# ABAP Course

Chapter 7: ABAP Objects

|  |  |
| --- | --- |
| **Content**  This chapter explains the main concepts of objects-orientation and their adaption in ABAP. | **Prerequisites**  You should be familiar with ABAP and the navigation in the SAP system. |
| **Motivation**  This chapter explains how to create simple programs in ABAP Objects. | **Lecture notes**  The fundamental understanding of the ABAP development in the SAP system is a prerequisite for the students. Students can go on with their account from chapter 1. |

* **Product**: All
* **Level**: Beginner
* **Focus**: Programming
* **Version**: 1.1
* **Author**: UCC Technische Universität München

**Task 1: Login into the SAP system**

**Short description**: Use SAPGui to login into the SAP system with your username and password

Login

Start the SAPGui and login into the development system using the provided account and password. Please refer to chapter 1 for your username and your password.

**Task 2: Working with ABAP Objects**

**Short description:** Use the concept of ABAP Object within an ABAP program.

Please start the Object Navigator from the SAP Easy Access Menu by using the following path:

Menu path

**Tools • ABAP Workbench • Overview • Object Navigator**.

You may also use the transaction code **SE80** for direct access.

No TOP INCL

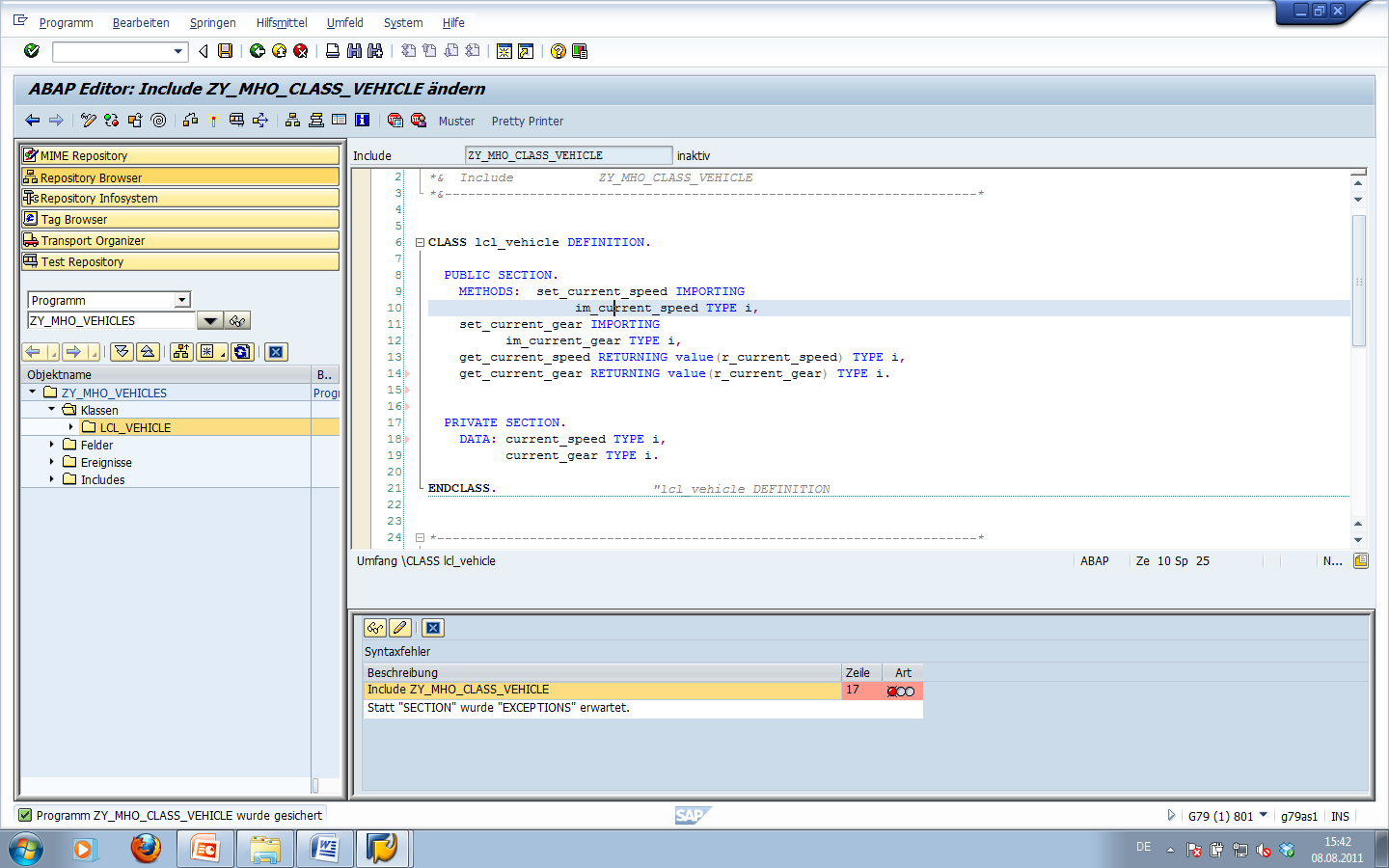
Create a new program named ‘**ZY\_##\_VEHICLES**’. Do not use a TOP-Include and declare the program as test program.

You will create a new local class with the purpose of saving vehicles. This class will be called in the program and several new instances of this class will be created.

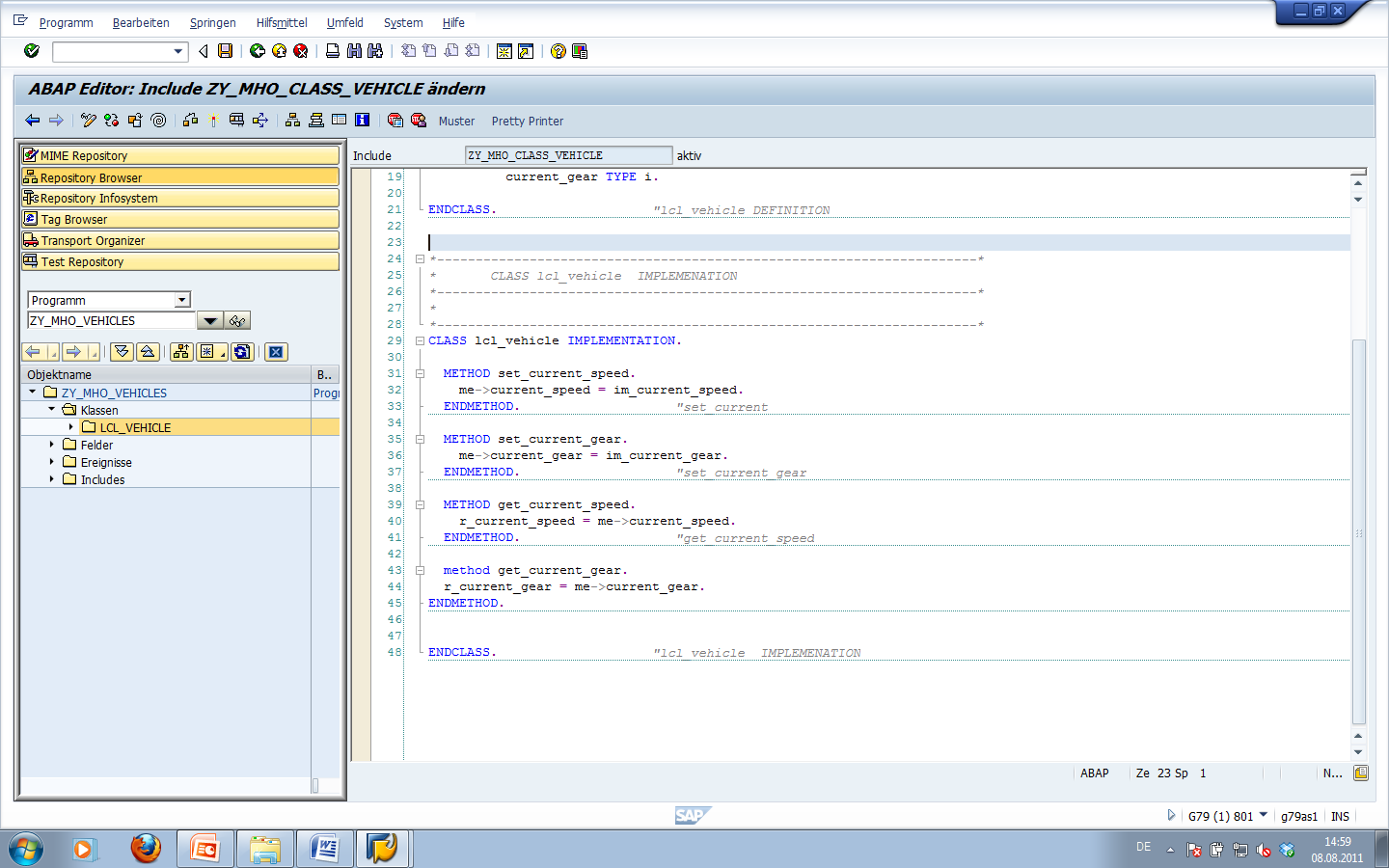
First add command ‘**INCLUDE ZY\_##\_CLASS\_VEHICLE.**’ in your program and create the corresponding include program by the use of forward navigation (double click). The newly created include program will include definition and implementation of the local class **LCL\_VEHICLE**.

Declare methods

Please start with the class declaration. You will require two setter-methods: **set\_current\_speed** and **set\_current\_gear** as well as the two getter methods **get\_current\_speed and get\_current\_gear** with public access. Each setter-method has an importing parameter and each getter-method has a return value. The corresponding attributes current\_speed and current\_gear have to be defined in the private section as attributes. The declaration of the local class is shown in the following code sample:

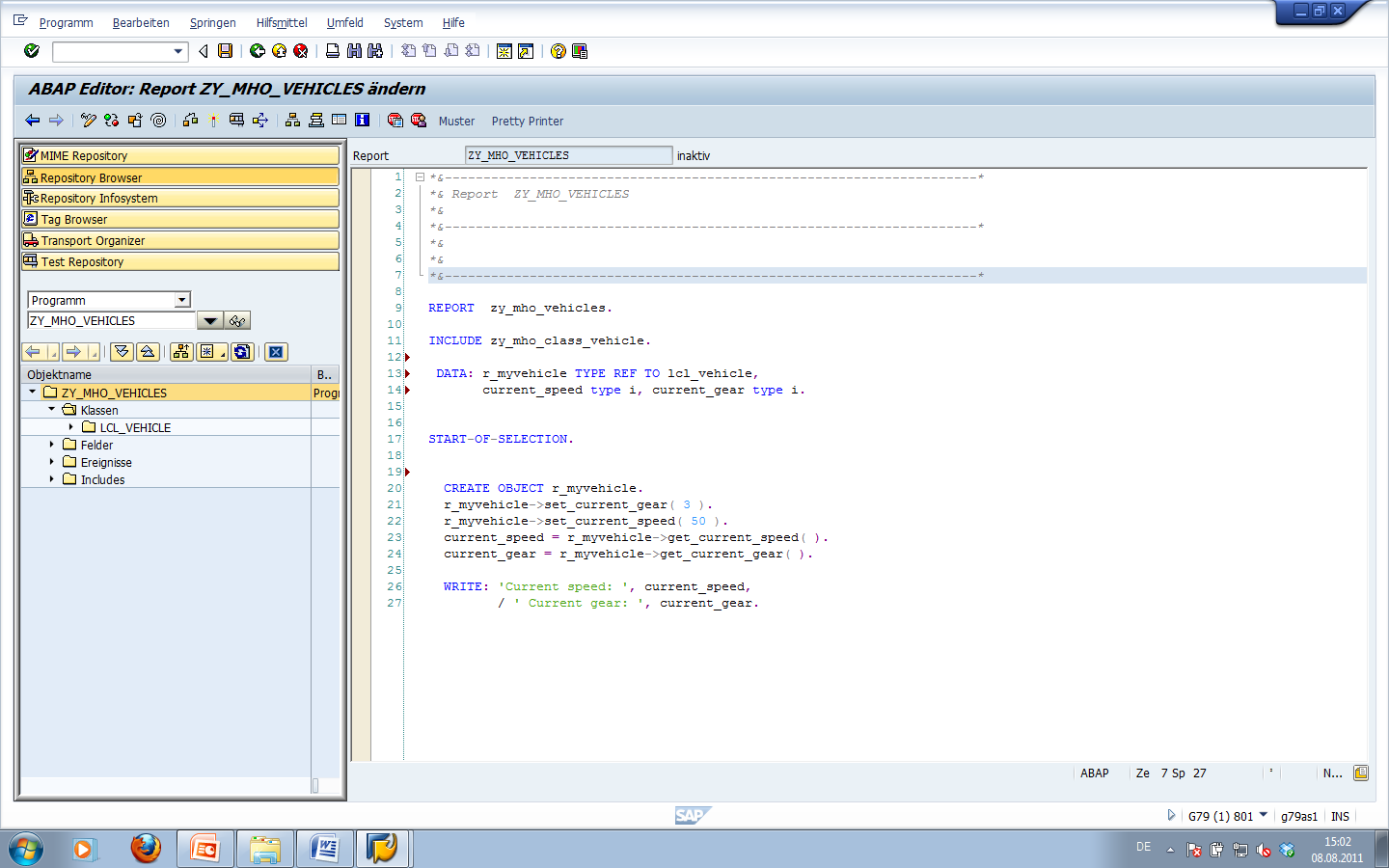


The setter-methods provide the functionality to set the private attributes according to the importing parameters. The getter-methods simply return the current value of the attributes. The Implementation of the methods is shown in the following code sample:



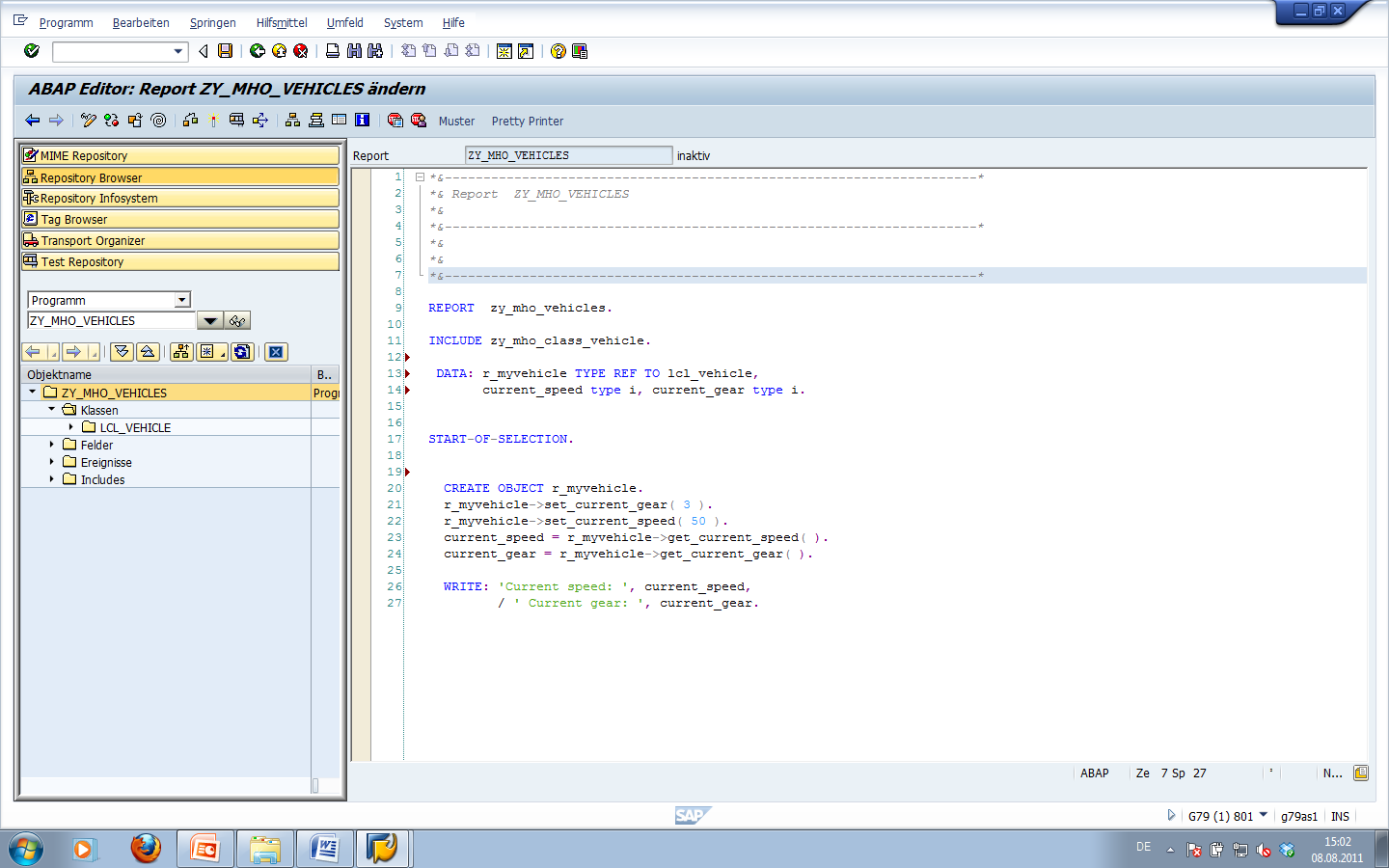
Implementation

Please return to the main program in order to implement the use of the class. You will now create a vehicle object using an object reference. The reference **r\_myvehicle** will refer to a new object from the local class **LCL\_VEHICLE** by the use of **TYPE REF TO**. The data declaration of program ‘**ZY\_##\_VEHICLES**’ therefore looks as shown in the figure bellow:



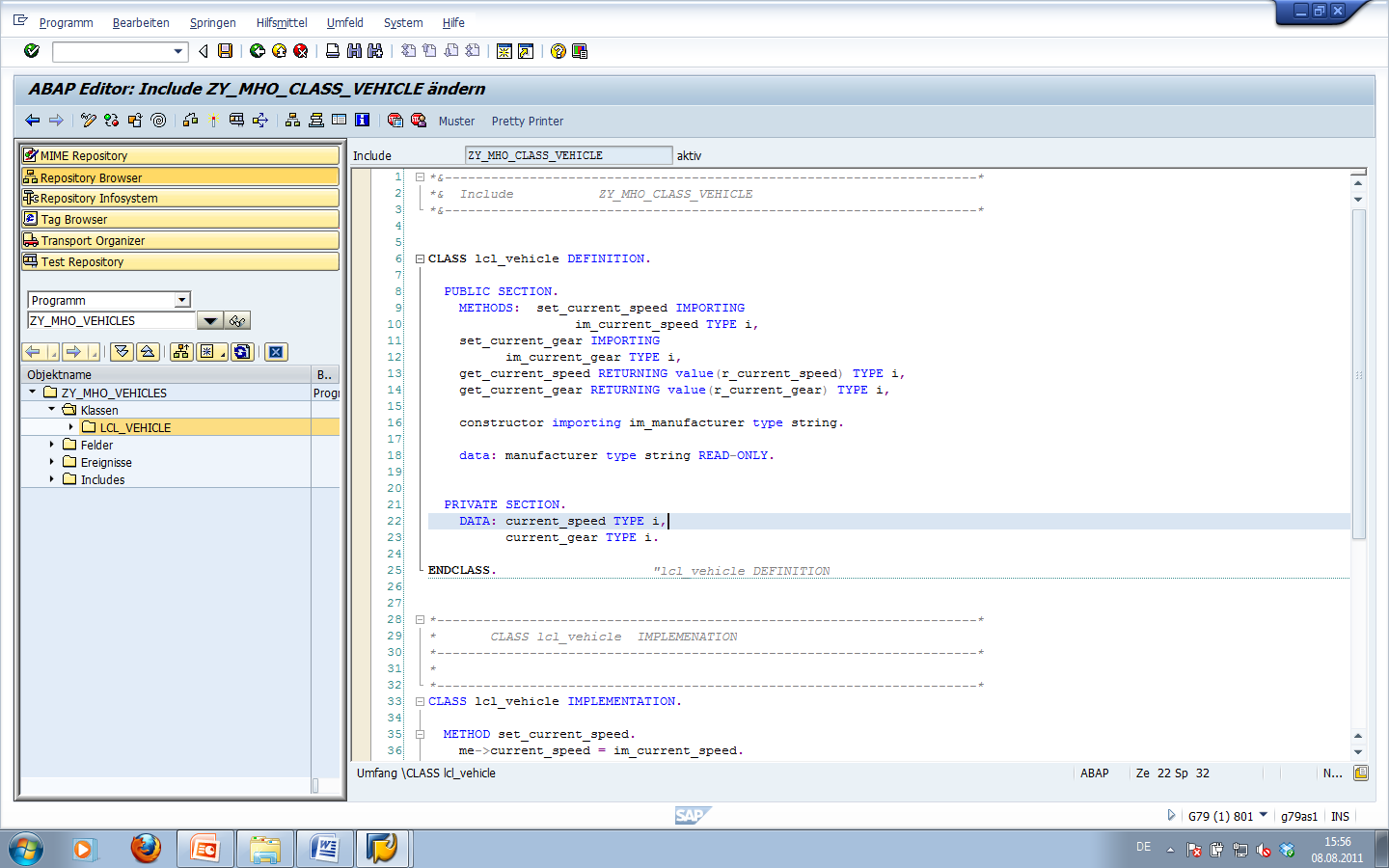
Now start the block ‘**START-OF-SELECTION**’ by creating an object with command **CREATE OBJECT** that is referenced by variable **r\_myvehicle**. Call the methods **set\_current\_gear** and **set\_current\_speed** using this reference and assign values to the object attributes. Afterwards you can test your value assignment using the WRITE statement.

Create an objects instance



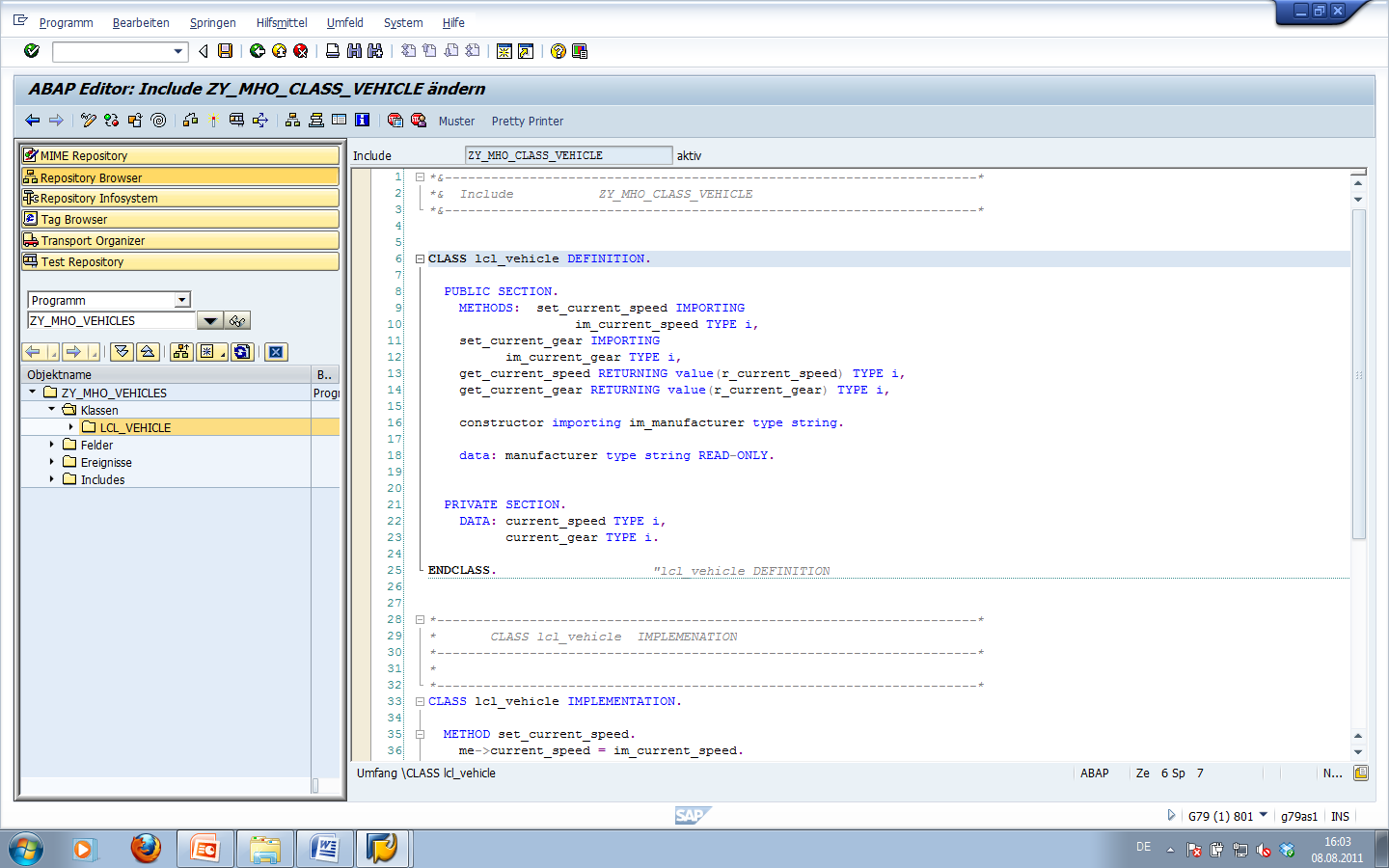
Create a READ-ONLY attribute

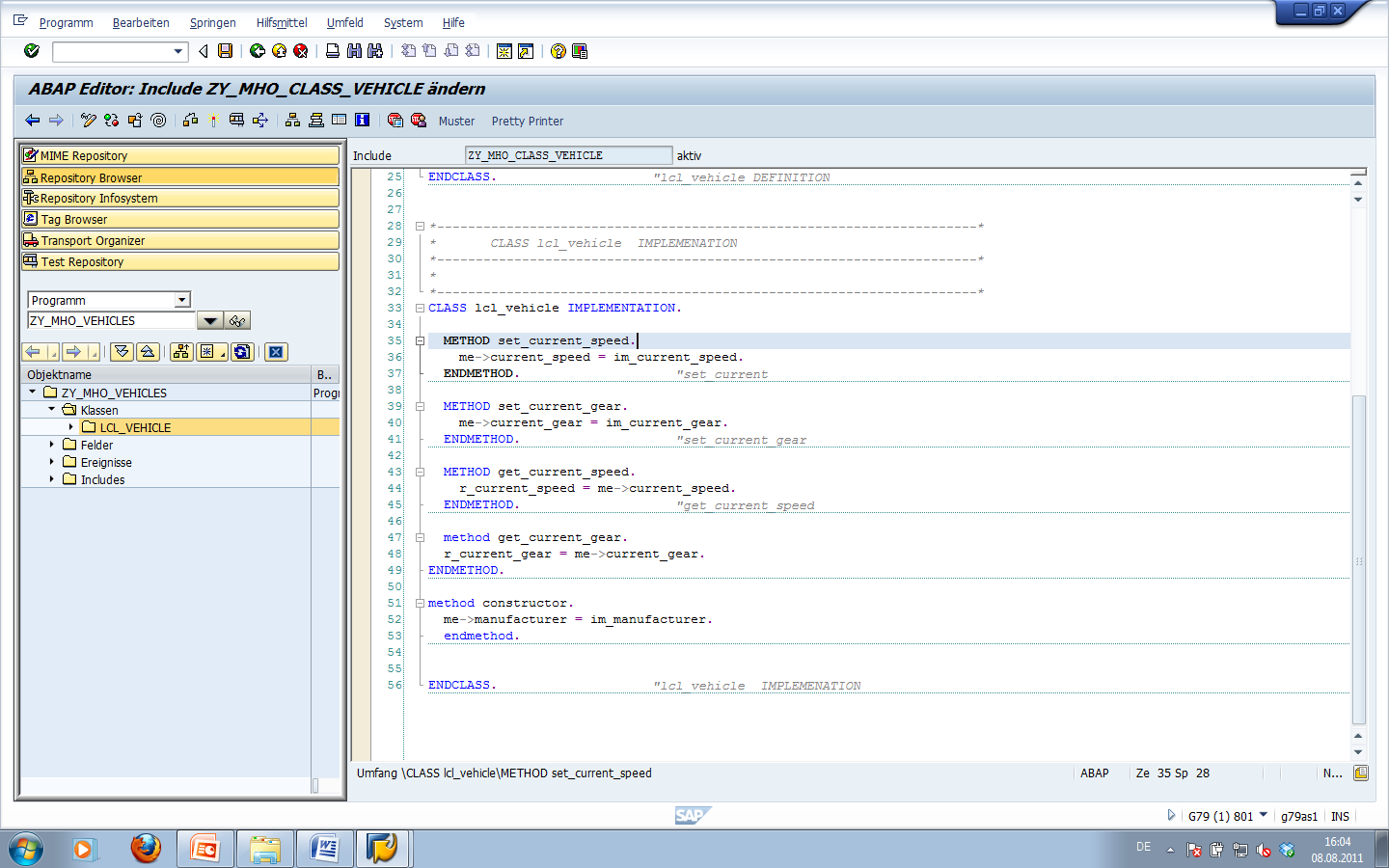
Now define an additional attribute manufacturer in the public section of the class. Use READ-ONLY as additional qualifier. This allows to access the attribute without a separate getter-method, but the value of the attribute can only be changed inside the class.



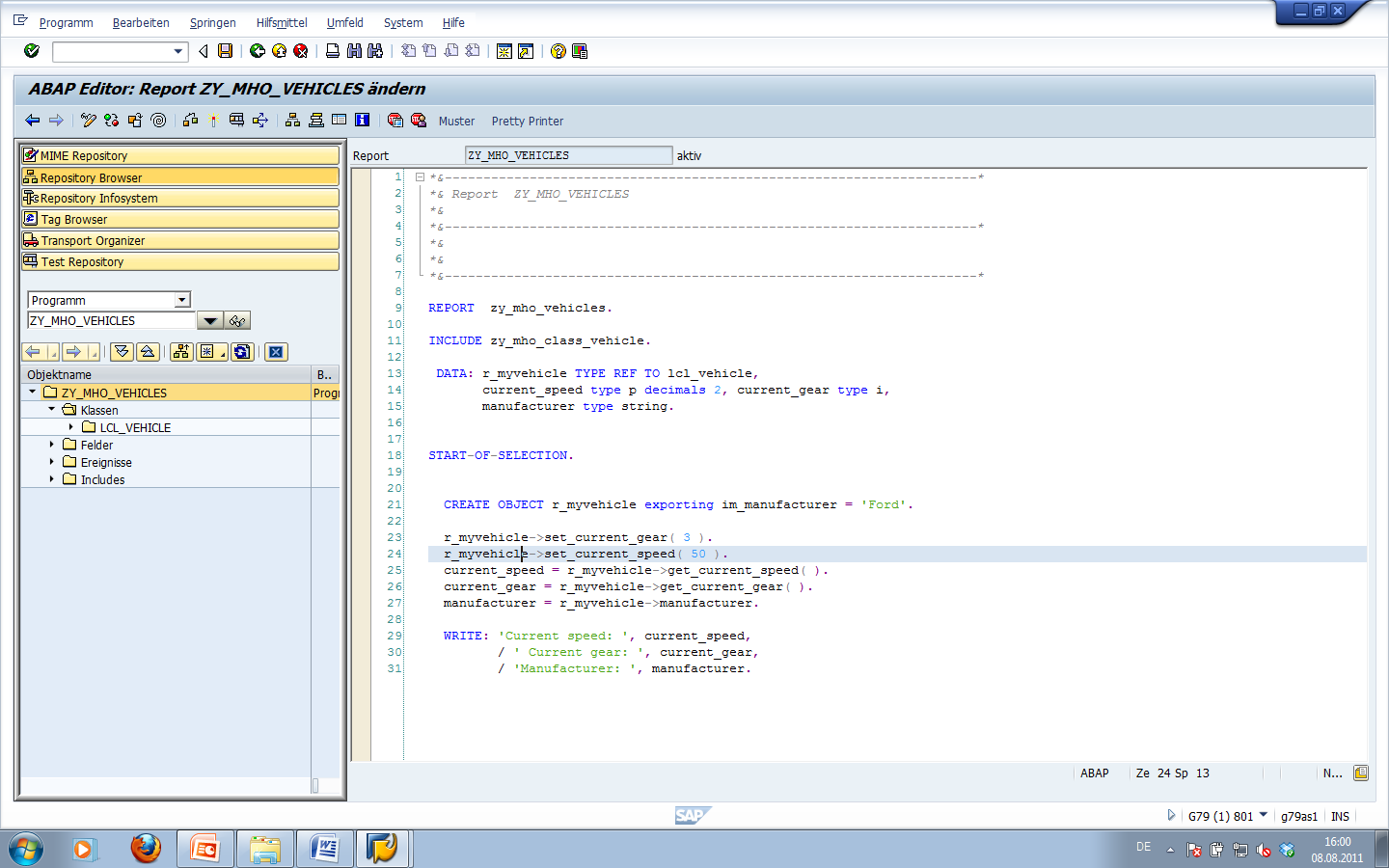
Create a constructur with importing parameter

Now define and implement a constructor and set the value of the manufacturing attribute with an according import parameter within the constructor.

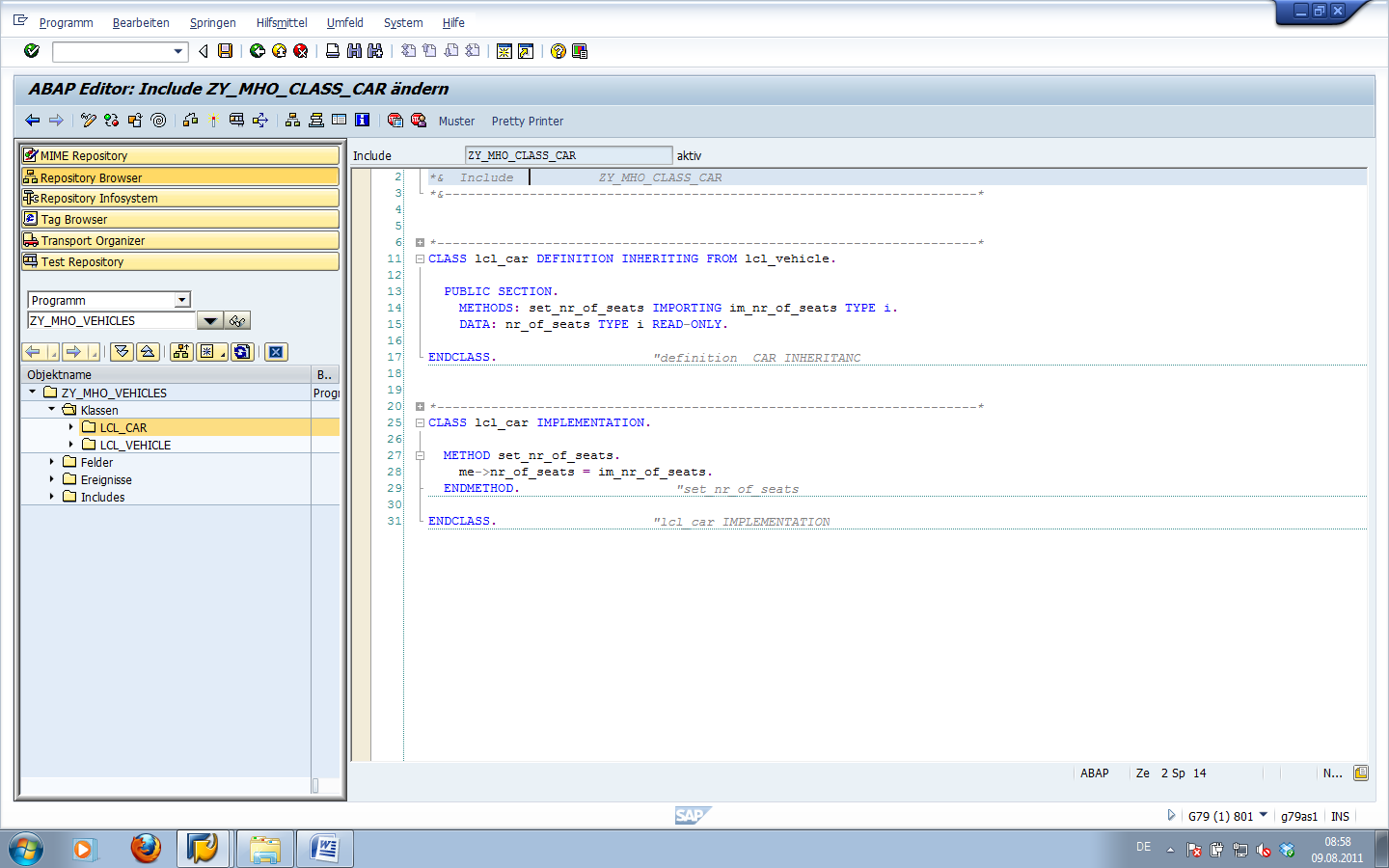




Afterwards modify your test program to test the constructor.



Now we will create a subclass lcl\_car that inherits from lcl\_vehicle. First insert a new include statement ‘**INCLUDE ZY\_##\_CLASS\_CAR’** in your program **ZY\_##\_VEHICLES**’ (after including ZY\_##\_CLASS\_VEHICLE) and create the corresponding include program by means of forward navigation. The lcl\_car class should have an additional attribute “nr\_of\_seats” and a setter-method “set\_nr\_of\_seats”.



After you have implemented the new class you may create an object instance from the new class in your main program and test its functionality.

