camos.

camos Develop
Developer training

Basics OCL

Prerequisites

- Carconfigurator on the state 3. day modeler training
- Database „OfferData.mdb“
- DSN = „DataCarConfigurator“

- Contents:
- Creating a SQL-statement (tyre details)

Training targets

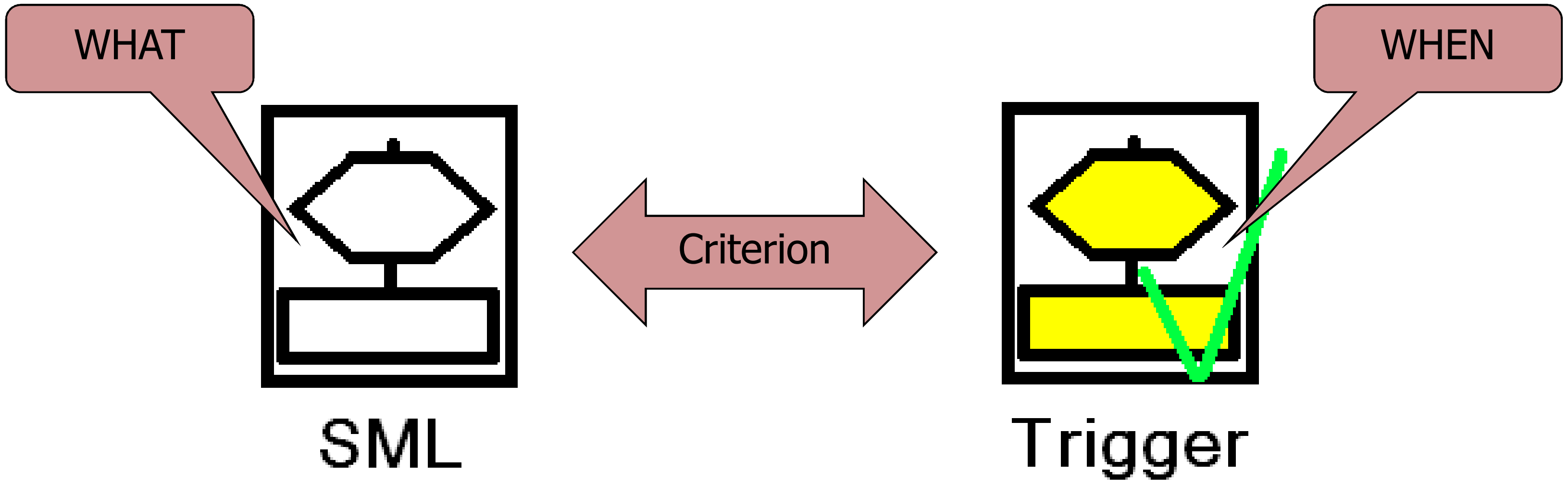
- **After these exercises you should …**
  - Know the advantages of the use of OCLs
  - Define and outprogram OCLs
  - Know when a trigger is firing

OCL

- **Object characteristics list**
  - An OCL determines the selection criteria for standard parts, e.g. screws are classified by length and diameter
  - Advantage: Avoidance of doublets and speeding up the parts search
  - In camos Develop OCL is a trigger mechanism

OCL

- **OCL consists of two parts**
  - The OCL defines the action, e.g. search in database
  - The OCL-trigger controls when this action is executed
  - Communication between OCL and trigger is carried out via criteria



OCL prerequisites

- **OCL-license**
  - Definition of the OCL and OCL-triggers always possible
  - Processing in the KIF only carried out with rule license
    - camos.Configurator, camos.CAP, camos.CAPP
  - Special license only for OCL, but without rules
    - camos.OCL

- **Processing in the interpreter**
  - The procedure code of an OCL is processed if the value of a Wasele that is allocated in the OCL-trigger changes

    -> „Multi-assignment trigger“

OCL Basics

- **OCL …**
  - are defined in the component classes
  - consist of a list of criteria
  - define an action (side effects possible)

- **OCL-trigger …**
  - are defined under assigned component values
  - link the criteria of the OCL to cause variables
  - can be ruled
  - cannot be overloaded

Exercise: Database search

- **Target**
  - The final selection of the tyres is carried out from a database with tyres of a different size
  - Only the tyres can be selected that correspond to the selected tyre width and the rim type
  - The selection is applied to the component tree and the result

- **Usage OCL**
  - For generating the database statement
  - The Where-clause is formed from the lists CriterionName[] and CriterionRange[]

Exercise: Database search

- **Preparations: Connect to database**
  - Create the numerical feature ODBCHnd in the class „start“
  - Create the method DBConnect()

```
# Establish connection to the database
ODBCHnd := SQLConnect('DSN=DataCarConfigurator');
# With unsuccessful Connect -> Display error message
# and return 0
IF ODBCHnd THEN
  RETURN 1;
ELSE
  WinMessage('ERROR', GetLastError());
  RETURN 0;
ENDIF;
```

  - Create the method Delete() in order to close the database connection with exiting the application:

```
SQLDisconnect(ODBCHnd);
```

Exercise: Database Connect

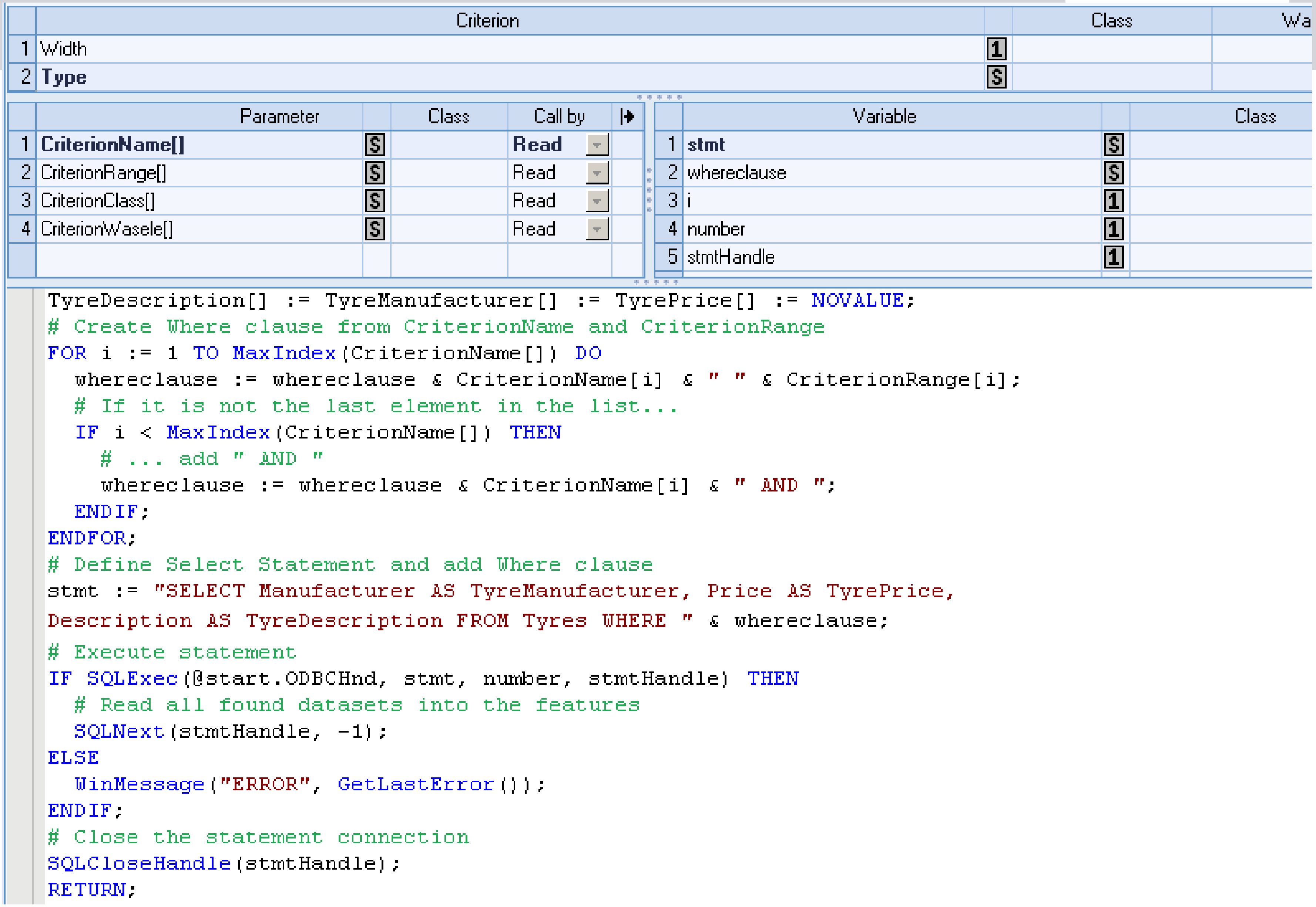
- **Call DBConnect() in new()**
  - The form should only be opened if the Connect to the database was successful:

```
IF DBConnect() THEN
  WinOpen('MainForm');
ENDIF;
```

- **Create the following features and components in the class „Tyres“**
  - TyreDescription[] (string list feature)
  - TyreManufacturer[] (string list feature)
  - TyrePrice[] (currency list feature)
  - @start (predecessor component on „start“)

Exercise: Database search

- **What should happen in the OCL?**
  - The criteria of the OCL are named like the DB-columns whose runtime values are involved in the formulating of the query
  - In a FOR-loop the WHERE-clause is formed from the column names (CriterionName[]) and search criteria (CriterionRange[])
  - The tyre properties that have to be determined do not depend on the transferred criteria, i.e. the SELECT-statement can be formulated fix in the source code
  - The complete SQL-statement (SELECT + WHERE) is executed and the found data records are written to the list features



	Criterion		Class	Wa
1	Width	1		
2	**Type**	S		

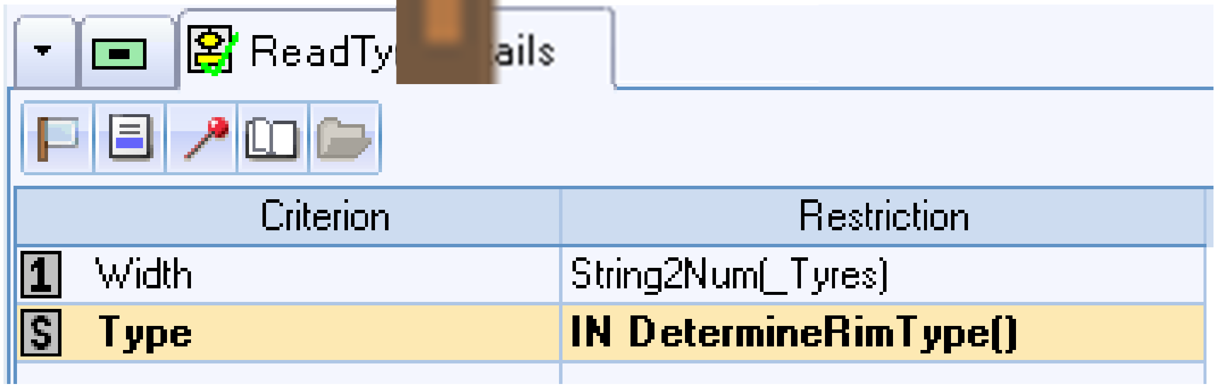
	Parameter		Class	Call by
1	**CriterionName[]**	S		**Read**
2	CriterionRange[]	S		Read
3	CriterionClass[]	S		Read
4	CriterionWasele[]	S		Read

	Variable		Class
1	**stmt**	S	
2	whereclause	S	
3	i	1	
4	number	1	
5	stmtHandle	1	

```
TyreDescription[] := TyreManufacturer[] := TyrePrice[] := NOVALUE;
# Create Where clause from CriterionName and CriterionRange
FOR i := 1 TO MaxIndex(CriterionName[]) DO
  whereclause := whereclause & CriterionName[i] & " " & CriterionRange[i];
  # If it is not the last element in the list...
  IF i < MaxIndex(CriterionName[]) THEN
    # ... add " AND "
    whereclause := whereclause & CriterionName[i] & " AND ";
  ENDIF;
ENDFOR;
# Define Select Statement and add Where clause
stmt := "SELECT Manufacturer AS TyreManufacturer, Price AS TyrePrice,
Description AS TyreDescription FROM Tyres WHERE " & whereclause;
# Execute statement
IF SQLExec(@start.ODBCHnd, stmt, number, stmtHandle) THEN
  # Read all found datasets into the features
  SQLNext(stmtHandle, -1);
ELSE
  WinMessage("ERROR", GetLastError());
ENDIF;
# Close the statement connection
SQLCloseHandle(stmtHandle);
RETURN;
```

Exercise: Database search

- **Create OCL-trigger**
  - In „Wheels": Assign „Tyres" as value to _Tyres
  - Under this you create an OCL-trigger for „ReadTyreDetails"
  - The currently selecte nsferred as restriction to the criterion „Width"
  - The restriction for th is returned by the method DetermineRimType()



	Criterion	Restriction
1	Width	String2Num(_Tyres)
S	**Type**	**IN DetermineRimType()**

Exercise: Database search

- **Create the method DetermineRimType() in Wheels**
  - Disable side effects, return type String list

```
# Column Type in the DB contains the information
# for which rim types a tyre is suitable:
# S=Steel rims, A=Alu rims, AS=Alu- and steel rims
CASE _Rims
  IS 'SteelRims' DO
    RETURN {'S', 'AS'};
  IS 'AlloyRims' DO
    RETURN {'A', 'AS'};
ENDCASE;
```

- **Restrict run of the OCL-trigger**
  - Trigger should not fire if no rims are selected
  - MayNot rule under the OCL-trigger:

```
_Rims = NOVALUE
```

Exercise: Database search

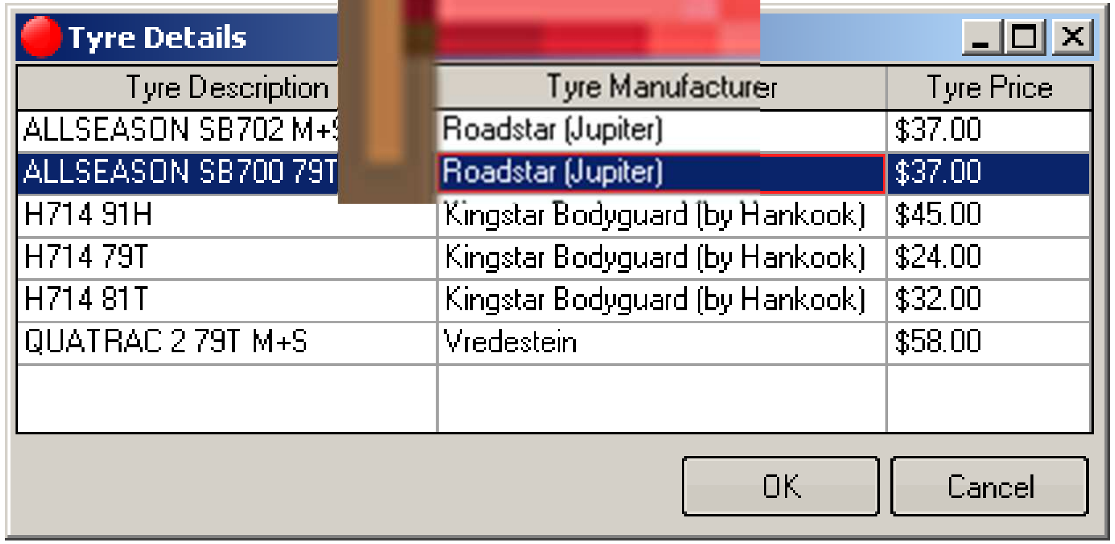
- **Create the form „TyreDetails“ in „Tyres“**
  - Table with two label columns and one currency column
  - Assign TyreDescription[], TyreManufacturer[] and TyrePrice[] as cause variables
  - Deposit column headings
  - Create numerical feature „selTab“
    - Reset SelTab at the beginning of the OCL-procedure
  - Deposit „selTab“ in the field „Selected“ of the table
  - Create the pushbutton „Cancel“

    ```
    WinClose(WinGetHandle());
    ```
  - Create a pushbutton „OK“

Exercise: Database search

- Define the following code in the selection trigger of the OK-button and additionally in the double-click trigger of the table:

```
IF selTab THEN
  SelectionTyres();
  WinClose(WinGetH
ENDIF;
```



Tyre Description	Tyre Manufacturer	Tyre Price
ALLSEASON SB702 M+	Roadstar (Jupiter)	$37.00
ALLSEASON SB700 79T	Roadstar (Jupiter)	$37.00
H714 91H	Kingstar Bodyguard (by Hankook)	$45.00
H714 79T	Kingstar Bodyguard (by Hankook)	$24.00
H714 81T	Kingstar Bodyguard (by Hankook)	$32.00
QUATRAC 2 79T M+S	Vredestein	$58.00

Exercise: Database search

- **Create the method SelectionTyres() in „Tyres“**

```
# Set object naming
ObjSetNaming(°Tyres of° & ' ' & TyreManufacturer[selTab] & ':
   ' & TyreDescription[selTab]);
# Deduct retail price of the tyres from the list price
@Car.ListPrice := @Auto.ListPrice - Price;
# special price * add 4 to list price
Price := TyrePrice[selTab] * 4;
@Car.ListPrice := @Auto.ListPrice + Price;
```

- **Create the method OpenTyreDetails()**

```
WinOpen('TyreDetails');
```

Exercise: Database search

- **Open form with tyre table**
  - Create new menu “TyreContext” in Car
  - Create menu trigger “Tyre Details”
  - Deposit the following expression in the field “Enabled”:

    ```
    _Wheels._Tyres <> NOVALUE and _Wheels._Rims <> NOVALUE
    ```

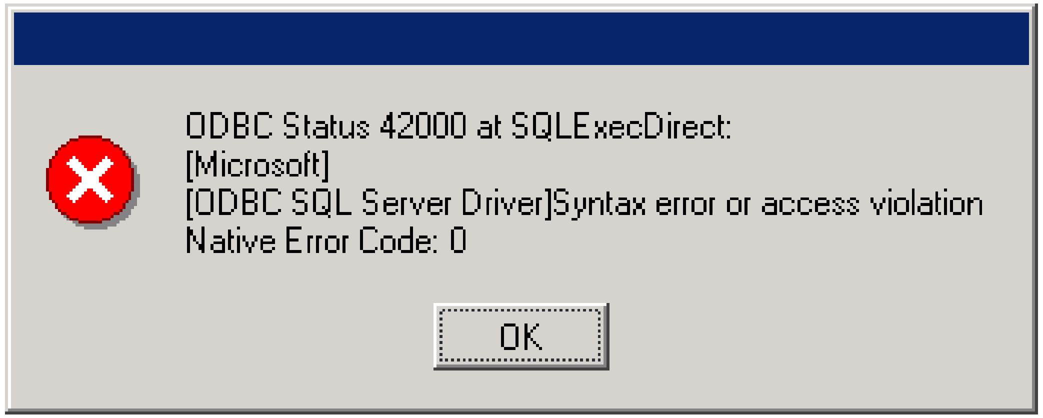
  - Procedure:

    ```
    _Wheels._Tyres.OpenTyreDetails();
    ```

- **Assign context menu**
  - Allocate the menu „TyreContext‟ as popup menu to configbox „_Tyres‟ in the „Detail form‟
  - Don‵t forget the overloaded form in Golf and Passat!

Exercise: Database search

- **Test**
  - Select a tyre and a rim type during runtime
  - -> The OCL-trigger fires and starts the OCL-procedure
  - -> Then a SQL-error is displayed:

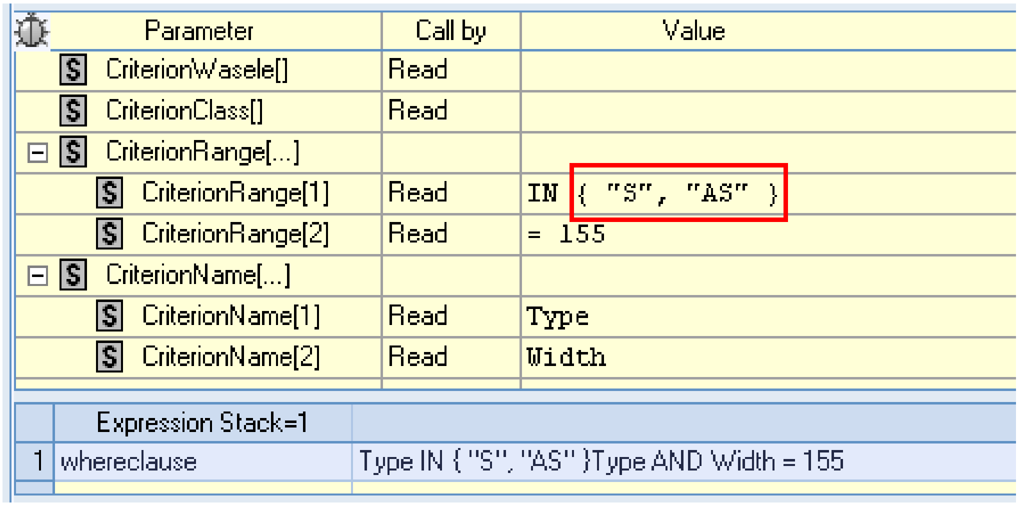


```
ODBC Status 42000 at SQLExecDirect:
[Microsoft]
[ODBC SQL Server Driver]Syntax error or access violation
Native Error Code: 0
```

  - Where is the error?

Exercise: Database error

- **Debugging**
  - Set breakpoint in OCL-procedure
  - Debug generation of the SQL-statement



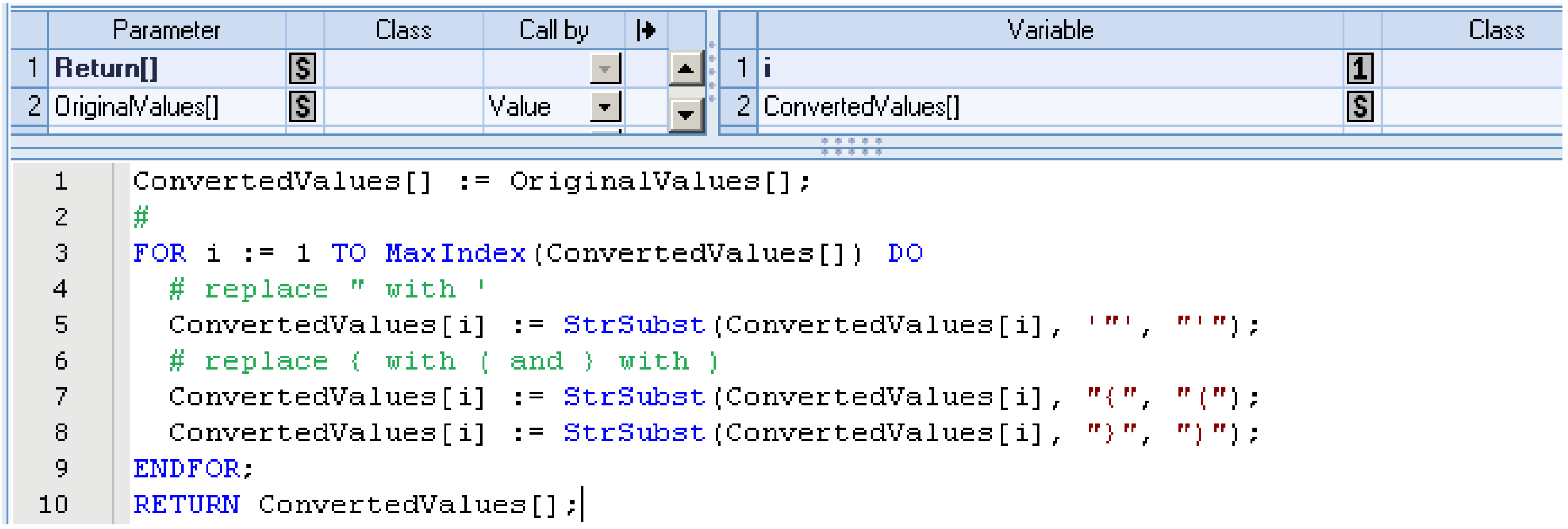
Parameter	Call by	Value
CriterionWasele[]	Read	
CriterionClass[]	Read	
CriterionRange[...]		
CriterionRange[1]	Read	IN { "S", "AS" }
CriterionRange[2]	Read	= 155
CriterionName[...]		
CriterionName[1]	Read	Type
CriterionName[2]	Read	Width

	Expression Stack=1	
1	whereclause	Type IN { "S", "AS" }Type AND Width = 155

- You will notice that the SQL-statement contains double inverted commas and braces
- -> These characters cause the SQL-error

Exercise: Database search

- **Solution**
  - The quotation marks have to be converted to simple inverted commas and the braces to parentheses
  - Create method ConvertValuesForDB() in „Tyres“



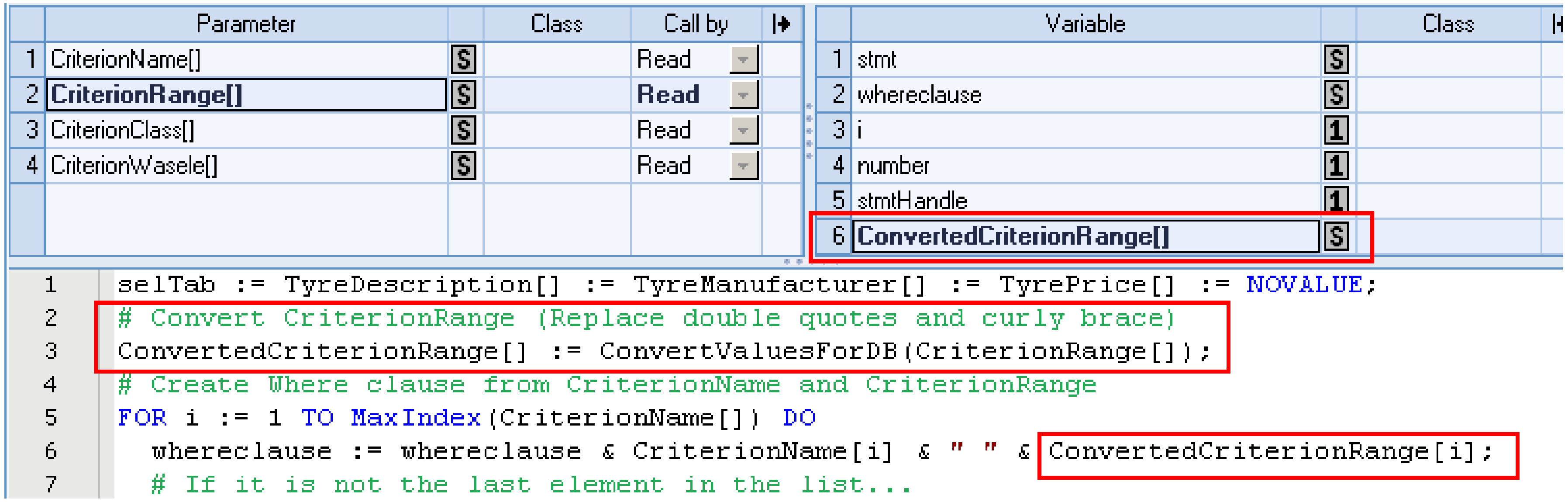
	Parameter		Class	Call by	
1	**Return[]**	S			
2	OriginalValues[]	S		Value	

	Variable		Class
1	i	1	
2	ConvertedValues[]	S	

```
ConvertedValues[] := OriginalValues[];
#
FOR i := 1 TO MaxIndex(ConvertedValues[]) DO
  # replace " with '
  ConvertedValues[i] := StrSubst(ConvertedValues[i], '"', "'");
  # replace { with ( and } with )
  ConvertedValues[i] := StrSubst(ConvertedValues[i], "{", "(");
  ConvertedValues[i] := StrSubst(ConvertedValues[i], "}", ")");
ENDFOR;
RETURN ConvertedValues[];
```

Exercise: Database search

- **Call of the conversion**
  - Now the conversion method has to be called in the OCL-procedure and then you have to work with the converted Range list
  - The changes on the OCL are marked in red:



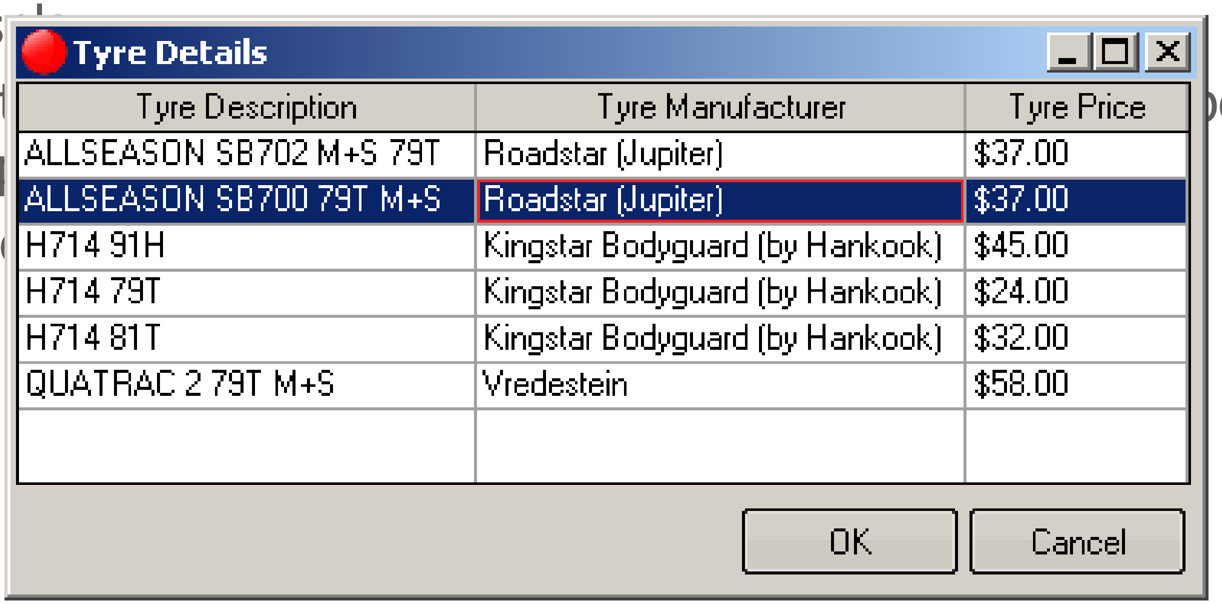
	Parameter		Class	Call by
1	CriterionName[]	S		Read
2	CriterionRange[]	S		Read
3	CriterionClass[]	S		Read
4	CriterionWasele[]	S		Read

	Variable		Class
1	stmt	S	
2	whereclause	S	
3	i	1	
4	number	1	
5	stmtHandle	1	
6	ConvertedCriterionRange[]	S	

```
selTab := TyreDescription[] := TyreManufacturer[] := TyrePrice[] := NOVALUE;
# Convert CriterionRange (Replace double quotes and curly brace)
ConvertedCriterionRange[] := ConvertValuesForDB(CriterionRange[]);
# Create Where clause from CriterionName and CriterionRange
FOR i := 1 TO MaxIndex(CriterionName[]) DO
  whereclause := whereclause & CriterionName[i] & " " & ConvertedCriterionRange[i];
  # If it is not the last element in the list...
```

Exercise: Database search

- **Test**
  - Select again a tyre and a rim type
  - -> The OCL runs, the date records are read out from the database
  - Open the tyre table via the context menu and check the dis
  - Wi e app
  - Th



Tyre Description	Tyre Manufacturer	Tyre Price
ALLSEASON SB702 M+S 79T	Roadstar (Jupiter)	$37.00
ALLSEASON SB700 79T M+S	Roadstar (Jupiter)	$37.00
H714 91H	Kingstar Bodyguard (by Hankook)	$45.00
H714 79T	Kingstar Bodyguard (by Hankook)	$24.00
H714 81T	Kingstar Bodyguard (by Hankook)	$32.00
QUATRAC 2 79T M+S	Vredestein	$58.00