camos.

camos Develop 12.1 Developer training

Basics DKL

Prerequisites and contents

- **Prerequisites**
  - Car configurator on the state 3. day modeling training

- **Target**
  - Dividing the configurator into two knowledge bases

  → Outsourcing the special accessories into an own knowledge base

- **After these exercises you should …**
  - Know the advantages of DKL
  - Be able to use external classes
  - Be able to start slaves in three different types
  - Be able to carry out knowledge base overlapping ruling
  - Be able to display positions of the slave in the result
  - Know which knowledge base is used with the debugging

Dynamic Knowledgebase Loader (DKL)

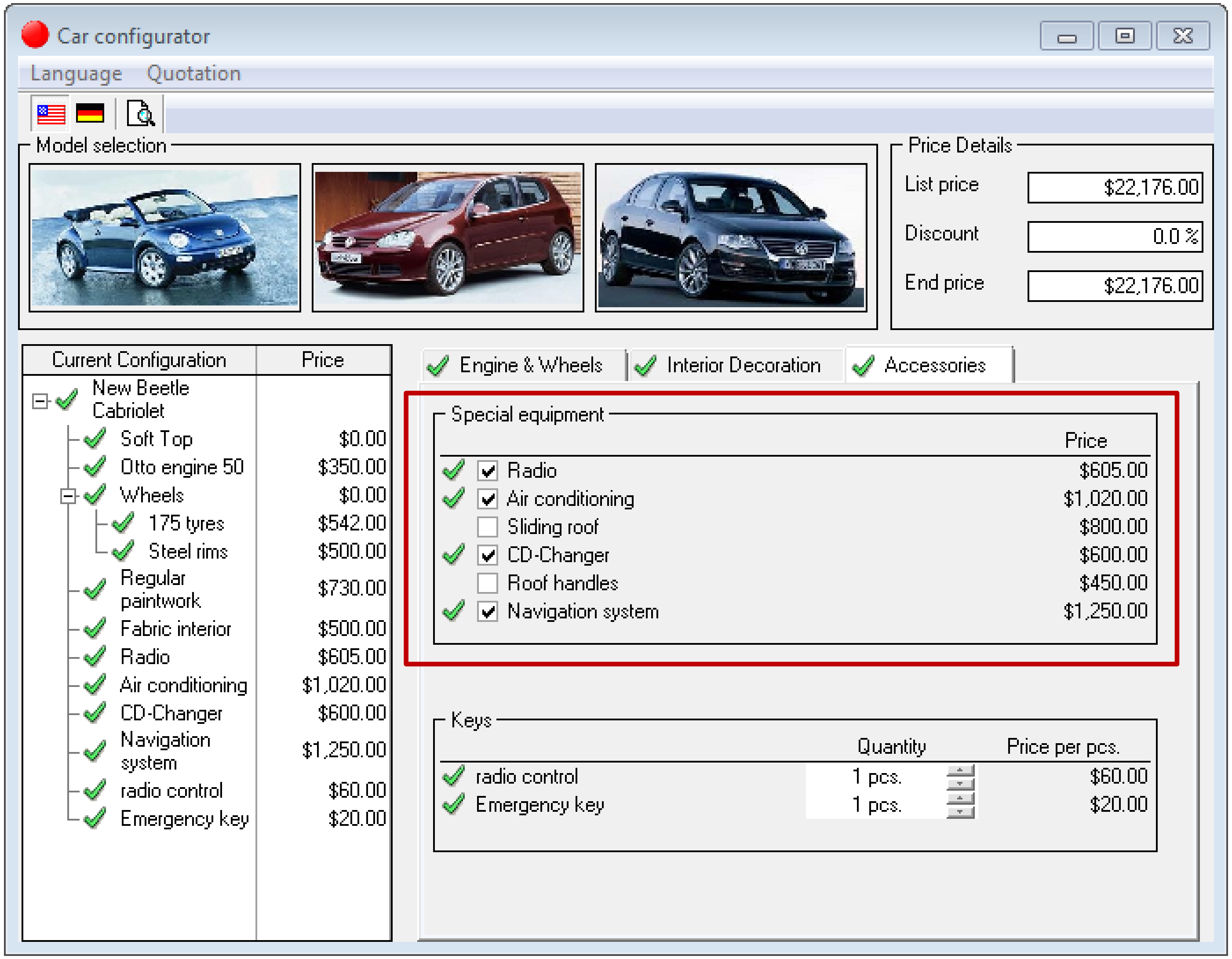
- **Definition:**
  - Dynamic reloading of KIFs in a Runner process

- **Advantages:**
  - Communication between master <-> slave or slave <-> master does not need special functions
  - Debugging is carried out in a common debugger
  - Runtime saving, because the communication is not carried out via the frontend
  - Display in a form element (e.g. component tree) is possible

Dynamic Knowledgebase Loader (DKL)

- **External classes**
  - There are two types of external classes
    - Pure external classes
      - Represent an object in the slave.
      - The slave is started immediately with the creating of an object of a pure external class
    - Hybrid external classes
      - Can represent an object in the master as well as in the slave.
      - I.e. a hybrid external class can be used in a master KNB like a normal object class.
      - An object of the master KNB is created with the creating of an object of a hybrid external class.
      - Via ObjAlter() the object can be converted to an object of a slave KNB.

Target of the training



Price Details

List price	$22,176.00
Discount	0.0 %
End price	$22,176.00

Current Configuration	Price
✓ New Beetle Cabriolet	
✓ Soft Top	$0.00
✓ Otto engine 50	$350.00
✓ Wheels	$0.00
✓ 175 tyres	$542.00
✓ Steel rims	$500.00
✓ Regular paintwork	$730.00
✓ Fabric interior	$500.00
✓ Radio	$605.00
✓ Air conditioning	$1,020.00
✓ CD-Changer	$600.00
✓ Navigation system	$1,250.00
✓ radio control	$60.00
✓ Emergency key	$20.00

✓ Engine & Wheels | ✓ Interior Decoration | ✓ Accessories

Special equipment

		Price
✓	☑ Radio	$605.00
✓	☑ Air conditioning	$1,020.00
	☐ Sliding roof	$800.00
✓	☑ CD-Changer	$600.00
	☐ Roof handles	$450.00
✓	☑ Navigation system	$1,250.00

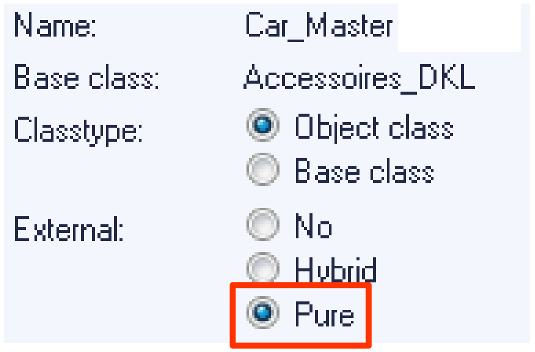
Keys

	Quantity	Price per pcs.
✓ radio control	1 pcs.	$60.00
✓ Emergency key	1 pcs.	$20.00

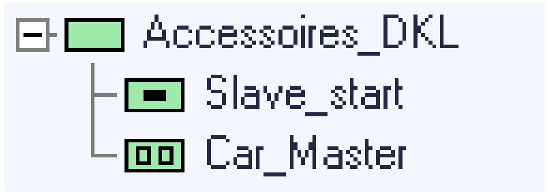
Preparation slave knowledge base

- **Creating a new knowledge base TrainingExample_Slave**
  - The knowledge base has to use the same frame as the TrainingExample KNB.
    - Create base class Accessoires_DKL

    Under this:
    - Object class Slave_start
    - Object class Car_Master
      - Set option External in the class workbench to Pure
      - -> Object class becomes a „pure external class“



Name:	Car_Master
Base class:	Accessoires_DKL
Classtype:	[✓ Object class] Base class
External:	No Hybrid [✓ Pure]

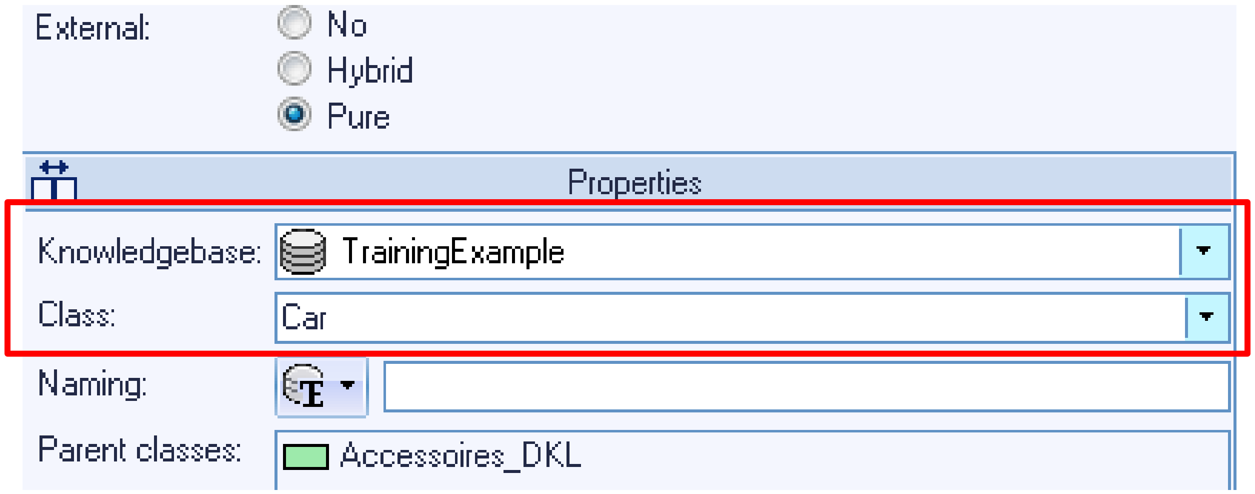


Accessoires_DKL
- Slave_start
- Car_Master

Preparation slave knowledge base

In the external class:

- Select knowledge base: TrainingExample
- Select class: Car


External: No Hybrid [✓ Pure]

Properties

Knowledgebase:	TrainingExample
Class:	Car
Naming:	
Parent classes:	🟩 Accessoires_DKL

Important:

- Here no version of the knowledge base is specified!
- The version has to be defined later!

Preparation slave knowledge base

- **Use of text elements**
  - Problem: The text elements that are used in the master should also be used in the slave knowledge base.

  - Solution:
    - Text elements have to be present in Public

    or

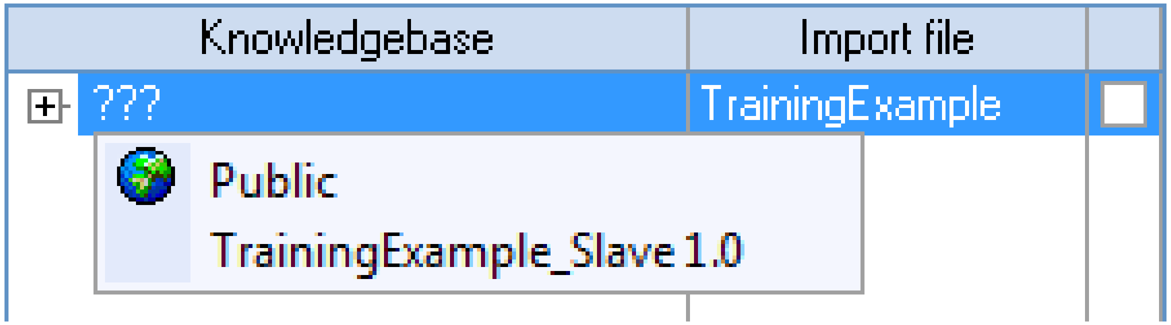
    - have to be present in the master as well as in the slave

  - Solution 2 is carried out via export / import of the text elements.

Preparation slave knowledge base

- Export of the text elements from the KNB TrainingExample
  - Textmanager → Extras → Export…
    - Select export directory
    - Remove flagging on Public
    - OK

- Import to KNB TrainingExample_Slave
  - Textmanager → Extras → Import
    - Select import file
    - With ??? you select TrainingExample_Slave in the context menu



Knowledgebase	Import file	
???	TrainingExample	☐

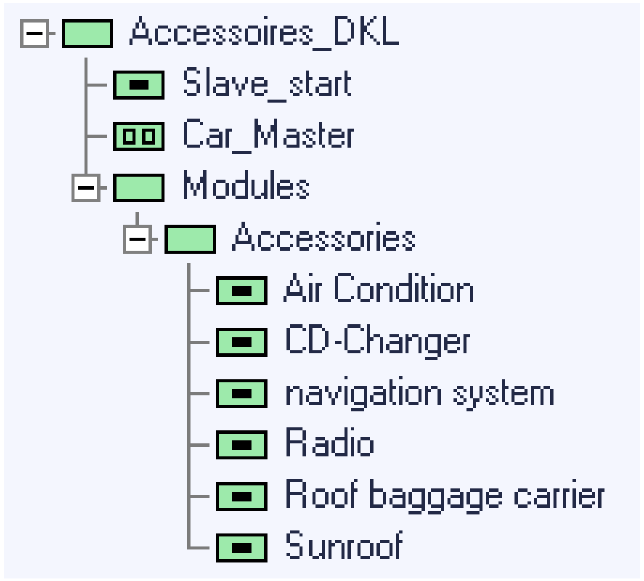
Preparation slave knowledge base

- **In the knowledge base TrainingExample**
  - Copy class Modules
- **In the knowledge base TrainingExample_Slave**
  - Insert copied class under Accessories_DKL

- **In the knowledge base TrainingExample**
  - Copy Accessories with all child classes
    - Context menu „Copy with children“
- **In the knowledge base TrainingExample_Slave**
  - Insert copied classes under Modules

Preparation slave knowledge base

Now the class tree should look like that:



- Accessoires_DKL
  - Slave_start
  - Car_Master
  - Modules
    - Accessories
      - Air Condition
      - CD-Changer
      - navigation system
      - Radio
      - Roof baggage carrier
      - Sunroof

Preparation slave knowledge base

- **In the class Accessories_DKL:**
  - In order to display the slave objects in the component tree (filter expression on the CompTree):
    - Create feature visible
      - Type: numeric
      - Init value: 1

  - Unfolding the branches in the component tree:
    - Create (system-)feature SubTreeOpen
      - Type: numeric
      - Init value: 1

Preparation slave knowledge base

- In the class Slave_start
  - List component _Accessories[]
    - External access = „KNB external full“
    - Apply values incl. NOVALUE to the structure tree
  - Create form „Form“
    - Width: 430
    - Height: 147
  - On this form -> Create configbox Component
    - Cause variable: _Accessories[]
    - X-position: 0
    - Y-position: 0
    - Width: 430
    - Height: 147
    - Display: List

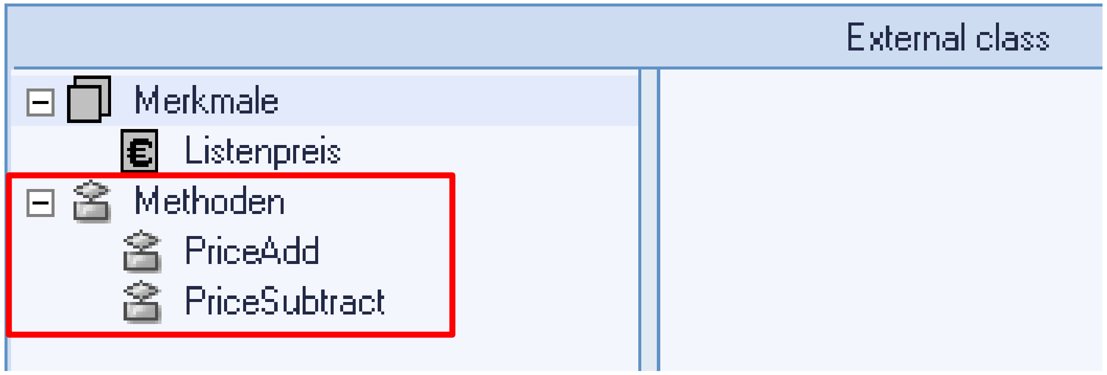
Preparation slave knowledge base

- In the class Modules
    - Structure tree –> Group Components
        - Predecessor @Car
            - Change component class to Car_Master

DKL – pure external class

- **In the master knowledge base TrainingExample**
  - Class Car
    - At methods PriceAdd() and PriceSubstract()
      - Set the property *KNB external*

        If this property is set, both methods are displayed in the area *External class* at the pure external class Car_Master in the slave knowledge base.


External class
Merkmale
Listenpreis
Methoden
PriceAdd
PriceSubtract

DKL – pure external class

- Below Modules
  - Create base class Accessories_DKL
    - Create new form Form
      - Width: 430
      - Height: 147

- Below Accessories_DKL
  - Create object class Accessories_pure
  - Option External = Pure
    - Knowledge base: TrainingExample_Slave
    - Class: Slave_start

DKL – pure external class

- In the class Car
  - Delete component _Accessories[]
  - Create component _Accessories_pure
  - Feature ListPrice: external access = „KNB external full"

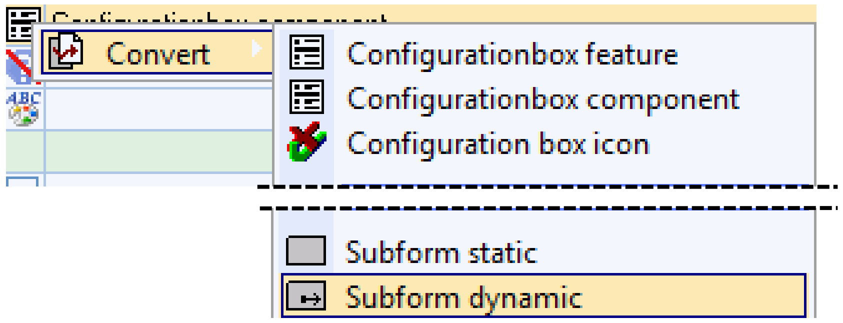
  - DetailForm – Tab page Accessories:
    - Selection trigger:

```
SystemSet('DKL_KNBVersion', ,TrainingExample_Slave', '1.0w');
#
_Accessories_pure := 'Accessories_pure';
RETURN;
```

    - The SystemSet defines which version of the slave knowledge base has to be used.

DKL – pure external class

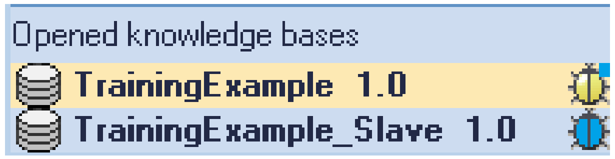
- A dynamic subform is required in order to display the slave
  - Convert configbox Component _Accessories to a dynamic subform



  - Form object: _Accessories_pure
  - Form name: Form

DKL – pure external class

- Test your application
  - Select vehicle Beetle
  - Select index tab Accessories
  - → Slave is started
    Can be recognized by the debug beetle next to the knowledge base TrainingExample_Slave



Opened knowledge bases
TrainingExample 1.0
TrainingExample_Slave 1.0

  - Accessories originates from the slave knowledge base

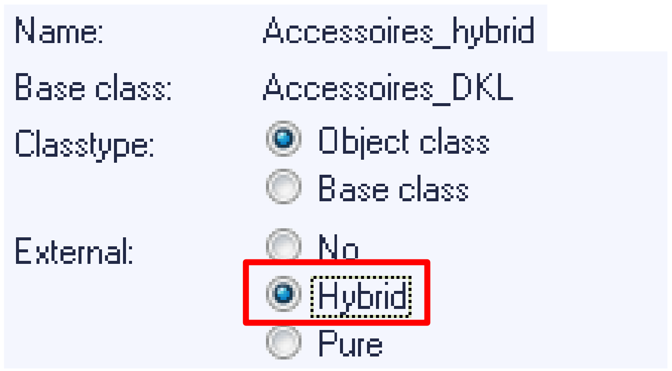
DKL – hybrid external class (ObjAlter())

- **Hybrid external classes**
  - Differ from pure external classes
  - In order to clarify these differences, a hybrid external class is used for the accessories of the Golf

- **Properties of hybrid external classes:**
  - Hybrid external classes can be:
    - Object in the master
    - Object in the slave
  - Hybrid external classes can have local Wasele

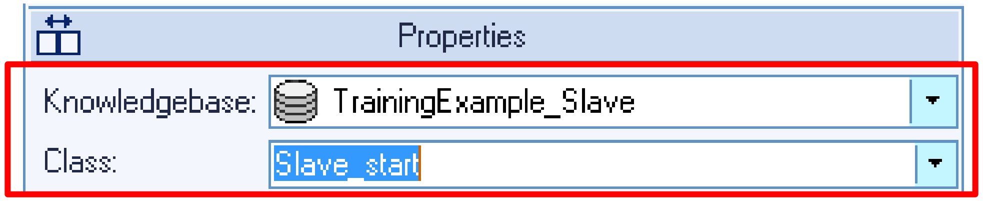
DKL – hybrid external class (ObjAlter())

- **In the KNB TrainingExample under Accessories_DKL**
  - Create object class Accessories_hybrid
    - Property External = hybrid
    - -> Class becomes a „hybrid external class"



Name: Accessoires_hybrid
Base class: Accessoires_DKL
Classtype: [✓ Object class] Base class
External: No [✓ Hybrid] Pure

- Knowledge base: TrainingExample_Slave
- Class: Slave_start



Properties
Knowledgebase: TrainingExample_Slave
Class: Slave_start

DKL – hybrid external class (ObjAlter())

- **In the class Accessories_hybrid:**
  - Create list component _Accessories[]
    - Component class: Accessories
    - Apply all values incl. NOVALUE to the structure tree
  - Overload form Form
    - Create configbox Component

Cause variable:	_Accessories[]
X-position:	0
Y-position:	0
Width:	430
Height:	147
Display:	List

DKL – hybrid external class (ObjAlter())

- **In the class Car**
  - Create component _Accessories_hybrid
    - Component class: Accessories_hybrid

- **In the class start -> Form MainForm**
- Complete selection trigger of !ImageBeetle by

```
_Car._Accessories_hybrid := NOVALUE;
```

- Complete selection trigger of !ImageGolf by

```
_Car._Accessories_hybrid := 'Accessories_hybrid';
```

- Complete selection trigger of !ImagePassat by

```
_Car._Accessories_hybrid := NOVALUE;
```

- → Therefore an object of the hybrid external class is only generated with the Golf = master object

DKL – hybrid external class (ObjAlter())

- **Switch in order to convert the master object to a slave object (and vice versa)**

- **In the class Hardtop**
  - Create feature Slave_start
    - Type: numeric
    - Init value: 0

- **In the class Golf**
  - DetailForm
    - Convert configbox Component _Accessories[] to a dynamic subform
      - Form object: _Accessories_hybrid
      - Form name: Form

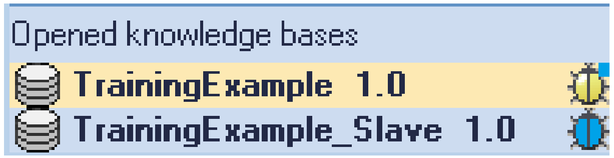
DKL – hybrid external class (ObjAlter())

- Create switchbox „Start/exit slave“
  - Position: under the dynamic subform
  - Text: Start/exit slave
  - Cause variable: Slave_start
  - Value: 1
  - Selection trigger:

```
SystemSet('DKL_KNBVersion', ,TrainingExample_Slave', '1.0w');
CASE Slave_start
  IS 0 DO
    ObjAlter(_Accessories_hybrid, 'INTERNAL');
  IS 1 DO
    ObjAlter(_Accessories_hybrid, 'EXTERNAL');
ENDCASE;
RETURN;
```

DKL – hybrid external class (ObjAlter())

- Test your application
  - Select Golf
  - Select tab page Accessories
  - Slave is not yet started => Accessories can still be selected
    → Master object of the hybrid external class
  - Enabling the switchbox „Start/exit slave"
  - Object of the master is converted to an object of the slave
    → Slave object of the hybrid external class
    → can be recognized by the beetle next to TrainingExample_Slave



Opened knowledge bases
TrainingExample 1.0
TrainingExample_Slave 1.0

  - Accessories that are selected in the master are applied to the slave and vice versa.

DKL – Component (ComponentCreate())

- **Without external classes:**
  - An object of a slave KNB can also be generated via the function ComponentCreate()
  - The knowledge base name and the class name are transferred to the function
  - The version of the KNB is also defined via SystemSet(„DKL_KNBVersion“)

- **In the class Car:**
  - Create component _Accessories_DKL
    - Component class: Accessories_DKL

DKL – Component (ComponentCreate())

- **In the class Passat -> DetailForm**
  - Tab page Accessories:
    - Selection trigger:

```
SystemSet('DKL_KNBVersion', 'TrainingExample_Slave', '1.0w');
#
IF not _Accessories_DKL THEN
  ComponentCreate(_Accessories_DKL, 'TrainingExample_Slave',
                  'Slave_start');
ENDIF;
RETURN;
```

    - Convert configbox Component _Accessories[] to a dynamic subform
      - Form object: _Accessories_DKL
      - Form name: Form
  - Test the application

DKL – Hiding the DKL-classes

- **Hiding the class Accessories_hybrid in the component tree**
  - Realization via skip-filter on the component tree:
    - In the knowledge base TrainingExample -> Class Configuration:
      - Create numeric feature Skip
      - Init value: 0
    - In the class Accessories_DKL:
      - Overload init value of Skip and set to 1
    - In the class start -> MainForm:
    - In the component tree:
        - Skip filter: Skip

DKL – Hiding the DKL-classes

- **Hiding the start class „Slave_start“ of the slave in the component tree**
  - In the knowledge base TrainingExample_Slave
    - In the class Accessories_DKL:
      - Create numeric feature Skip
      - Init value: 0
    - In the class Slave_start:
      - Overload init value of Skip and set to 1

DKL – Ruling in the slave

- **Special accessories should be faded in or hidden, depending on the selected vehicle**
  - Sunroof, roof baggage carrier and navigation system should be hidden with the Beetle

- **In order to be able to access the Car (in the master) out of the slave:**
  - Knowledge base TrainingExample:
    - Class start -> Component _Car:
      - Property „External access“ = KNB external full

DKL – Ruling in the slave

- In the knowledge base TrainingExample_Slave
  - Below the class Accessories_DKL
    Create pure external class start_Master
    - Knowledge base: TrainingExample
    - Class: start

  - In the class Slave_start
    - Create predecessor @start_Master
    - Enable „Rules allowed" on component _Accessories[]

    - New InvisibleOnly rule on the value Sunroof
    - In the rule editor – New expression:
      ```
      @start_Master._Car = 'Beetle'
      ```

DKL – Ruling in the slave

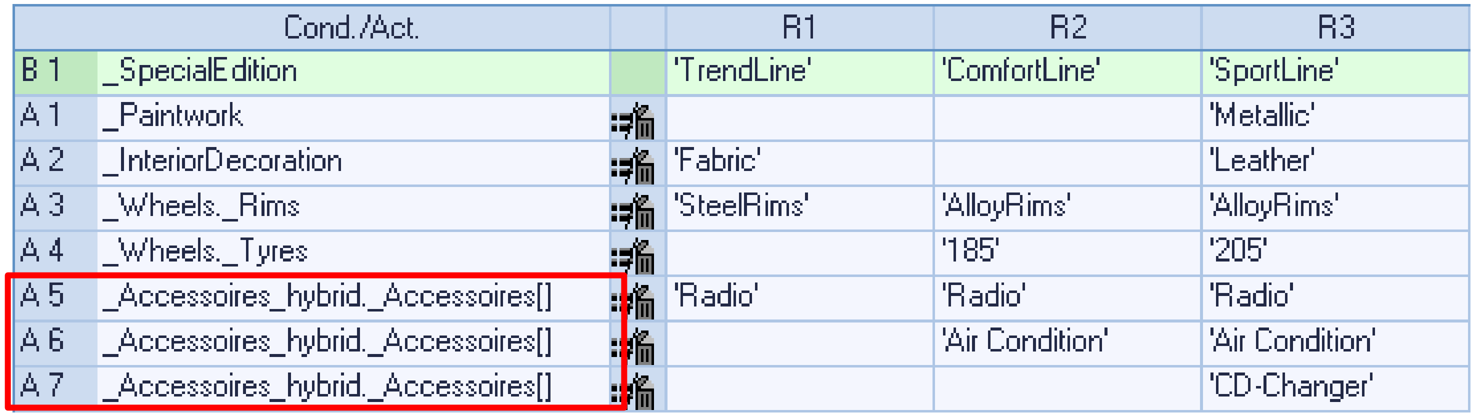
- InvisibleOnly rules with the same condition also on:
  - Roof baggage carrier
  - Navigation system

⚠ In the knowledge base TrainingExample:

- Class start → Component _Car
  - Enable property „Check relevant"
- Class Car → Component _Accessories_hybrid
  - Enable property „Check relevant"
- Class Accessories_hybrid → Component _Accessories[]
  - Enable property „Rules enabled"

DKL – Ruling in the slave

- Class Golf → Decision table „Special models"
  - Change the lines „_SpecialAccessories[] in the column „Cond./Act." to „**_Accessories_hybrid.**_Accessories[]"



Cond./Act.			R1	R2	R3
B 1	_SpecialEdition		'TrendLine'	'ComfortLine'	'SportLine'
A 1	_Paintwork				'Metallic'
A 2	_InteriorDecoration		'Fabric'		'Leather'
A 3	_Wheels._Rims		'SteelRims'	'AlloyRims'	'AlloyRims'
A 4	_Wheels._Tyres			'185'	'205'
A 5	_Accessoires_hybrid._Accessoires[]		'Radio'	'Radio'	'Radio'
A 6	_Accessoires_hybrid._Accessoires[]			'Air Condition'	'Air Condition'
A 7	_Accessoires_hybrid._Accessoires[]				'CD-Changer'

DKL – Ruling in the slave

- Further rules (in the master as well as in the slave)
  - Roof baggage carrier should be hidden with Golf
  - If CD changer is selected, then also radio has to be selected
  - Navigation system can only be selected if radio is selected

⚠ Rules in the master only in the class Accessories_hybrid
  - First a predecessor component on the class start has to be created in the class Accessories_hybrid

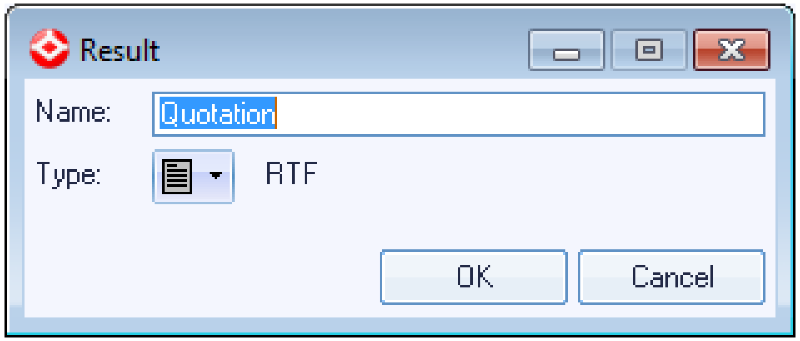
DKL – Ruling in the slave

- In the knowledge base TrainingExample
  - Class Car
    - Cut constraint „Accessories“

- In the knowledge base TrainingExample_Slave
  - Class Slave_start
    - Insert constraint „Accessories“

DKL – Display of the slave positions in the result

- **The selected accessories should be displayed in the result**
  - In the knowledge base TrainingExample_Slave
    - Create result Offer in the KNB properties
      - View Lock results
      - Create new result
        - Name: Quotation
        - Type: RTF

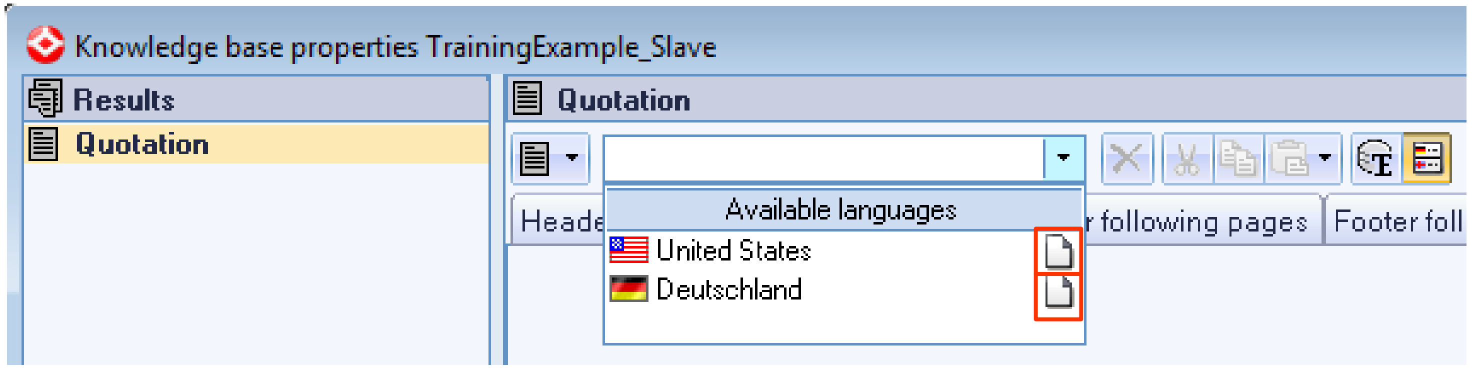


Name: Quotation

Type: RTF

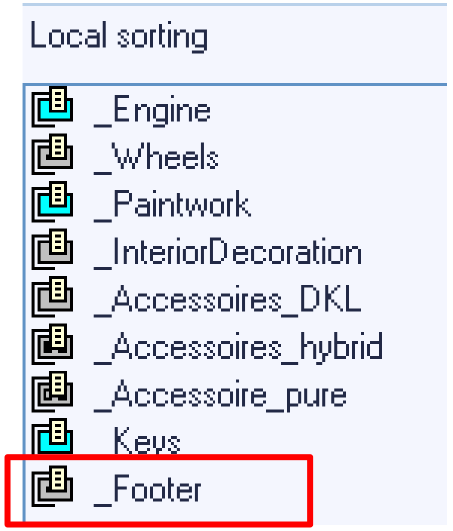
DKL – Display of the slave positions in the result

- **The selected accessories should be displayed in the result**
  - Create result in the following languages:
    - English (United States)
    - German (Deutschland)


Knowledge base properties TrainingExample_Slave

DKL – Display of the slave positions in the result

- **The selected accessories should be displayed in the result**
  - Knowledge base TrainingExample -> Class Car:
    - On _Accessories_pure, _Accessories_hybrid and _Accessories_DKL
      - Enable „Output to result/printout form“ and
      - „Sort“
    - Local sorting:
      Drag _Footer to last position


Local sorting
_Engine
_Wheels
_Paintwork
_InteriorDecoration
_Accessoires_DKL
_Accessoires_hybrid
_Accessoire_pure
_Keys
_Footer

⚠ Possibly the local sorting also has to be dragged in the car models later

DKL – Display of the slave positions in the result

- In the class Accessories_hybrid:
  - Enable the property „Output to result/printout form" on the component _Accessories[]
  - Overload the constant !Tablerow and remove contents
- In the knowledge base TrainingExample_Slave
  - In Slave_start:
    - On component _Accessories[]
      Enable the property „ Output to result/printout form"
- Test the application

DKL – Tips

- **Which knowledge base version is used with the debugging?**
  - Via SystemSet('DKL_KNBVersion') e.g. version ‚1.0w' is specified:
    - Slave knowledge base version 1.0 is opened in Develop
      → This version is used to load the slave
    - Slave knowledge base is not opened / opened in another version
      → A KIF of the KNB is used to load the slave.
  - Via SystemSet('DKL_KNBVersion') version ‚w' is specified as version:
    - Several versions of the slave knowledge base are opened in Develop
      → The last opened slave knowledge base is used
    - Slave knowledge base is not opened
      → The KIF with the highest work version of the knowledge base is used

DKL – Tips

- **Which version of the slave knowledge base is used with the syntax check?**
  - If several versions of the slave knowledge base are present in the project, then the highest version is used for the syntax check.