ABAP Objects

Adrian Streitz, Johannes Rank, Borys Levkovskyi

Prerequisites

- Basic experience with the ABAP Workbench and navigation in Eclipse
- Basic knowledge in one programming language

Agenda

I. Introduction

II. Principles of object-oriented programming in ABAP

III. Global classes – introduction to class builder

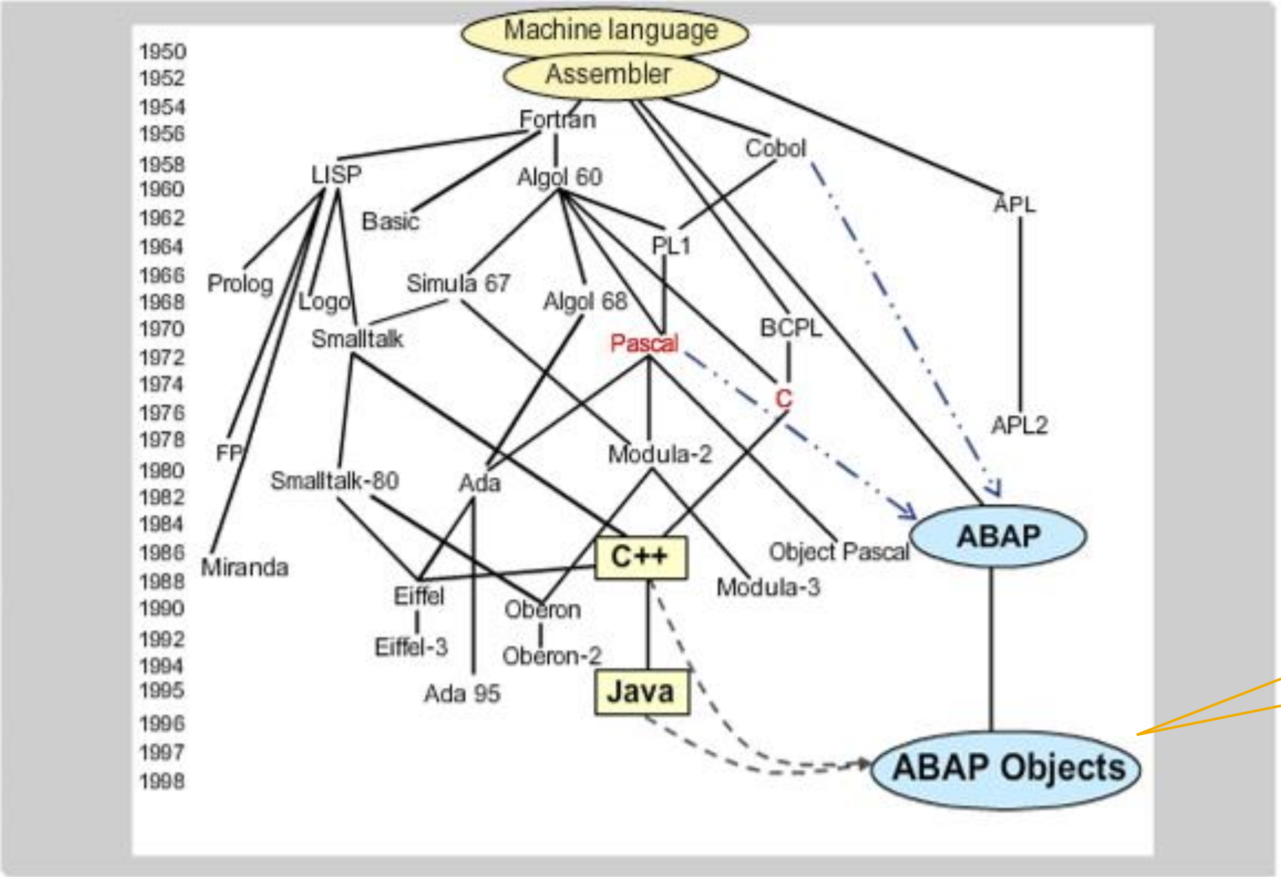
IV. Further principles of object-oriented programming

V. Applications of object-oriented programming

VI. Check your knowledge

I. Introduction

History of selected programming languages



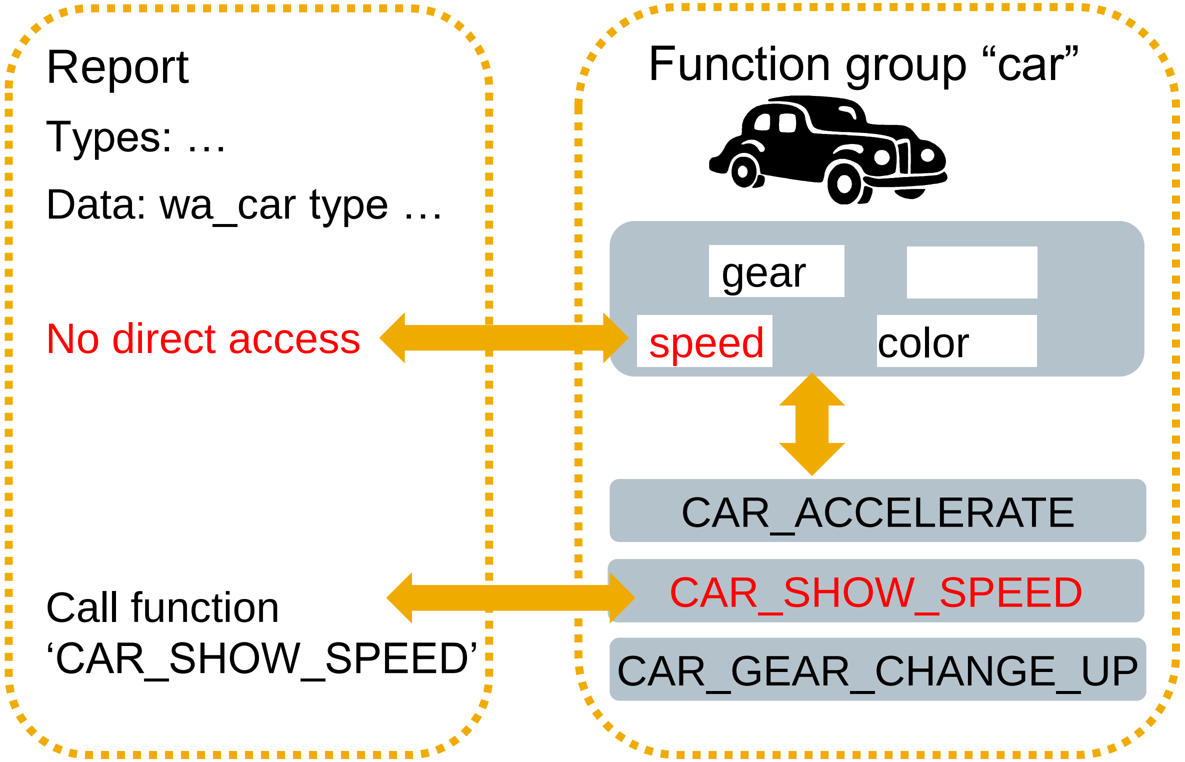
Source: BC401 ABAP Objects (2011)

- All extensions are upward compatible.
- The difference in ABAP Objects compared to other object-oriented languages is in the development environment. You can use the entire range of functions of the ABAP Workbench with ABAP Objects.

ABAP Objects = object-oriented extension of ABAP

Procedural vs. object-oriented approach

Procedural ABAP program



Report

Types: …

Data: wa_car type …

No direct access

Call function
‘CAR_SHOW_SPEED’

Function group “car”

gear

speed

color

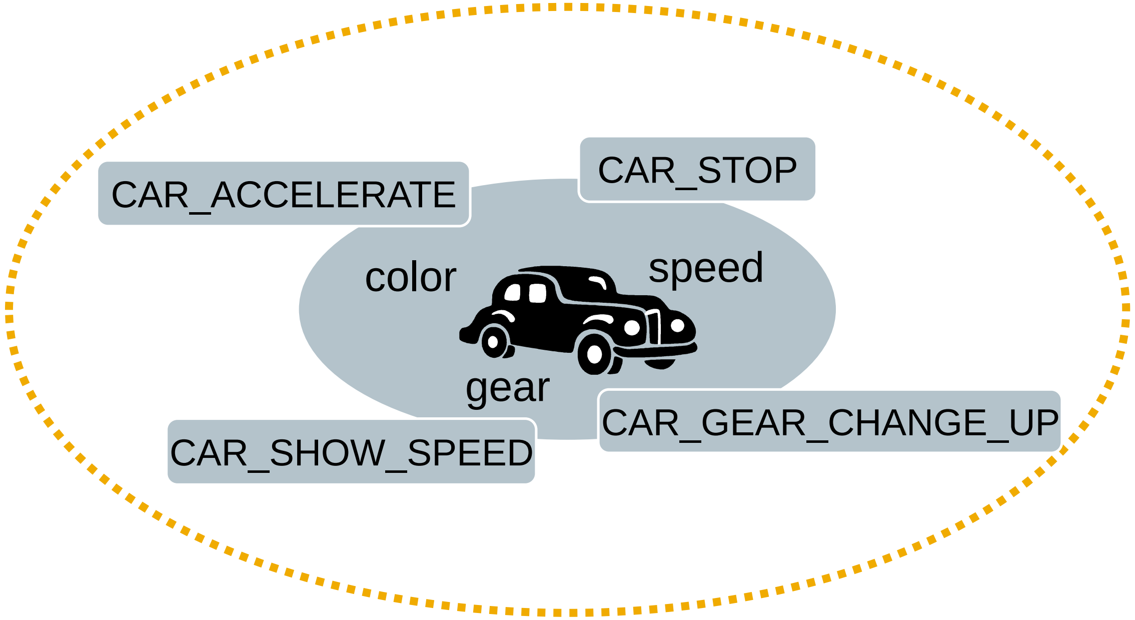
CAR_ACCELERATE

CAR_SHOW_SPEED

CAR_GEAR_CHANGE_UP

Separation of data and functions

Object-oriented approach



CAR_ACCELERATE

CAR_STOP

color

speed

gear

CAR_SHOW_SPEED

CAR_GEAR_CHANGE_UP

Encapsulation of data and functions

The implementation of a class is invisible outside the class. Interaction takes place only by a defined interface.

Classes and Objects

Classes

- General/abstract description of objects (“construction plan for cars”)
- Attributes of classes specify status data (e.g. speed)
- Methods describe the behavior (e.g. car_accelerate)

lcl_car

Attributes

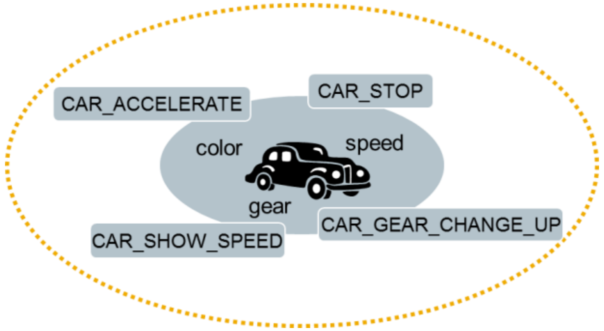
…

Methods

…

Objects

- Objects are instances of a class (i.e. one object corresponds to one car, it is built after the construction plan of its class)
- Objects represent a part of the real world (i.e. one car)
- Objects are units made up of data and functions

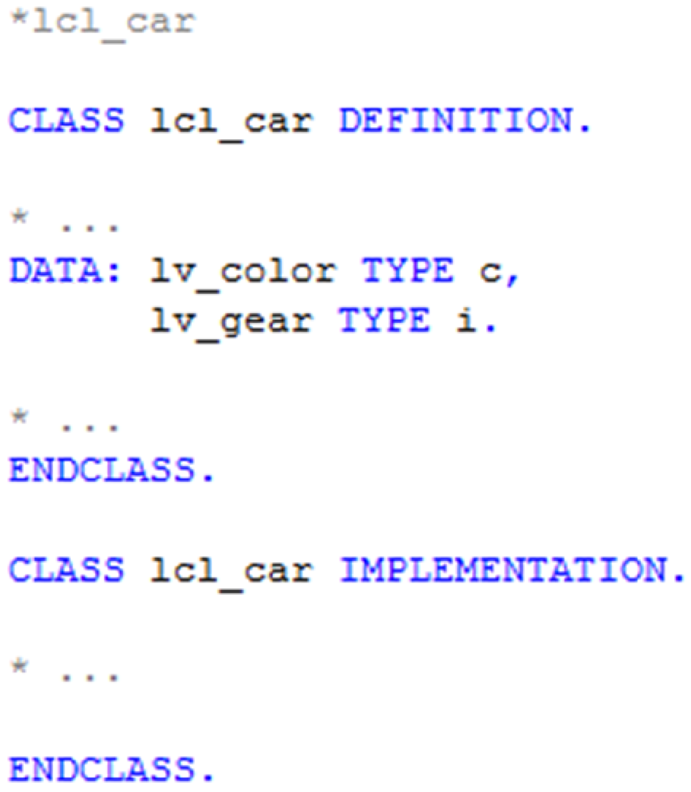


Elementary syntax elements – defining a class

- Classes can be created with the ABAP Editor (SE38) or the Class Builder (SE24)
- The declaration of a class is split into a definition and an implementation part

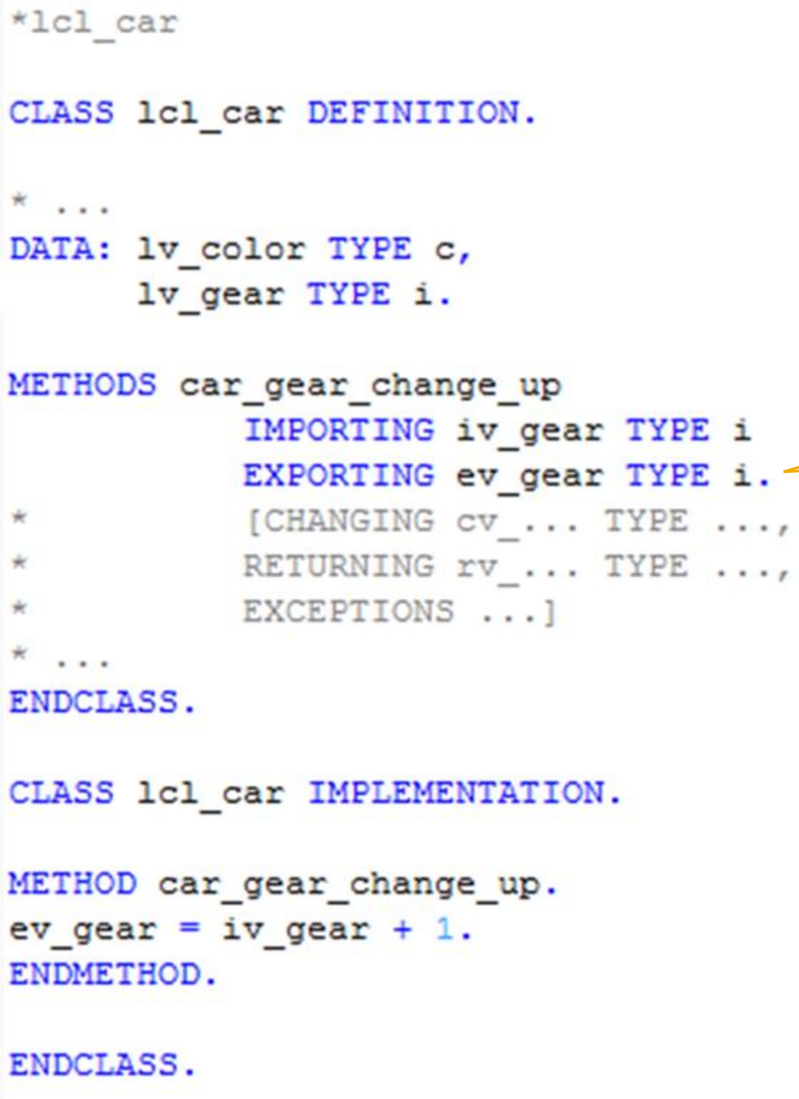
Declaration of all components of your class: attributes, methods, constants, types,…

Implementation of all methods of your class



```
*lcl_car

CLASS lcl_car DEFINITION.

* ...
DATA: lv_color TYPE c,
      lv_gear TYPE i.

* ...
ENDCLASS.

CLASS lcl_car IMPLEMENTATION.

* ...

ENDCLASS.
```

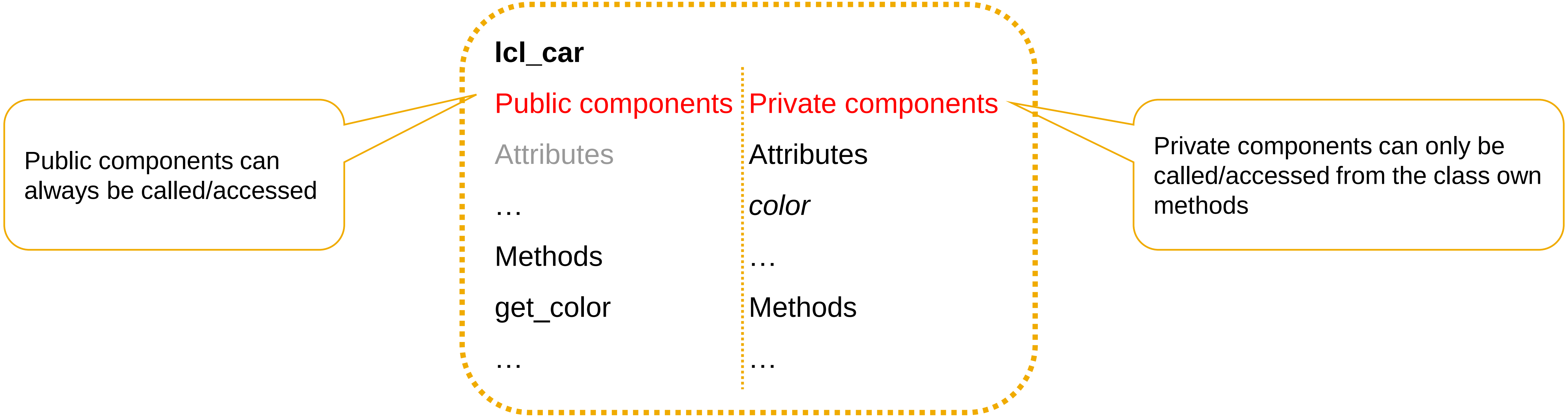
Elementary syntax elements – methods



```
*lcl_car

CLASS lcl_car DEFINITION.

* ...
DATA: lv_color TYPE c,
      lv_gear TYPE i.

METHODS car_gear_change_up
           IMPORTING iv_gear TYPE i
           EXPORTING ev_gear TYPE i.
*          [CHANGING cv_... TYPE ...,
*          RETURNING rv_... TYPE ...,
*          EXCEPTIONS ...]
* ...
ENDCLASS.

CLASS lcl_car IMPLEMENTATION.

METHOD car_gear_change_up.
ev_gear = iv_gear + 1.
ENDMETHOD.

ENDCLASS.
```

Methods have a signature, where parameters and exceptions can be passed.

Public and private components



Public components can always be called/accessed

lcl_car

Public components	Private components
Attributes	Attributes
…	*color*
Methods	…
get_color	Methods
…	…

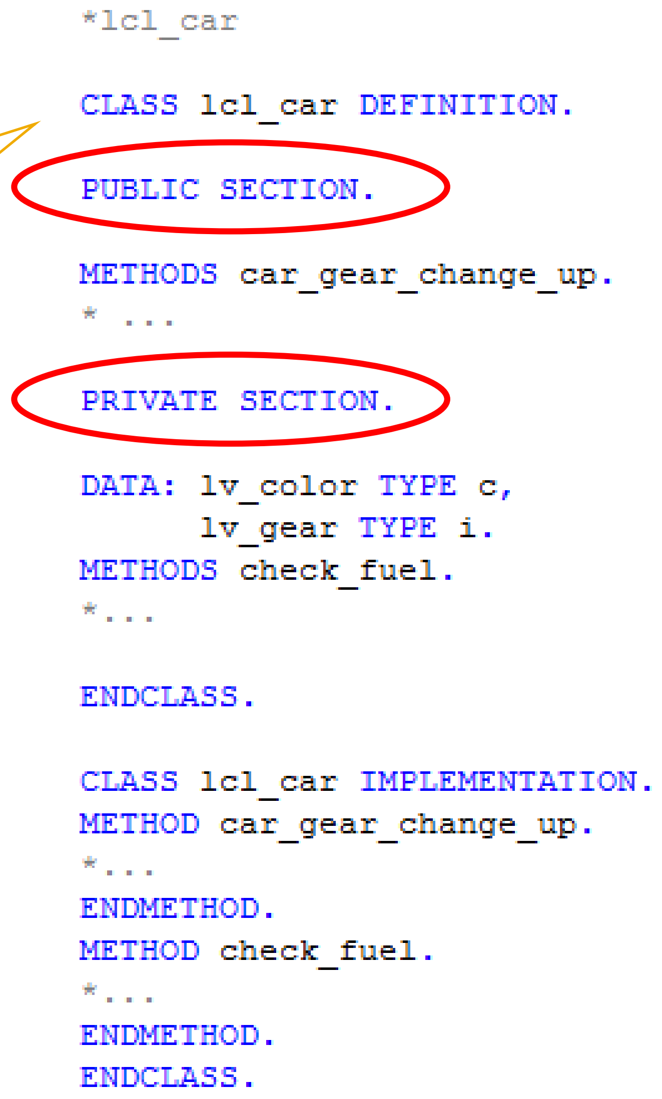
Private components can only be called/accessed from the class own methods

Design principle

Normally, data types and attributes are declared as private components. They are accessed via public methods. E.g. the private attribute color may be accessed via the public method get_color.

Elementary syntax elements – public and private components

By default all the members of a class are PRIVATE.



```
*lcl_car

CLASS lcl_car DEFINITION.

PUBLIC SECTION.

METHODS car_gear_change_up.
* ...

PRIVATE SECTION.

DATA: lv_color TYPE c,
      lv_gear TYPE i.
METHODS check_fuel.
*...

ENDCLASS.

CLASS lcl_car IMPLEMENTATION.
METHOD car_gear_change_up.
*...
ENDMETHOD.
METHOD check_fuel.
*...
ENDMETHOD.
ENDCLASS.
```

Static and instance components

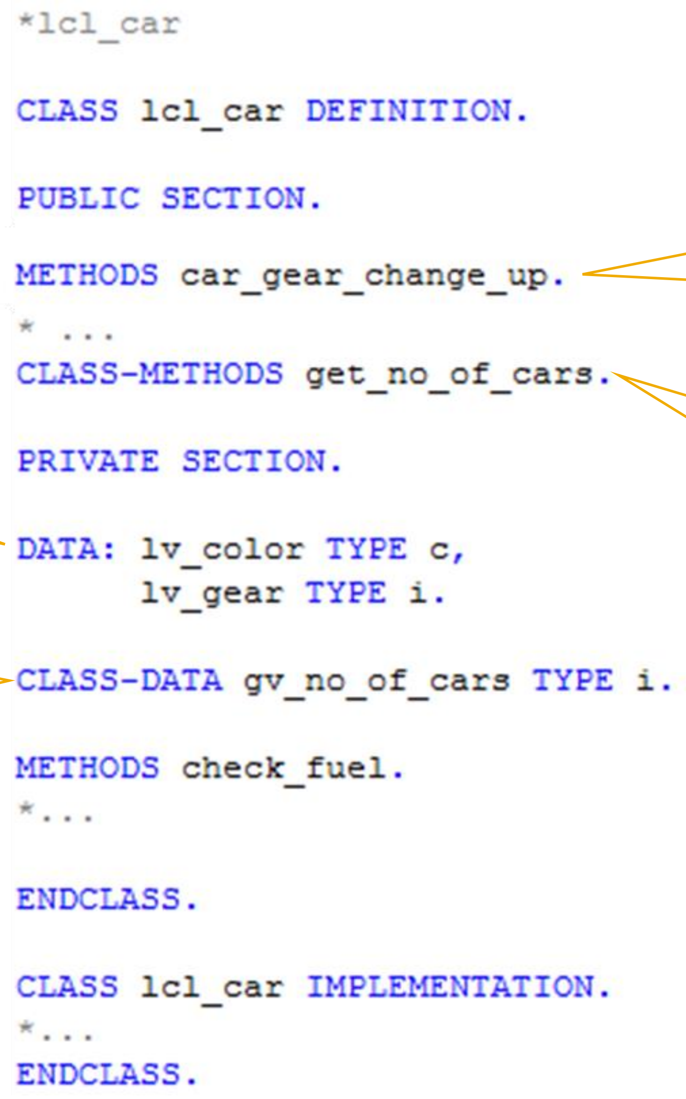
Attributes

- Instance attributes exist for every instance of a class, e.g. every car of class lcl_car has its own color
- Static attributes exist only once per class, e.g. attribute no_of_cars
- Attributes may also be declared as constants, e.g. no_of_wheels = 4

Methods

- Instance methods (e.g. get_color) may access static and instance components
- Static methods (e.g. get_no_of_cars) may only access static components

Elementary syntax elements – static and instance components



```
*lcl_car

CLASS lcl_car DEFINITION.

PUBLIC SECTION.

METHODS car_gear_change_up.

* ...
CLASS-METHODS get_no_of_cars.

PRIVATE SECTION.

DATA: lv_color TYPE c,
      lv_gear TYPE i.

CLASS-DATA gv_no_of_cars TYPE i.

METHODS check_fuel.
*...

ENDCLASS.

CLASS lcl_car IMPLEMENTATION.
*...
ENDCLASS.
```

Instance methods are defined by the expression METHOD(S).

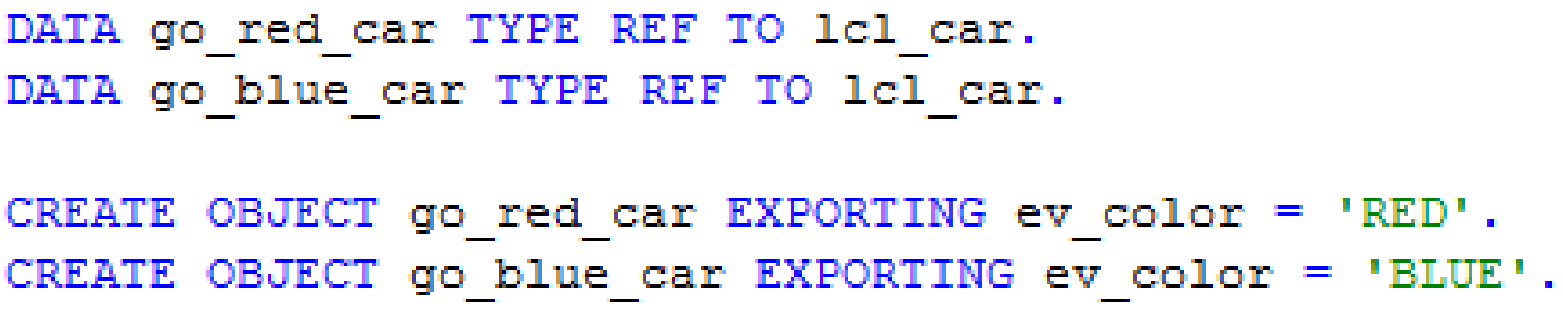
Instance attributes are defined by the expression DATA.

Static methods are defined by the expression CLASS-METHOD(S).

Static attributes are defined by the expression CLASS-DATA.

Instances of classes

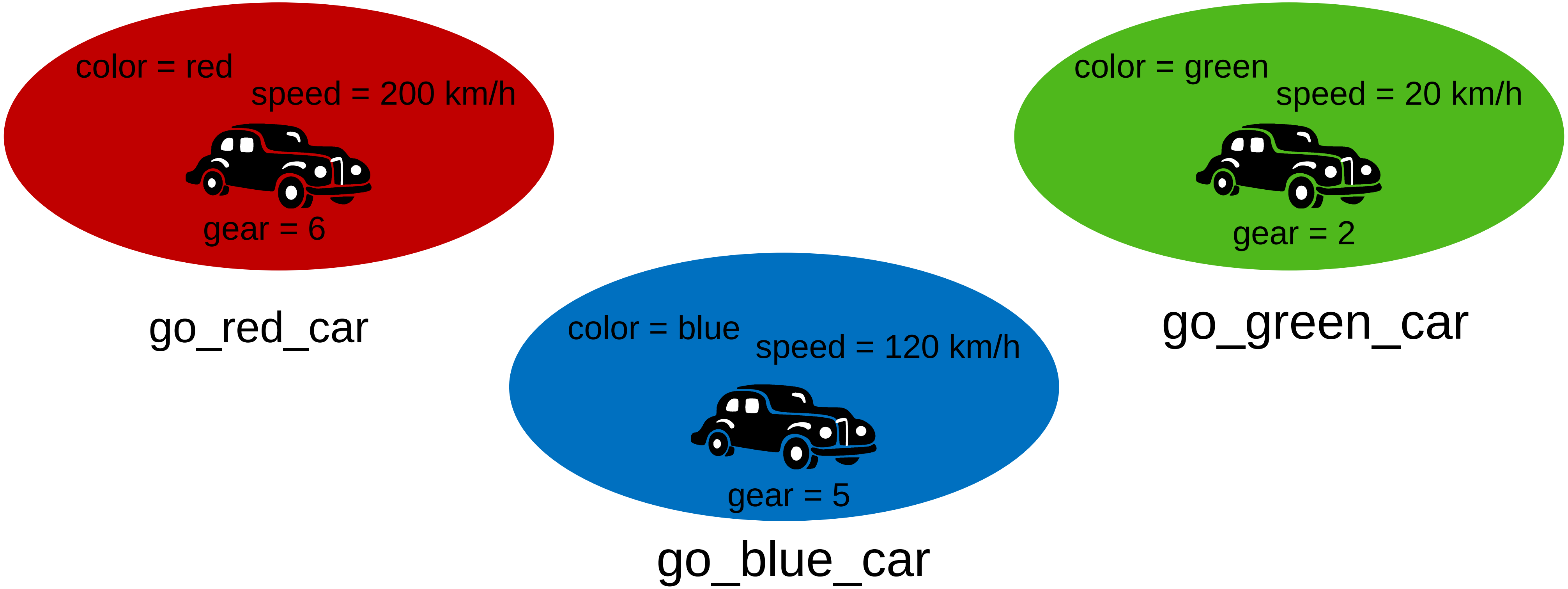
- One object of a class = one instance of a class
- The instantiation of an object is triggered by the expression CREATE OBJECT …



```
DATA go_red_car TYPE REF TO lcl_car.
DATA go_blue_car TYPE REF TO lcl_car.

CREATE OBJECT go_red_car EXPORTING ev_color = 'RED'.
CREATE OBJECT go_blue_car EXPORTING ev_color = 'BLUE'.
```

Instances of classes - multiple instantiation

- Multiple instances of a class may exist at the same time
- Important characteristic of object-orientated programming



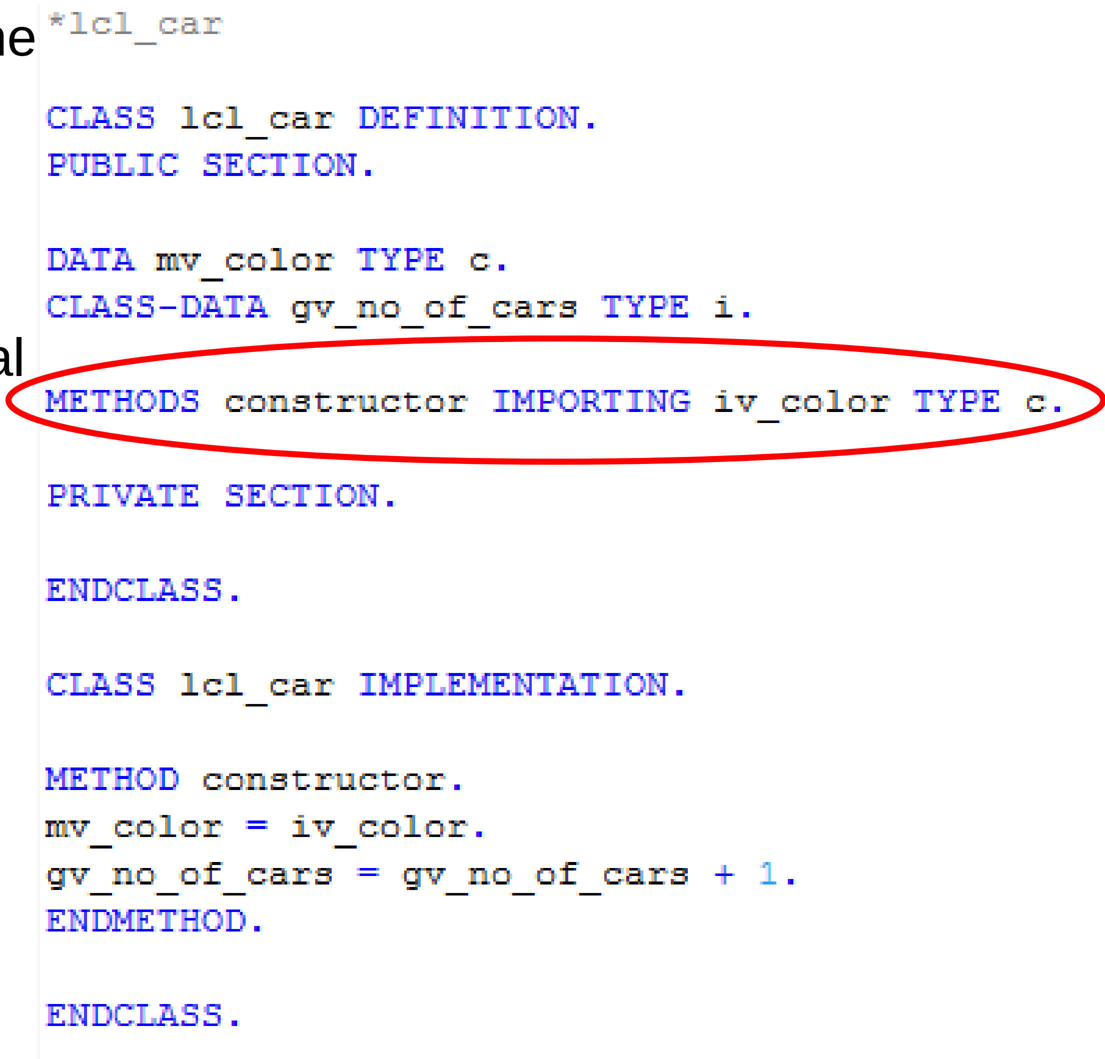
Instances of classes - constructor

- The expression CREATE OBJECT automatically calls the method constructor (reserved name)

- The constructor method is an instance method

- The implementation of the constructor method is optional (if the implementation is missing, simply a new instance is created)

- There is no destructor in ABAP

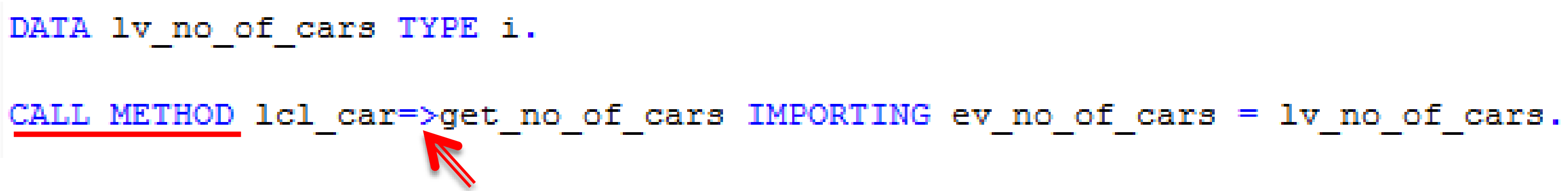


```
*lcl_car

CLASS lcl_car DEFINITION.
PUBLIC SECTION.

DATA mv_color TYPE c.
CLASS-DATA gv_no_of_cars TYPE i.

METHODS constructor IMPORTING iv_color TYPE c.

PRIVATE SECTION.

ENDCLASS.

CLASS lcl_car IMPLEMENTATION.

METHOD constructor.
mv_color = iv_color.
gv_no_of_cars = gv_no_of_cars + 1.
ENDMETHOD.

ENDCLASS.
```

Accessing methods and attributes

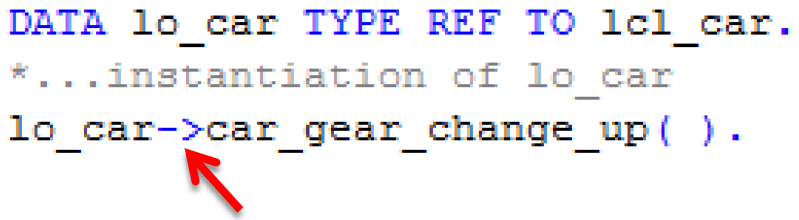
Methods

- Static methods:



```
DATA lv_no_of_cars TYPE i.

CALL METHOD lcl_car=>get_no_of_cars IMPORTING ev_no_of_cars = lv_no_of_cars.
```

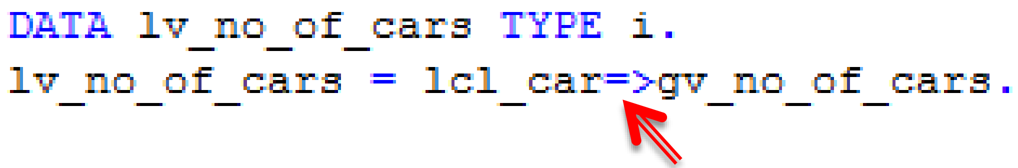
- Instance methods:



```
DATA lo_car TYPE REF TO lcl_car.
*...instantiation of lo_car
lo_car->car_gear_change_up( ).
```

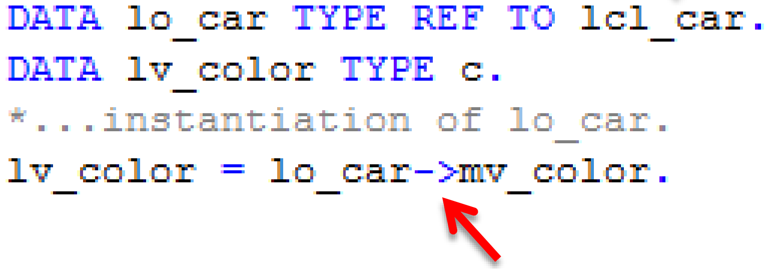
Attributes

- Static attributes:



```
DATA lv_no_of_cars TYPE i.
lv_no_of_cars = lcl_car=>gv_no_of_cars.
```

- Instance attributes:



```
DATA lo_car TYPE REF TO lcl_car.
DATA lv_color TYPE c.
*...instantiation of lo_car.
lv_color = lo_car->mv_color.
```

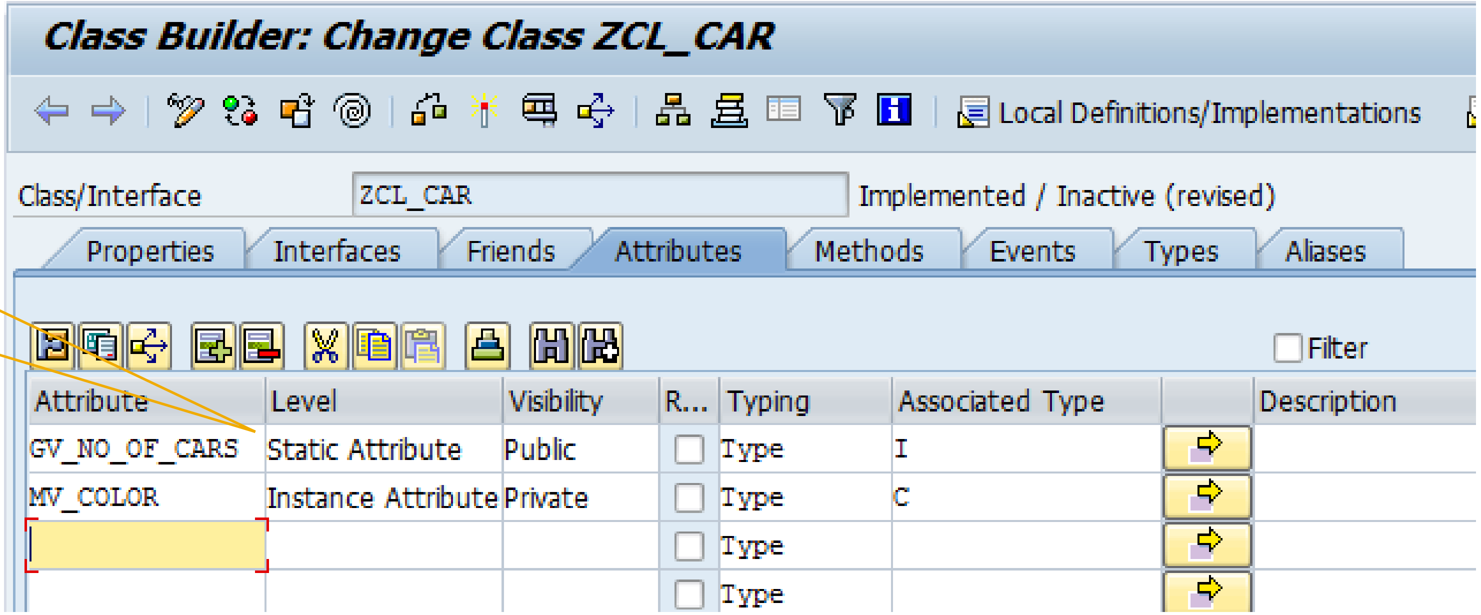
Now you are able to do create instances of classes and access methods.

To consolidate your knowledge, you can do tasks 3 and 4 of the ABAP Objects exercises.

The Class Builder

- Local classes can only be accessed within the program they have been defined and implemented.
- A global class is stored centrally in the class library of the repository and can be accessed from all ABAP programs.
- You can define global classes in the Class Builder (Transaction SE24).

Properties of attributes can be defined without coding.



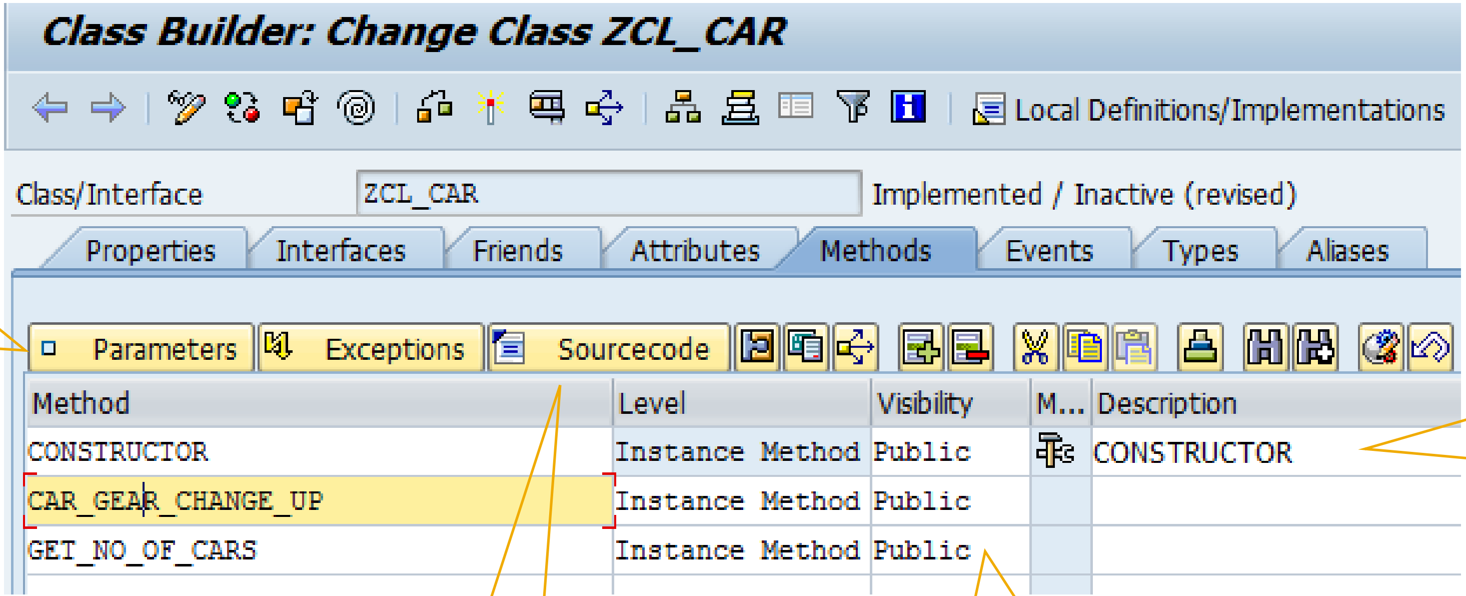
Class Builder: Change Class ZCL_CAR

Class/Interface ZCL_CAR Implemented / Inactive (revised)

Attribute	Level	Visibility	R...	Typing	Associated Type		Description
GV_NO_OF_CARS	Static Attribute	Public	□	Type	I	⇨	
MV_COLOR	Instance Attribute	Private	□	Type	C	⇨	
			□	Type		⇨	
			□	Type		⇨	

III. Global classes

The Class Builder



Class Builder: Change Class ZCL_CAR

Class/Interface: ZCL_CAR — Implemented / Inactive (revised)

Method	Level	Visibility	M...	Description
CONSTRUCTOR	Instance Method	Public		CONSTRUCTOR
CAR_GEAR_CHANGE_UP	Instance Method	Public		
GET_NO_OF_CARS	Instance Method	Public		

Click here to define parameters and exceptions for your method.

Every class may have only one constructor method

Click here to implement your method.

Properties of your method can be defined without coding

IV. Further principles of object-oriented programming

Polymorphism and inheritance

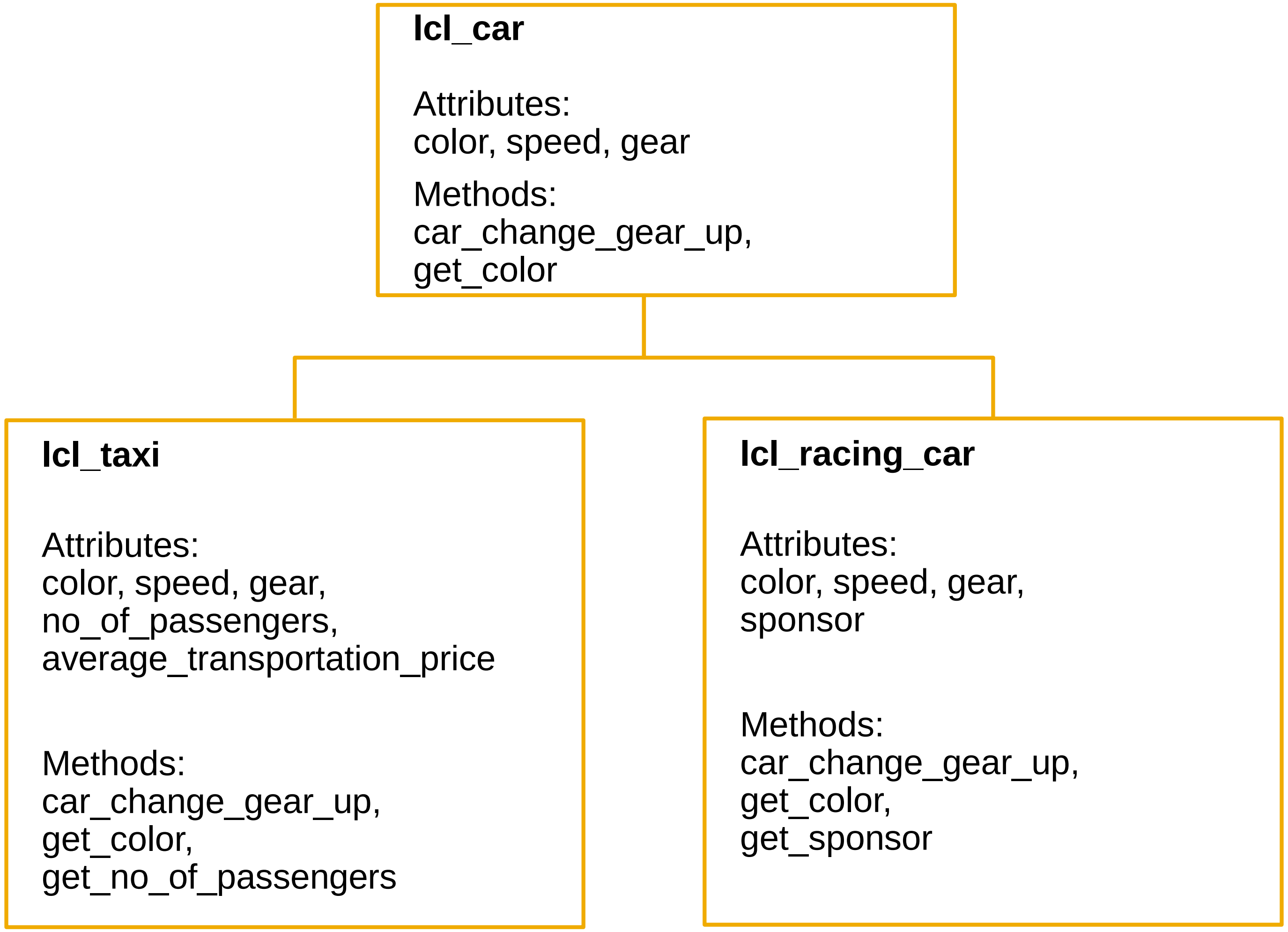
Beneath the encapsulation of data and functions, further principles of object-oriented programming exist.

Overloading	✗	(same method with different parameters)
Overriding	✓	(implementation of subclass replaces superclass’ one)

- **Polymorphism**
  Identically-named methods used for different classes respond according to an appropriate class-specific behavior.

- **Inheritance**
  A new class can inherit attributes and methods from an existing class that can be extended by additional, own attributes and methods.

Inheritance - example



lcl_car

Attributes:
color, speed, gear

Methods:
car_change_gear_up,
get_color

lcl_taxi

Attributes:
color, speed, gear,
no_of_passengers,
average_transportation_price

Methods:
car_change_gear_up,
get_color,
get_no_of_passengers

lcl_racing_car

Attributes:
color, speed, gear,
sponsor

Methods:
car_change_gear_up,
get_color,
get_sponsor

Examples

- Exception classes / exception handling
- Odata service implementation
- ALV-programming
- Business Add-Ins
- …

Now you can define global ABAP Classes with the Class Builder.

To consolidate your knowledge, you can do the Challenge of the ABAP Objects exercises.

Check your knowledge

Check your knowledge

- An object in object-oriented programming represents a “construction plan”.

  □ True □ False

- Explain the meaning of the term “class” in the ABAP Objects context!

- What is the difference between a static and an instance component of a class?

- A local class can be accessed from every report in your SAP system.

  □ True □ False

- What happens, if the implementation of the constructor is missing?

Solution

Solution

- An object in object-oriented programming represents a “construction plan”.

  □ True ☒ False

- Explain the meaning of the term “class” in the ABAP Objects context!

  See section *Principles of object-orientation*

- What is the difference between a static and an instance component of a class?

  See section *Principles of object-orientation*

- A local class can be accessed from every report in your SAP system.

  □ True ☒ False

- What happens, if the implementation of the constructor is missing?

  See section *Principles of object-orientation*

References

- BC401 ABAP Objects, Teilnehmerhandbuch. Version der Schulung: 92, 2011

© 2018 SAP SE or an SAP affiliate company. All rights reserved.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors.

National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP SE or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP SE or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, which speak only as of their dates, and they should not be relied upon in making purchasing decisions.