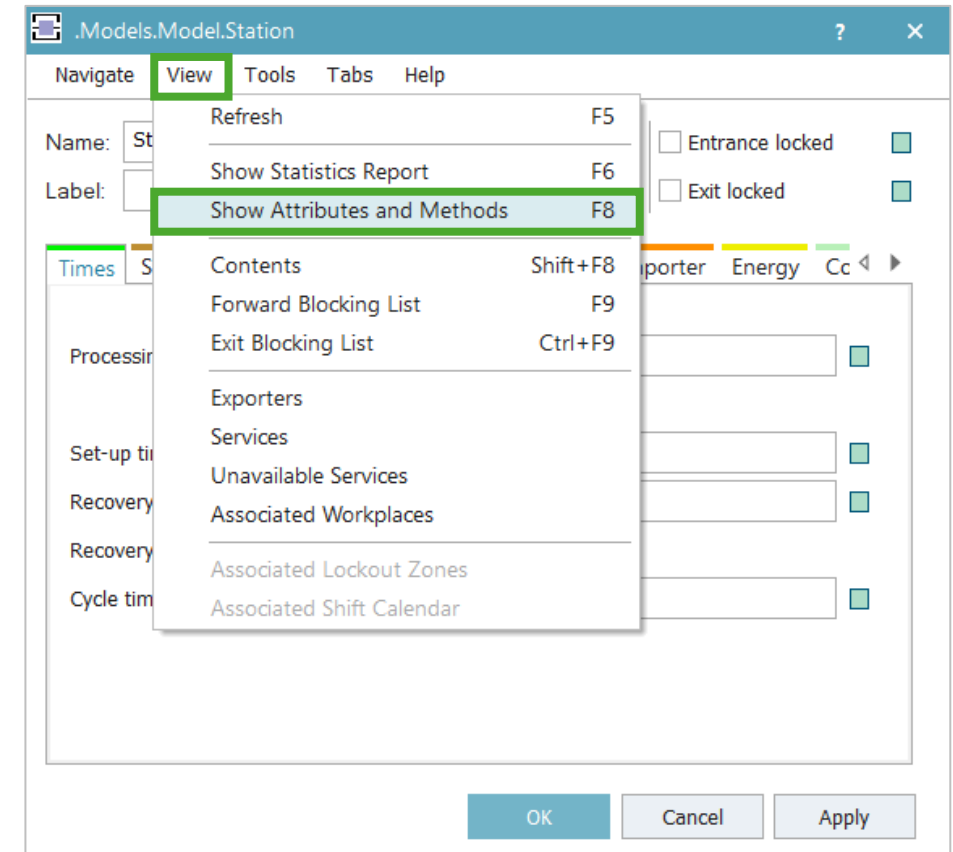
- Command **“print”** and result in **Console window**

```
.Models.Model.Method  
print root      --Models.Model  
print ~         --Models.Model.Model1  
print ~,~       --Models.Model  
print ~,~,~     --Models  
print root,~    --Models  
print root,~,~ --(prázdná cesta)
```

# Programming in SimTalk

## Value assignment

- `<Object>.<attribute> := <new value>`
- `<attribute>`:
  - It controls **behavior** or it represents **status** of objects.
  - Attributes are predefined for each object, their values can be **set** or **gained**.
  - All attributes of the selected item can be **shown** by F8 key (“Object” → View → Show Attributes and Methods).





# Programming in SimTalk

## Display of object's attributes and methods

- Window "Show Attributes and Methods".

**Absolute path to the selected object**

**Toolbox**

**Attribute's name**

**Actual attribute's value**

**Value inheritance YES/NO**

**Possibility to monitor attribute**

**Attribute's data type**

Name	Value	Inherited	Watchable	Signature
AddObserver				(AttributeName:string, ...
AssignedLockoutZones	[ ]			-> array
assignedWorkplaces	[ ]			([Workplaces:table]) ->...
attributeWatchable				(AttributeName:string) -...
AutomaticProcessing	true	✓	*	boolean
AutomaticSetup	true	✓	*	boolean
Availability	100.0			real
Capacity	1	✓	*	integer
ChangePathCtrl	VOID	✓		method
childN				(No:integer) -> object
connectAutomatically				
ConnectCtrl		✓		method
ConstructorCtrl	VOID	✓		method
Cont	VOID		*	-> object
contentsList	[ ]			([Contents:table]) -> any
Coordinate3D	[17.15m, -20.65m, 0m]	-		array
CostingActive	false	✓		boolean
createAttr				(NameOfUserDefinedAtt...
createIcon				([IconName:string, Widt...
CreateIn3D	true	✓		boolean
CurrIcon	"Operational"	✓	*	string



# Programming in SimTalk

## Meaning of some attributes

- **Object's attributes:**
  - Availability – device availability.
  - Capacity – capacity.
  - EntranceOpen – entrance is opened.
  - EntranceLocked – entrance is locked.
  - Pred – object's predecessor.
  - Succ – object's successor.
  - Empty – object is empty.
  - CurrIcon – name of currently used icon.
  - CurrIconNo – number of currently used icon.
  - ExitStrategy – set exit strategy.
  - FailureActive – failure activation.



# Programming in SimTalk

## Meaning of some attributes

- **Times:**
  - ProcTime – processing time.
  - MTTR – mean time to repair.
  - SetupTime – set-up time.
  - ShiftCalendarObject – shift mode activating.
- **Size of displayed object icon and its position:**
  - ZoomX – scale in the x-axis direction.
  - ZoomY – scale in the y-axis direction.
  - XPos – the object's icon position in the window in the x-axis direction.
  - YPos – the object's icon position in the window in the y-axis direction.



# Programming in SimTalk

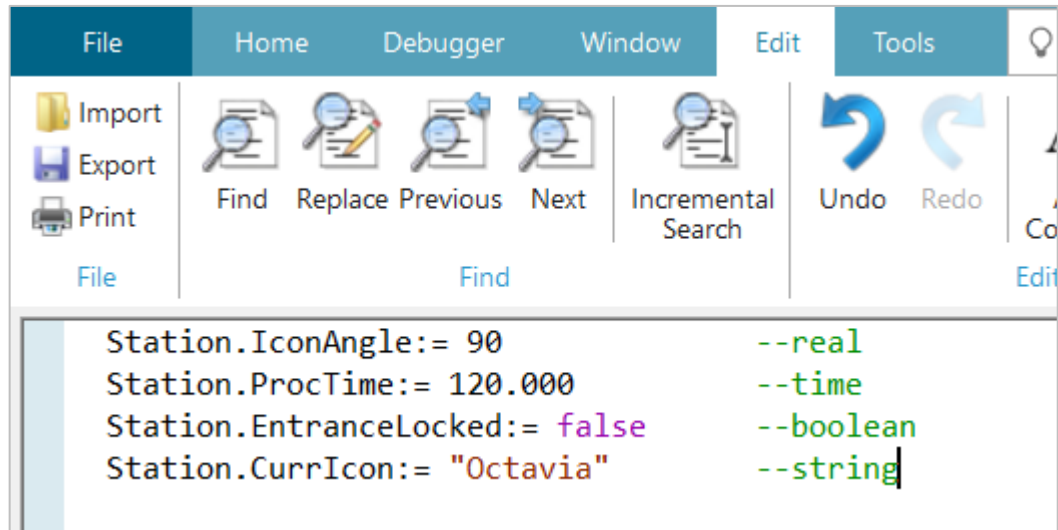
## Meaning of some attributes

- **Path:**
  - Class – absolute path to Class.
  - ~ – absolute path to Frame one level up in hierarchy.
  - RootFrame – absolute path to Frame.
- **Statistics:**
  - Attributes which begin with “stat”, e.g.:
    - statNumIn – number of all MUs, which entered the object.
    - statNumOut – number of all MUs, which exited the object.

# Programming in SimTalk

## Value assignment

- `<Object>.<attribute> := <new value>`
- `<new value>`:
  - User-defined.
  - The value must match the attribute data type.



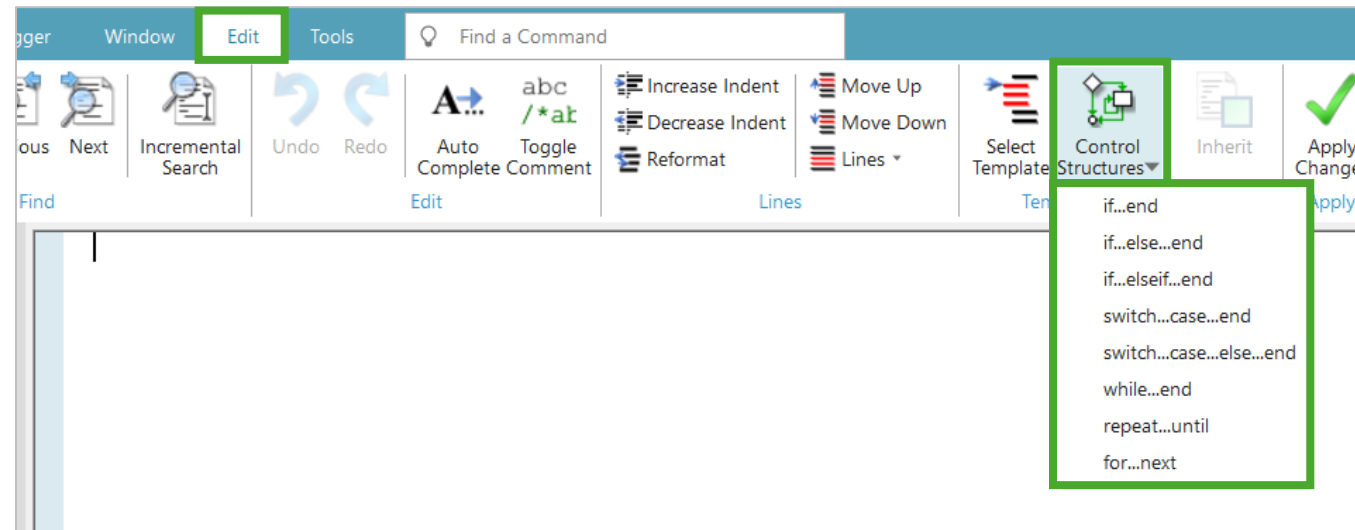
The screenshot shows the SimTalk IDE interface with the following code in the editor:

```
Station.IconAngle:= 90           --real
Station.ProcTime:= 120.000       --time
Station.EntranceLocked:= false  --boolean
Station.CurrIcon:= "Octavia"    --string
```

# Programming in SimTalk

## Conditional commands

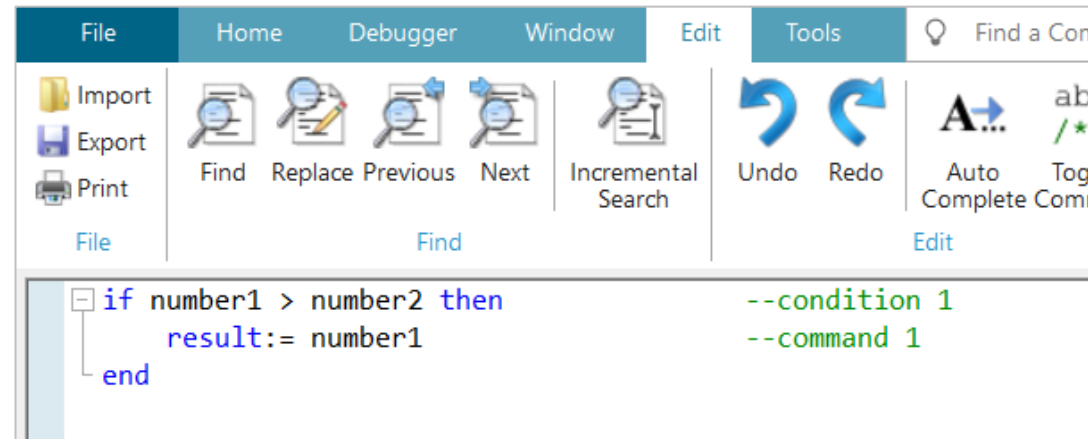
- Commands that are **dependent on conditions**, i.e. they allow to execute required command based on condition/conditions.
- Basic **types of conditional commands**:
  - if ... end
  - if ... else ... end
  - if ... elseif ... end
  - switch ... case ... end
  - switch ... case ... else ... end
  - while ... end
  - repeat ... until
  - for ... next



# Programming in SimTalk

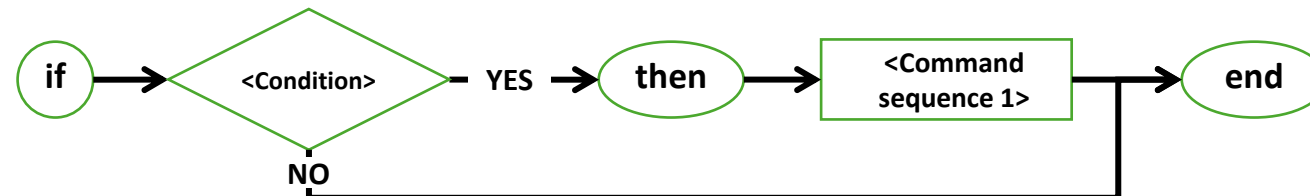
## Conditional commands with “if”

- `if <condition> then`  
    <command>  
`end`



```
if number1 > number2 then --condition 1
  result:= number1      --command 1
end
```

- Syntax diagram:



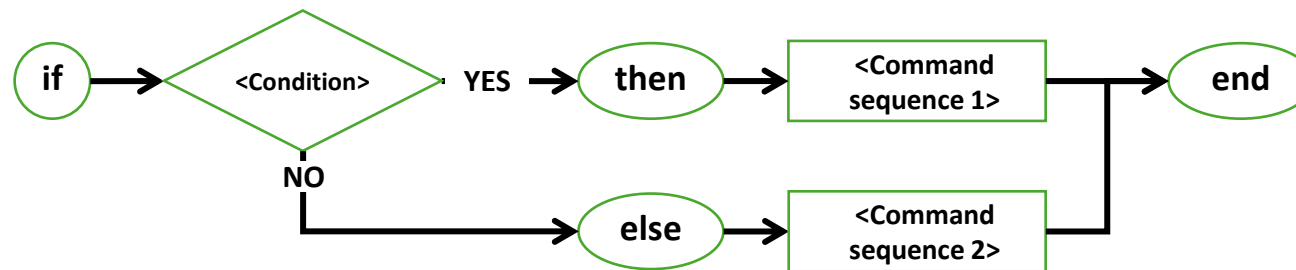
# Programming in SimTalk

## Conditional commands with “if”

- `if <condition> then`  
    `<command1>`  
`else`  
    `<command2>`  
`end`

```
if number1 > number2 then           --condition 1
    result:= number1                 --command 1
else
    result:= number2                 --command 2
end
```

- Syntax diagram:





# Programming in SimTalk

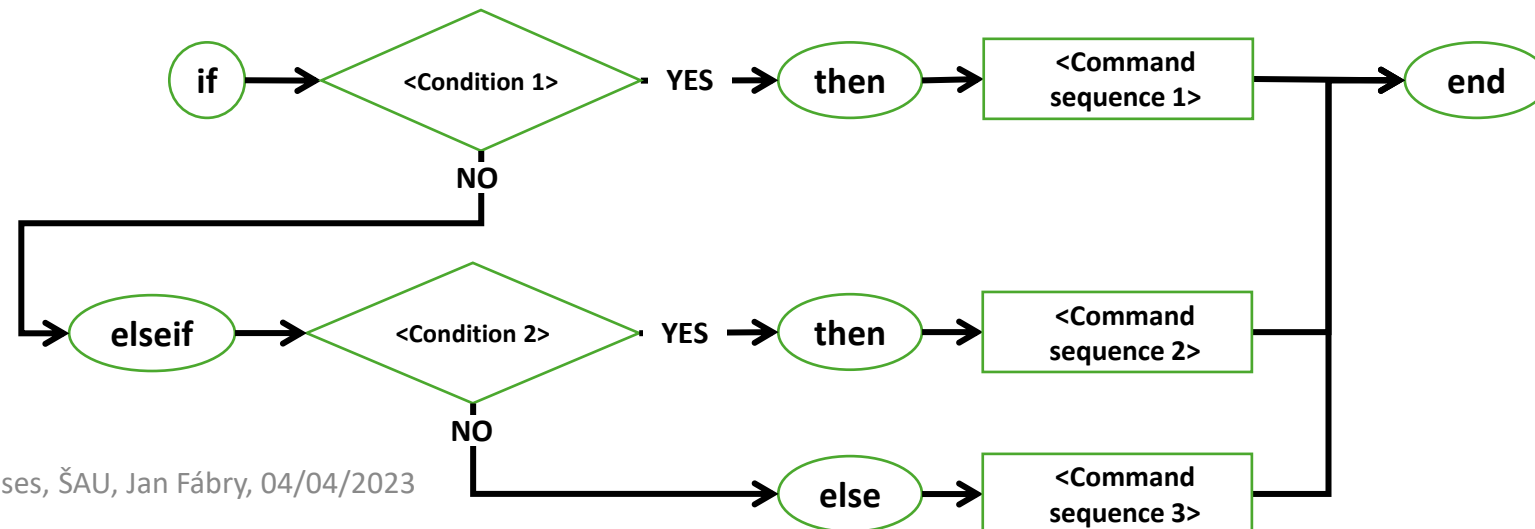
## Conditional commands with “if”

- if** <condition 1> **then**  
 <command1>  
**elseif** <condition 2> **then**  
 <command2>  
**else**  
 <command3>  
**end**

```

File Home Debugger Window Edit Tools Find a Command
Import Find Replace Previous Next Incremental Search Undo Redo Auto Complete Toggle Comment Ince
Export Print File Find Edit
if @.Model = "Fabia" then --condition 1
  FabiaCelkem := FabiaCelkem + 1 --command 1
elseif @.Model = "Yeti" then --condition 2
  YetiCelkem := YetiCelkem + 1 --command 2
else
  debug --command 2 (unknown model)
end
  
```

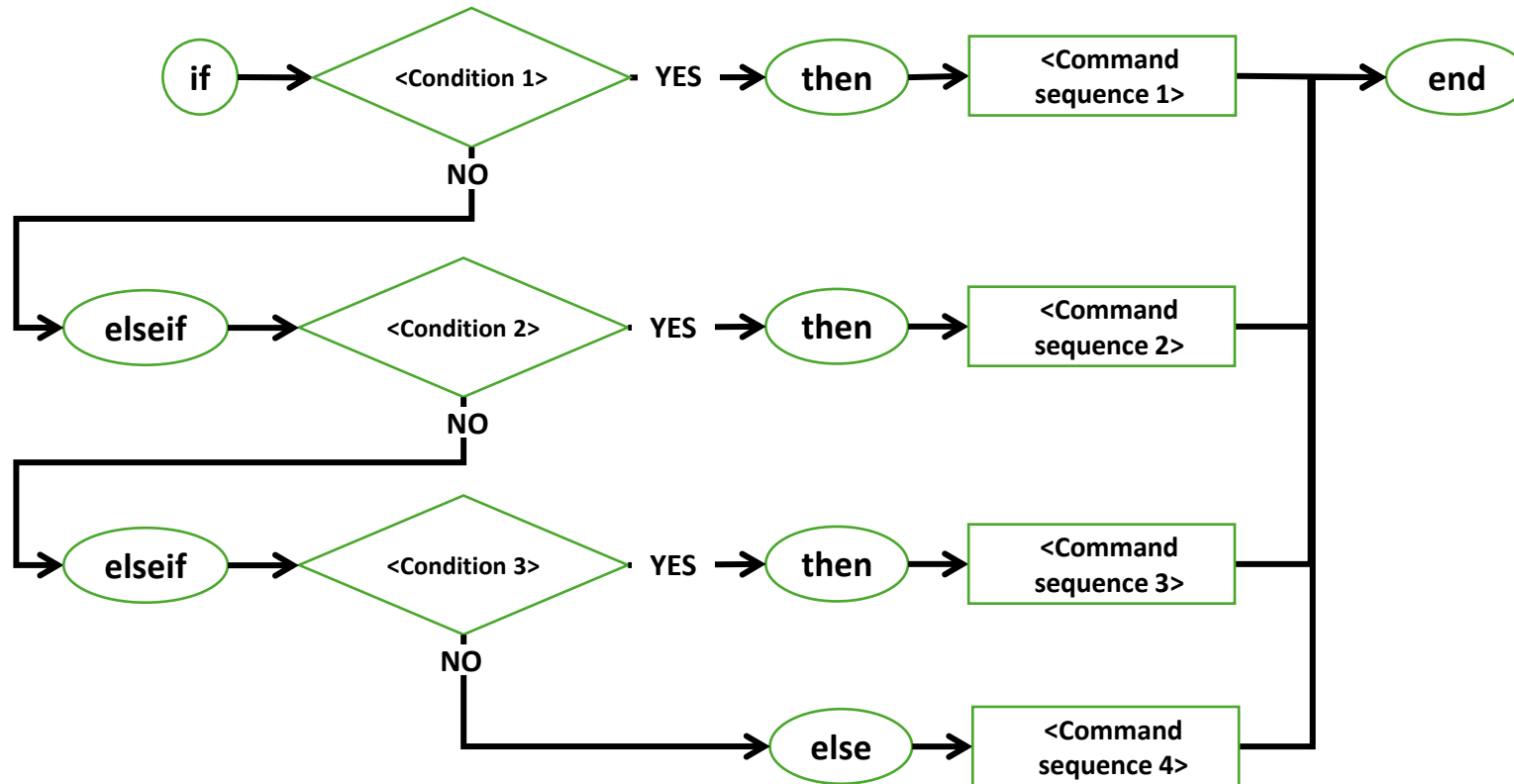
- Syntax diagram:



# Programming in SimTalk

## Conditional commands with “if”

- Syntax diagram:





ŠKODA AUTO University

# Thank you for attention

**Jan Fábry**

Department of Production, Logistics and Quality Management

✉ [fabry@savs.cz](mailto:fabry@savs.cz)

🌐 [www.janfabry.cz](http://www.janfabry.cz)

[www.savs.cz](http://www.savs.cz)