ABAP OBJECTS – BASICS

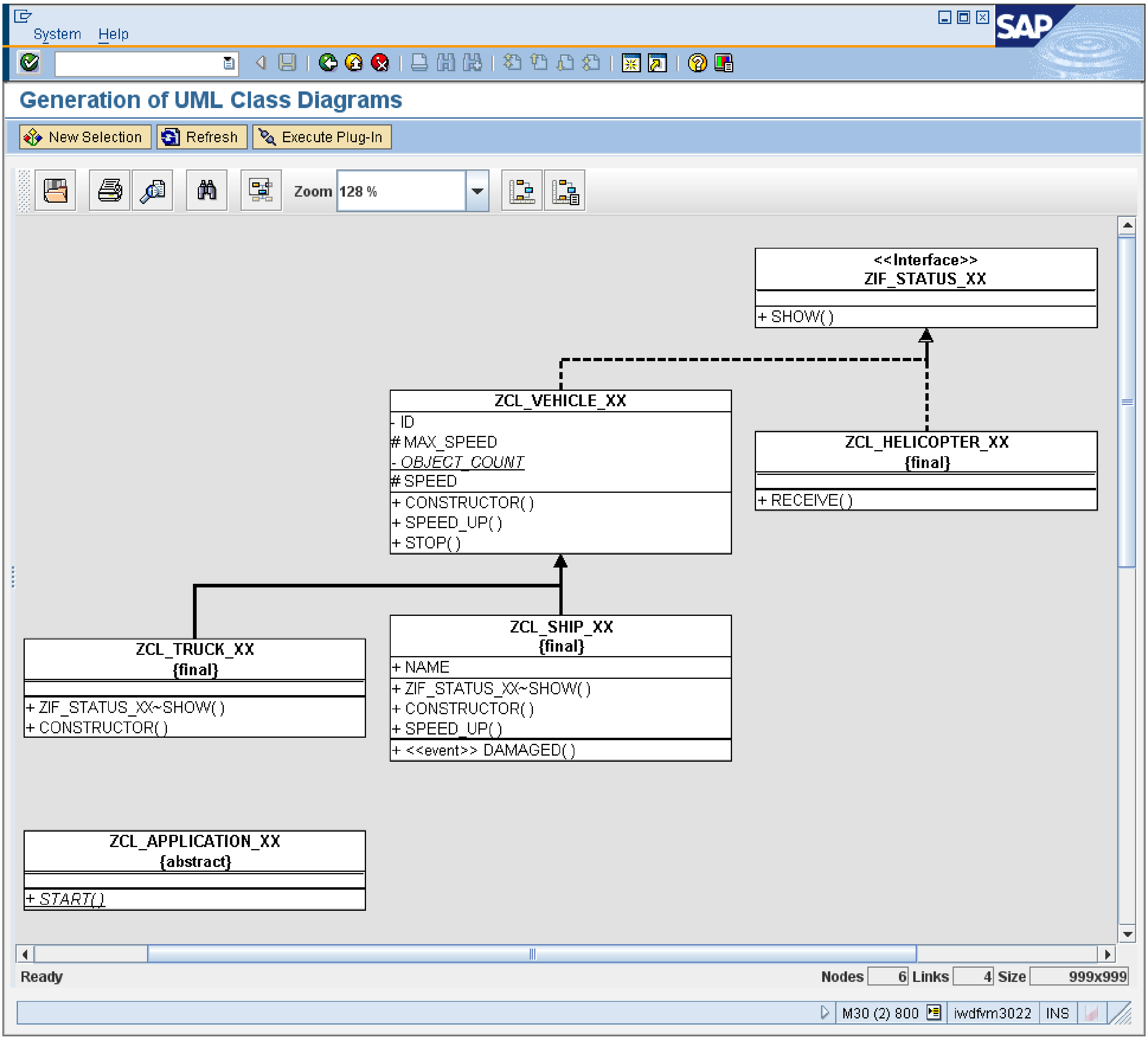
GUIDED TUTORIAL

Exercises / Solutions by Peter McNulty, Horst Keller / SAP

Introduction

This exercise is a tutorial or guided tour through the fundamental language elements of ABAP Objects and the usage of the respective ABAP Workbench tools. The tutorial is designed for developers who have had little or no experience with ABAP and ABAP Objects until now.

The following class diagram shows the scope of the exercise.



(After finishing the exercise, you should be able to display this diagram for your own classes).

Exercise 1, Classes and Objects

(Solution in Package Z_ABAP_OBJECTS_INTRODUCTION_A)

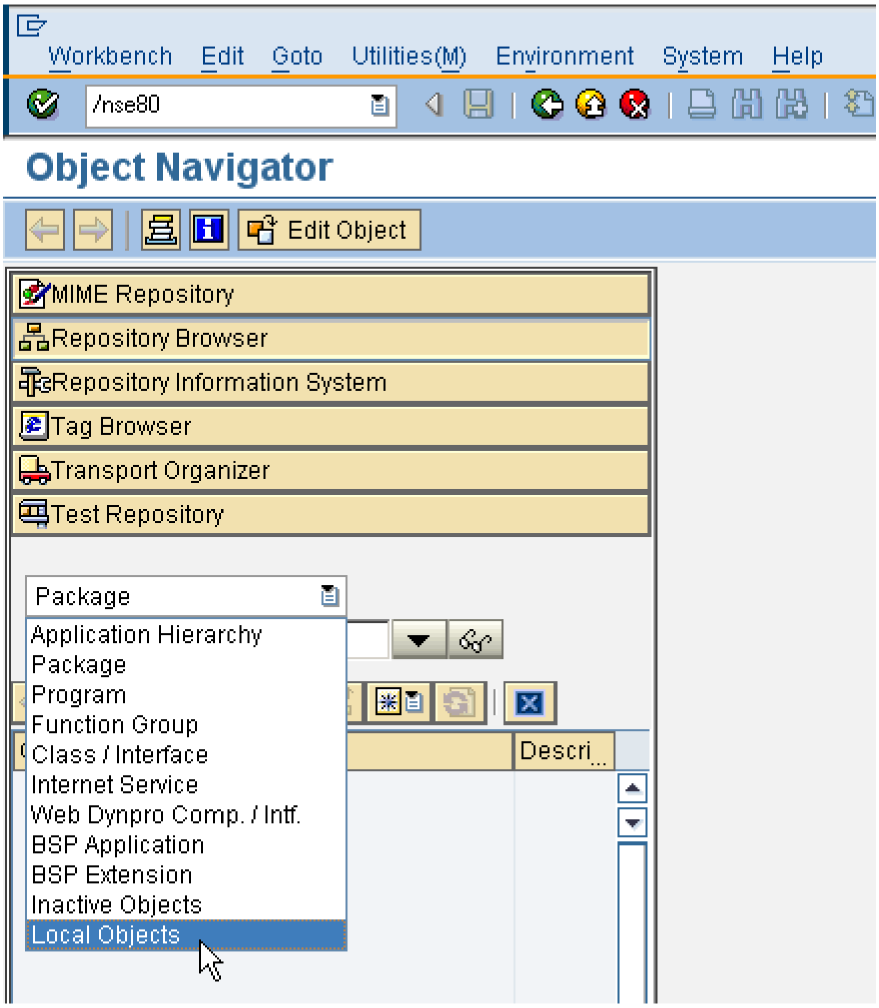
Create a vehicle class

Create a class ZCL_VEHICLE_XX (where XX is your group number).

- The class should have the protected instance attributes SPEED and MAX_SPEED for its speed and maximum speed, and the public methods SPEED_UP, STOP, and SHOW. Furthermore there should be a private attribute that contains a running number for an object ID.
- SPEED_UP should have an IMPORTING parameter STEP. The method should increase the speed by STEP, but not allow it to exceed the maximum speed.
- STOP should reset the speed to zero.
- WRITE should display a message containing the speed and the maximum speed.

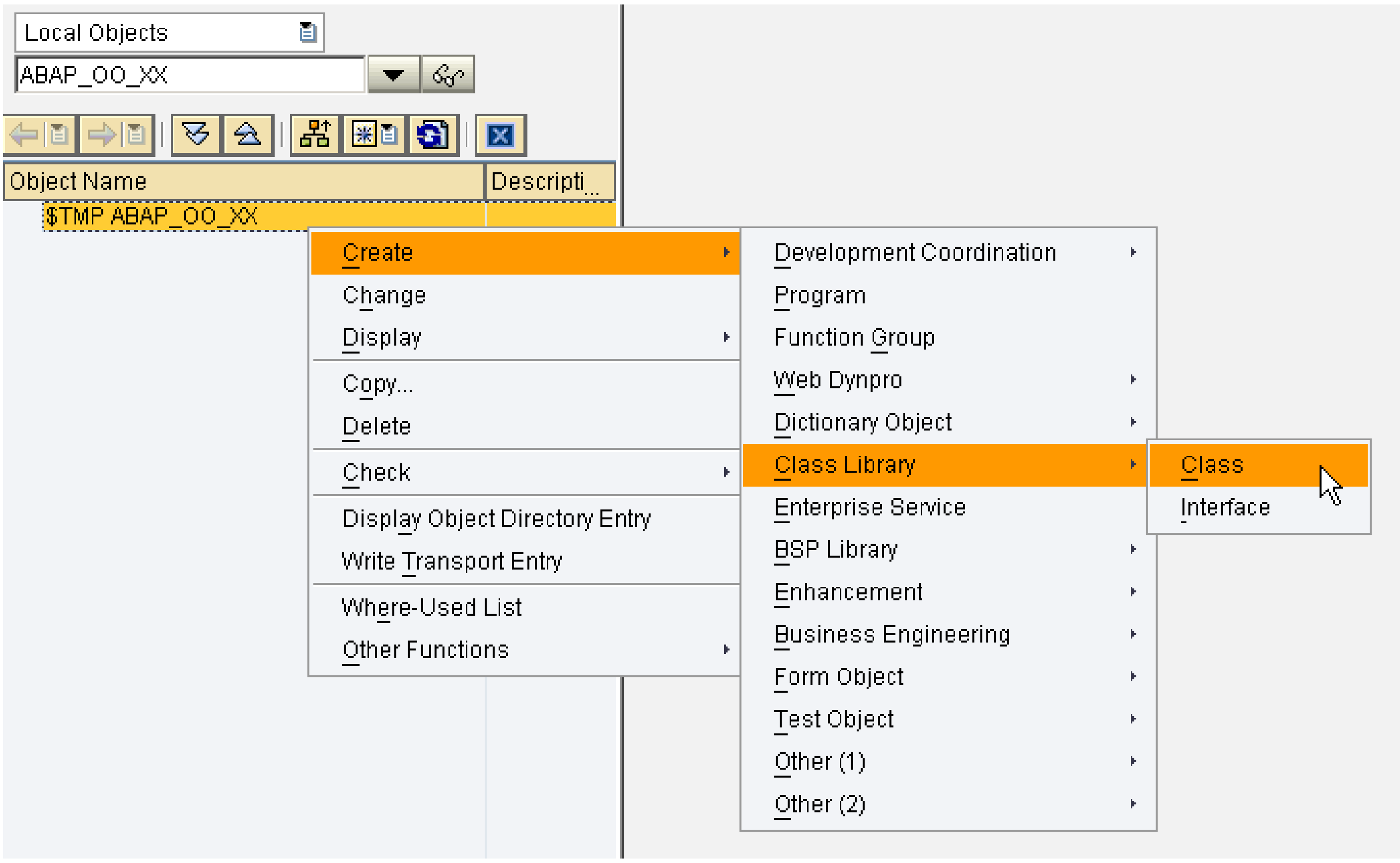
Solution

1. Logon to the system and open the Object Navigator of the ABAP Workbench (Transaction SE80, enter /nSE80 in the command field of the system task bar).
2. Select **Local Objects** in order to work in a test package that is not transported to other systems.

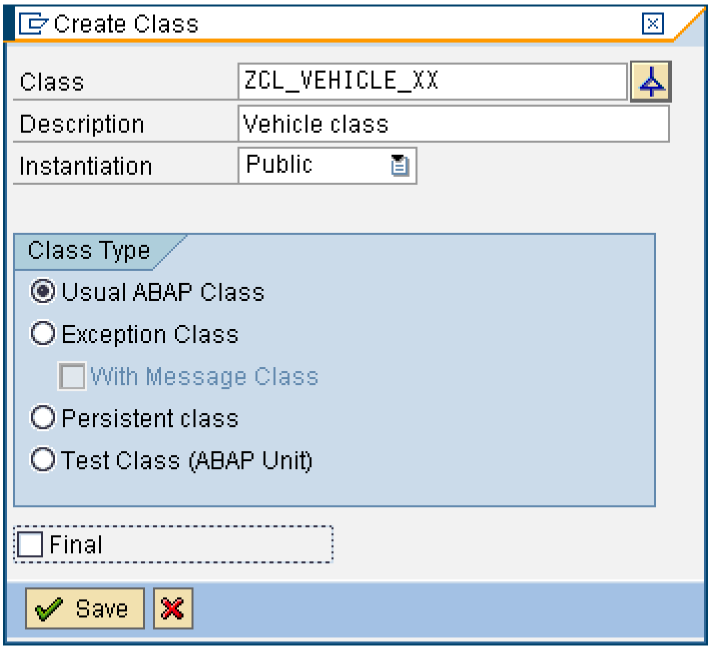


Hit **Enter.**

3. Right Click the name of the local package and navigate to the creation of a class (where XX is your group number).



4. Fill the pop-up as follows ZCL_VEHICLE_XX (where XX is your group number) and select **Save**.



Create Class

Class	ZCL_VEHICLE_XX
Description	Vehicle class
Instantiation	Public

Class Type

● Usual ABAP Class

○ Exception Class

□ With Message Class

○ Persistent class

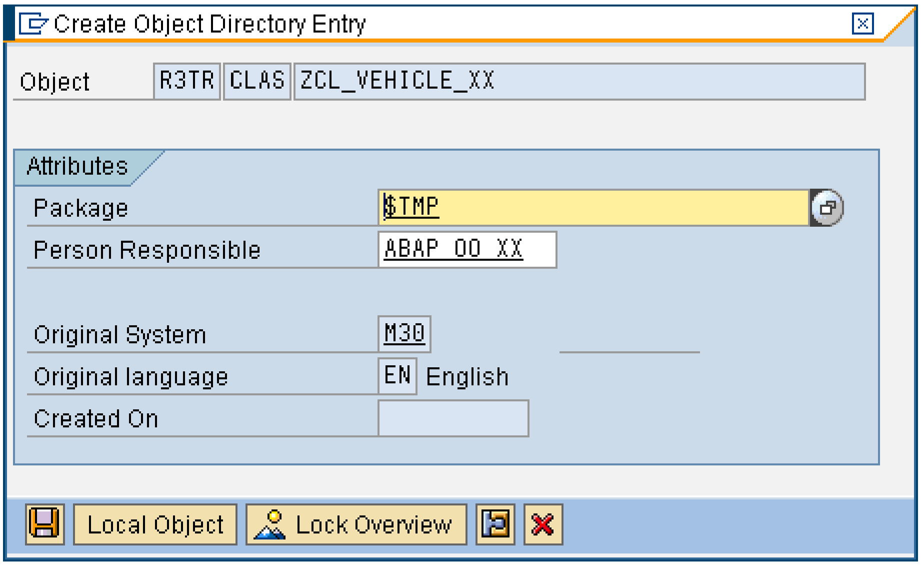
○ Test Class (ABAP Unit)

□ Final

Save

(Don't forget to uncheck **Final**)

5. Acknowledge the following window without changes (select either **Save** or **Local Object**).



Object R3TR CLAS ZCL_VEHICLE_XX

Package $TMP

Person Responsible ABAP_00_XX

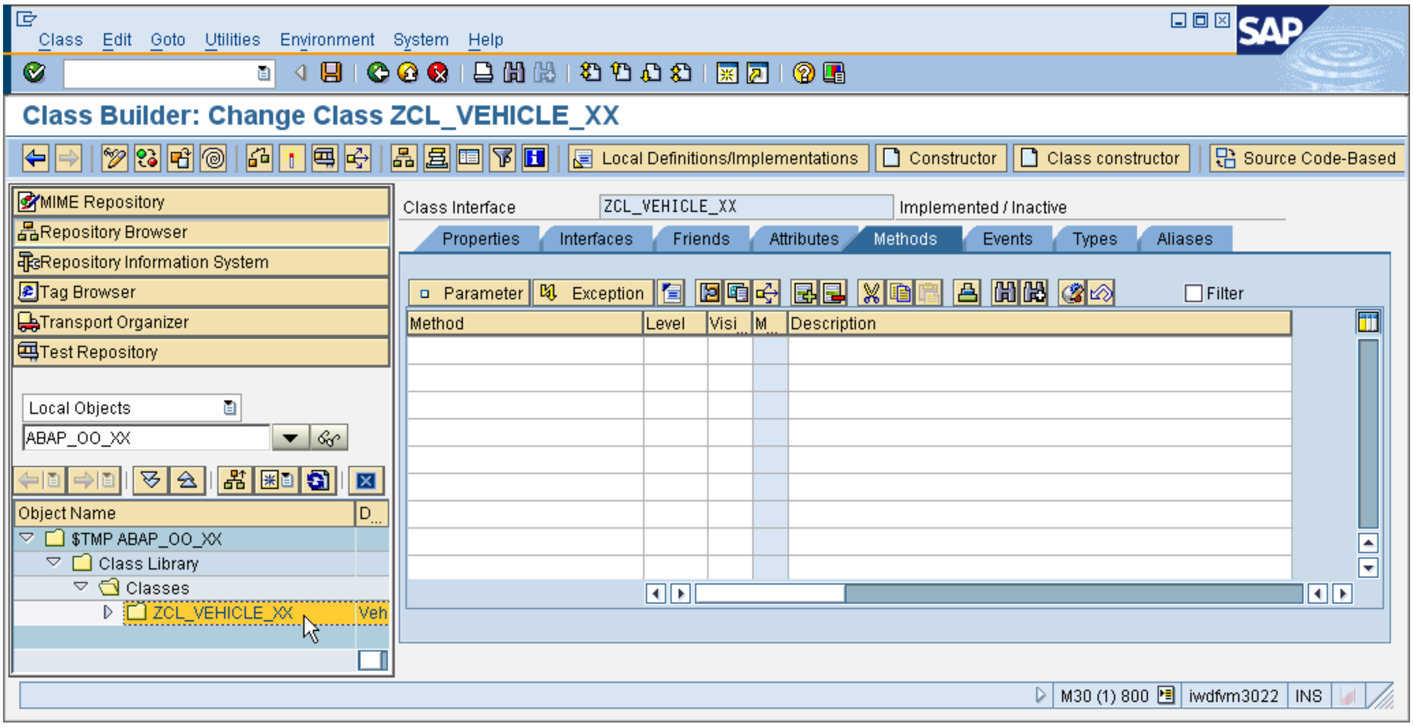
Original System M30

Original language EN English

Created On

The same holds for all other development objects during this exercise.

6. Now you enter the Class Builder



Class Interface ZCL_VEHICLE_XX Implemented / Inactive

Here you can edit the class either in **Form-Based** mode (default) or in **Source Code-Based** mode. Use the respective button to toggle between the modes.

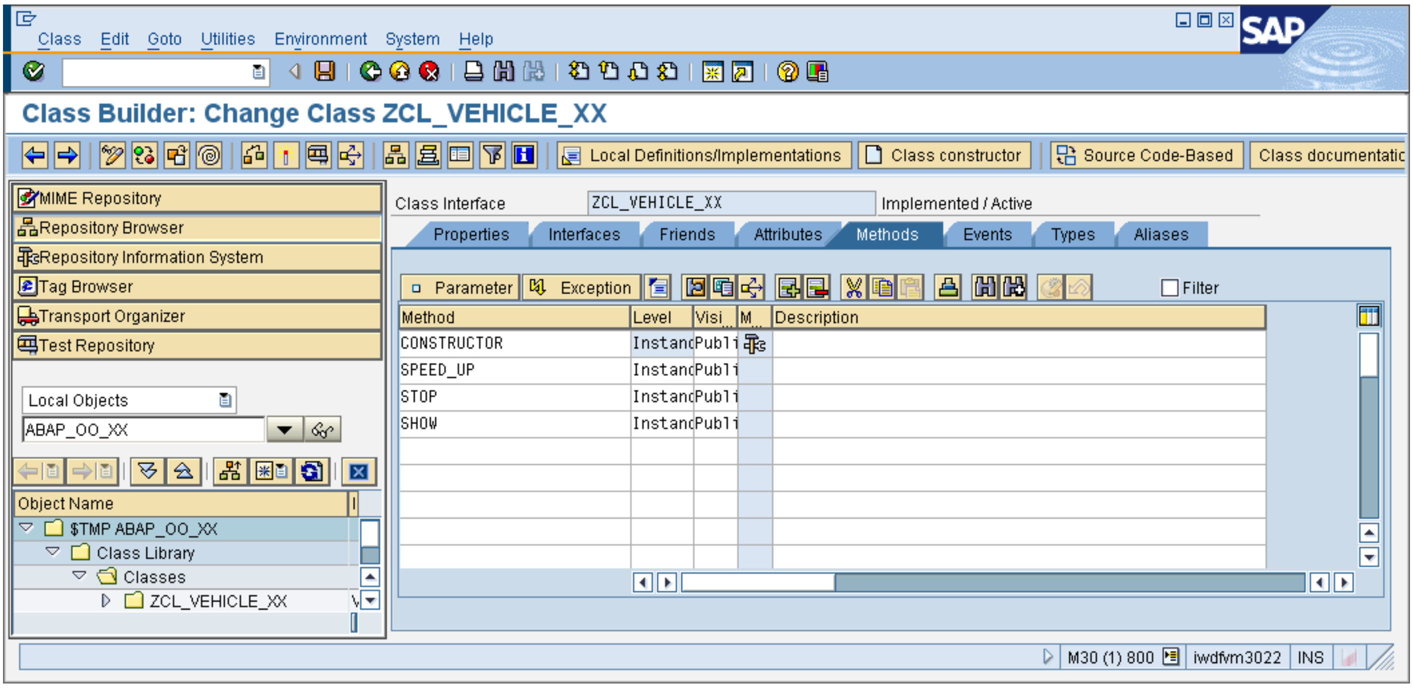
7. Switch to **Source Code-Based** mode Source Code-Based, switch to **Change** mode and replace the existing template with the following code (where XX is your group number).

```
CLASS zcl_vehicle_xx DEFINITION PUBLIC CREATE PUBLIC.
  PUBLIC SECTION.
    METHODS constructor.
    METHODS speed_up
      IMPORTING
```

```
        step TYPE i.
    METHODS stop.
    METHODS show.
  PROTECTED SECTION.
    DATA: speed     TYPE i,
          max_speed TYPE i VALUE 50.
  PRIVATE SECTION.
    DATA id TYPE i .
    CLASS-DATA object_count TYPE i.
ENDCLASS.

CLASS zcl_vehicle_xx IMPLEMENTATION.
  METHOD constructor.
    object_count = object_count + 1.
    id = object_count.
  ENDMETHOD.
  METHOD show.
    DATA msg TYPE string.
    msg =  `Vehicle `      && |{ id }| &&
          `, Speed = `     && |{ speed }| &&
          `, Max-Speed = ` && |{ max_speed }|.
    MESSAGE msg TYPE 'I'.
  ENDMETHOD.
  METHOD speed_up.
    speed = speed + step.
    IF speed > max_speed.
      speed = max_speed.
    ENDIF.
  ENDMETHOD.
  METHOD stop.
    speed = 0.
  ENDMETHOD.
ENDCLASS.
```

Check, **Save** and **Activate** the class (acknowledge all entries of the activation pop-up).

8. Switch back to **Form-Based** mode Form-Based and play around in that mode by double clicking the class components.



Class Builder: Change Class ZCL_VEHICLE_XX

Class Interface ZCL_VEHICLE_XX Implemented / Active

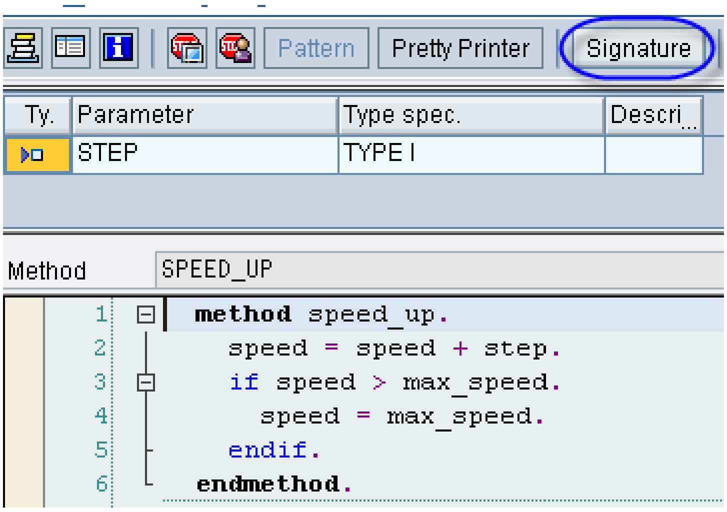
ABAP_OO_XX

Object Name
$TMP ABAP_OO_XX
Class Library
Classes
ZCL_VEHICLE_XX

Method	Level	Visi...	M...	Description
CONSTRUCTOR	Instanc	Publi		
SPEED_UP	Instanc	Publi		
STOP	Instanc	Publi		
SHOW	Instanc	Publi		

Tip: Select Signature when displaying the implementation of a method.

9.



Ty.	Parameter	Type spec.	Descri...
	STEP	TYPE I	

Method SPEED_UP

```
method speed_up.
  speed = speed + step.
  if speed > max_speed.
    speed = max_speed.
  endif.
endmethod.
```

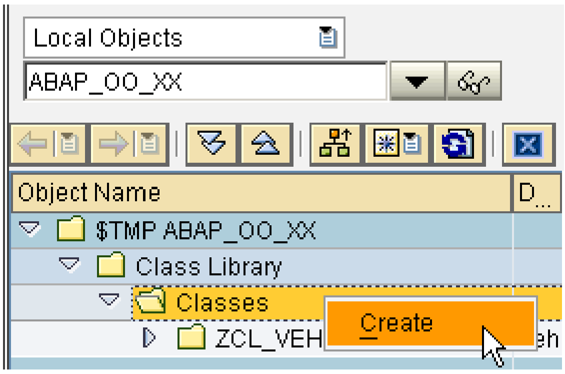
Create an application class

Create a class ZCL_APPLICATION_XX (where XX is your group number).

- This class should use your vehicle class.
- It should have simply one static method START that creates and uses objects of ZCL_VEHICLE_XX.

Solution

1. Create the class in the object navigator, where you can directly right click **Classes** now

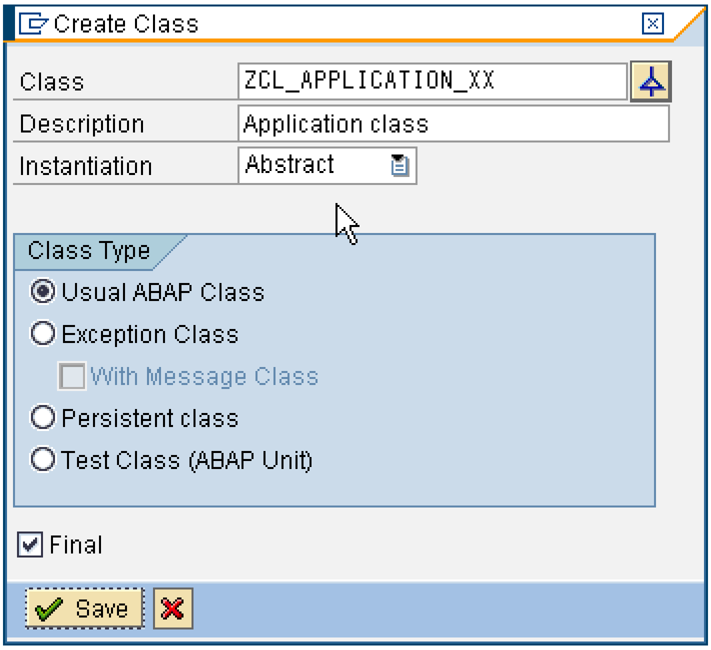


Local Objects

ABAP_OO_XX

Object Name	D...
$TMP ABAP_OO_XX	
Class Library	
Classes	
ZCL_VEH	

2. Fill the pop-up as follows (where XX is your group number).



Create Class

Class	ZCL_APPLICATION_XX
Description	Application class
Instantiation	Abstract

Class Type

● Usual ABAP Class

○ Exception Class

□ With Message Class

○ Persistent class

○ Test Class (ABAP Unit)

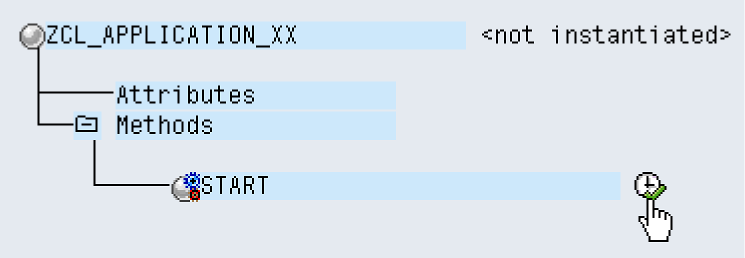
☑ Final

3. Implement the class as follows:

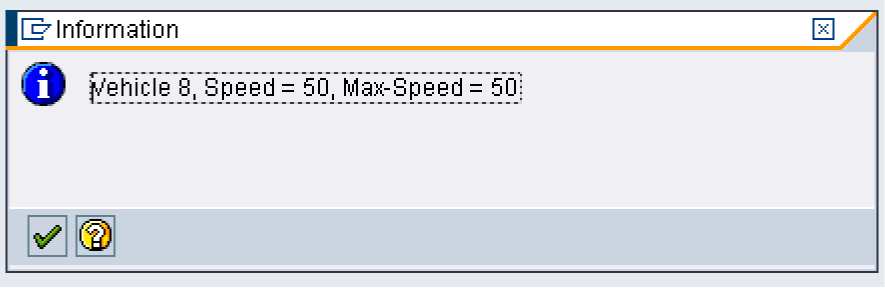
```
CLASS zcl_application_xx DEFINITION PUBLIC ABSTRACT FINAL CREATE PUBLIC.
  PUBLIC SECTION.
    CLASS-METHODS start.
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.

CLASS ZCL_APPLICATION_XX IMPLEMENTATION.
  METHOD start.
    DATA vehicle     TYPE REF TO zcl_vehicle_xx.
    DATA vehicle_tab LIKE TABLE OF vehicle.
    DATA tabindex TYPE i.
    DO 10 TIMES.
      CREATE OBJECT vehicle.
      APPEND vehicle TO vehicle_tab.
    ENDDO.
    LOOP AT vehicle_tab INTO vehicle.
      tabindex = sy-tabix * 10.
      vehicle->speed_up( tabindex ).
      vehicle->show( ).
    ENDLOOP.
  ENDMETHOD.
ENDCLASS.
```

Check, **Save** and **Activate** the class (acknowledge all entries of the activation pop-up).

4. Select (**F8**) in the Class Builder and execute method START.



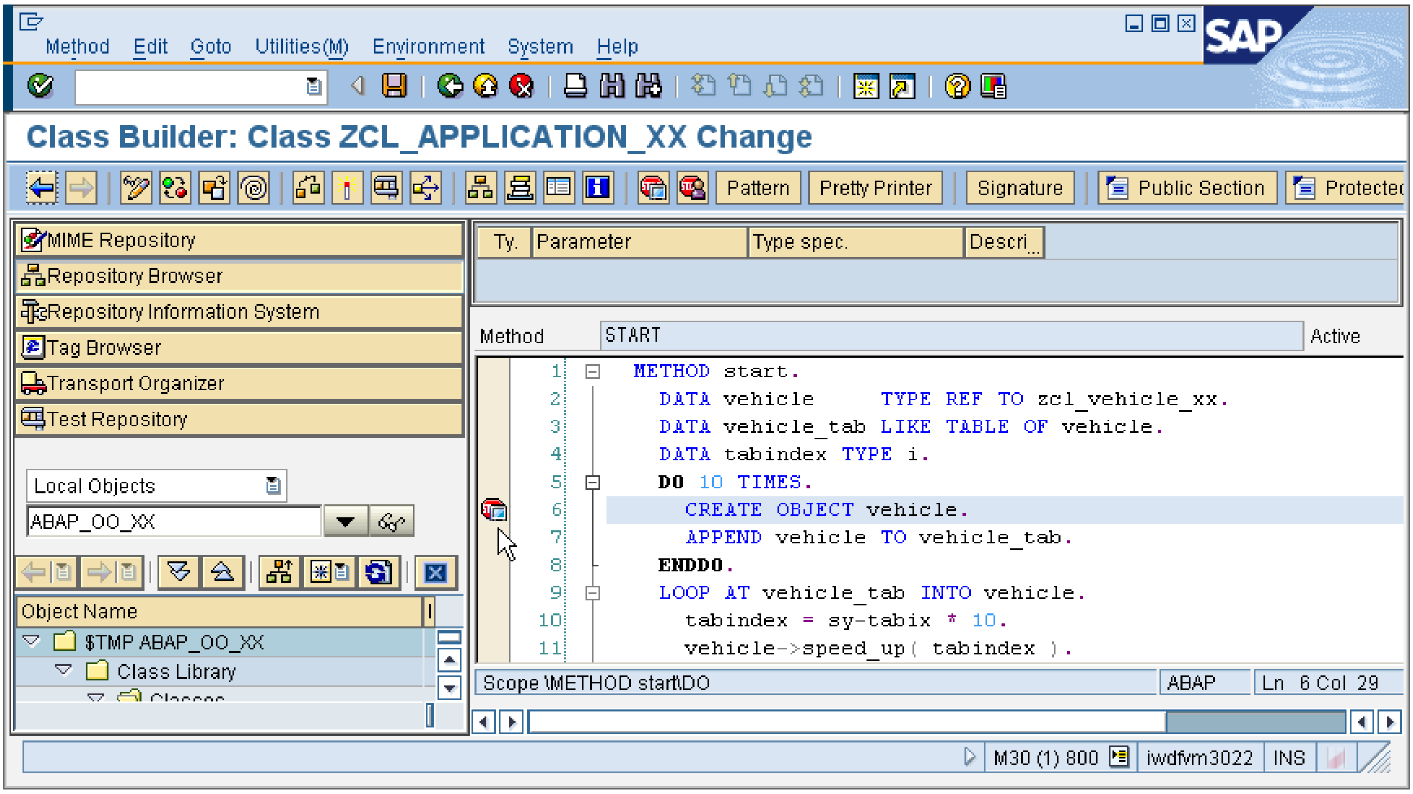
Now you can see the Ids and speed of the objects created.



Vehicle 8, Speed = 50, Max-Speed = 50

5. To examine further, navigate to the source code of method START and create a breakpoint at an appropriate position. To add a breakpoint at a certain line, double-click in the left margin on the line that you want a breakpoint. A little should appear in the margin on that line. Similarly, this is the same way you remove a breakpoint.

6.



Class Builder: Class ZCL_APPLICATION_XX Change

Method START Active

```
METHOD start.
  DATA vehicle      TYPE REF TO zcl_vehicle_xx.
  DATA vehicle_tab LIKE TABLE OF vehicle.
  DATA tabindex TYPE i.
  DO 10 TIMES.
    CREATE OBJECT vehicle.
    APPEND vehicle TO vehicle_tab.
  ENDDO.
  LOOP AT vehicle_tab INTO vehicle.
    tabindex = sy-tabix * 10.
    vehicle->speed_up( tabindex ).
```

Scope \METHOD start\DO | ABAP | Ln 6 Col 29

M30 (1) 800 | iwdfvm3022 | INS

Test the method again and play around with the ABAP Debugger.

Exercise 2, Inheritance

(Solution in Package Z_ABAP_OBJECTS_INTRODUCTION_B)

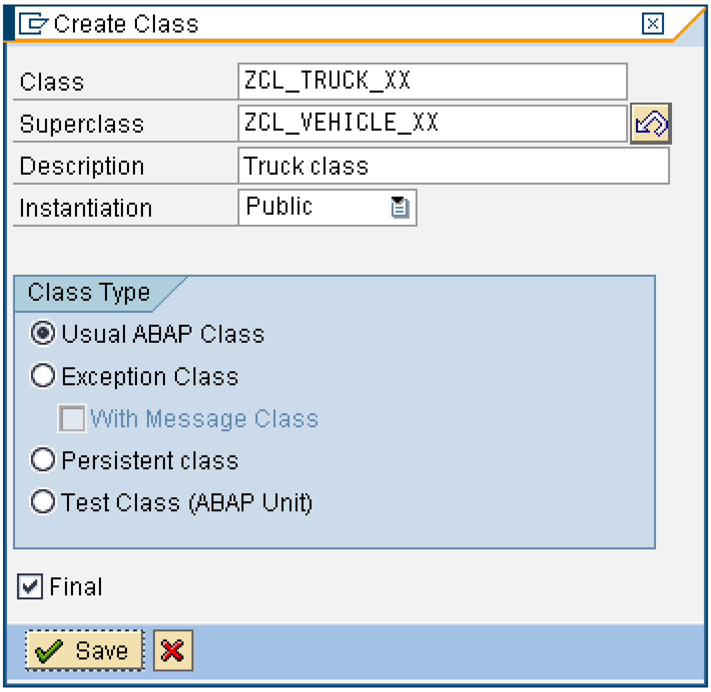
Create a truck subclass

Create a subclass ZCL_TRUCK_XX (where XX is your group number).

- The class should have an instance constructor that sets the maximal speed to 100
- It should redefine the method SHOW to produce some specific output.

Solution

1. Create the class in the object navigator as before, but select **Create inheritance** ( ) this time in order to enter a **Superclass**.



Create Class

Class	ZCL_TRUCK_XX
Superclass	ZCL_VEHICLE_XX
Description	Truck class
Instantiation	Public

Class Type

● Usual ABAP Class

○ Exception Class

□ With Message Class

○ Persistent class

○ Test Class (ABAP Unit)

☑ Final

Save 💾 and **Activate** .

2. Enter the truck's own constructor in the **Form-based** mode (type it or select **Create constructor** Constructor), double click it and implement it as follows:

```
METHOD constructor.
  super->constructor( ).
  max_speed = 100.
ENDMETHOD.
```

Save 💾 and **Activate** .

3. Select method SHOW in the **Form-based** mode and redefine it by selecting . Replace the implementation as follows:

```
METHOD show.
  DATA msg TYPE string.
  msg = `Truck `           &&
        `, Speed = `       && |{ speed }| &&
        `, Max-Speed = `   && |{ max_speed }|.
  MESSAGE msg TYPE 'I'.
ENDMETHOD.
```

4. **Save** and **Activate** .
   Check out the syntax in the **Source code-based** mode.

Create a ship subclass

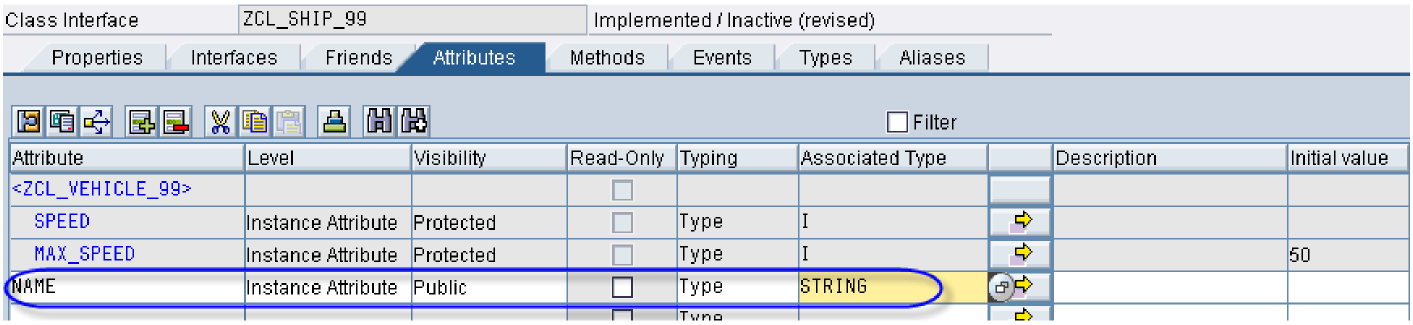
Create a subclass ZCL_SHIP_XX (where XX is your group number).

- The class should have an instance constructor that sets the maximal speed to 30 and that has an import parameter to set an additional read-only attribute to the ship's name.
- It should redefine the method SHOW to produce some specific output (including the ship's name).

Solution

1. Create the subclass ZCL_SHIP_XX in the object navigator as you did before for the truck class (where XX is your group number)..

2. Add a new public instance attribute NAME of type STRING either in **Form-based** mode or in **Source code-based** mode.

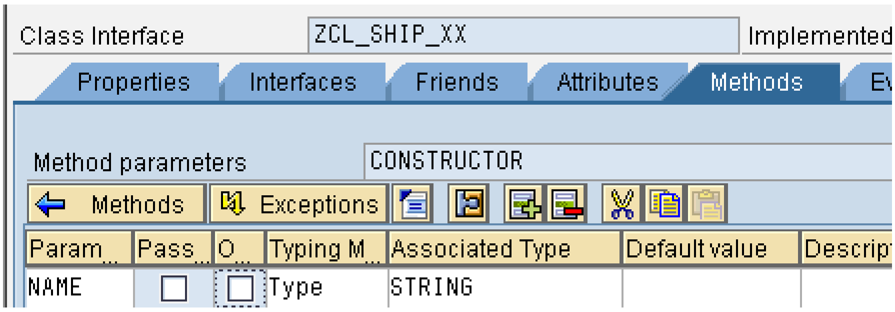


Class Interface ZCL_SHIP_99 Implemented / Inactive (revised)

Attribute	Level	Visibility	Read-Only	Typing	Associated Type		Description	Initial value
<ZCL_VEHICLE_99>			□					
SPEED	Instance Attribute	Protected	□	Type	I	⇨		
MAX_SPEED	Instance Attribute	Protected	□	Type	I	⇨		50
NAME	Instance Attribute	Public	□	Type	STRING	⇨		
			□	Type		⇨		

3. Insert the ship's own constructor as you did for the truck class.

4. Create an importing parameter for the constructor either by selecting **Parameter** in **Form-based** mode



Class Interface ZCL_SHIP_XX Implemented

Method parameters CONSTRUCTOR

Param...	Pass...	O...	Typing M...	Associated Type	Default value	Descrip
NAME	□	□	Type	STRING		

or by adding it in **Source code-based** mode:

```
METHODS constructor
  IMPORTING
    name TYPE string.
```

5. Implement the constructor as follows:

```
METHOD CONSTRUCTOR.
  super->constructor( ).
  max_speed = 30.
  me->name = name.
ENDMETHOD.
```

6. Redefine method SHOW as follows:

```
METHOD show.
  DATA msg TYPE string.
  msg = me->name &&
        `, Speed = `     && |{ speed }| &&
        `, Max-Speed =`  && |{ max_speed }|.
  MESSAGE msg TYPE 'I'.
ENDMETHOD.
```

7. **Save** and **Activate** ..
Check out the syntax in the **Source code-based** mode.

Adjust the application class

The code of the START method should demonstrate the usage of the subclasses now.

- Declare extra reference variables TRUCK and SHIP for the new classes.
- You can delete the code that creates objects for VEHICLE. Instead, create one instance of each of your new subclasses and place the corresponding reference into VEHICLE_TAB.
- Call the method SPEED_UP for both classes using the respective subclass reference, and SHOW using a superclass reference.

Solution

1. Replace the code of method START of ZCL_APPLICATION_XX with the following (where XX is your group number):

```
METHOD start.
  DATA: vehicle  TYPE REF TO zcl_vehicle_xx,
        vehicle_tab LIKE TABLE OF vehicle,
        truck TYPE REF TO zcl_truck_xx,
        ship  TYPE REF TO zcl_ship_xx.
  CREATE OBJECT: truck,
                 ship EXPORTING name = 'Titanic'.
  APPEND: truck TO vehicle_tab,
          ship  TO vehicle_tab.
  truck->speed_up( 30 ).
  ship->speed_up( 10 ).
  LOOP AT vehicle_tab INTO vehicle.
    vehicle->show( ).
  ENDLOOP.
ENDMETHOD.
```

Note the polymorphic method call `vehicle->show( )`.

2. Execute method START from the Class Builder again.

Exercise 3, Interfaces

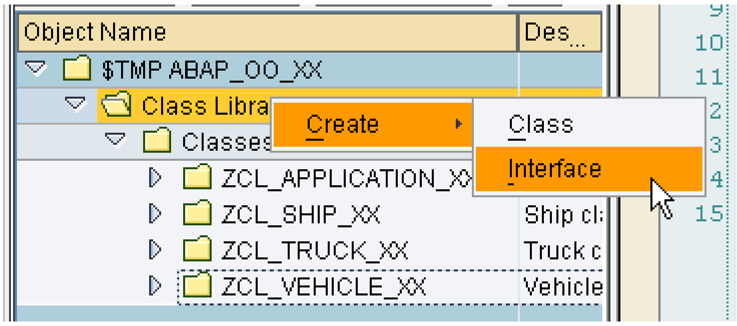
(Solution in Package Z_ABAP_OBJECTS_INTRODUCTION_C)

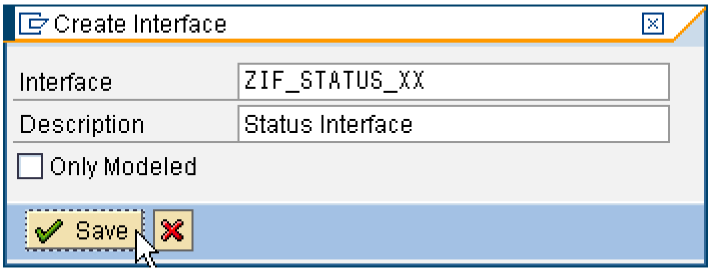
Create a status interface

- Create an interface ZIF_STATUS_XX (where XX is your group number).
- The interface should have one instance method SHOW.

Solution

1. Create the interface in the object navigator as follows:

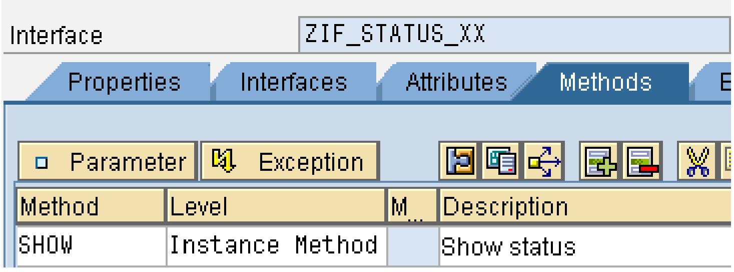




Interface	ZIF_STATUS_XX
Description	Status Interface

□ Only Modeled

2. Define one Method without parameters:



Interface ZIF_STATUS_XX

Method	Level	M...	Description
SHOW	Instance Method		Show status

Save and **Activate** ..

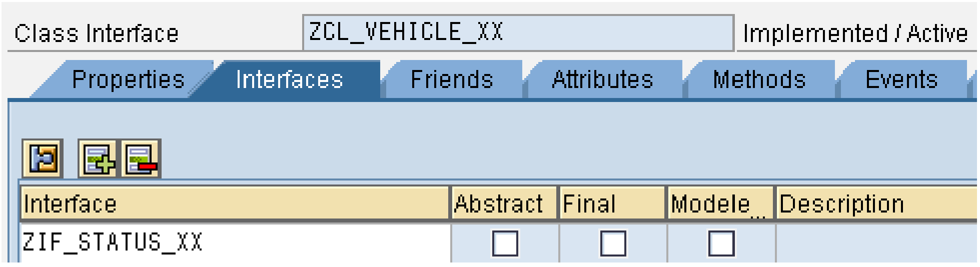
Implement the interface in the superclass

Implement ZIF_STATUS_XX in ZCL_VEHICLE_XX (where XX is your group number).

- Copy the implementation of the class method SHOW to the interface method SHOW.
- Delete class method SHOW
- Create an alias name SHOW for the interface method in order to keep the subclasses valid.

Solution

1. Open ZCL_VEHICLE_XX and enter the interface either in **Form-based** mode:



Class Interface ZCL_VEHICLE_XX Implemented / Active

Interface	Abstract	Final	Modele...	Description
ZIF_STATUS_XX	□	□	□	

or in **Source code-based** mode:

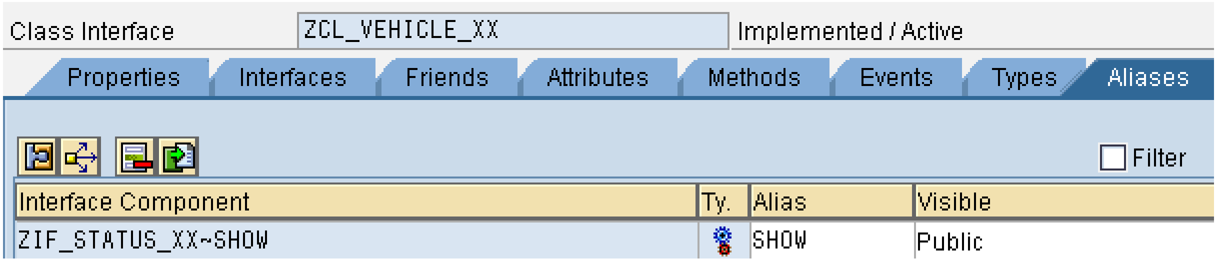
```
  PUBLIC SECTION.
    INTERFACES zif_status_xx.
```

2. Implement the interface method with the code from SHOW:

```
METHOD zif_status_xx~show.
  DATA msg TYPE string.
  msg =  `Vehicle `       && |{ id }| &&
        `, Speed = `      && |{ speed }| &&
        `, Max-Speed = `  && |{ max_speed }|.
  MESSAGE msg TYPE 'I'.
ENDMETHOD.
```

3. Delete ( ) method SHOW.

4. Create a public alias SHOW:



Class Interface ZCL_VEHICLE_XX Implemented / Active

Interface Component	Ty.	Alias	Visible
ZIF_STATUS_XX~SHOW		SHOW	Public

5. **Save** and **Activate** ..

6. Check ZCL_TRUCK_XX and ZCL_SHIP_XX (where XX is your group number). They redefine the superclass interface method SHOW via the alias SHOW now and there should be no errors.

Create a new helicopter class that also implements the interface

Create a new class ZCL_HELICOPTER_XX (where XX is your group number) that is not part of the ZCL_VEHICLE_XX inheritance tree but that also implements STATUS.

Solution

1. Create class ZCL_HELICOPTER_XX (where XX is your group number) as follows:

```
CLASS zcl_helicopter_xx DEFINITION PUBLIC FINAL  CREATE PUBLIC .
  PUBLIC SECTION.
    INTERFACES zif_status_xx.
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.

CLASS zcl_helicopter_xx IMPLEMENTATION.
  METHOD zif_status_xx~show.
    DATA msg TYPE string.
    msg = `Helicopter, idle`.
    MESSAGE msg TYPE 'I'.
  ENDMETHOD.
ENDCLASS.
```

2. **Save** and **Activate** .

Adjust the application class

The code of the START method should demonstrate the usage of the interface now.

- Declare a reference variable HELI for the class ZCL_HELICOPTER_XX following (where XX is your group number) and create a corresponding object.
- Replace the reference variable VEHICLE and the table VEHICLE_TAB with an interface reference STATUS and an internal table STATUS_TAB.
- Insert the reference variables for truck, ship and helicopter into the table STATUS_TAB.

Solution

1. Replace the code of method START of ZCL_APPLICATION_XX with the following (where XX is your group number):

```
  METHOD start.
    DATA: status     TYPE REF TO zif_status_xx,
          status_tab LIKE TABLE OF status,
          truck   TYPE REF TO zcl_truck_xx,
          ship    TYPE REF TO zcl_ship_xx,
          heli    TYPE REF TO zcl_helicopter_xx.
    CREATE OBJECT: truck,
                   ship EXPORTING name = 'Titanic',
                   heli.
    APPEND: truck TO status_tab,
            ship  TO status_tab,
            heli  TO status_tab.
    truck->speed_up( 30 ).
    ship->speed_up( 10 ).
    LOOP AT status_tab INTO status.
      status->show( ).
    ENDLOOP.
  ENDMETHOD.
```

Note the polymorphic method call `status->show( )`.

2. Execute method START from the Class Builder again.

Exercise 4, Events

(Solution in Package Z_ABAP_OBJECTS_INTRODUCTION_D)

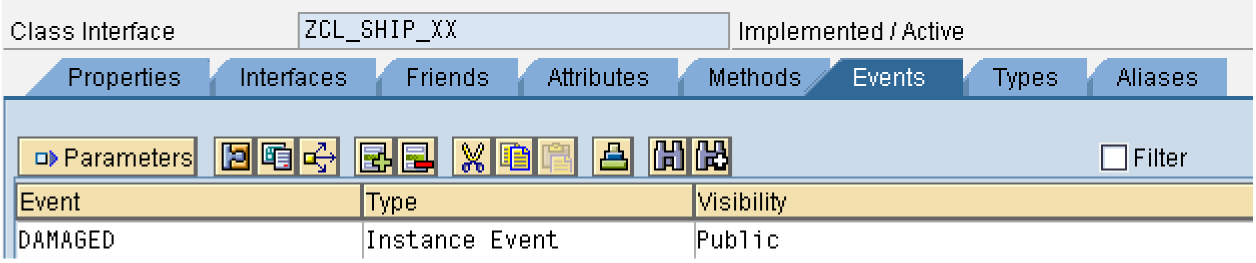
Define an event in the ship class

Objects of ZCL_SHIP_XX (where XX is your group number) should raise an event if the speed becomes higher than the maximal speed.

- Define an instance event DAMAGED in ZCL_SHIP_XX
- Raise the event in method SPEED_UP.

Solution

1. Open ZCL_SHIP_XX (where XX is your group number) in the Class Builder and define an instance event DAMAGED either in **Form-based** mode:



Class Interface ZCL_SHIP_XX Implemented / Active

Event	Type	Visibility
DAMAGED	Instance Event	Public

or in **Source code-based** mode:

```
PUBLIC SECTION.
  ...
  EVENTS damaged.
```

2. Redefine ( ) method SPEED_UP and implement it as follows:

```
METHOD SPEED_UP.
  speed = speed + step.
    IF speed > max_speed.
      max_speed = 0.
      CALL METHOD stop.
      RAISE EVENT damaged.
    ENDIF.
ENDMETHOD.
```

3. **Save** and **Activate** .

Define an event handler method in the helicopter class

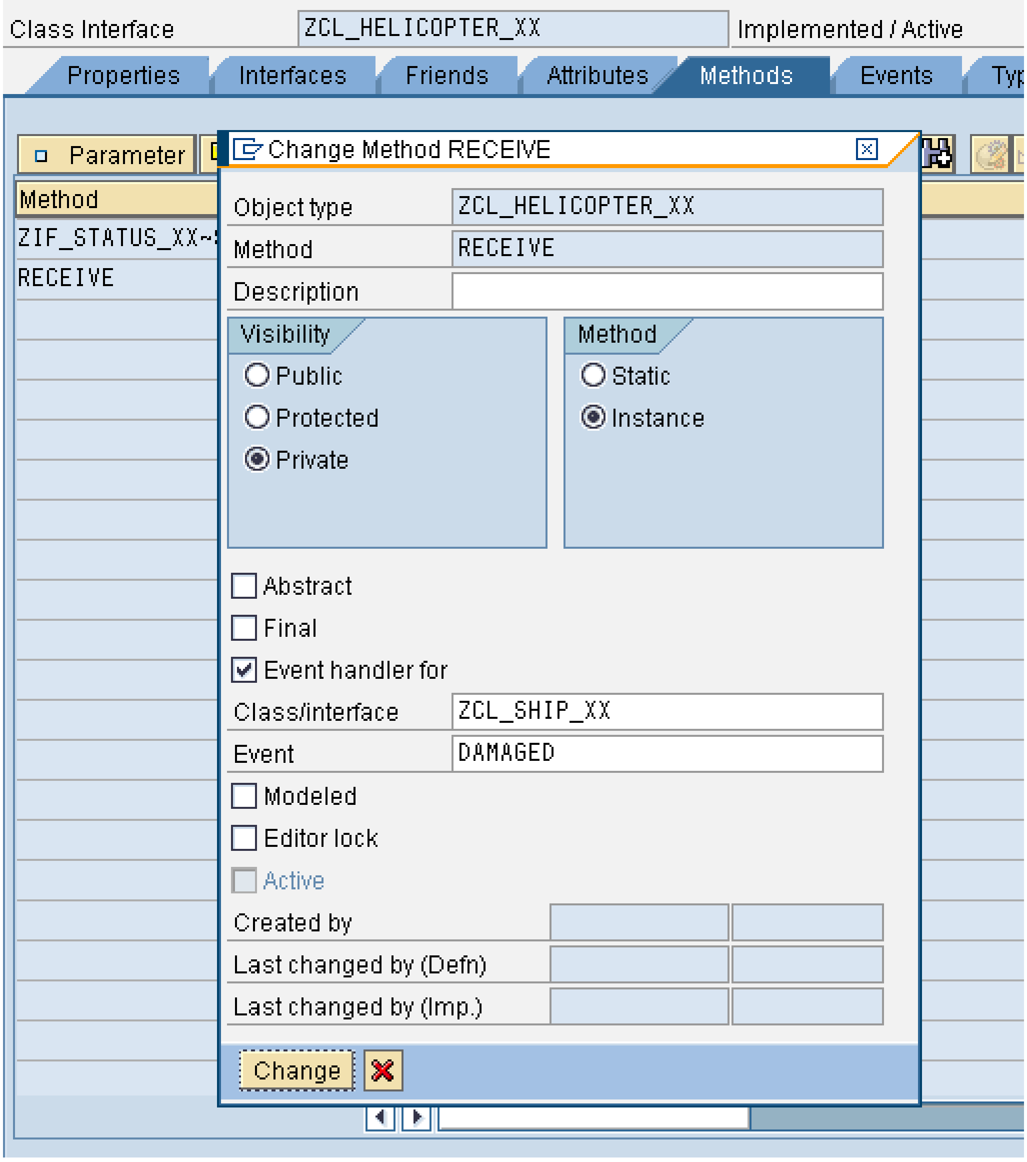
The helicopter class should be able to handle the event DAMAGED of the ship class.

- Define a public method RECEIVE as an event handler for event DAMAGED of ZCL_SHIP_XX in ZCL_HELICOPTER_XX (where XX is your group number).
- Implement the event handler in a way that is simply sends a message that uses the default event parameter SENDER to address the name of the damaged ship .

Solution

1. Open ZCL_HELICOPTER_XX (where XX is your group number) in the class builder.

   In **Form-based** mode, enter a new public method RECEIVE and select **Detail view** (⧉) where you can define the method as an **Event handler**:



Class Interface ZCL_HELICOPTER_XX Implemented / Active

Change Method RECEIVE

Object type	ZCL_HELICOPTER_XX
Method	RECEIVE
Description	

Visibility: ○ Public ○ Protected ● Private

Method: ○ Static ● Instance

□ Abstract

□ Final

☑ Event handler for

Class/interface	ZCL_SHIP_XX
Event	DAMAGED

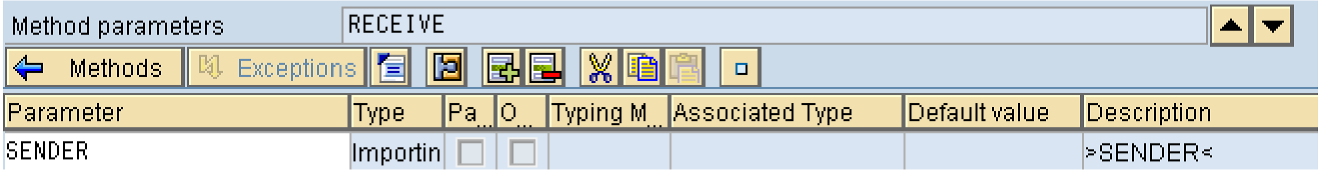
□ Modeled

□ Editor lock

□ Active

Created by		
Last changed by (Defn)		
Last changed by (Imp.)		

and define the predefined importing parameter SENDER:



Method parameters RECEIVE

Parameter	Type	Pa...	O...	Typing M...	Associated Type	Default value	Description
SENDER	Importin	□	□				>SENDER<

In **Source code-based** mode it is enough to type:

```
METHODS receive
  FOR EVENT damaged OF zcl_ship_xx.
    IMPORTING sender.
```

2. Implement method RECEIVE as follows:

```
METHOD receive.
  DATA msg TYPE string.
  msg = `Helicopter received call from ` && sender->name.
  MESSAGE msg TYPE 'I'.
ENDMETHOD.
```

3. **Save** and **Activate** .

Adjust the application class

The code of the START method should register the instances of the helicopter class for the events of the ship class.

- Use statement SET HANDLER to register the event handler.
- Call method SPEED_UP of the ship object in order to raise the event.

Solution

1. Replace the code of method START of ZCL_APPLICATION_XX with the following (where XX is your group number):

```
METHOD start.
  DATA: status     TYPE REF TO zif_status_xx,
            status_tab LIKE TABLE OF status,
            truck  TYPE REF TO zcl_truck_xx,
            ship   TYPE REF TO zcl_ship_xx,
            heli   TYPE REF TO zcl_helicopter_xx.
  CREATE OBJECT: truck,
                  ship EXPORTING name = 'Titanic',
                  heli.
  APPEND: truck TO status_tab,
          ship  TO status_tab,
          heli  TO status_tab.
  SET HANDLER heli->receive FOR ALL INSTANCES.
  truck->speed_up( 30 ).
  ship->speed_up( 10 ).
  LOOP AT status_tab INTO status.
    status->show( ).
  ENDLOOP.
  DO 5 TIMES.
    ship->speed_up( 10 ).
  ENDDO.
ENDMETHOD.
```

2. Execute method START from the Class Builder again.

Exercise 5, Exceptions

(Solution in Package Z_ABAP_OBJECTS_INTRODUCTION_E)

Define an exception class

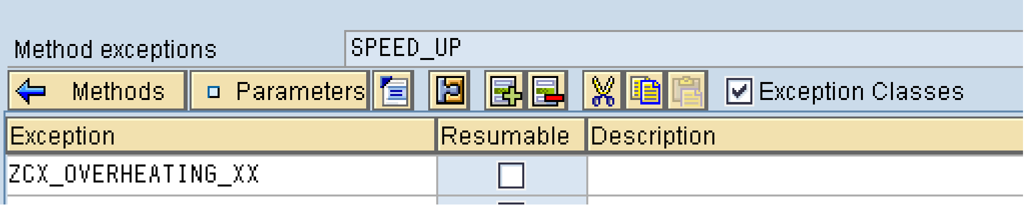
Objects of ZCL_VEHICLE_XX should raise an exception ZCX_OVERHEATING_XX (where XX is your group number) if the speed becomes higher than the maximal speed.

- Define an exception ZCX_OVERHEATING_XX (where XX is your group number)
- Raise the exception in method SPEED_UP.

Solution

1. Open ZCL_VEHICLE_XX (where XX is your group number) in the Class Builder, place the cursor on method SPEED_UP in **Form-based** mode and select the button **Exception**.

2. Mark **Exception Classes** and enter ZCX_OVERHEATING_XX (where XX is your group number).

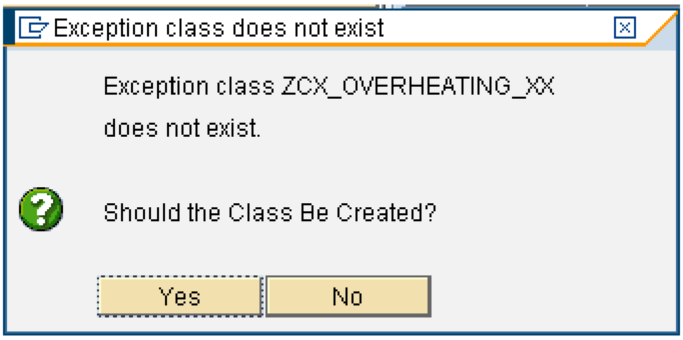


Method exceptions: SPEED_UP

☑ Exception Classes

Exception	Resumable	Description
ZCX_OVERHEATING_XX	□	

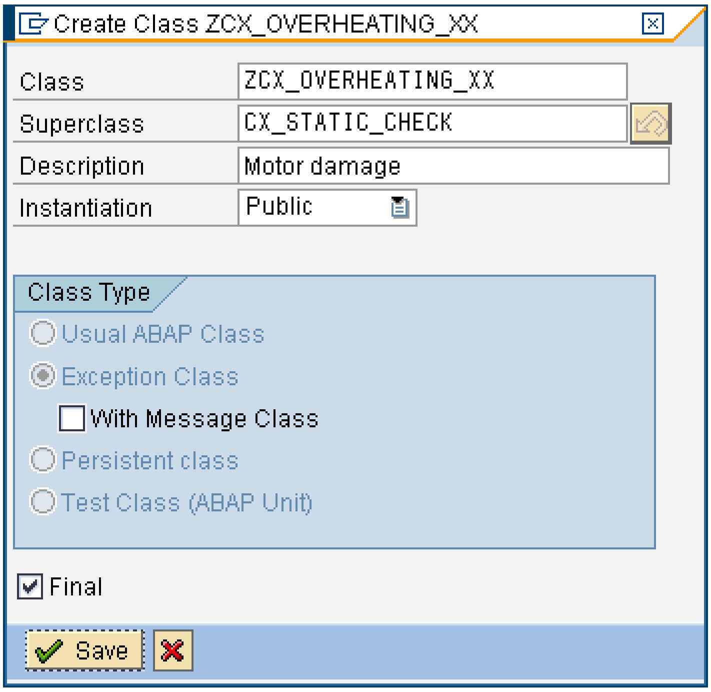
3. **Save** and select **Yes** and **Save** on the following pop-ups with respectively.



Exception class does not exist

Exception class ZCX_OVERHEATING_XX does not exist.

Should the Class Be Created?



Create Class ZCX_OVERHEATING_XX

Class	ZCX_OVERHEATING_XX
Superclass	CX_STATIC_CHECK
Description	Motor damage
Instantiation	Public

Class Type

○ Usual ABAP Class

● Exception Class

□ With Message Class

○ Persistent class

○ Test Class (ABAP Unit)

☑ Final

4. Double click the name of exception class, **Activate** it and return to the vehicle class.

5. Change the implementation if method SPEED_UP as follows (where XX is your group number).:

```
METHOD speed_up.
  speed = speed + step.
  IF speed > max_speed.
    RAISE EXCEPTION TYPE zcx_overheating_xx.
  ENDIF.
ENDMETHOD.
```

6. **Save** and **Activate** .

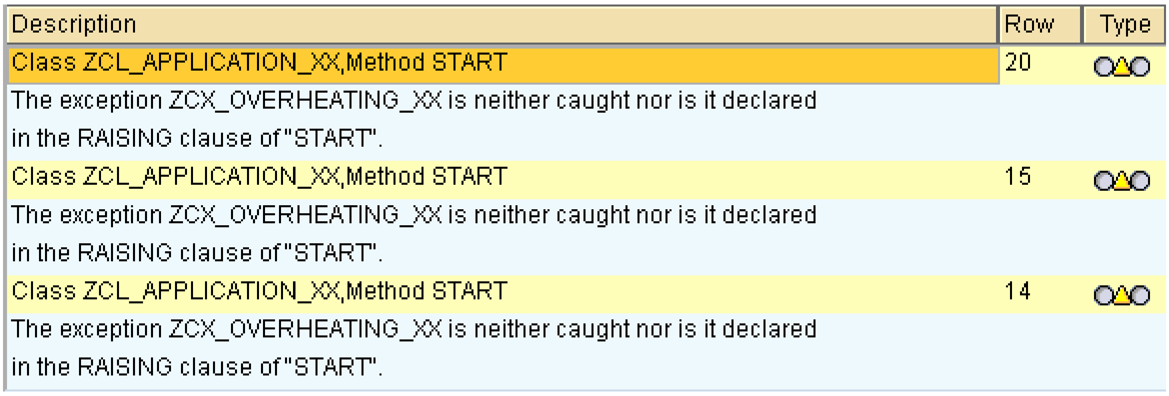
Adjust the application class

The code of the START method must handle the exception now.

- Use a TRY control structure to handle the exception ZCX_OVERHEATING_XX (where XX is your group number)
- Call method SPEED_UP of the truck object in order to raise the exception.

Solution

1. Check the syntax of method START of ZCL_APPLICATION_XX (where XX is your group number). You should get the following warnings:



Description	Row	Type
Class ZCL_APPLICATION_XX,Method START	20	⚠
The exception ZCX_OVERHEATING_XX is neither caught nor is it declared in the RAISING clause of "START".		
Class ZCL_APPLICATION_XX,Method START	15	⚠
The exception ZCX_OVERHEATING_XX is neither caught nor is it declared in the RAISING clause of "START".		
Class ZCL_APPLICATION_XX,Method START	14	⚠
The exception ZCX_OVERHEATING_XX is neither caught nor is it declared in the RAISING clause of "START".		

2. Adjust the implementation of method START as follows (where XX is your group number):

```
METHOD start.
  DATA: status      TYPE REF TO zif_status_xx,
        status_tab LIKE TABLE OF status,
        truck   TYPE REF TO zcl_truck_xx,
        ship    TYPE REF TO zcl_ship_xx,
        heli    TYPE REF TO zcl_helicopter_xx.
  CREATE OBJECT: truck,
                  ship EXPORTING name = 'Titanic',
                  heli.
  APPEND: truck TO status_tab,
          ship  TO status_tab,
          heli  TO status_tab.
  SET HANDLER heli->receive FOR ALL INSTANCES.
  TRY.
      truck->speed_up( 30 ).
      ship->speed_up( 10 ).
      LOOP AT status_tab INTO status.
        status->show( ).
      ENDLOOP.
      DO 5 TIMES.
```

```
        ship->speed_up( 10 ).
      ENDDO.
      DO 5 TIMES.
        truck->speed_up( 30 ).
        truck->show( ).
      ENDDO.
    CATCH zcx_overheating_xx.
      MESSAGE 'Truck overheated' TYPE 'I'.
      EXIT.
  ENDTRY.
ENDMETHOD.
```

No syntax warnings should occur any more.

3. Execute method START from the Class Builder again.

Exercise 6, Unit Tests

(Solution in Package Z_ABAP_OBJECTS_INTRODUCTION_F)

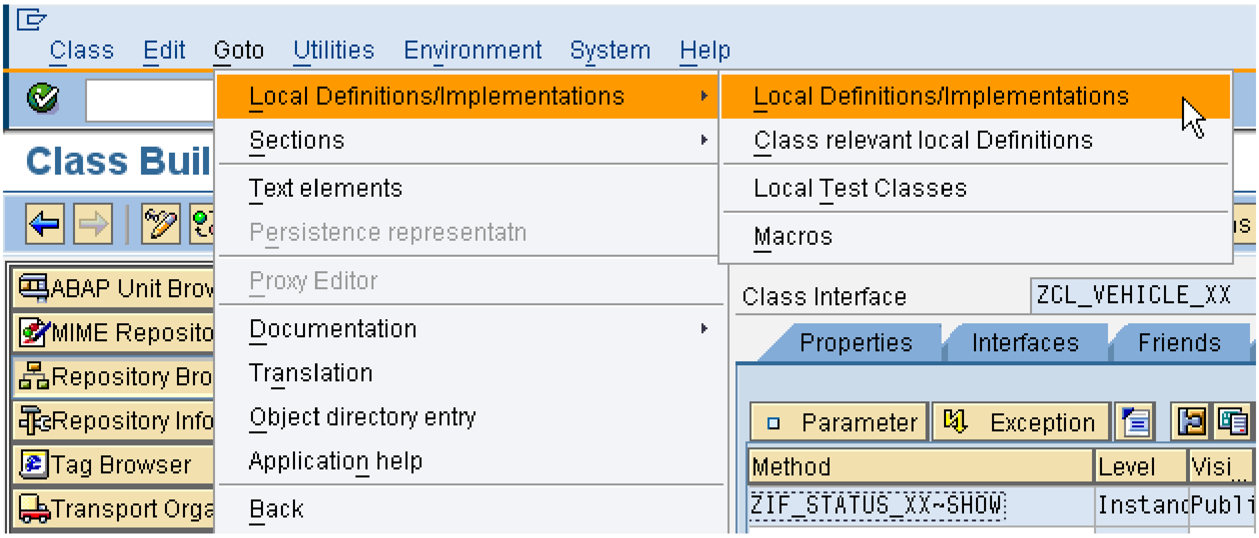
Define a test class with a test method for the vehicle class

Class ZCL_VEHICLE_XX (where XX is your group number) should contain a test class that tests the SPEED_UP method completely.

- Declare and implement a local test class TEST_VEHICLE in the vehicle class.

Solution

1. Open ZCL_VEHICLE_XX (where XX is your group number) in the Class Builder an select the following:



2. Enter the following code there (where XX is your group number):

```
CLASS test_vehicle DEFINITION DEFERRED.
CLASS zcl_vehicle_xx DEFINITION LOCAL FRIENDS test_vehicle.
```

The second statement is the important one. It declares the test class as friend of the global class in order to have access of the vehicle’s private components. The first statement is necessary for the ABAP Compiler. (select F1 to learn more).

3. Navigate back to the Class Builder’s main screen and select **Local Test Classes** in the above menu, answer the pop-up with **Yes**, and implement your test class as follows (where XX is your group number):

```
CLASS test_vehicle DEFINITION FOR TESTING
                   RISK LEVEL HARMLESS
                   DURATION SHORT.
  PRIVATE SECTION.
    DATA vehicle TYPE REF TO zcl_vehicle_xx.
    METHODS: test_speed_up FOR TESTING,
             setup,
             teardown.
ENDCLASS.

CLASS test_vehicle IMPLEMENTATION.
  METHOD setup.
    CREATE OBJECT vehicle.
  ENDMETHOD.
  METHOD test_speed_up.
    TRY.
        vehicle->speed_up( 50 ).
        cl_abap_unit_assert=>assert_equals(
          EXPORTING
            exp   =  50
            act   =  vehicle->speed
            msg   =  'Speed not as expexted'
            level = if_aunit_constants=>critical ).
      CATCH zcx_overheating_xx.
        cl_abap_unit_assert=>fail(
          EXPORTING
            msg   = 'No exception expected'
            level = if_aunit_constants=>critical ).
    ENDTRY.
    TRY.
        vehicle->speed_up( 1000 ).
        cl_abap_unit_assert=>fail(
          EXPORTING
            msg   = 'Exception expected'
            level = if_aunit_constants=>critical ).
      CATCH zcx_overheating_xx.
    ENDTRY.
  ENDMETHOD.
  METHOD teardown.
    CLEAR vehicle.
  ENDMETHOD.
ENDCLASS.
```

Save

4. Navigate back to the Class Builder’s main screen, **Save** and **Activate**.

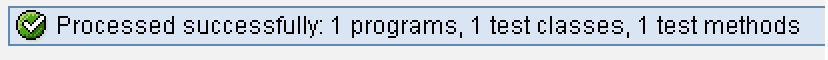
5. Carry out the test method by

Class -> Unit Test in the Class Builder menu

or

Test -> Unit Test from the context menu of the class in the Repository Browser

The result should be:



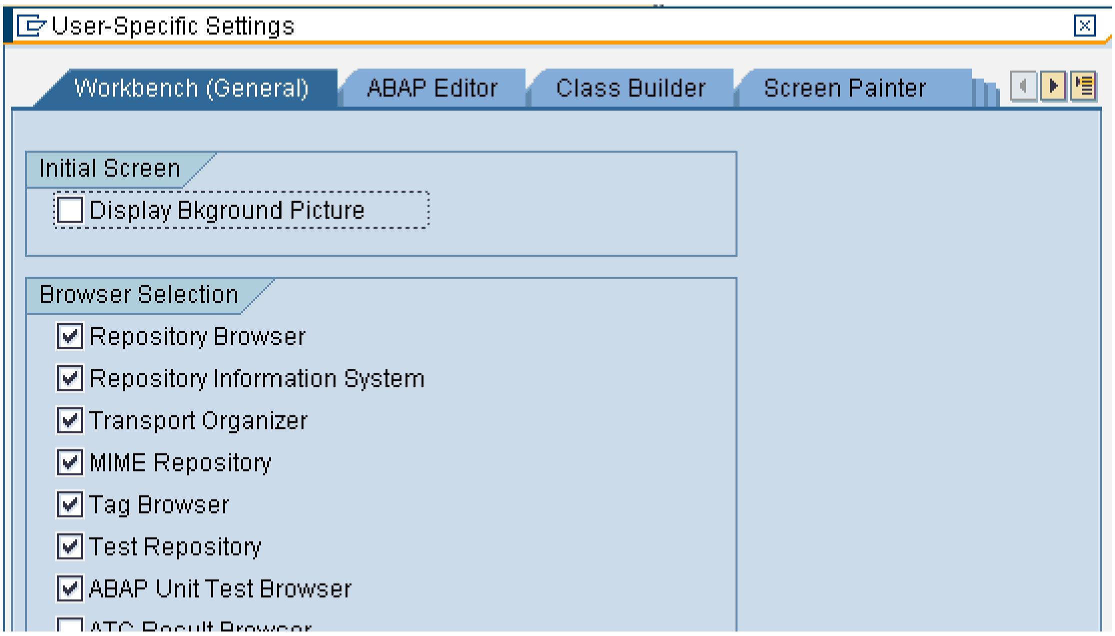
Processed successfully: 1 programs, 1 test classes, 1 test methods

Use the ABAP Unit Browser

Examine the possibilities of the ABAP Unit Browser embedded in the Object Navigator

Solution

1. Select **Utilities -> Settings** in the Object Navigator and additionally select the **ABAP Unit Test Browser** under **Workbench (General)**:



User-Specific Settings

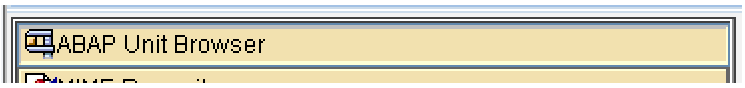
Initial Screen

□ Display Bkground Picture

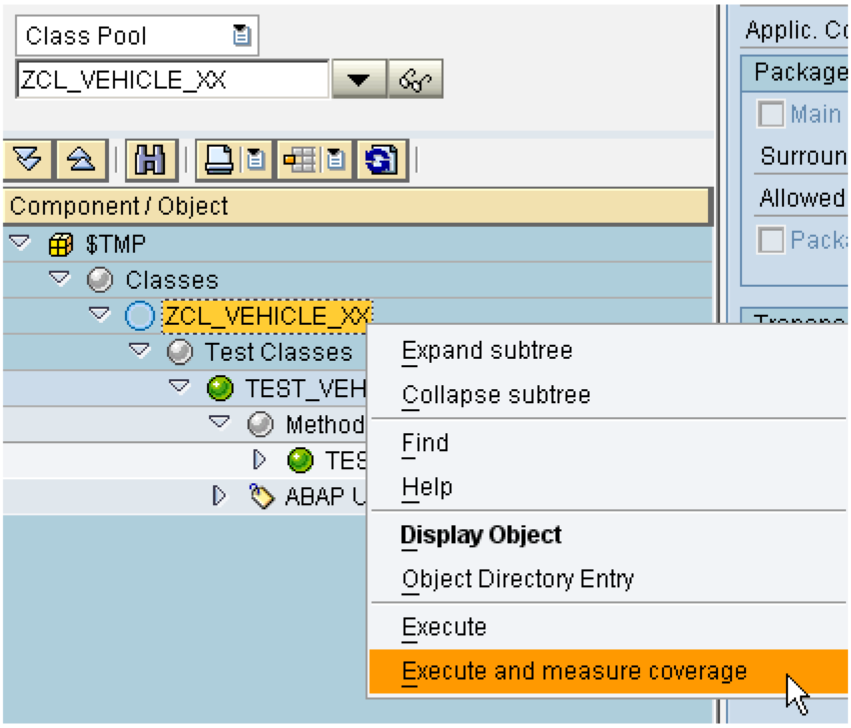
Browser Selection

☑ Repository Browser
☑ Repository Information System
☑ Transport Organizer
☑ MIME Repository
☑ Tag Browser
☑ Test Repository
☑ ABAP Unit Test Browser
□ ATC Result Browser

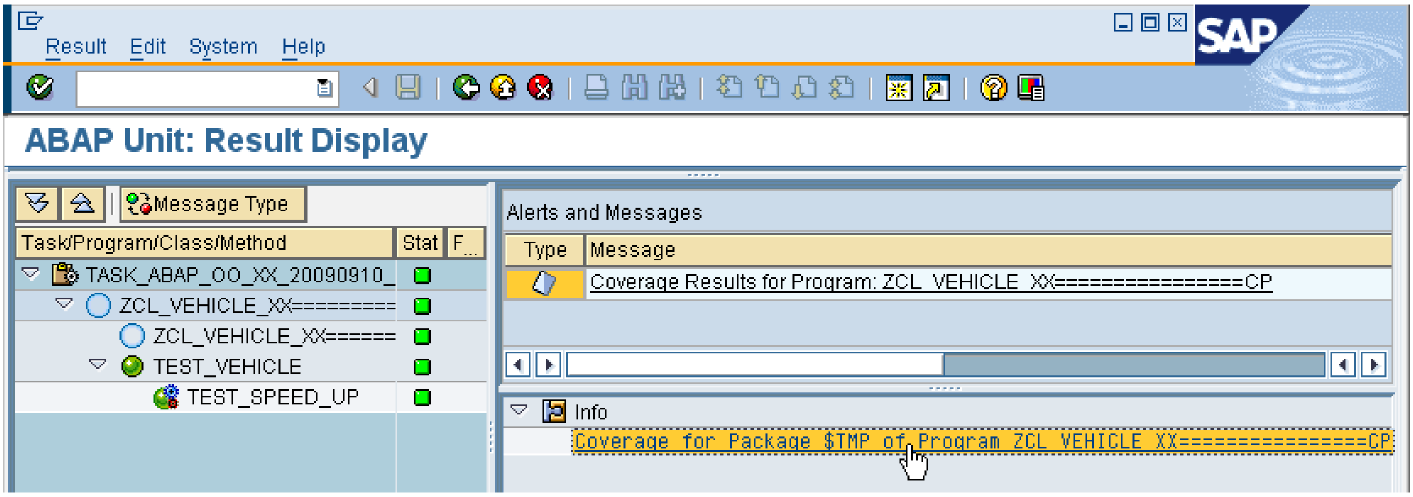
2. Select the ABAP Unit Browser:



3. Select Class Pool, enter ZCL_VEHICLE_XX (where XX is your group number), and carry out the test with measuring the test coverage:

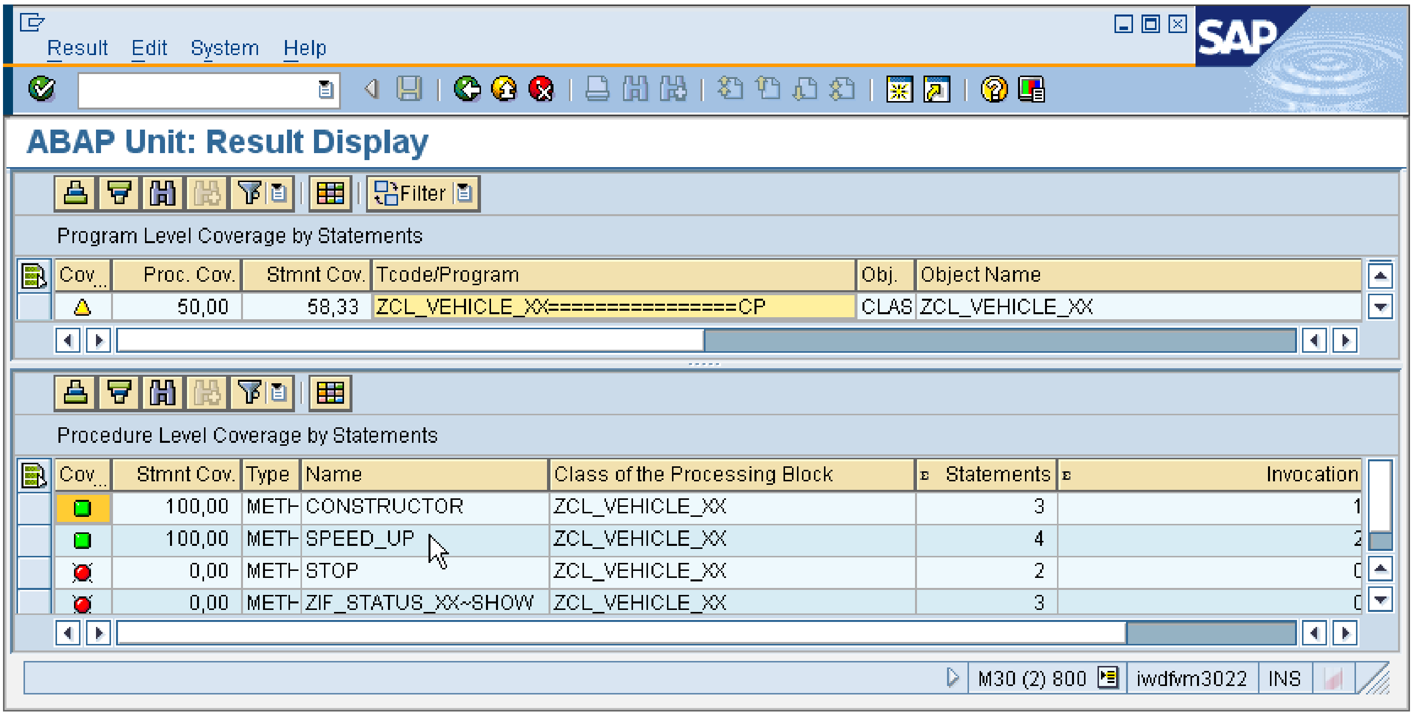


4. After successful execution you can navigate through the results to view the test coverage:



Coverage Results for Program: ZCL_VEHICLE_XX===============CP

Coverage for Package $TMP of Program ZCL_VEHICLE_XX===============CP



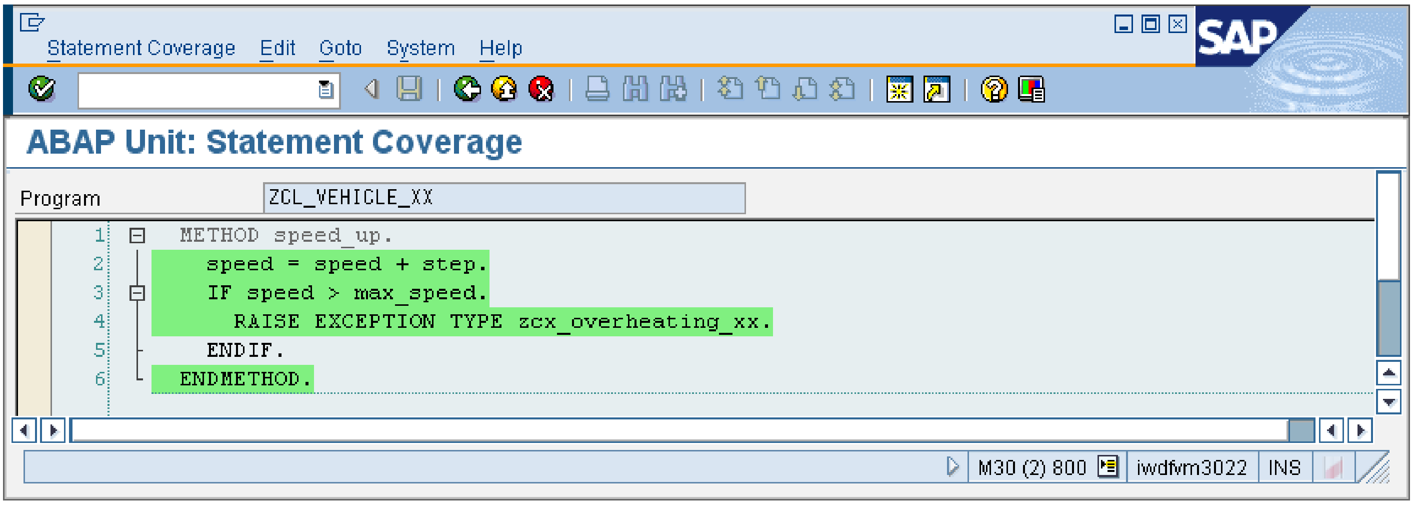
ABAP Unit: Result Display

Program Level Coverage by Statements

Cov...	Proc. Cov.	Stmnt Cov.	Tcode/Program	Obj.	Object Name
⚠	50,00	58,33	ZCL_VEHICLE_XX================CP	CLAS	ZCL_VEHICLE_XX

Procedure Level Coverage by Statements

Cov...	Stmnt Cov.	Type	Name	Class of the Processing Block	Σ Statements	Σ Invocation
🟩	100,00	METH	CONSTRUCTOR	ZCL_VEHICLE_XX	3	1
🟩	100,00	METH	SPEED_UP	ZCL_VEHICLE_XX	4	2
🔴	0,00	METH	STOP	ZCL_VEHICLE_XX	2	0
🔴	0,00	METH	ZIF_STATUS_XX~SHOW	ZCL_VEHICLE_XX	3	0



ABAP Unit: Statement Coverage

Program ZCL_VEHICLE_XX

```
METHOD speed_up.
  speed = speed + step.
  IF speed > max_speed.
    RAISE EXCEPTION TYPE zcx_overheating_xx.
  ENDIF.
ENDMETHOD.
```

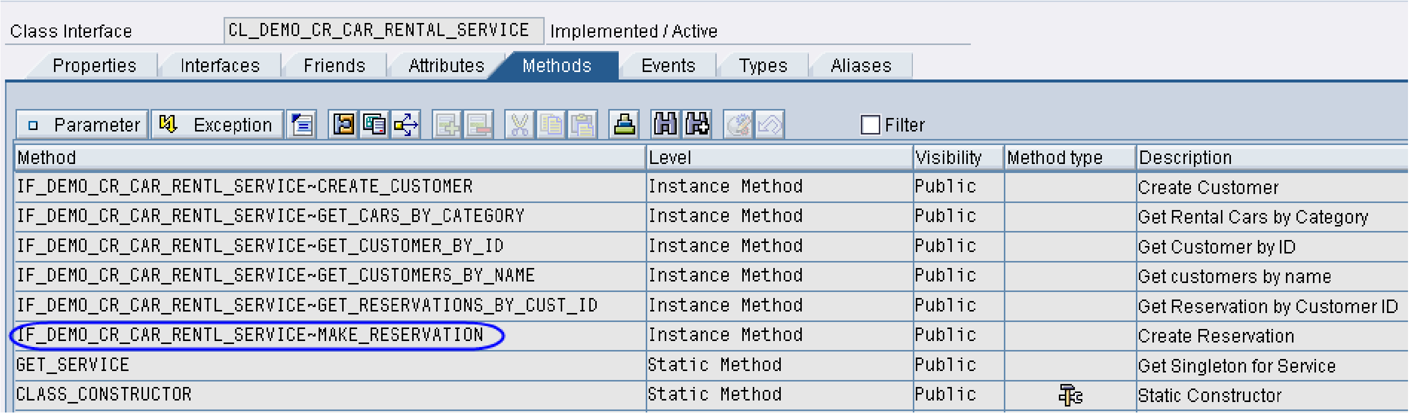
Exercise 7, Service Enablement

(Solution in Package ZABAP_OO_SERVICEENABLEMENT

Expose Method As Web Service (Inside-Out Approach)

In this exercise,

- You will create a RFC-enabled Function Module to invoke the IF_DEMO_CR_CAR_RENTL_SERVICE~MAKE_RESERVATION method from Class CL_DEMO_CR_CAR_RENTAL_SERVICE
- Copy existing RFC-enabled Function Module into your Function Group
- Expose the RFC-enabled Function Modules in the Function Group as a Web Service
- Configure/Test the web service using SOAManager and Web Service Navigator
- Optionally, debug using an external breakpoint

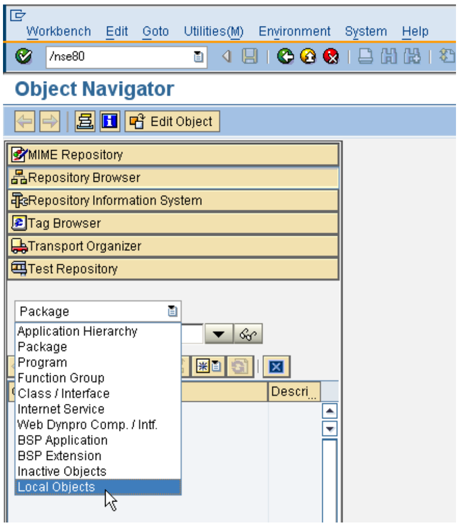


Class Interface CL_DEMO_CR_CAR_RENTAL_SERVICE Implemented / Active

Method	Level	Visibility	Method type	Description
IF_DEMO_CR_CAR_RENTL_SERVICE~CREATE_CUSTOMER	Instance Method	Public		Create Customer
IF_DEMO_CR_CAR_RENTL_SERVICE~GET_CARS_BY_CATEGORY	Instance Method	Public		Get Rental Cars by Category
IF_DEMO_CR_CAR_RENTL_SERVICE~GET_CUSTOMER_BY_ID	Instance Method	Public		Get Customer by ID
IF_DEMO_CR_CAR_RENTL_SERVICE~GET_CUSTOMERS_BY_NAME	Instance Method	Public		Get customers by name
IF_DEMO_CR_CAR_RENTL_SERVICE~GET_RESERVATIONS_BY_CUST_ID	Instance Method	Public		Get Reservation by Customer ID
IF_DEMO_CR_CAR_RENTL_SERVICE~MAKE_RESERVATION	Instance Method	Public		Create Reservation
GET_SERVICE	Static Method	Public		Get Singleton for Service
CLASS_CONSTRUCTOR	Static Method	Public		Static Constructor

Solution

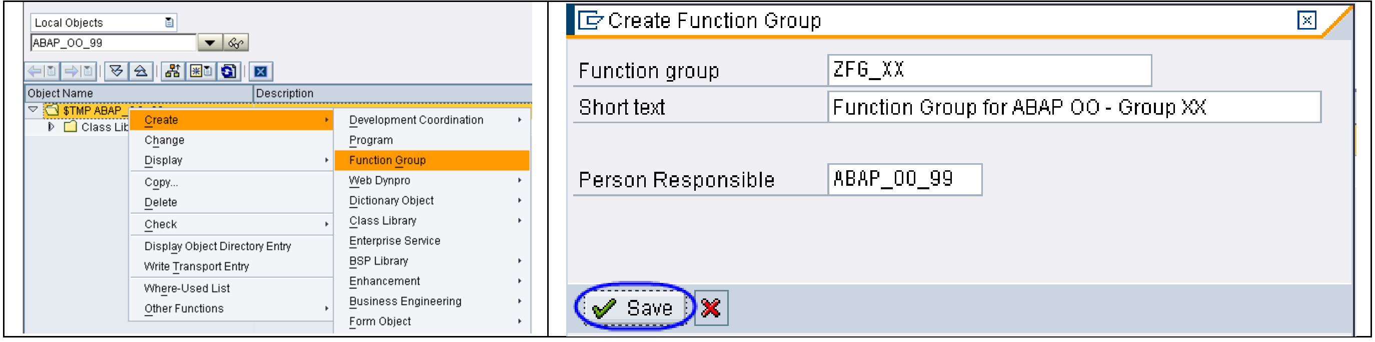
1. Logon to the system and open the Object Navigator of the ABAP Workbench (Transaction SE80, enter /nSE80 in the command field of the system task bar).
2. Select **Local Objects** in order to work in a test package that is not transported to other systems.



/nse80

Hit **Enter.**

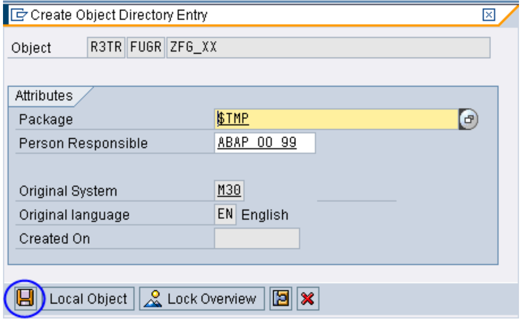
3. Create a Function Group
   - Right Click the name of the local package and navigate to the creation of a Function Group.
   - Enter **ZFG_XX** (where XX is your group number) and short text. Click **Save**.



Create Function Group

Function group: ZFG_XX
Short text: Function Group for ABAP OO - Group XX
Person Responsible: ABAP_OO_99

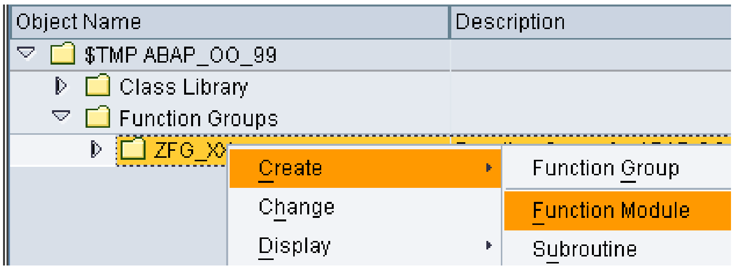
   - Acknowledge the following window without changes (select either Save or Local Object).


Create Object Directory Entry

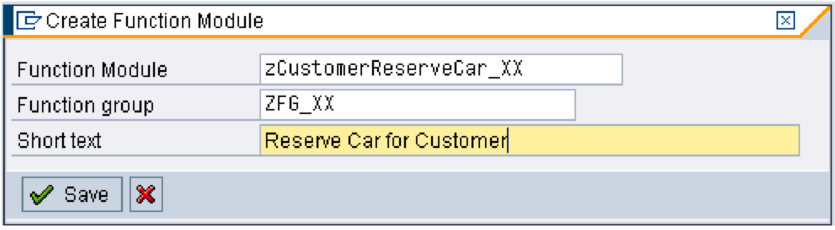
Object: R3TR FUGR ZFG_XX

Attributes
Package: $TMP
Person Responsible: ABAP_OO_99
Original System: M30
Original language: EN English
Created On:

4. Create the function module.
   - Expand the Function Groups Folder. Right click on the function group, **ZFG_XX**, and choose Create->Function Module

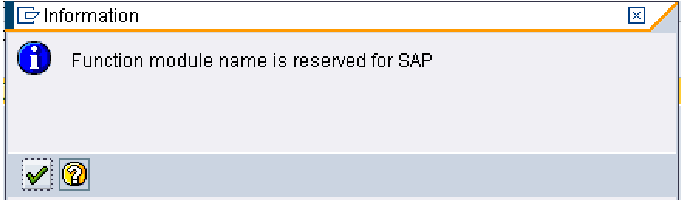


   - Enter **zCustomerReserveCar_XX** (where XX is your group number) and short text. Click **Save**.


Create Function Module

Function Module: zCustomerReserveCar_XX
Function group: ZFG_XX
Short text: Reserve Car for Customer

   - You can disregard the following informational message whenever it may appear

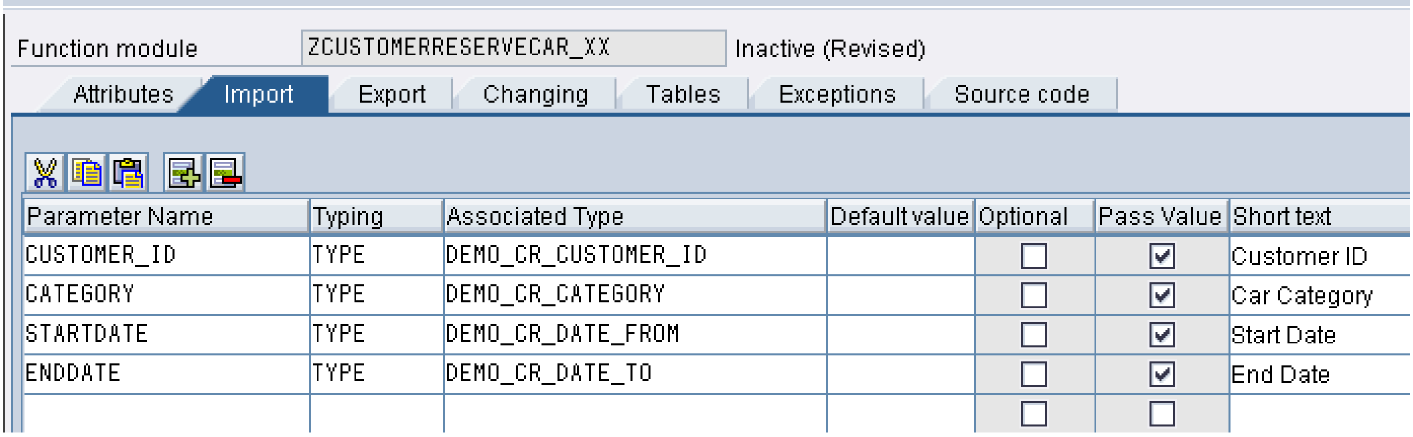

Information

Function module name is reserved for SAP

5. Define the function module interface by entering its parameters

- Provide the import parameters for the function module under **Import** tab

Parameter Name	Typing	Associated Type	Pass Value
CUSTOMER_ID	TYPE	DEMO_CR_CUSTOMER_ID	☑
CATEGORY	TYPE	DEMO_CR_CATEGORY	☑
STARTDATE	TYPE	DEMO_CR_DATE_FROM	☑
ENDDATE	TYPE	DEMO_CR_DATE_TO	☑

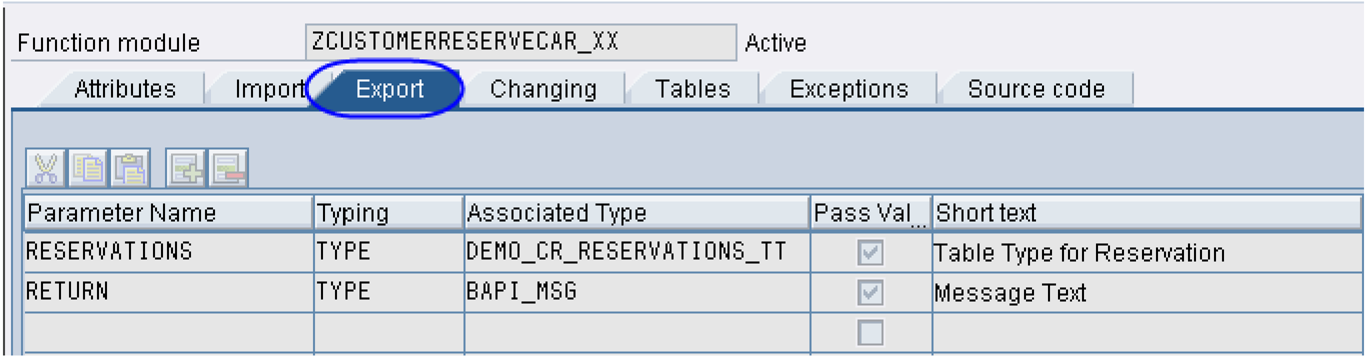


Function module ZCUSTOMERRESERVECAR_XX Inactive (Revised)

Parameter Name	Typing	Associated Type	Default value	Optional	Pass Value	Short text
CUSTOMER_ID	TYPE	DEMO_CR_CUSTOMER_ID		□	☑	Customer ID
CATEGORY	TYPE	DEMO_CR_CATEGORY		□	☑	Car Category
STARTDATE	TYPE	DEMO_CR_DATE_FROM		□	☑	Start Date
ENDDATE	TYPE	DEMO_CR_DATE_TO		□	☑	End Date
				□	□	

- Provide the export parameters for the function module under **Export** tab

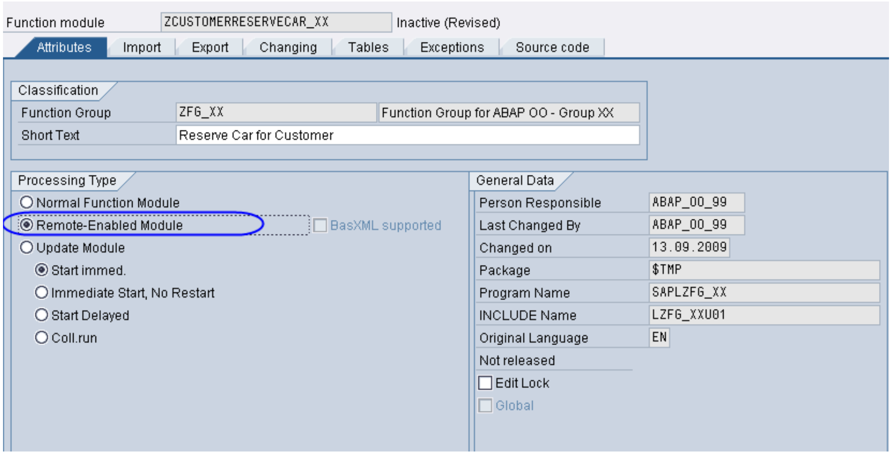
Parameter Name	Typing	Associated Type	Pass Value
RESERVATIONS	TYPE	DEMO_CR_RESERVATIONS_TT	☑
RETURN	TYPE	BAPI_MSG	☑



Function module ZCUSTOMERRESERVECAR_XX Active

Parameter Name	Typing	Associated Type	Pass Val...	Short text
RESERVATIONS	TYPE	DEMO_CR_RESERVATIONS_TT	☑	Table Type for Reservation
RETURN	TYPE	BAPI_MSG	☑	Message Text
			□	

- Please mark “Remote-enabled Module” radio button under **Attributes** tab.



Function module ZCUSTOMERRESERVECAR_XX Inactive (Revised)

Classification

Function Group ZFG_XX Function Group for ABAP OO - Group XX

Short Text Reserve Car for Customer

Processing Type

○ Normal Function Module

● Remote-Enabled Module □ BasXML supported

○ Update Module

● Start immed.

○ Immediate Start, No Restart

○ Start Delayed

○ Coll.run

General Data

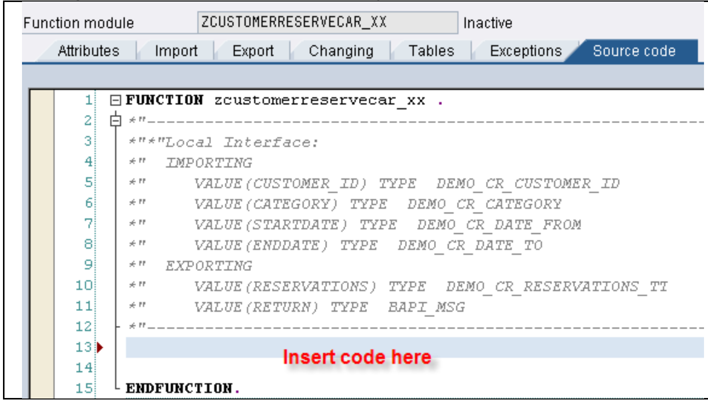
Person Responsible	ABAP_00_99
Last Changed By	ABAP_00_99
Changed on	13.09.2009
Package	$TMP
Program Name	SAPLZFG_XX
INCLUDE Name	LZFG_XXU01
Original Language	EN

Not released

□ Edit Lock

□ Global

6. Complete the source code for your function module under Source Code tab.



```
FUNCTION zcustomerreservecar_xx .
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(CUSTOMER_ID) TYPE  DEMO_CR_CUSTOMER_ID
*"     VALUE(CATEGORY) TYPE  DEMO_CR_CATEGORY
*"     VALUE(STARTDATE) TYPE  DEMO_CR_DATE_FROM
*"     VALUE(ENDDATE) TYPE  DEMO_CR_DATE_TO
*"  EXPORTING
*"     VALUE(RESERVATIONS) TYPE  DEMO_CR_RESERVATIONS_TT
*"     VALUE(RETURN) TYPE  BAPI_MSG
*"----------------------------------------------------------------------


ENDFUNCTION.
```

You can toggle between Display and Change mode using the icon.

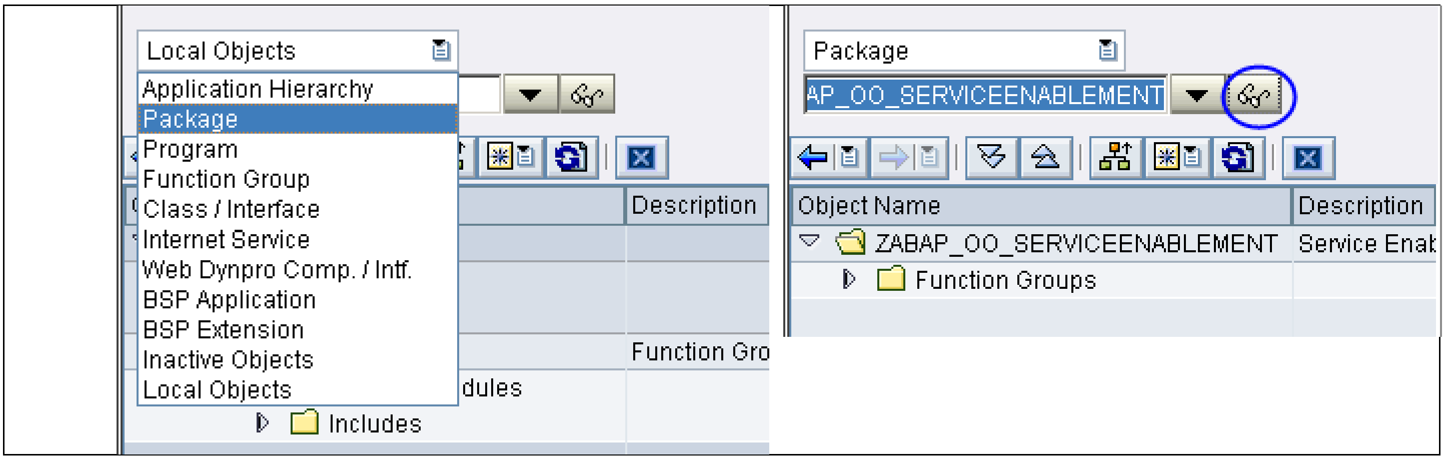
- Add the ABAP source code

```
  DATA: lr_service TYPE REF TO if_demo_cr_car_rentl_service,
        lo_exception TYPE REF TO cx_root,
        l_customer TYPE demo_cr_scustomer,
        l_reservation TYPE demo_cr_sreservation.

  lr_service = cl_demo_cr_car_rental_service=>get_service( ).

  TRY.
      lr_service->make_reservation(
          EXPORTING
            i_customer_id = customer_id
            i_category    = category
            i_date_from   = startdate
            i_date_to     = enddate ).
    CATCH cx_demo_cr_no_customer  cx_demo_cr_lock  cx_demo_cr_reservation INTO lo_exception.
      return = lo_exception->get_text( ).
      EXIT.
    CATCH cx_root  INTO lo_exception.
      return = lo_exception->get_text( ).
      EXIT.
  ENDTRY.

  l_customer = lr_service->get_customer_by_id( customer_id ).

* Method returns ALL reservations for this customer
  reservations = lr_service->get_reservations_by_cust_id( customer_id ).
* Delete the records not matching the requested start and end dates
  DELETE reservations WHERE date_from <> startdate OR date_to <> enddate.
* We still may have the situation where the same reservation request was booked multiple times
  CASE lines( reservations ).
    WHEN 0.
      return = `Unable to confirm reservation - Contact Help Desk`.
    WHEN 1.
      READ TABLE reservations INTO l_reservation INDEX 1.
      return = `Reservation ` && l_reservation-reservation_id  && ` Booked for ` &&  l_customer-name.
    WHEN OTHERS.
      sort reservations by reservation_id ascending.
      return = `Multipe reservation exist for ` && l_customer-name  && ` in the selected time period`.
  ENDCASE.
```

- Check, Save and Activate the Function Module

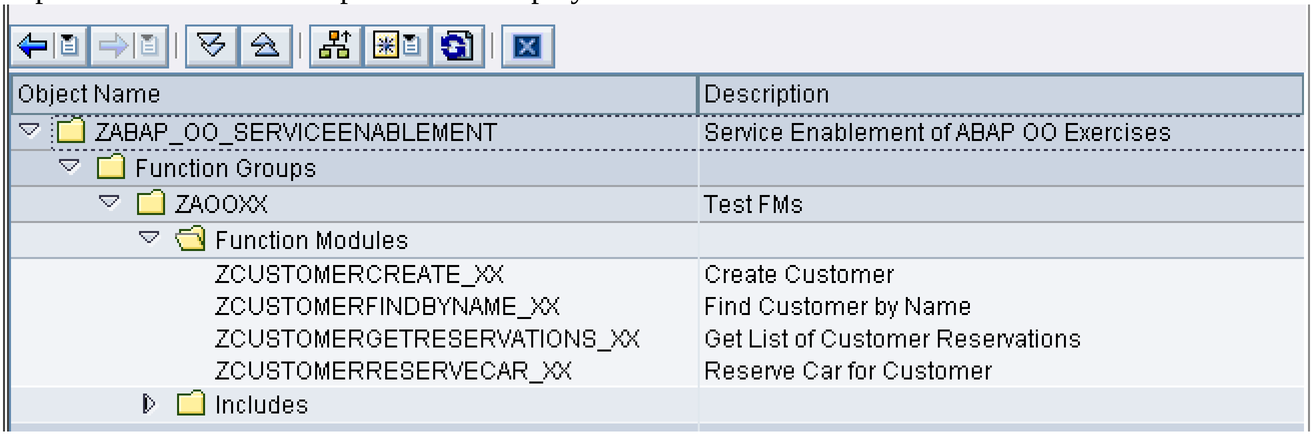
7. Copy 3 additional Function Modules from a different package into your Function Group

- Navigate to Package - ZABAP_OO_SERVICEENABLEMENT click Display



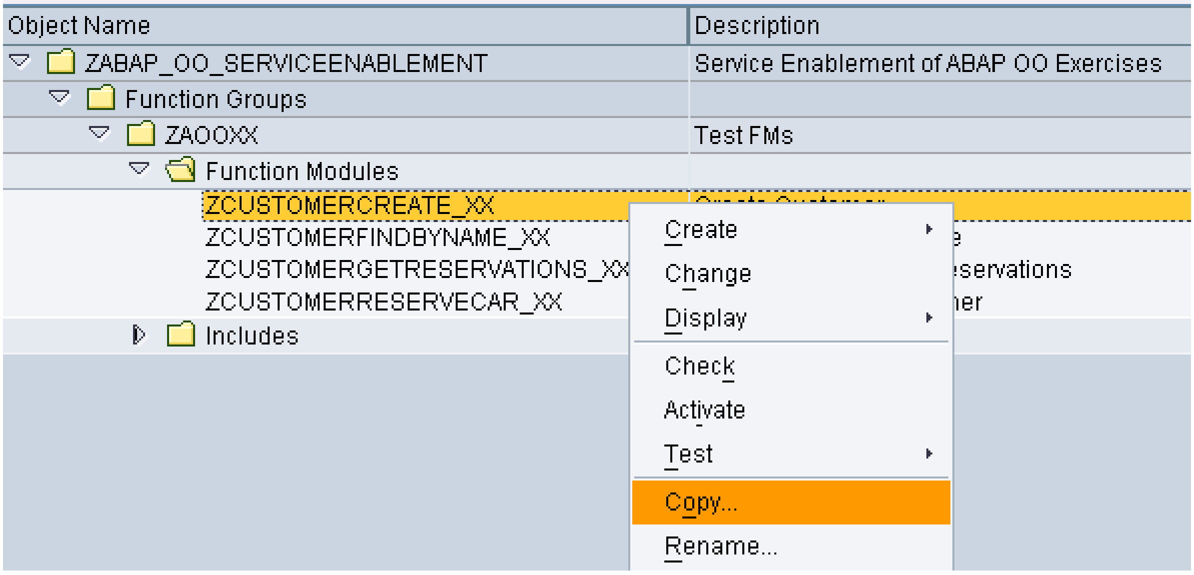
Object Name	Description
ZABAP_OO_SERVICEENABLEMENT	Service Enab
Function Groups	

- Expand the Function Groups folder to display the Function Modules



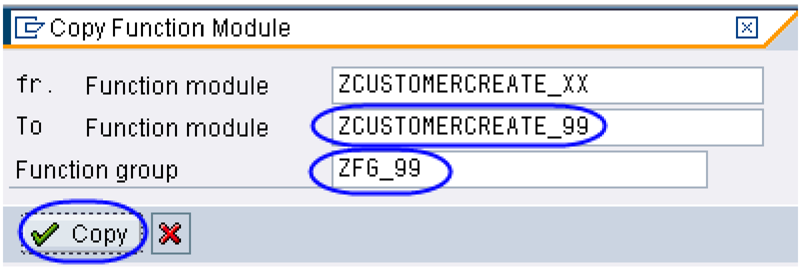
Object Name	Description
ZABAP_OO_SERVICEENABLEMENT	Service Enablement of ABAP OO Exercises
Function Groups	
ZAOOXX	Test FMs
Function Modules	
ZCUSTOMERCREATE_XX	Create Customer
ZCUSTOMERFINDBYNAME_XX	Find Customer by Name
ZCUSTOMERGETRESERVATIONS_XX	Get List of Customer Reservations
ZCUSTOMERRESERVECAR_XX	Reserve Car for Customer
Includes	

- Right-click on the ZCUSTOMERCREATE_XX Function Module and Select Copy



Object Name	Description
ZABAP_OO_SERVICEENABLEMENT	Service Enablement of ABAP OO Exercises
Function Groups	
ZAOOXX	Test FMs
Function Modules	
ZCUSTOMERCREATE_XX	
ZCUSTOMERFINDBYNAME_XX	e
ZCUSTOMERGETRESERVATIONS_XX	servations
ZCUSTOMERRESERVECAR_XX	ner
Includes	

- Enter the Following
  - To Function Module **ZCUSTOMERCREATE_99** (replace 99 with your group number)
  - Function Group **ZFG_99** (replace 99 with your group number)
  - Select **Copy**



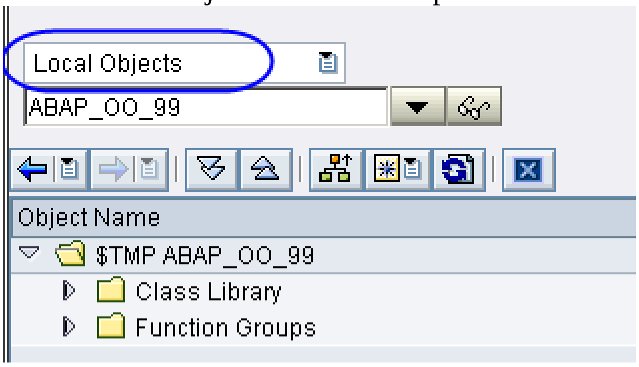
Copy Function Module

fr.	Function module	ZCUSTOMERCREATE_XX
To	Function module	ZCUSTOMERCREATE_99
Function group		ZFG_99

- You can ignore any pop-up informational messages
- Repeat the same steps above to copy 2 more Function Modules remembering to replace 99 with your group number

Copy From	Copy To
ZCUSTOMERFINDBYNAME_XX	ZCUSTOMERFINDBYNAME_99
ZCUSTOMERGETRESERVATIONS_XX	ZCUSTOMERGETRESERVATIONS_99

- Select Local Objects from the dropdown box

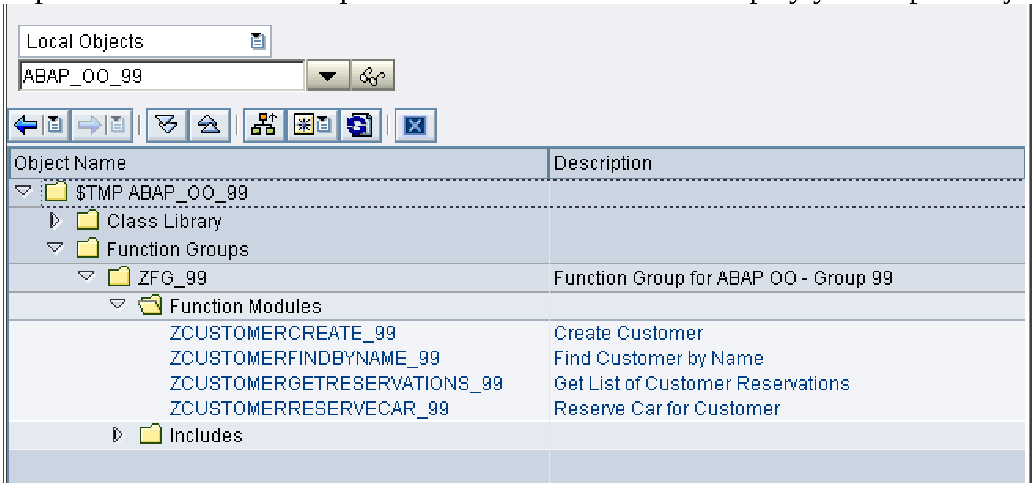


Local Objects

ABAP_OO_99

Object Name
- $TMP ABAP_OO_99
  - Class Library
  - Function Groups

- Expand the Function Groups and Function Modules to Display your copied objects

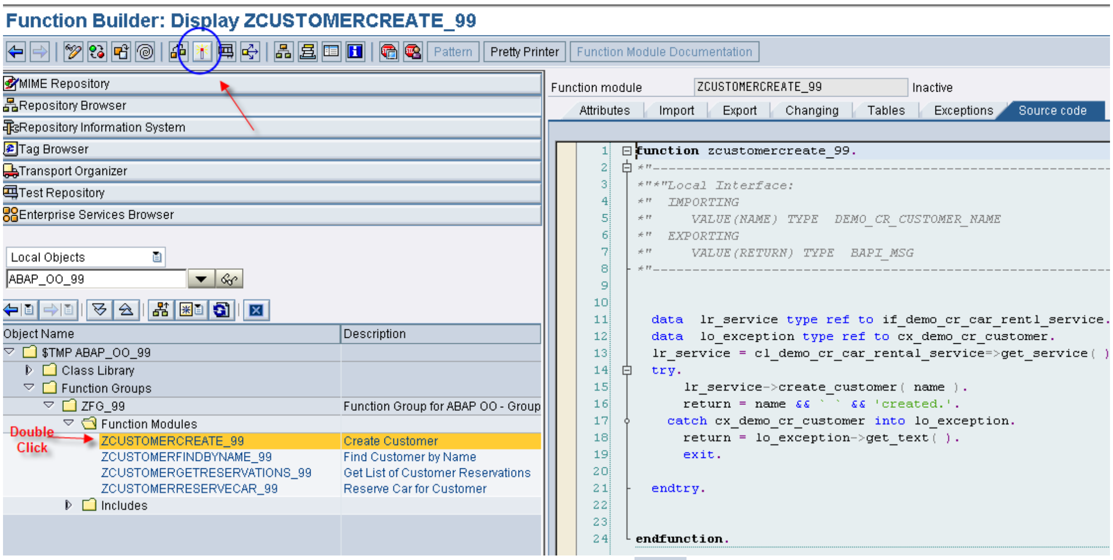


Local Objects

ABAP_OO_99

Object Name	Description
$TMP ABAP_OO_99	
Class Library	
Function Groups	
ZFG_99	Function Group for ABAP OO - Group 99
Function Modules	
ZCUSTOMERCREATE_99	Create Customer
ZCUSTOMERFINDBYNAME_99	Find Customer by Name
ZCUSTOMERGETRESERVATIONS_99	Get List of Customer Reservations
ZCUSTOMERRESERVECAR_99	Reserve Car for Customer
Includes	

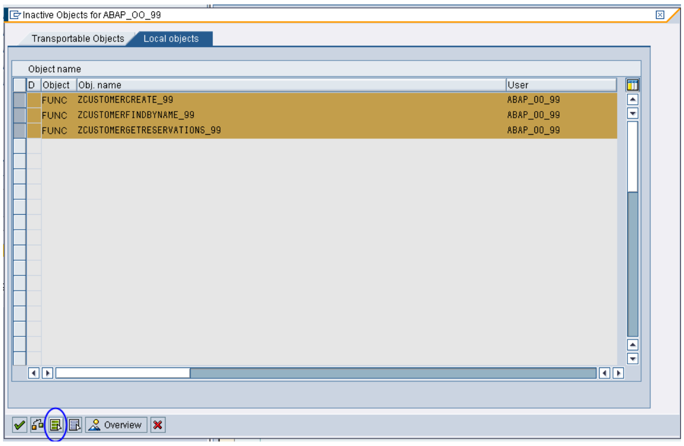
- Double-click on **ZCUSTOMERCREATE_XX** (where XX is your group number) and then click *Activate*



Object Name	Description
$TMP ABAP_OO_99	
Class Library	
Function Groups	
ZFG_99	Function Group for ABAP OO - Group
Function Modules	
ZCUSTOMERCREATE_99	Create Customer
ZCUSTOMERFINDBYNAME_99	Find Customer by Name
ZCUSTOMERGETRESERVATIONS_99	Get List of Customer Reservations
ZCUSTOMERRESERVECAR_99	Reserve Car for Customer
Includes	

```
function zcustomercreate_99.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(NAME) TYPE  DEMO_CR_CUSTOMER_NAME
*"  EXPORTING
*"     VALUE(RETURN) TYPE  BAPI_MSG
*"----------------------------------------------------------------------


  data  lr_service type ref to if_demo_cr_car_rentl_service.
  data  lo_exception type ref to cx_demo_cr_customer.
  lr_service = cl_demo_cr_car_rental_service=>get_service( ).
  try.
      lr_service->create_customer( name ).
      return = name && ` ` && 'created.'.
    catch cx_demo_cr_customer into lo_exception.
      return = lo_exception->get_text( ).
      exit.

  endtry.


endfunction.
```

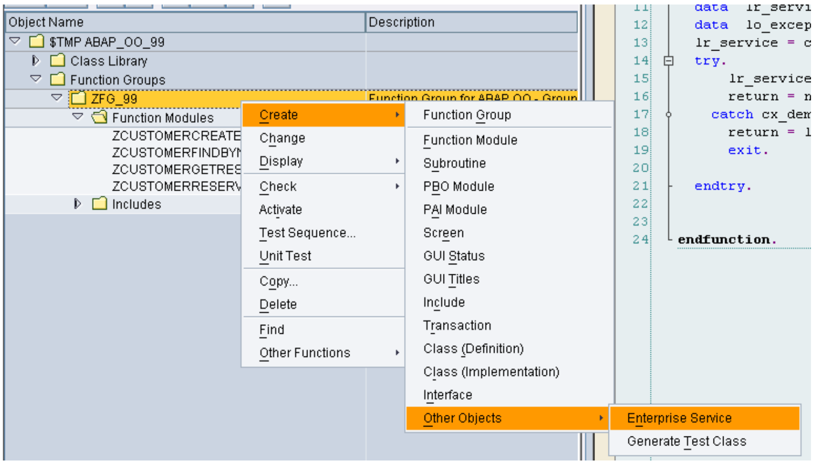
- Select the entire worklist and click continue



D	Object	Obj. name	User
	FUNC	ZCUSTOMERCREATE_99	ABAP_OO_99
	FUNC	ZCUSTOMERFINDBYNAME_99	ABAP_OO_99
	FUNC	ZCUSTOMERGETRESERVATIONS_99	ABAP_OO_99

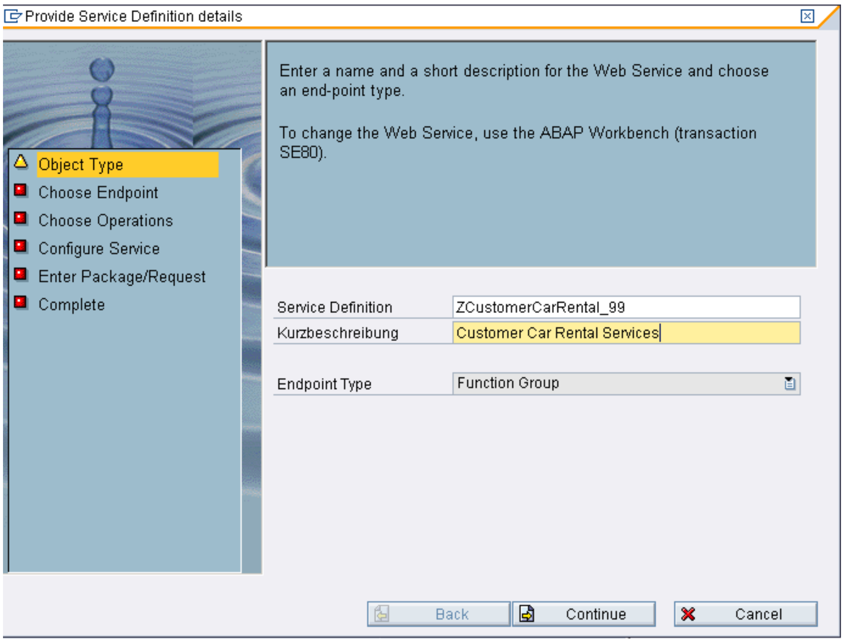
8. Using the Web Service Wizard, generate an Enterprise Service Definition for the function group ZFG_XX. (where XX is your Group Number).

- Right mouse click on the function group, **ZFG_XX**, and choose Create->Other Objects->Enterprise Service



```
  data  lr_servi
  data  lo_excep
  lr_service = c
  try.
      lr_service
      return = n
    catch cx_dem
      return = l
      exit.

  endtry.


endfunction.
```

- On the Provide Service Definition Details step, input the name of your service definition – **ZCustomerCarRental_XX**(where XX is your Group Number). You can also enter a short description (Kurzbeschreibung) and set the Endpoint Type to Function Group. Click **Continue**



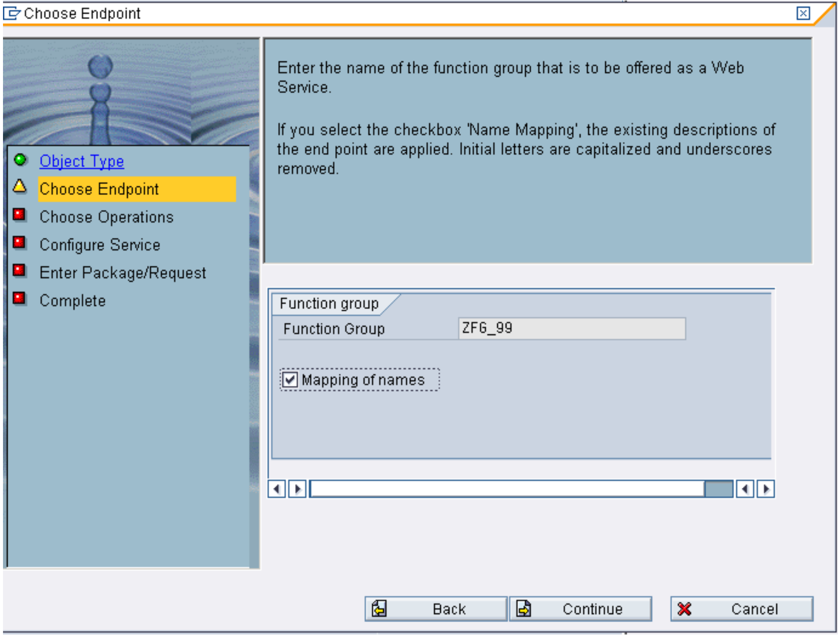
Provide Service Definition details

Enter a name and a short description for the Web Service and choose an end-point type.

To change the Web Service, use the ABAP Workbench (transaction SE80).

Service Definition	ZCustomerCarRental_99
Kurzbeschreibung	Customer Car Rental Services
Endpoint Type	Function Group

- On the Choose Endpoint step, you have to specify the name of the function group which will serve as the implementation for this service definition. If you want to use Name Mapping (underscores are removed and changed to camel case) you can **check the Mapping of names** option.



Choose Endpoint

Enter the name of the function group that is to be offered as a Web Service.

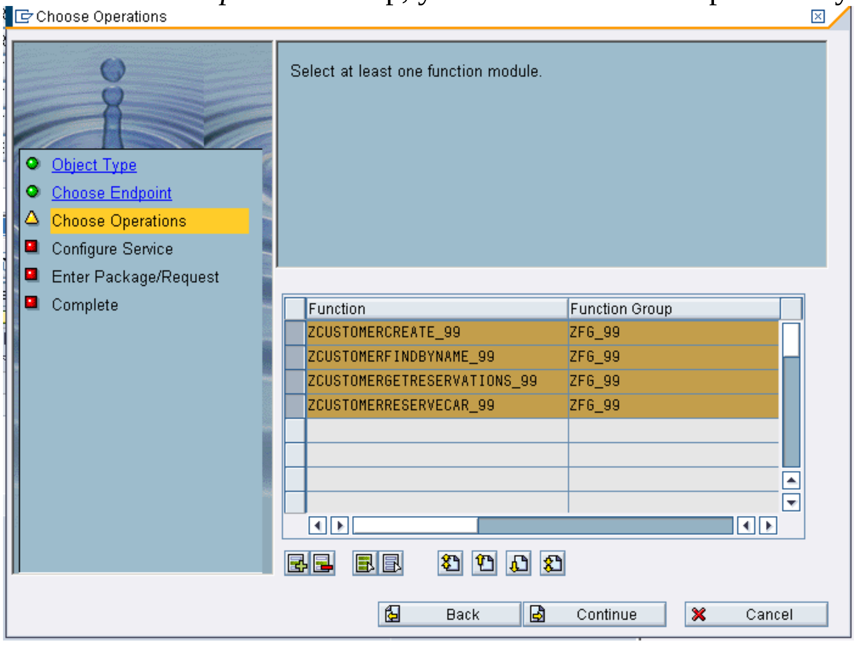
If you select the checkbox 'Name Mapping', the existing descriptions of the end point are applied. Initial letters are capitalized and underscores removed.

Function group

Function Group	ZF6_99

☑ Mapping of names

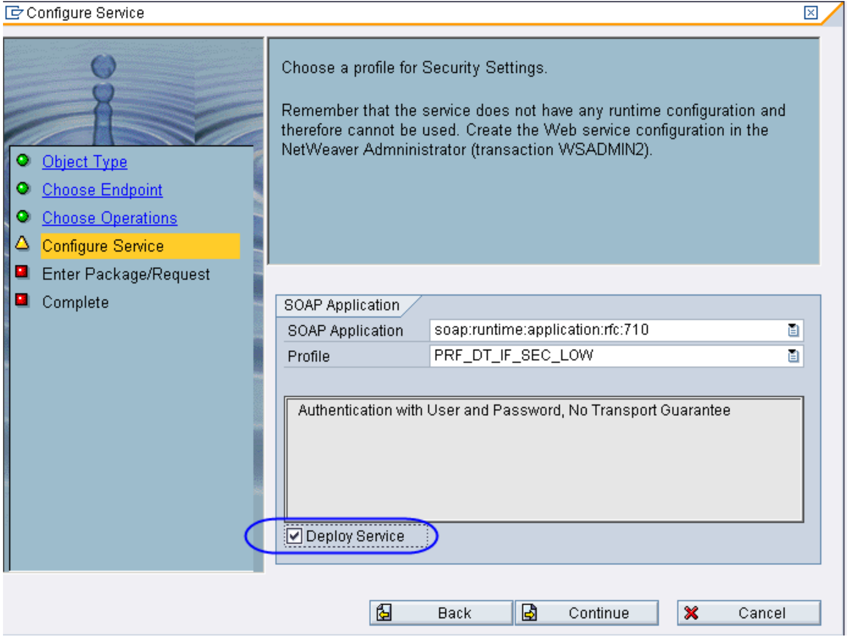
- On the *Choose Operations* step, you can select which operations you wish to expose. Click Continue



Select at least one function module.

Function	Function Group
ZCUSTOMERCREATE_99	ZFG_99
ZCUSTOMERFINDBYNAME_99	ZFG_99
ZCUSTOMERGETRESERVATIONS_99	ZFG_99
ZCUSTOMERRESERVECAR_99	ZFG_99

- On the *Configure Service* step, set the **PRF_DT_IF_SEC_LOW** profile in order to set the lowest security level for this service definition. Be sure to check *Deploy Service*. If you forget to check this box, you can complete this step later, manually from transaction code **SOAMANAGER** by creating an Endpoint.



Choose a profile for Security Settings.

Remember that the service does not have any runtime configuration and therefore cannot be used. Create the Web service configuration in the NetWeaver Admninistrator (transaction WSADMIN2).

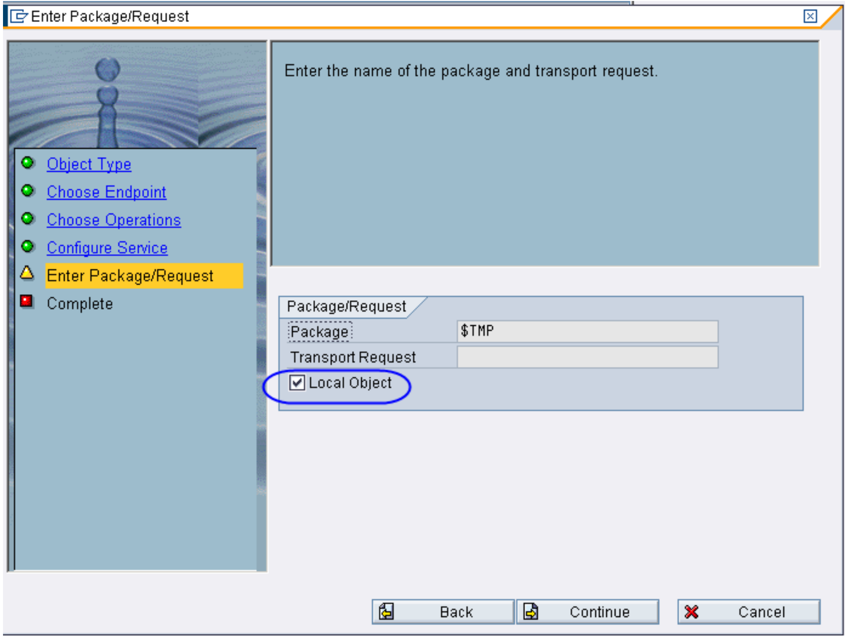
SOAP Application

SOAP Application	soap:runtime:application:rfc:710
Profile	PRF_DT_IF_SEC_LOW

Authentication with User and Password, No Transport Guarantee

☑ Deploy Service

- On the *Enter Package/Request* step, please check *Local Object* to save the generated Service Definition as local/private.



Enter Package/Request

Enter the name of the package and transport request.

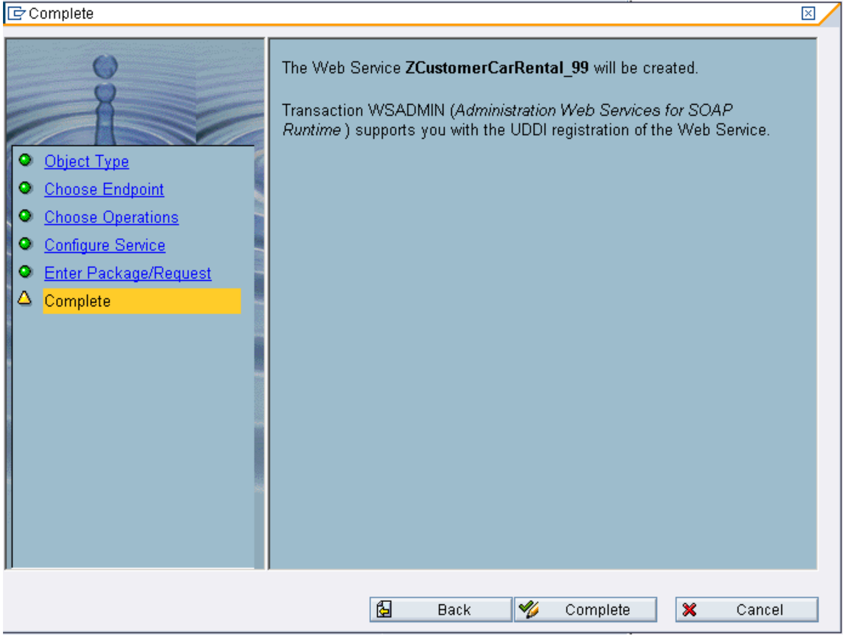
Package/Request

Package: $TMP

Transport Request:

☑ Local Object

- On the final step choose *Complete*.

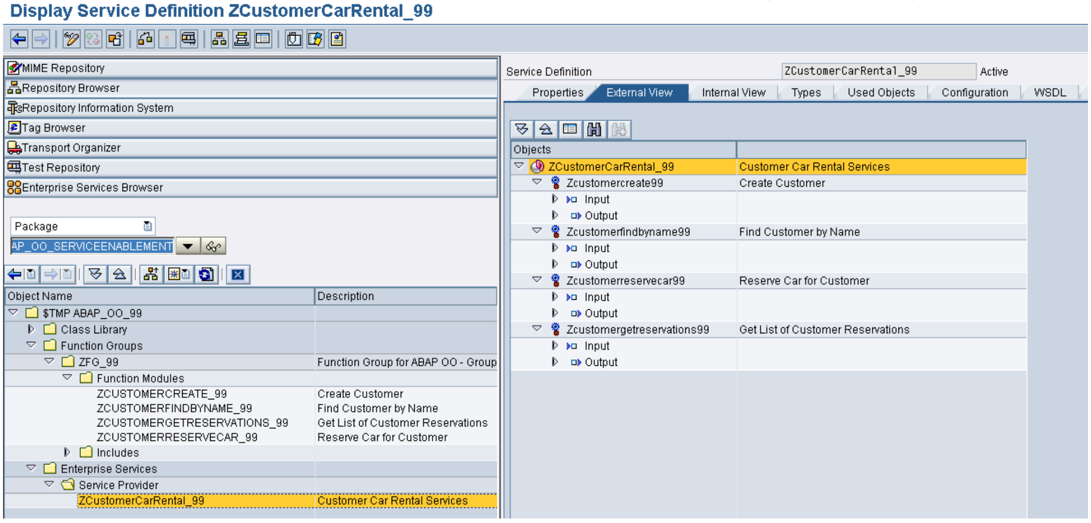


Complete

The Web Service **ZCustomerCarRental_99** will be created.

Transaction WSADMIN (*Administration Web Services for SOAP Runtime*) supports you with the UDDI registration of the Web Service.

- You can optionally explore the Service Definition that was generated by the Wizard.



Display Service Definition ZCustomerCarRental_99

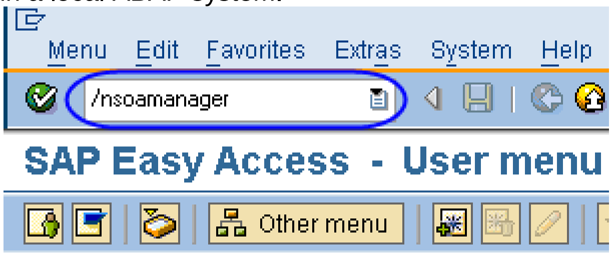
Package: AP_OO_SERVICEENABLEMENT

Object Name	Description
$TMP ABAP_OO_99	
Class Library	
Function Groups	
ZFG_99	Function Group for ABAP OO - Group
Function Modules	
ZCUSTOMERCREATE_99	Create Customer
ZCUSTOMERFINDBYNAME_99	Find Customer by Name
ZCUSTOMERGETRESERVATIONS_99	Get List of Customer Reservations
ZCUSTOMERRESERVECAR_99	Reserve Car for Customer
Includes	
Enterprise Services	
Service Provider	
ZCustomerCarRental_99	Customer Car Rental Services

Service Definition: ZCustomerCarRental_99 Active

Objects	
ZCustomerCarRental_99	Customer Car Rental Services
Zcustomercreate99	Create Customer
Input	
Output	
Zcustomerfindbyname99	Find Customer by Name
Input	
Output	
Zcustomerreservecar99	Reserve Car for Customer
Input	
Output	
Zcustomergetreservations99	Get List of Customer Reservations
Input	
Output	

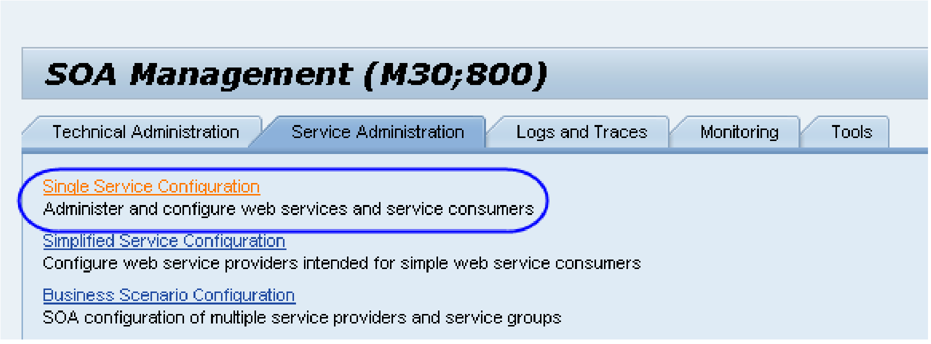
9. To start the SOA Manager, use the transaction code `SOAMANAGER`. (enter /n SOAManager in the command field of the system task bar). SOA Manager is used to complete the configuration of service providers and consumer proxies in a local ABAP system.



/nsoamanager

SAP Easy Access - User menu

- The transaction **SOAMANAGER** should launch a Web Dynpro ABAP based application in your Internet Browser. Choose the *Service Administration* tab and then click on *Single Service Administration.*



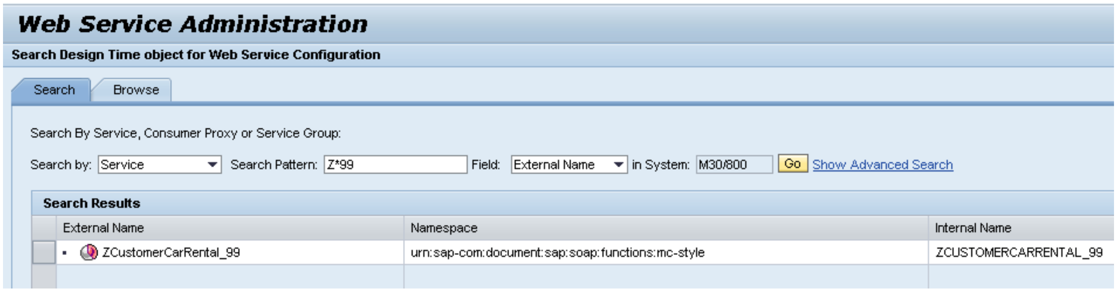
SOA Management (M30;800)

Single Service Configuration
Administer and configure web services and service consumers

Simplified Service Configuration
Configure web service providers intended for simple web service consumers

Business Scenario Configuration
SOA configuration of multiple service providers and service groups

- In the Web Service Administration screen that comes up, you can search for your Service Definition (Hint use wildcard **Z*XX** where XX is your group number)



Web Service Administration

Search Design Time object for Web Service Configuration

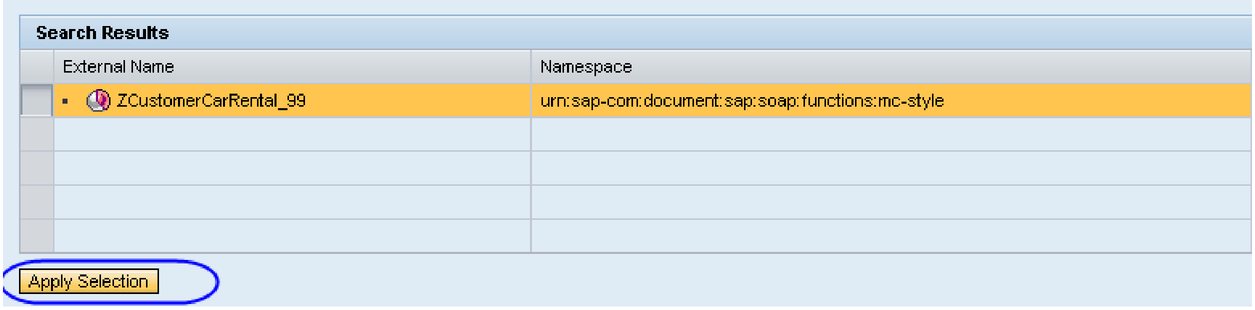
Search By Service, Consumer Proxy or Service Group:

Search by: Service Search Pattern: Z*99 Field: External Name in System: M30/800 Go Show Advanced Search

Search Results

External Name	Namespace	Internal Name
ZCustomerCarRental_99	urn:sap-com:document:sap:soap:functions:mc-style	ZCUSTOMERCARRENTAL_99

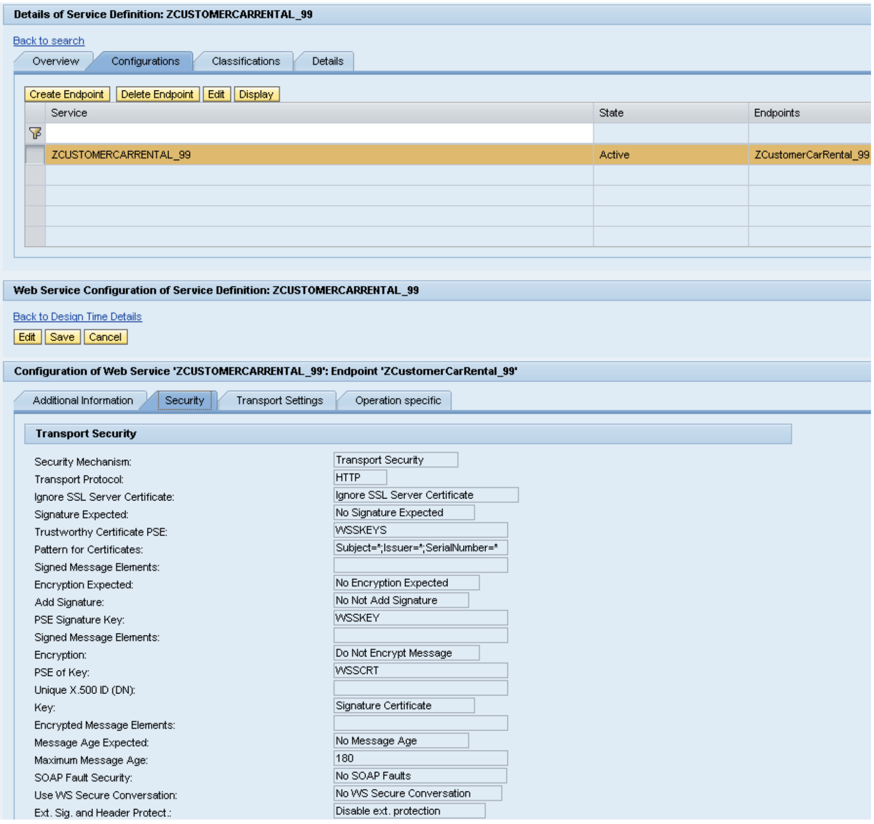
- Select the row in the Search Results for your Service Definition and then click the *Apply Selection button*. The bottom half of the screen will now show the details for the selected Service Definition.



Search Results

External Name	Namespace
ZCustomerCarRental_99	urn:sap-com:document:sap:soap:functions:mc-style

- There are lots of changes that can be made to the Service Definition from this screen. You can see a summary of all the settings from the *Details* tab. You can alter the published classification settings from the *Classification* tab. From the *Configurations* tab, you see the Endpoints for this service. If you had forgotten to Deploy Service during the wizard, you would now have to create this Endpoint manually. The settings for the Endpoint were generated for us based upon the security profile we choose during the wizard.
-



Details of Service Definition: ZCUSTOMERCARRENTAL_99

Service	State	Endpoints
ZCUSTOMERCARRENTAL_99	Active	ZCustomerCarRental_99

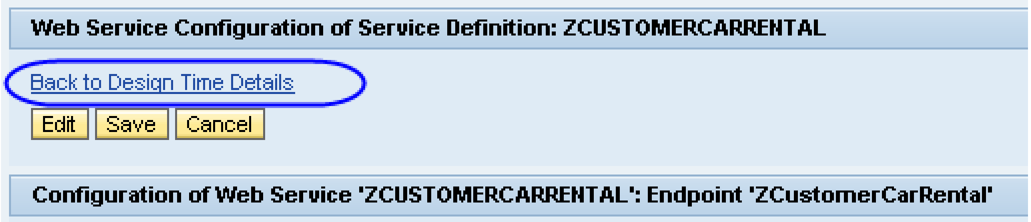
Web Service Configuration of Service Definition: ZCUSTOMERCARRENTAL_99

Configuration of Web Service 'ZCUSTOMERCARRENTAL_99': Endpoint 'ZCustomerCarRental_99'

Transport Security

Security Mechanism:	Transport Security
Transport Protocol:	HTTP
Ignore SSL Server Certificate:	Ignore SSL Server Certificate
Signature Expected:	No Signature Expected
Trustworthy Certificate PSE:	WSSKEYS
Pattern for Certificates:	Subject=*;Issuer=*;SerialNumber=*
Signed Message Elements:	
Encryption Expected:	No Encryption Expected
Add Signature:	No Not Add Signature
PSE Signature Key:	WSSKEY
Signed Message Elements:	
Encryption:	Do Not Encrypt Message
PSE of Key:	WSSCRT
Unique X.500 ID (DN):	
Key:	Signature Certificate
Encrypted Message Elements:	
Message Age Expected:	No Message Age
Maximum Message Age:	180
SOAP Fault Security:	No SOAP Faults
Use WS Secure Conversation:	No WS Secure Conversation
Ext. Sig. and Header Protect.:	Disable ext. protection

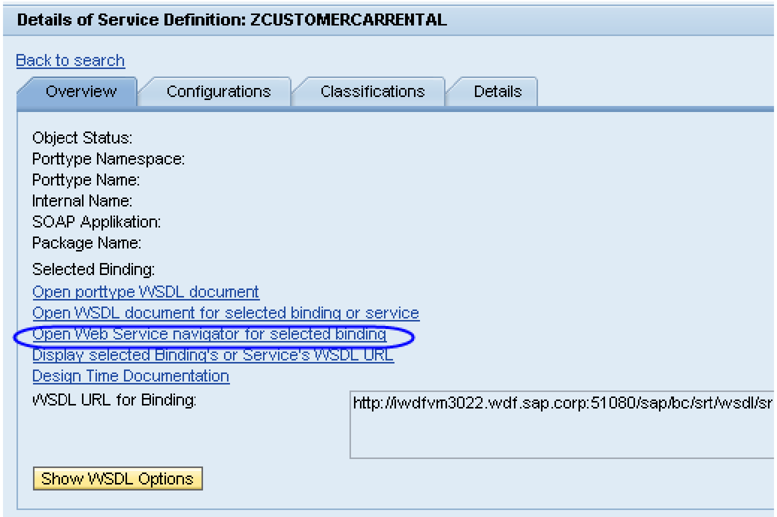
- Go Back to Design Time Details



Web Service Configuration of Service Definition: ZCUSTOMERCARRENTAL

Configuration of Web Service 'ZCUSTOMERCARRENTAL': Endpoint 'ZCustomerCarRental'

- Click on the **Overview** tab
- Click on **Open Web Service Navigator for the selected binding**



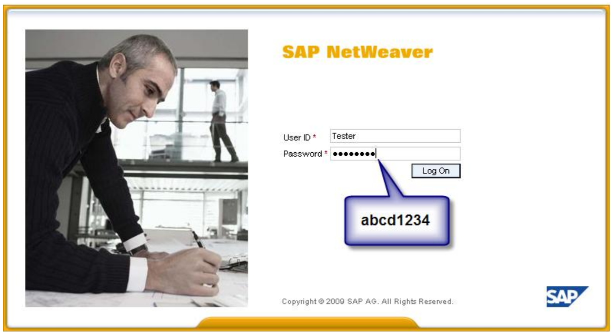
Details of Service Definition: ZCUSTOMERCARRENTAL

Object Status:
Porttype Namespace:
Porttype Name:
Internal Name:
SOAP Applikation:
Package Name:
Selected Binding:

Open porttype WSDL document
Open WSDL document for selected binding or service
Open Web Service navigator for selected binding
Display selected Binding's or Service's WSDL URL
Design Time Documentation

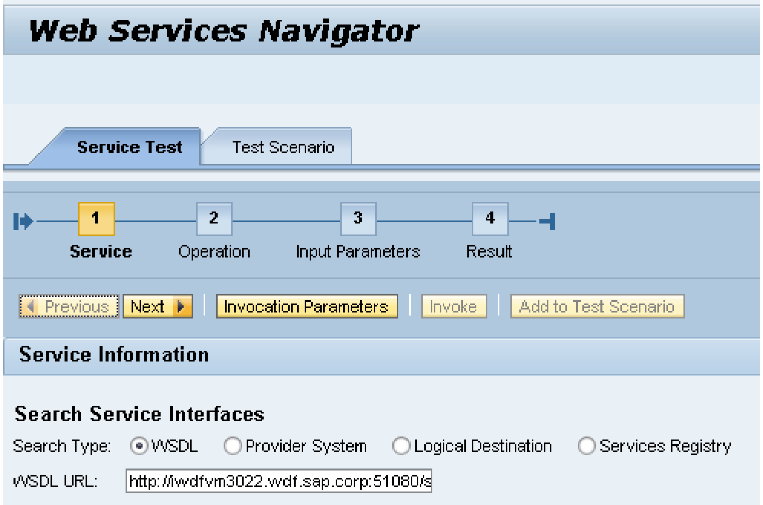
WSDL URL for Binding: http://iwdfvm3022.wdf.sap.corp:51080/sap/bc/srt/wsdl/sr

- A Separate Browser Window is started. If you are asked to login, enter the credentials Tester / abcd1234



User ID * Tester
Password * ••••••••

- The url of wsdl is displayed. Click Next



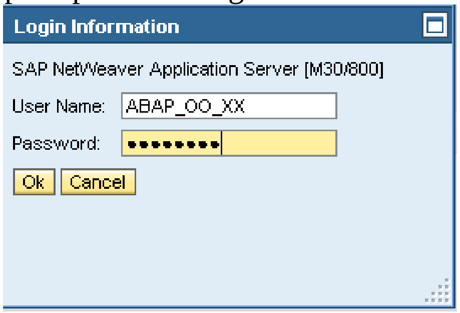
Service Information

Search Service Interfaces

Search Type: ● WSDL ○ Provider System ○ Logical Destination ○ Services Registry

WSDL URL: http://iwdfvm3022.wdf.sap.corp:51080/s

- If prompted for a login to download the wsdl – Please enter your userid and pwd for M30 and click **OK**



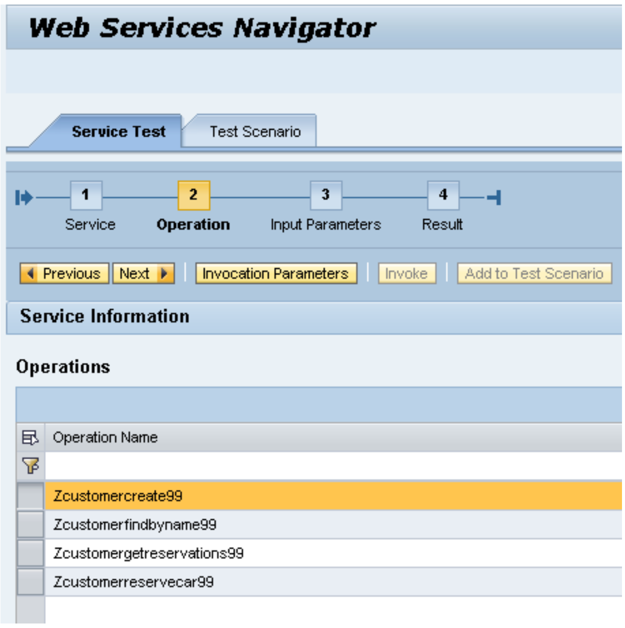
Login Information

SAP NetWeaver Application Server [M30/800]

User Name: ABAP_OO_XX

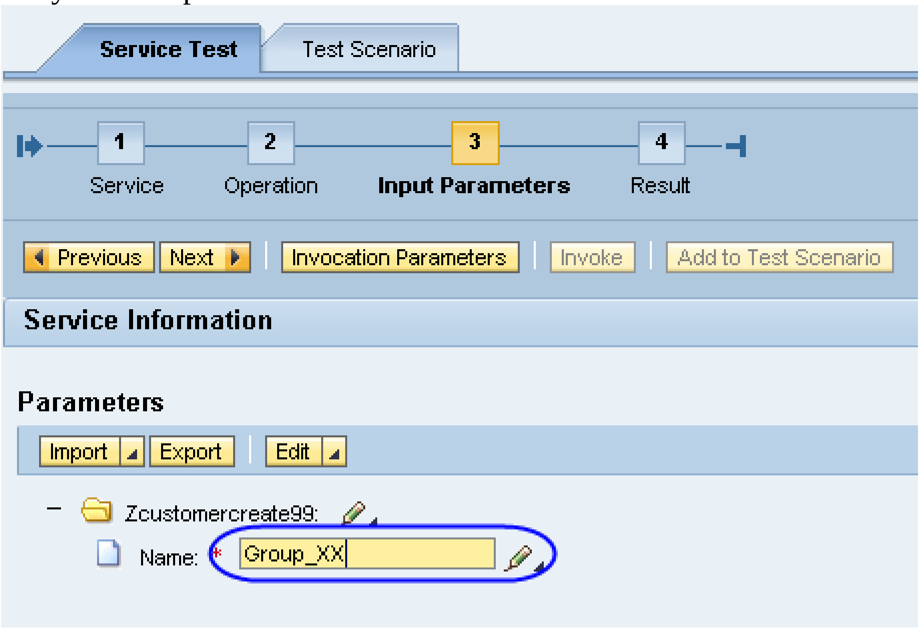
Password: ••••••••

- Select the **ZCUSTOMERCREATE_XX** operation (where XX is your group number) and Click **Next**



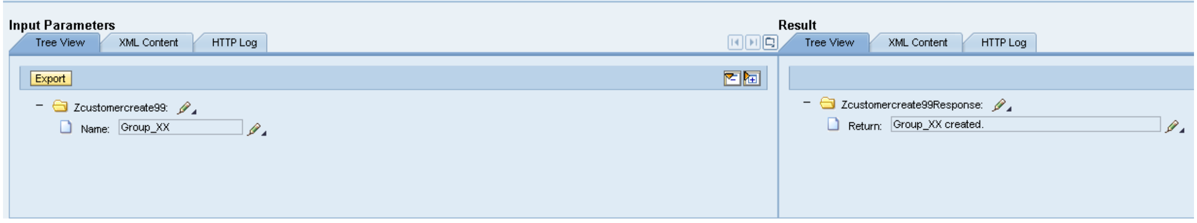
Zcustomercreate99
Zcustomerfindbyname99
Zcustomergetreservations99
Zcustomerreservecar99

- Enter your Group Number for the Customer Name and Click **Next**



Zcustomercreate99:
Name: * Group_XX

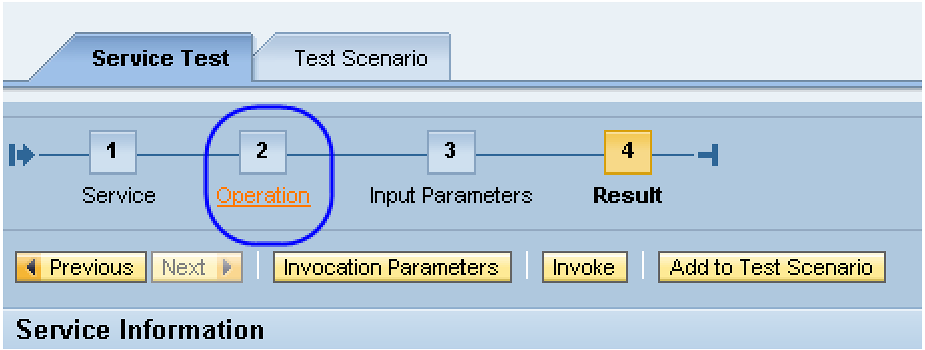
- The system displays the input parameters and the result of the test.



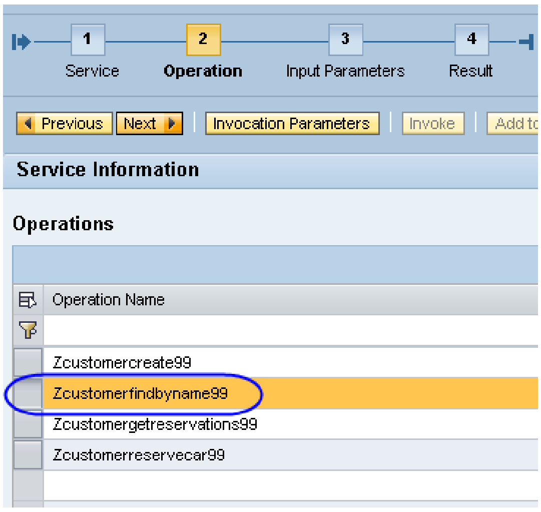
Zcustomercreate99:
Name: Group_XX

Zcustomercreate99Response:
Return: Group_XX created.

- Click on the **Operation** link

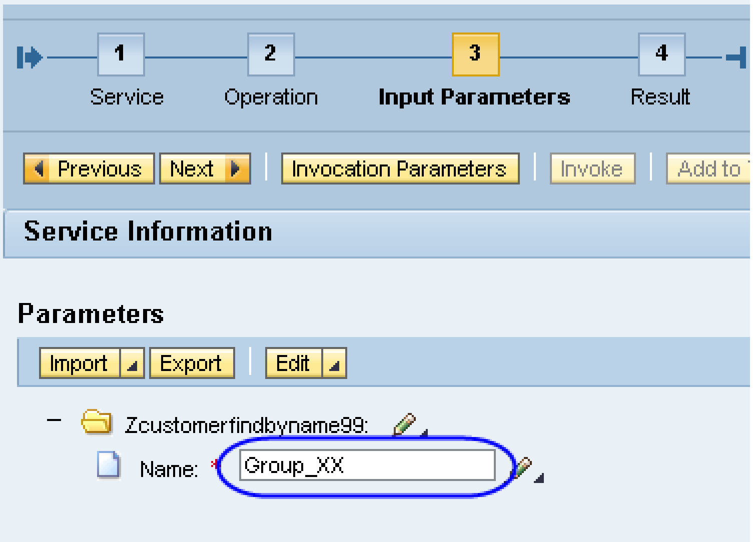


- Select the **ZCUSTOMERFINDBYNAME_XX** operation (where XX is your group number) and Click **Next**



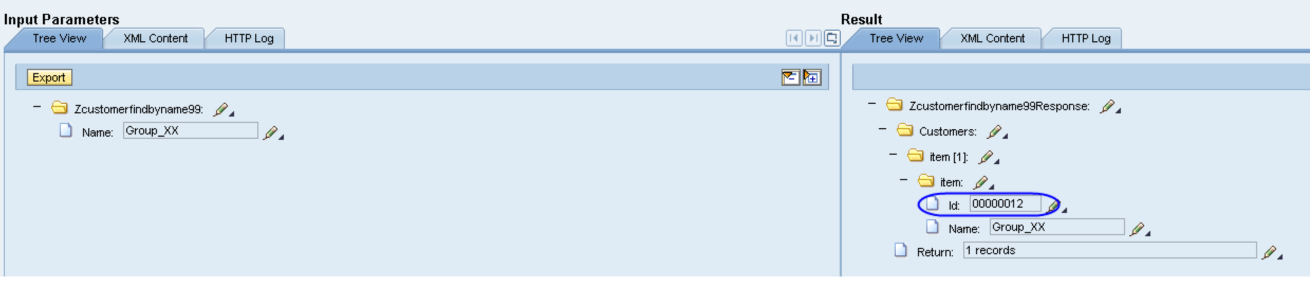
Operation Name
Zcustomercreate99
Zcustomerfindbyname99
Zcustomergetreservations99
Zcustomerreservecar99

- Enter your Group Number for the Customer Name Click **Next**



Zcustomerfindbyname99:
Name: * Group_XX

- The system displays the input parameters and the result of the test.

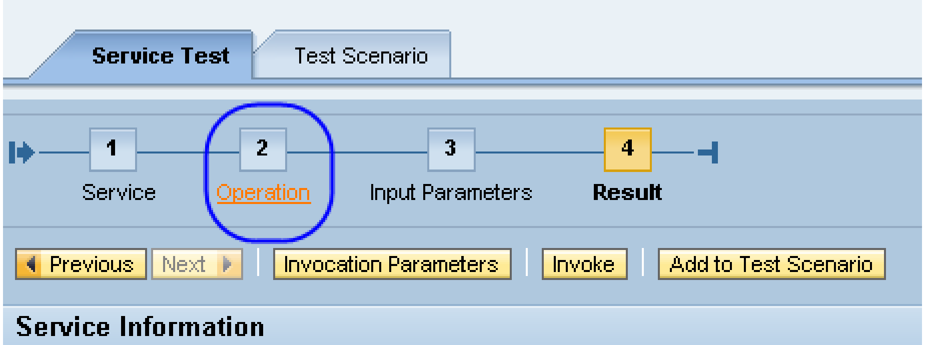


Zcustomerfindbyname99:
Name: Group_XX

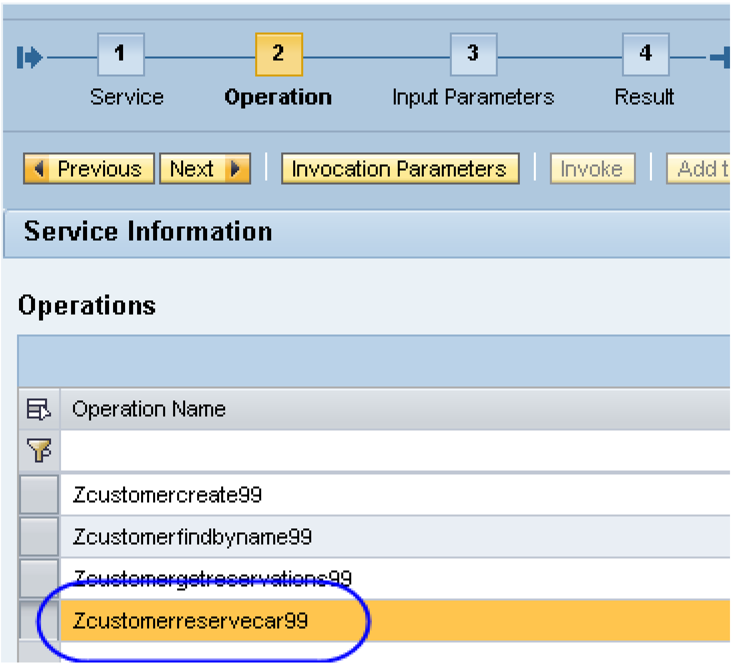
Zcustomerfindbyname99Response:
Customers:
item [1]:
item:
Id: 00000012
Name: Group_XX
Return: 1 records

- **Important** Please make note of your Customer ID (we will use this later).

- Click on the **Operation** link

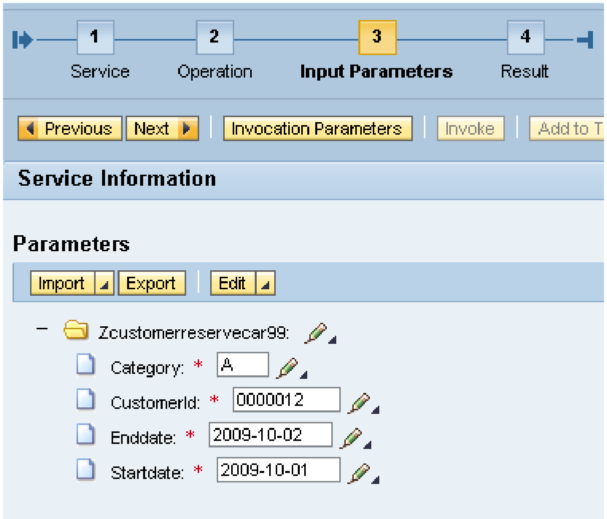


- Select the **ZCUSTOMERRESERVECAR_XX** operation (where XX is your group number) and Click **Next**



Operation Name
Zcustomercreate99
Zcustomerfindbyname99
Zcustomergetreservations99
Zcustomerreservecar99

- Enter a Category A and the CustomerID returned from the **ZCUSTOMERFINDBYNAME_XX** operation. Please be sure to enter the Start and End dates using the **YYYY-MM-DD** ISO date format, Click **Next**



− Zcustomerreservecar99:

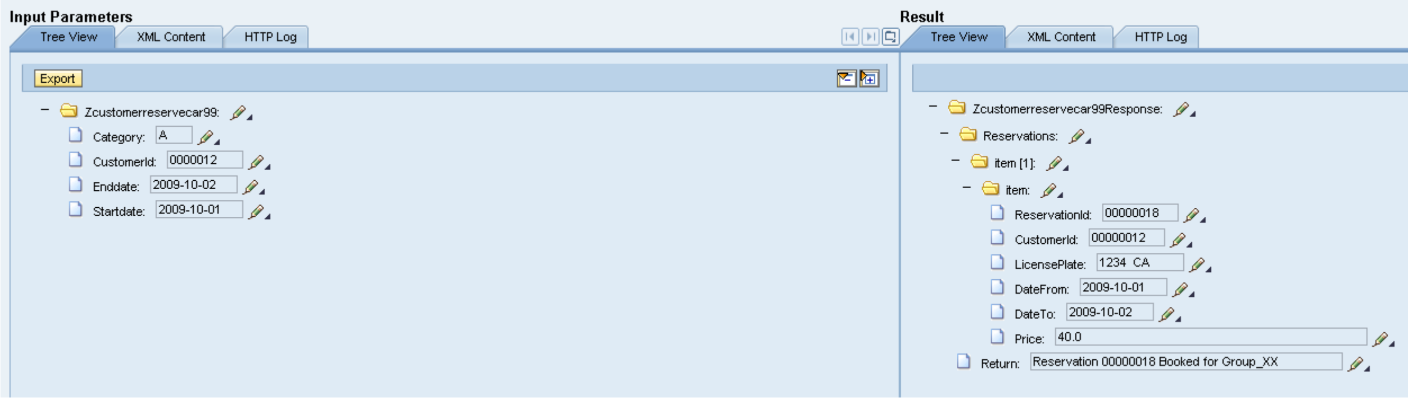
Category: * A

CustomerId: * 0000012

Enddate: * 2009-10-02

Startdate: * 2009-10-01

- The system displays the input parameters and the result of the test.



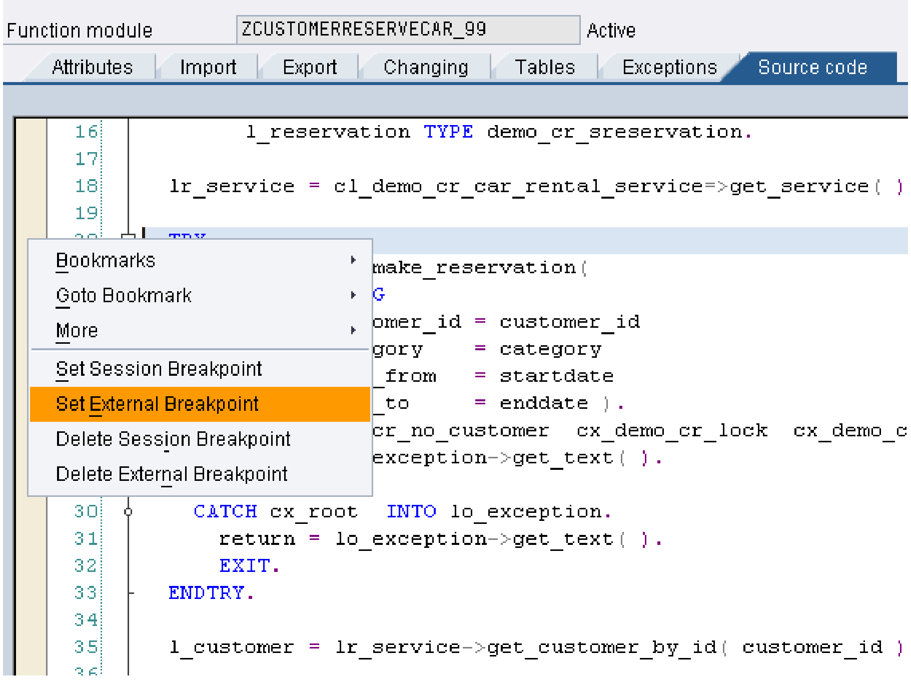
Input Parameters

- Zcustomerreservecar99:
  - Category: A
  - CustomerId: 0000012
  - Enddate: 2009-10-02
  - Startdate: 2009-10-01

Result

- Zcustomerreservecar99Response:
  - Reservations:
    - item [1]:
      - item:
        - ReservationId: 00000018
        - CustomerId: 00000012
        - LicensePlate: 1234 CA
        - DateFrom: 2009-10-01
        - DateTo: 2009-10-02
        - Price: 40.0
  - Return: Reservation 00000018 Booked for Group_XX

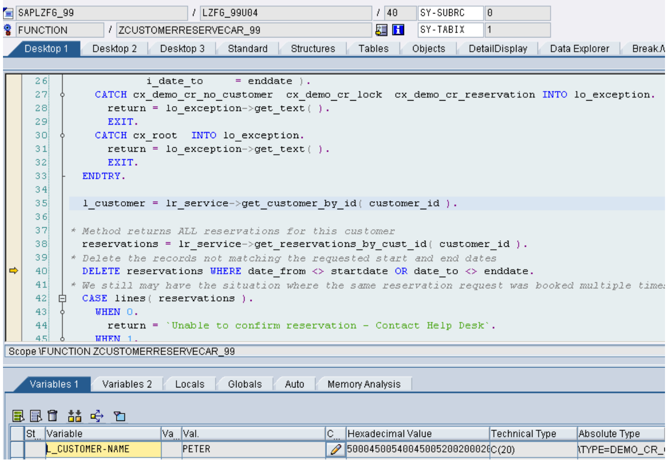
10. **Optional** – Set an External Breakpoint in Function Module ZCUSTOMERRESERVECAR_XX (where XX is your group number) by right clicking in the left margin and **Set External Breakpoint** after which you can debug from the WS Navigator into the ABAP system



Function module ZCUSTOMERRESERVECAR_99 Active

```
        l_reservation TYPE demo_cr_sreservation.

  lr_service = cl_demo_cr_car_rental_service=>get_service( )

  TRY.
                 make_reservation(
                 G
                 omer_id = customer_id
                 gory     = category
                 _from    = startdate
                 _to      = enddate ).
                 cr_no_customer  cx_demo_cr_lock  cx_demo_c
                 exception->get_text( ).
    CATCH cx_root  INTO lo_exception.
      return = lo_exception->get_text( ).
      EXIT.
  ENDTRY.

  l_customer = lr_service->get_customer_by_id( customer_id )
```

- You can step thru the code using . The ABAP debugger has many powerful features and learning how to use it is essential for Java programmers accessing ABAP Business Logic.



SAPLZFG_99 / LZFG_99U04 / 40 SY-SUBRC 0

FUNCTION / ZCUSTOMERRESERVECAR_99 SY-TABIX 1

```
             i_date_to      = enddate ).
    CATCH cx_demo_cr_no_customer  cx_demo_cr_lock  cx_demo_cr_reservation INTO lo_exception.
      return = lo_exception->get_text( ).
      EXIT.
    CATCH cx_root  INTO lo_exception.
      return = lo_exception->get_text( ).
      EXIT.
  ENDTRY.

  l_customer = lr_service->get_customer_by_id( customer_id ).

* Method returns ALL reservations for this customer
  reservations = lr_service->get_reservations_by_cust_id( customer_id ).
* Delete the records not matching the requested start and end dates
  DELETE reservations WHERE date_from <> startdate OR date_to <> enddate.
* We still may have the situation where the same reservation request was booked multiple time
  CASE lines( reservations ).
    WHEN 0.
      return = `Unable to confirm reservation - Contact Help Desk`.
    WHEN 1.
```

Scope \FUNCTION ZCUSTOMERRESERVECAR_99

St	Variable	Va	Val.	C	Hexadecimal Value	Technical Type	Absolute Type
	L_CUSTOMER-NAME		PETER		500045005400450052002000200	C(20)	\TYPE=DEMO_CR_