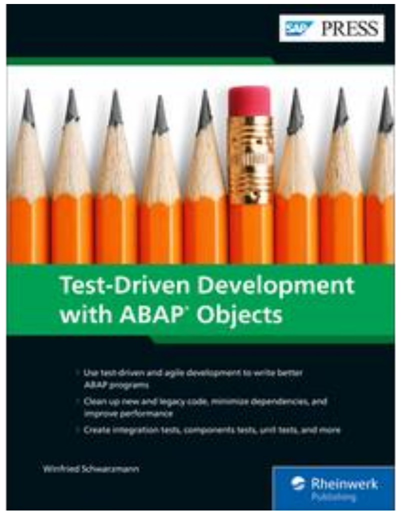
ABAPConf 2024 South Africa

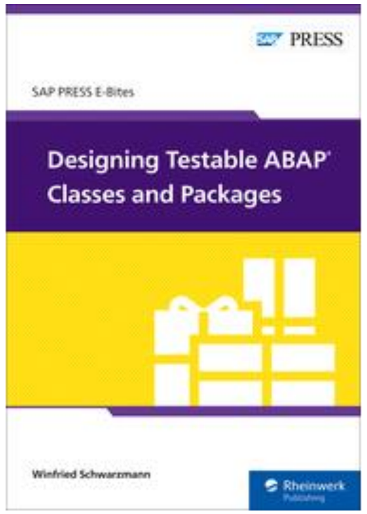
Designing Testable ABAP Classes and Packages

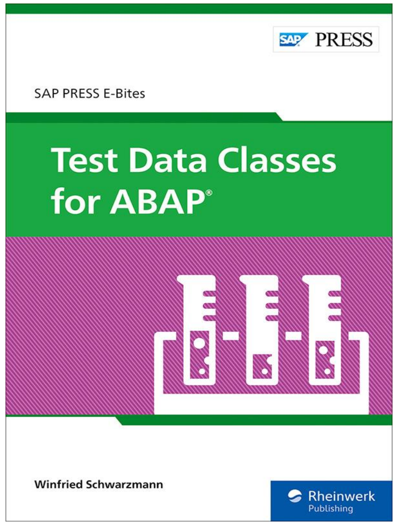
Winfried Schwarzmann, SAP SE | November 13, 2024

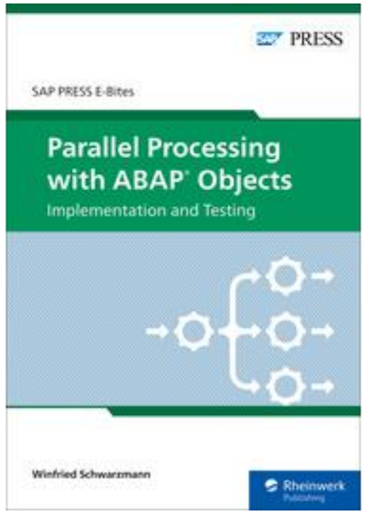
PUBLIC

Introducing Winfried Schwarzmann


SAP PRESS
Test-Driven Development with ABAP Objects
Use test-driven and agile development to write better ABAP programs
Clean up new and legacy code, minimize dependencies, and improve performance
Create integration tests, components tests, unit tests, and more
Winfried Schwarzmann
Rheinwerk Publishing



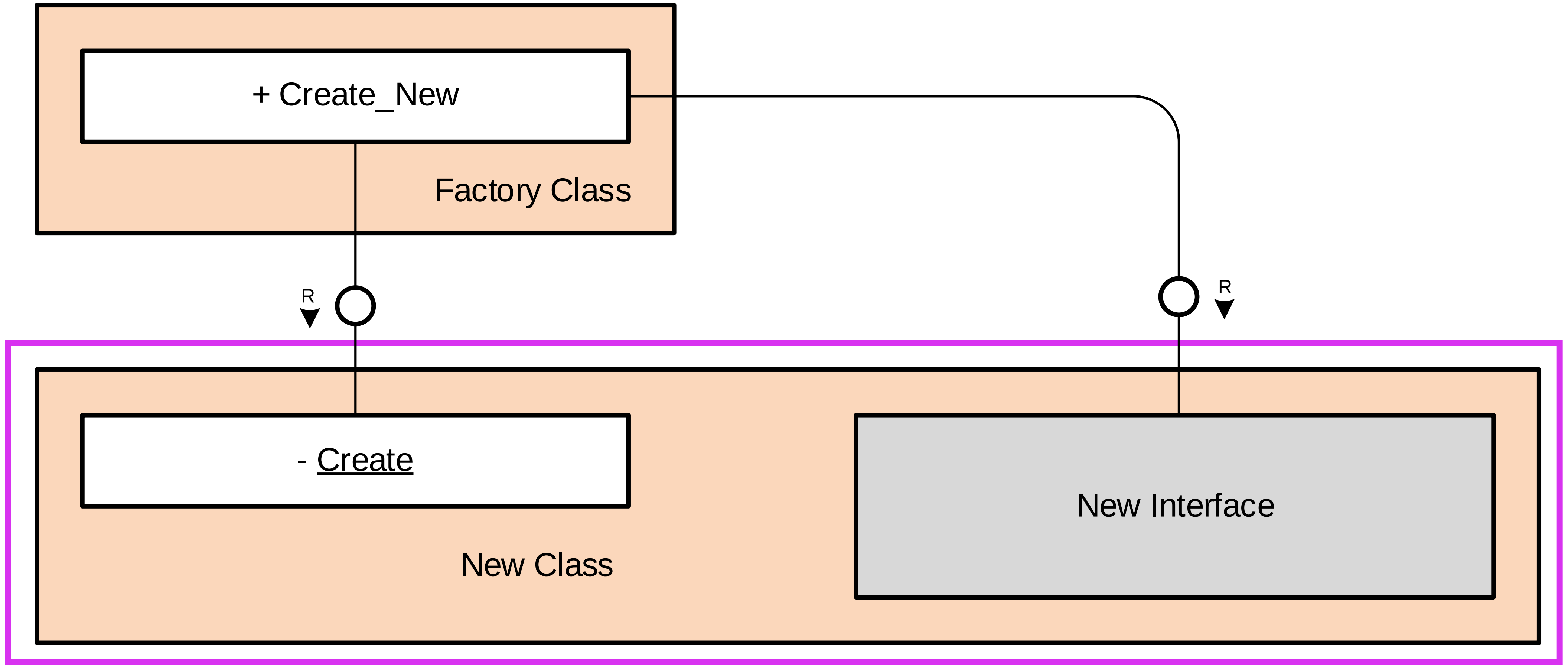




Working for SAP for more than 25 years as:

- Developer and Architect
- Agile Software Engineering Coach

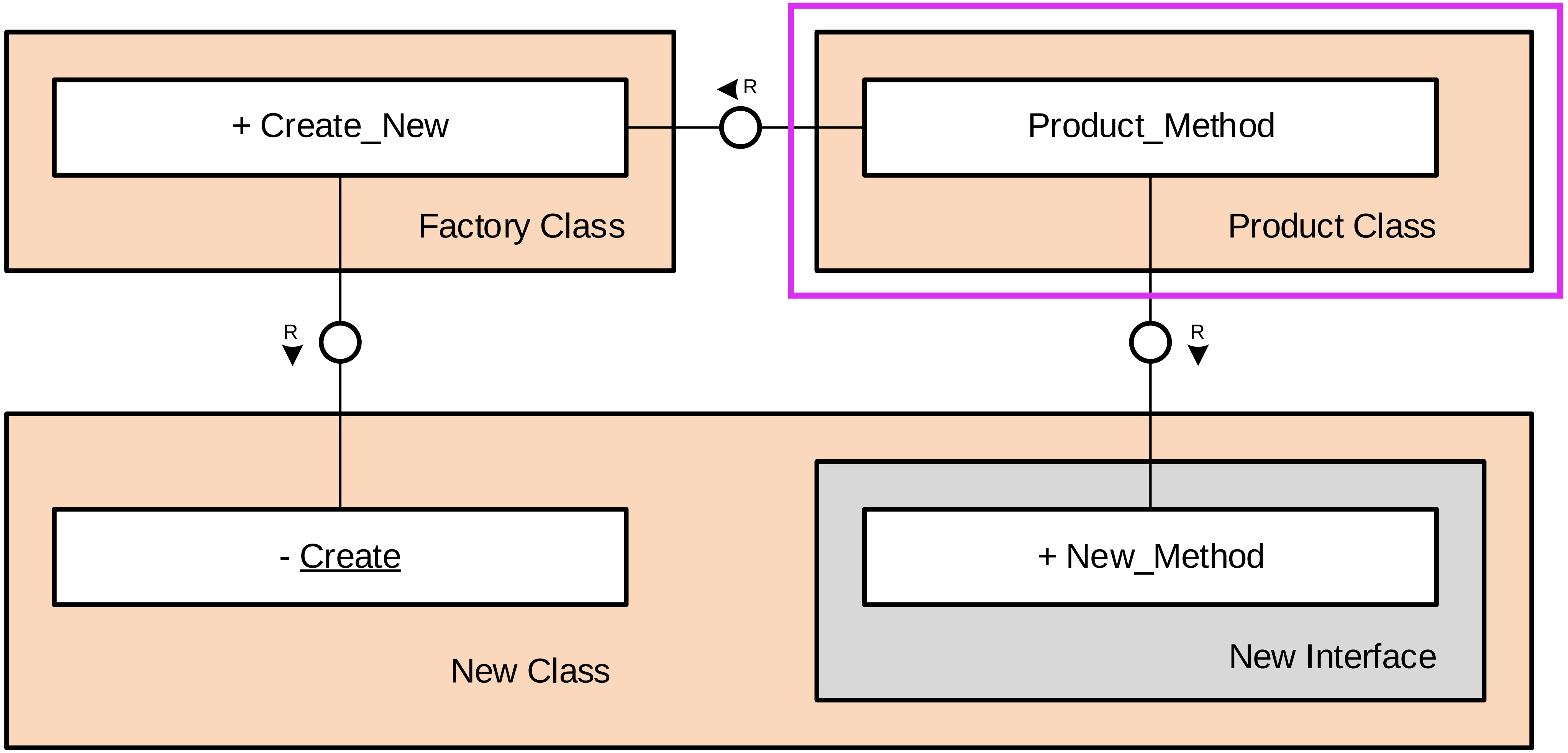
Designing a New Class



Implementing the Skeleton of a New Class

```
CLASS cl_new DEFINITION PUBLIC FINAL CREATE PRIVATE
GLOBAL FRIENDS cl_factory.
  PUBLIC SECTION.
    INTERFACES if_new.

  PRIVATE SECTION.
    CLASS-METHODS create
      RETURNING VALUE(ro_object) TYPE REF TO cl_new.
ENDCLASS.

CLASS cl_new IMPLEMENTATION.
  METHOD create.
    ro_object = NEW cl_new( ).
  ENDMETHOD.
ENDCLASS.
```

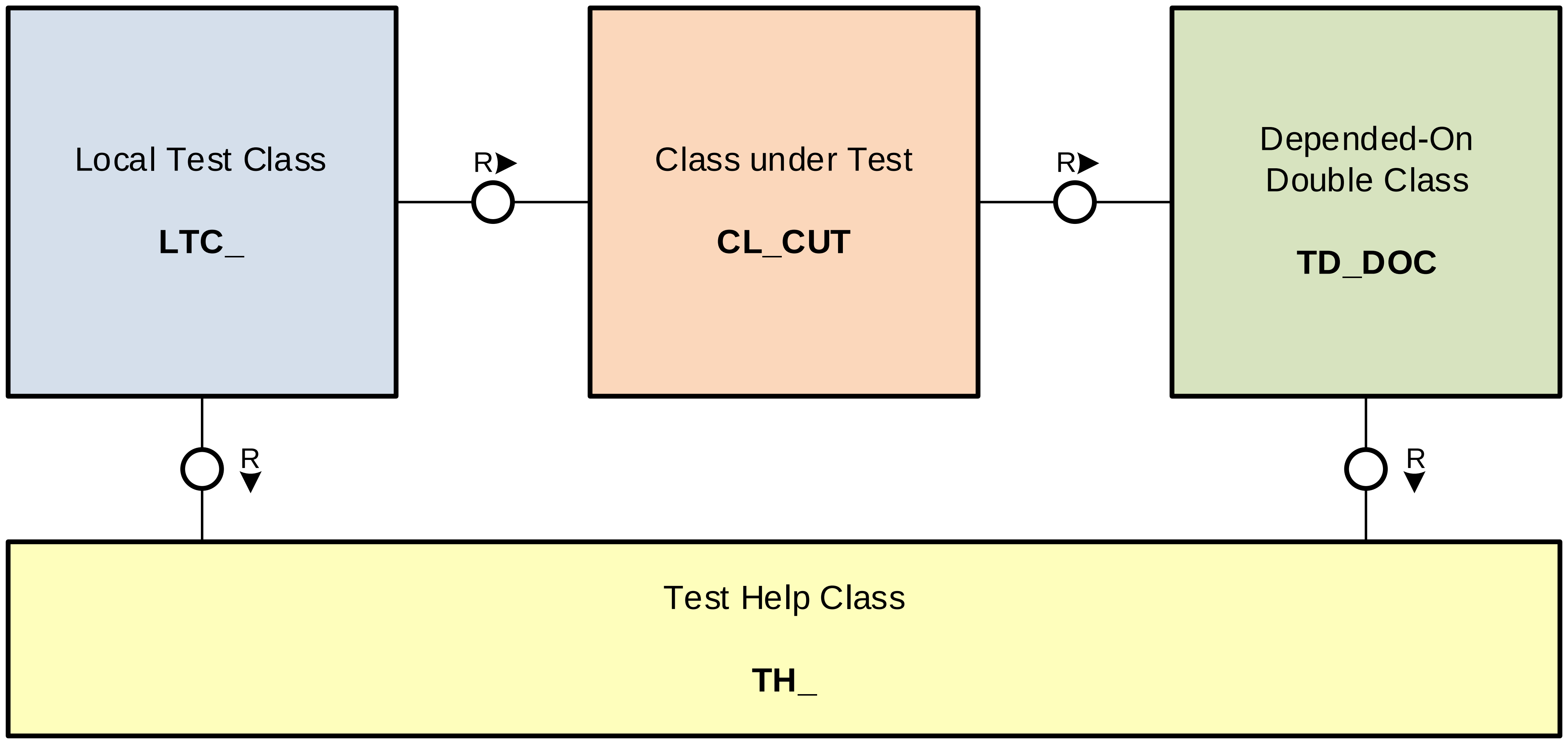
Decoupling from New Class



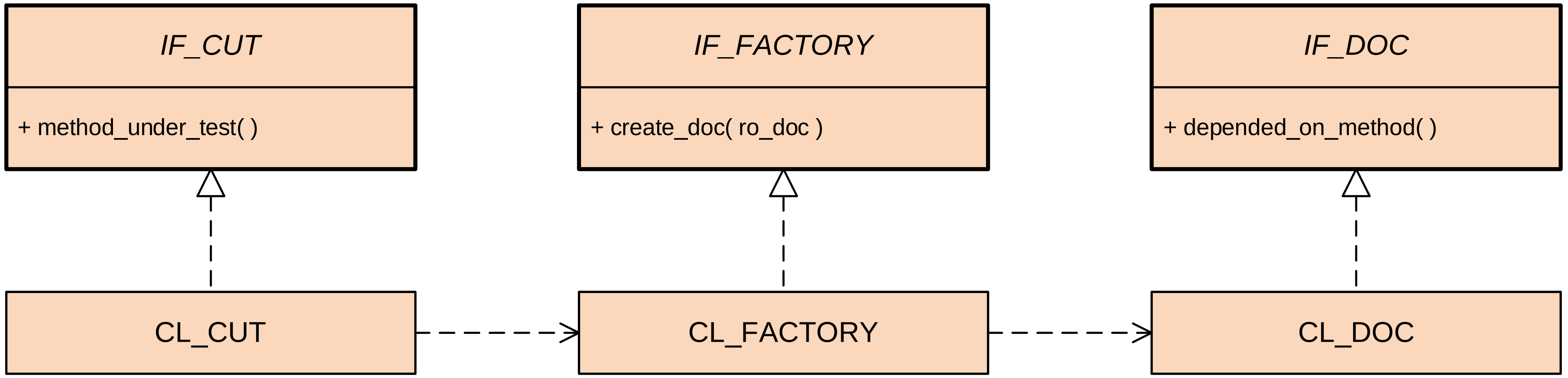
Using the New Class

```
CLASS cl_product DEFINITION PUBLIC CREATE PUBLIC.
  PUBLIC SECTION.
    METHODS product_method.
ENDCLASS.


CLASS cl_product IMPLEMENTATION.
 METHOD product_method.
    …
    DATA(lo_new_object) = cl_factory=>get( )->create_new( ).
    lo_new_object->new_method( ).
    …
  ENDMETHOD.
ENDCLASS.
```

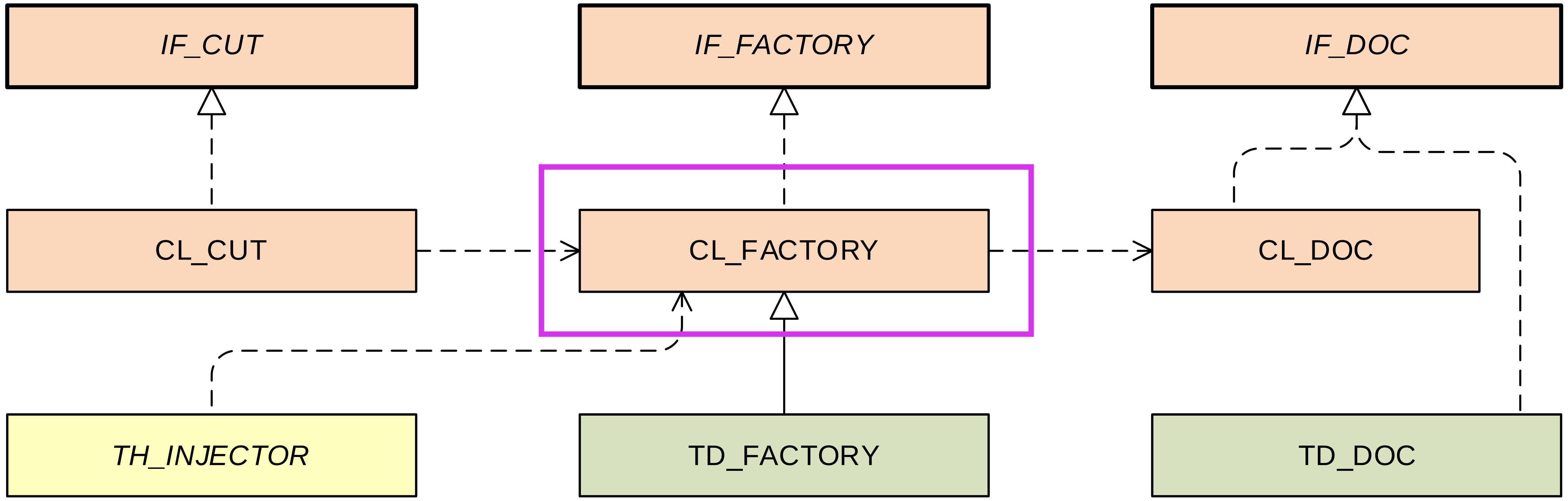
Test Abbreviations



Clean Design: Product Classes



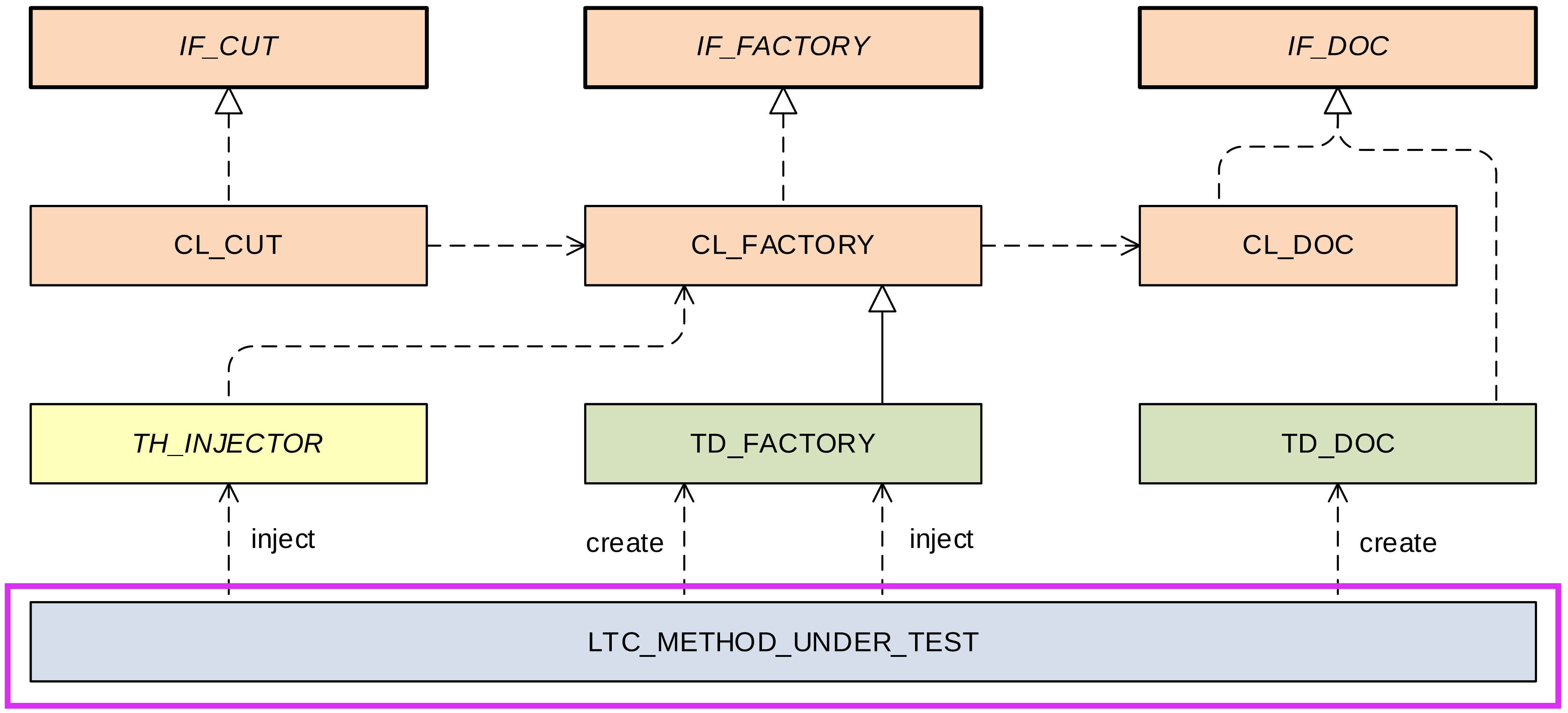
Clean Design: Double Classes



Singleton Factory

```
CLASS cl_factory DEFINITION PUBLIC CREATE PROTECTED
GLOBAL FRIENDS th_injector.
  PUBLIC SECTION.
    CLASS-METHODS get
      RETURNING VALUE(ro_factory) TYPE REF TO if_factory.

    INTERFACES if_factory.

  PRIVATE SECTION.
    CLASS-DATA so_factory TYPE REF TO if_factory.
ENDCLASS.
```

Clean Design: Local Test Class



Local Test Class: Injecting Singleton Double (1)

```
CLASS ltc method under test IMPLEMENTATION.
  METHOD setup.
    DATA(lo_factory_double) = td_factory=>create( ).
    th_factory_injector=>inject_factory( lo_factory_double ).

    mo_doc_double = td_doc=>create( ).
    lo_factory_double->inject_doc( mo_doc_double ).
  ENDMETHOD.

  METHOD test_method.
    mo_doc_double->configure( ... ). ”late configuration
    ...
  ENDMETHOD.
ENDCLASS.
```

Local Test Class: Injecting Singleton Double (2)

```
CLASS ltc_method_under_test DEFINITION FINAL
FOR TESTING DURATION SHORT RISK LEVEL HARMLESS.
  PRIVATE SECTION.
    METHODS setup.
    METHODS test_method FOR TESTING.

    DATA mo_doc_double TYPE REF TO td_doc.
ENDCLASS.
```

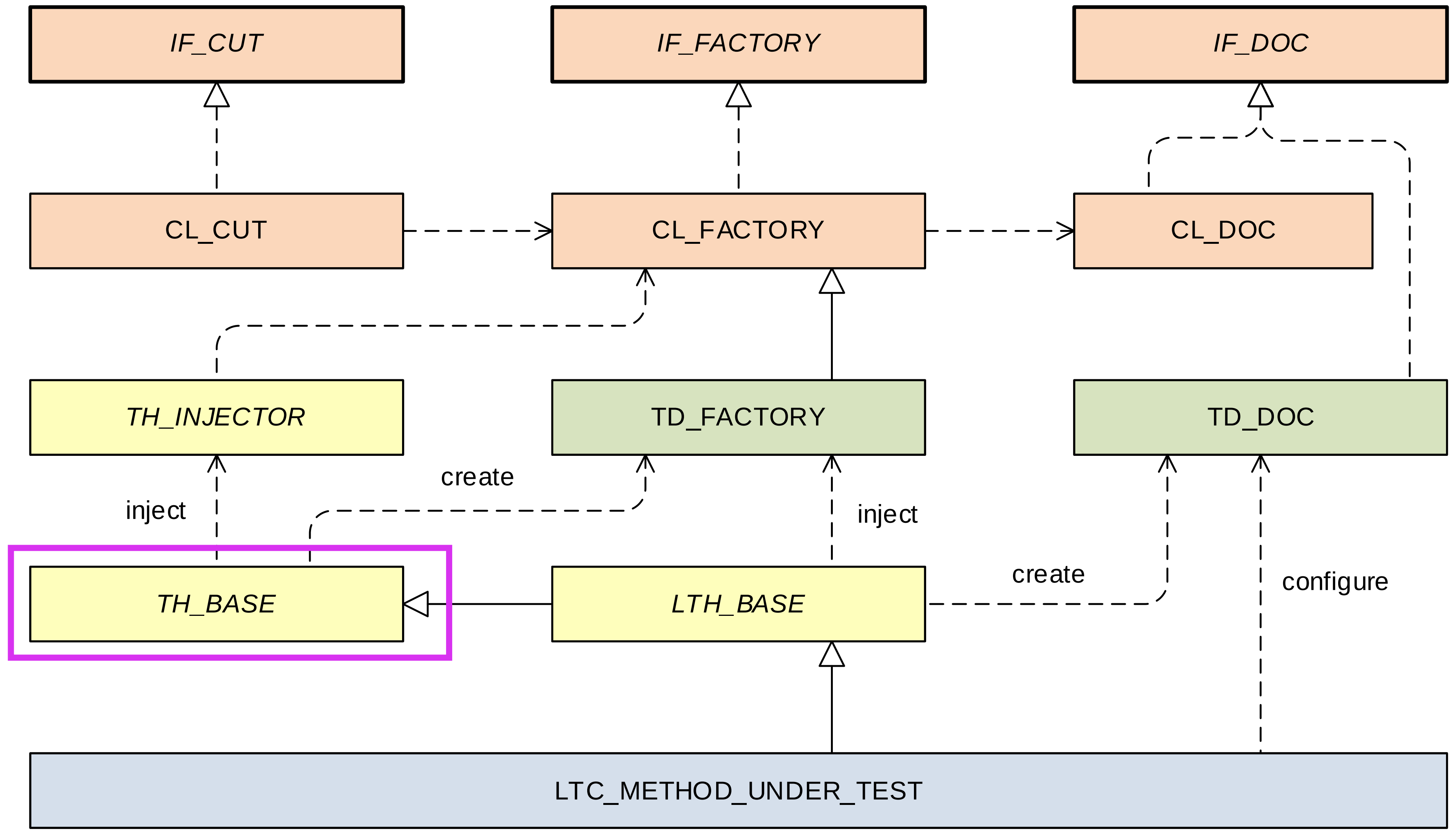
Local Test Class: Injecting Non-Singleton Doubles (1)

```
CLASS ltc_method_under_test IMPLEMENTATION.
  METHOD setup.
    mo_factory_double = td_factory=>create( ).
    th_factory_injector=>inject_factory( mo_factory_double ).
  ENDMETHOD.

  METHOD test_method.
    DATA(lo_doc_double_1) = td_doc=>create(...). "configuration
    DATA(lo_doc_double_2) = td_doc=>create(...). "configuration
    mo_factory_double->inject_doc( lo_doc_double_1 ).
    mo_factory_double->inject_doc( lo_doc_double_2 ).
    ...
  ENDMETHOD.
ENDCLASS.
```

Local Test Class: Injecting Non-Singleton Doubles (2)

```
CLASS ltc_method_under_test DEFINITION FINAL
FOR TESTING DURATION SHORT RISK LEVEL HARMLESS.
  PRIVATE SECTION.
    METHODS setup.
    METHODS test_method FOR TESTING.

    DATA mo_factory_double TYPE REF TO td_factory.
ENDCLASS.
```

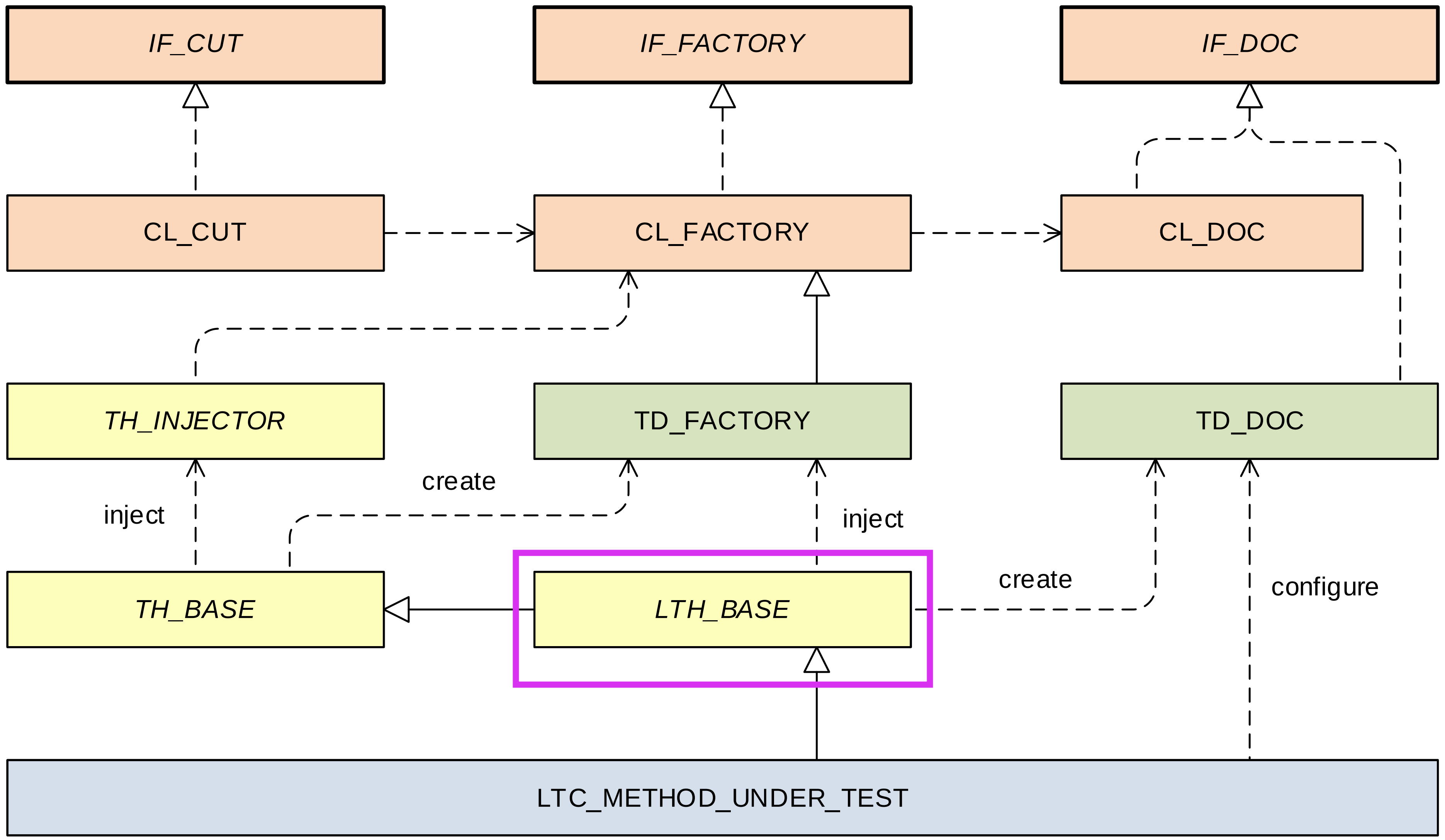
Clean Design: Test Base Classes



Global Test Base Class: Injecting Factory Double

```
CLASS th_base DEFINITION PUBLIC ABSTRACT
FOR TESTING DURATION SHORT RISK LEVEL HARMLESS.
  PROTECTED SECTION.
    DATA mo_factory_double TYPE REF TO td_factory.

  PRIVATE SECTION.
    METHODS setup.
ENDCLASS.

CLASS th_base IMPLEMENTATION.
  METHOD setup.
    mo_factory_double = td_factory=>create( ).
    th_factory_injector=>inject_factory( mo_factory_double ).
  ENDMETHOD.
ENDCLASS.
```

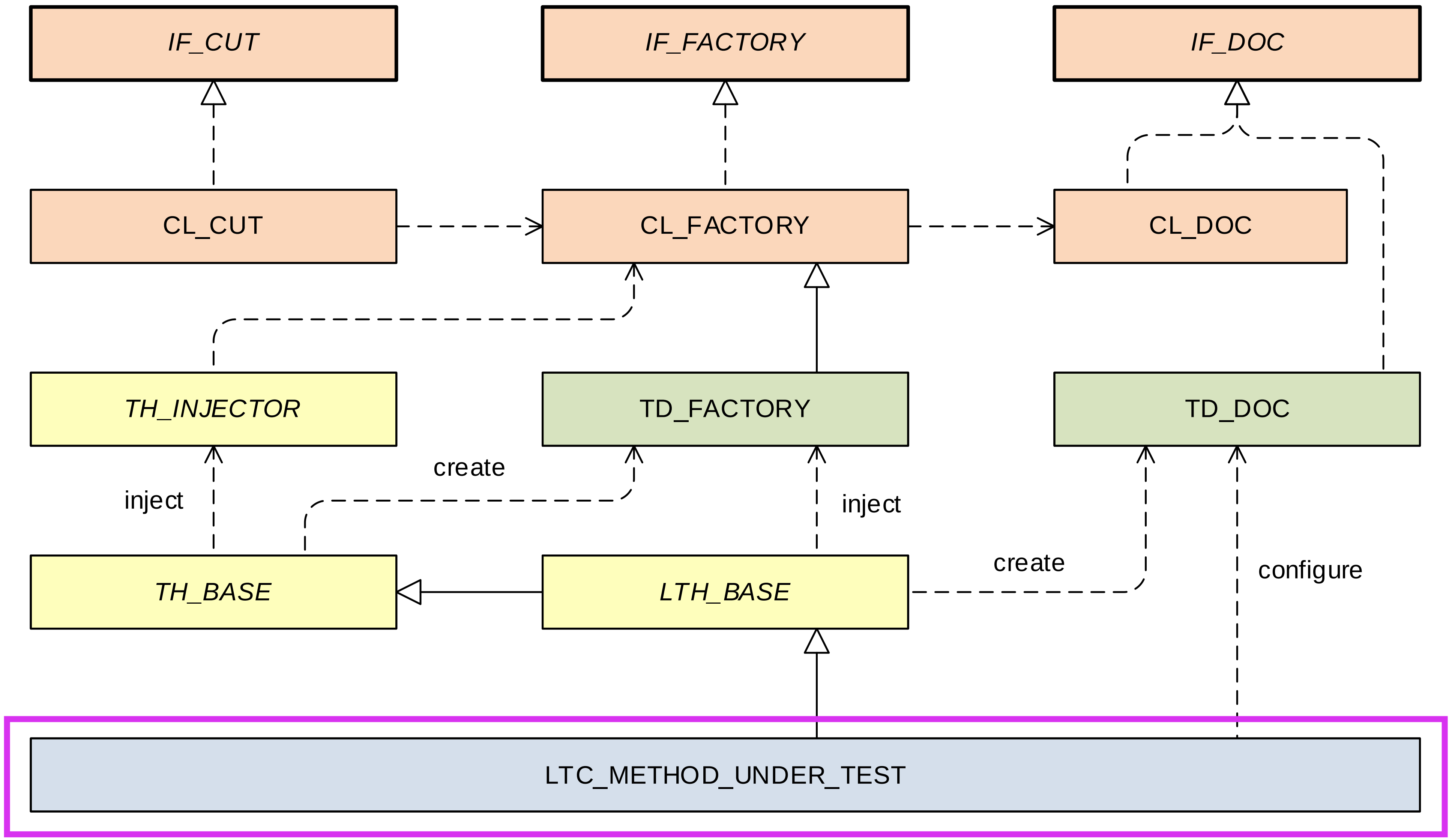
Clean Design: Test Base Classes



Local Test Base Class: Injecting Singleton Double

```
CLASS lth_base DEFINITION ABSTRACT
INHERITING FROM th_base
FOR TESTING DURATION SHORT RISK LEVEL HARMLESS.
  PROTECTED SECTION.
    DATA mo_doc_double TYPE REF TO td_doc.

  PRIVATE SECTION.
    METHODS setup.
ENDCLASS.

CLASS lth_base IMPLEMENTATION.
  METHOD setup.
    mo_doc_double = td_doc=>create( ).
    mo_factory_double->inject_doc( mo_doc_double ).
  ENDMETHOD.
ENDCLASS.
```

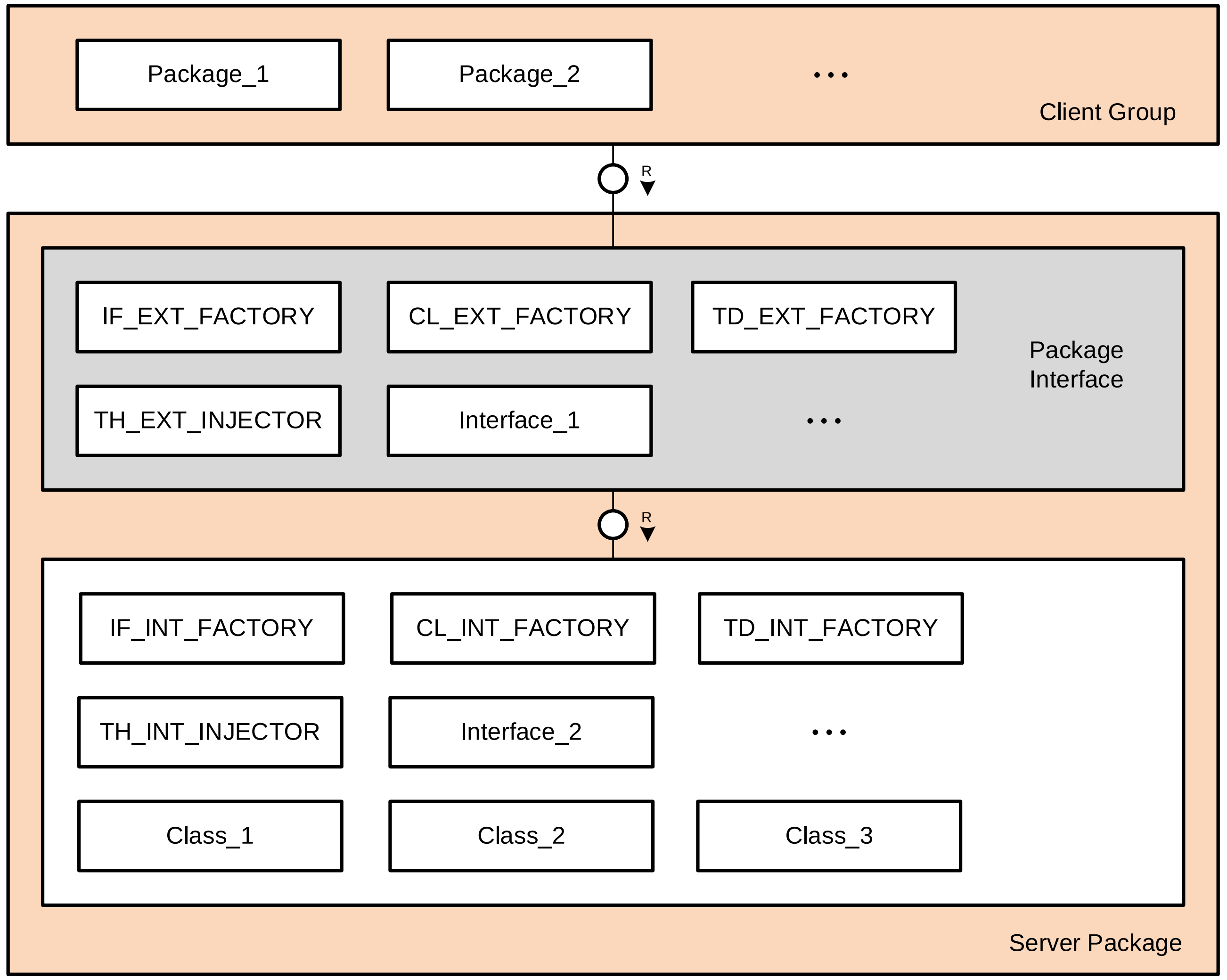
Clean Design: Test Base Classes



Local Test Class: Configuring Singleton Double

```
CLASS ltc method under test DEFINITION
INHERITING FROM lth_base FINAL
FOR TESTING DURATION SHORT RISK LEVEL HARMLESS.
 PRIVATE SECTION.
    METHODS test_method FOR TESTING.
ENDCLASS.


CLASS ltc_method_under_test IMPLEMENTATION.
  METHOD test_method.
    mo_doc_double->configure( ... ). ”late configuration
    ...
  ENDMETHOD.
ENDCLASS.
```

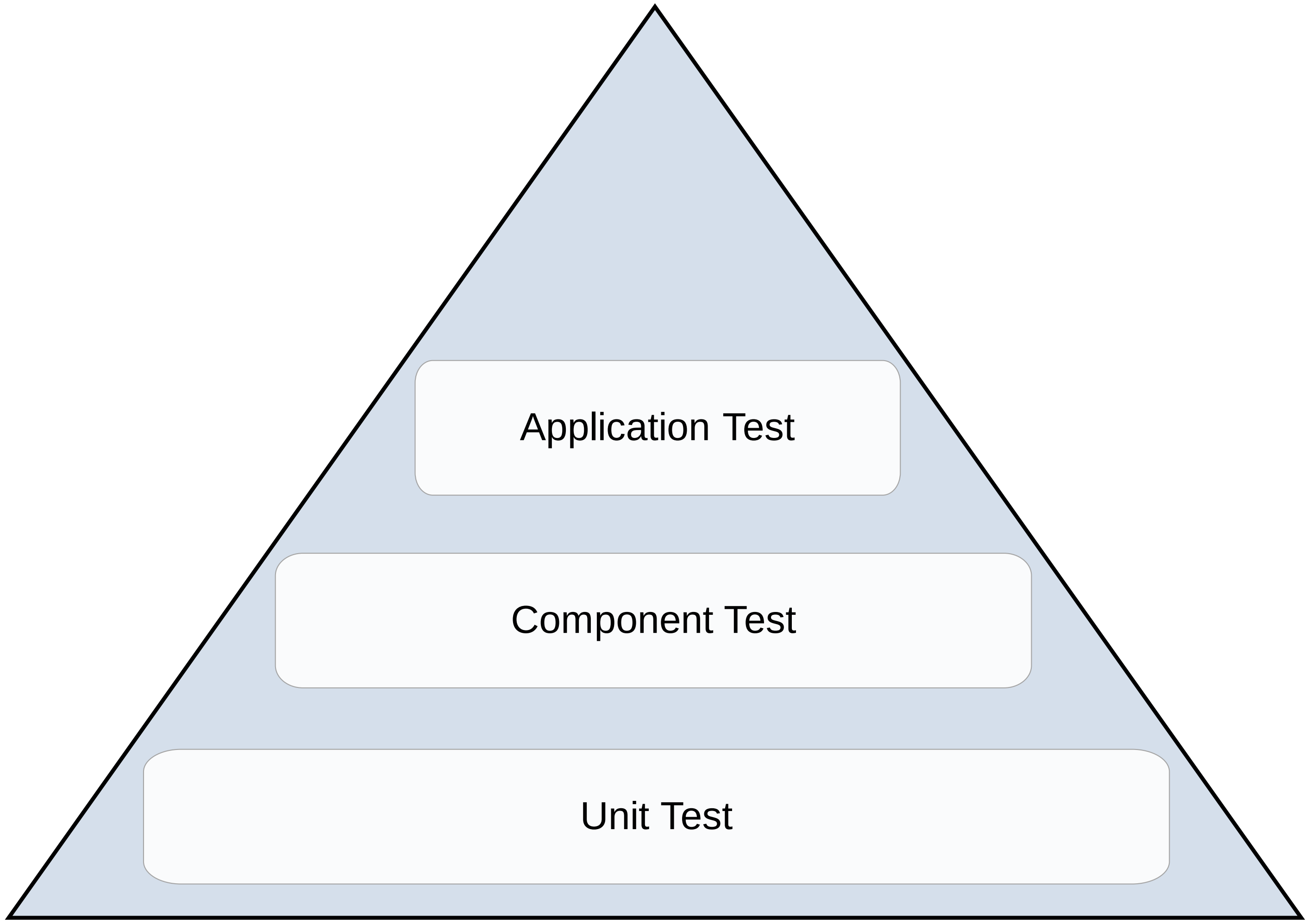
Clean Package



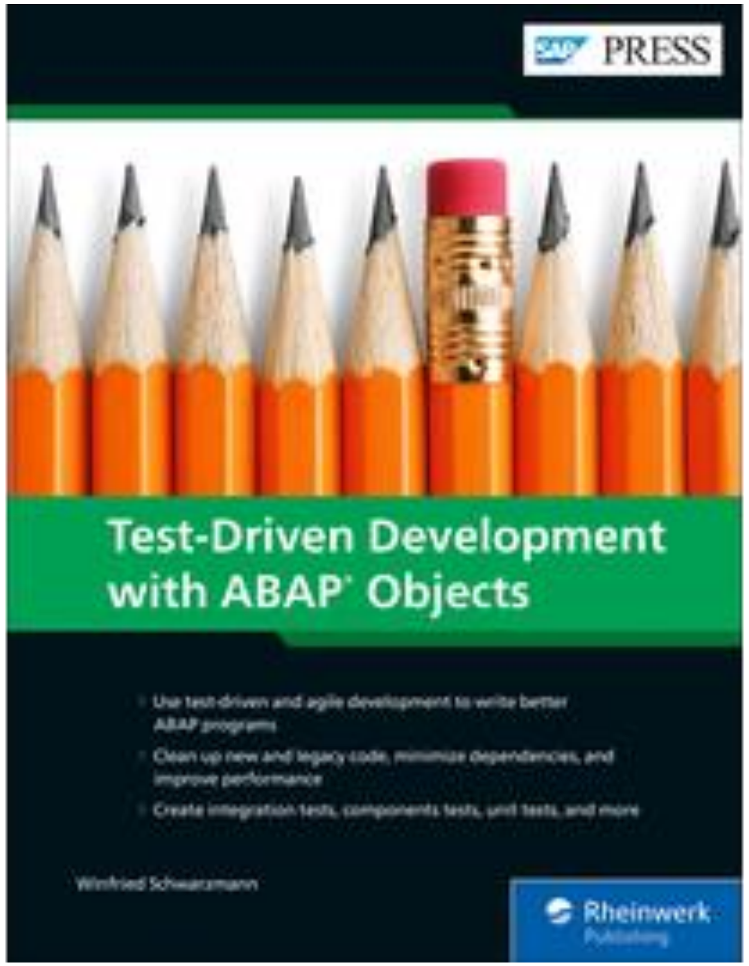
External factory provides access to the interfaces of the API Units.

Internal factory decouples encapsulated units.

Test Pyramid



Test-Oriented Improvement Process


SAP PRESS
Test-Driven Development with ABAP Objects
Use test-driven and agile development to write better ABAP programs
Clean up new and legacy code, minimize dependencies, and improve performance
Create integration tests, components tests, unit tests, and more
Winfried Schwarzmann
Rheinwerk Publishing

Book & E-Book
SAP Press
2019

Content:

Part I:
Modernization of legacy code

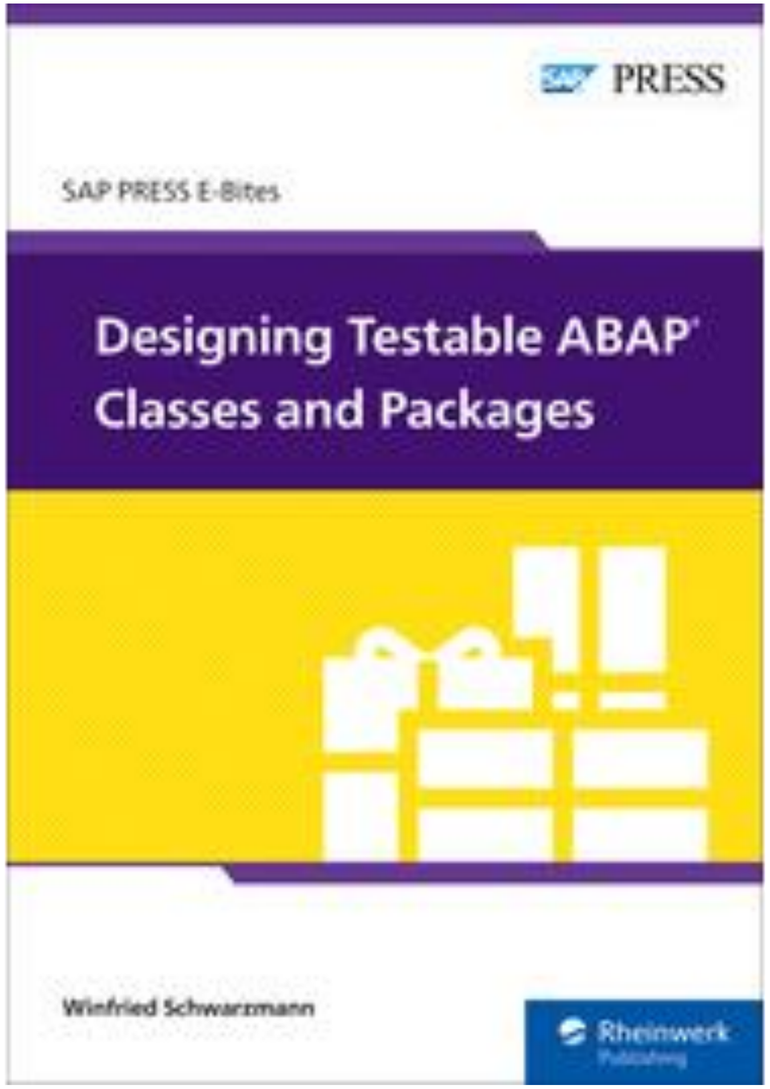
Part II:
Test infrastructure

Part III:
Test-driven development for new code

Part IV:
Agile software engineering

Part V:
Development & test tools

Clean Design



E-Book
SAP Press
2022

Content:

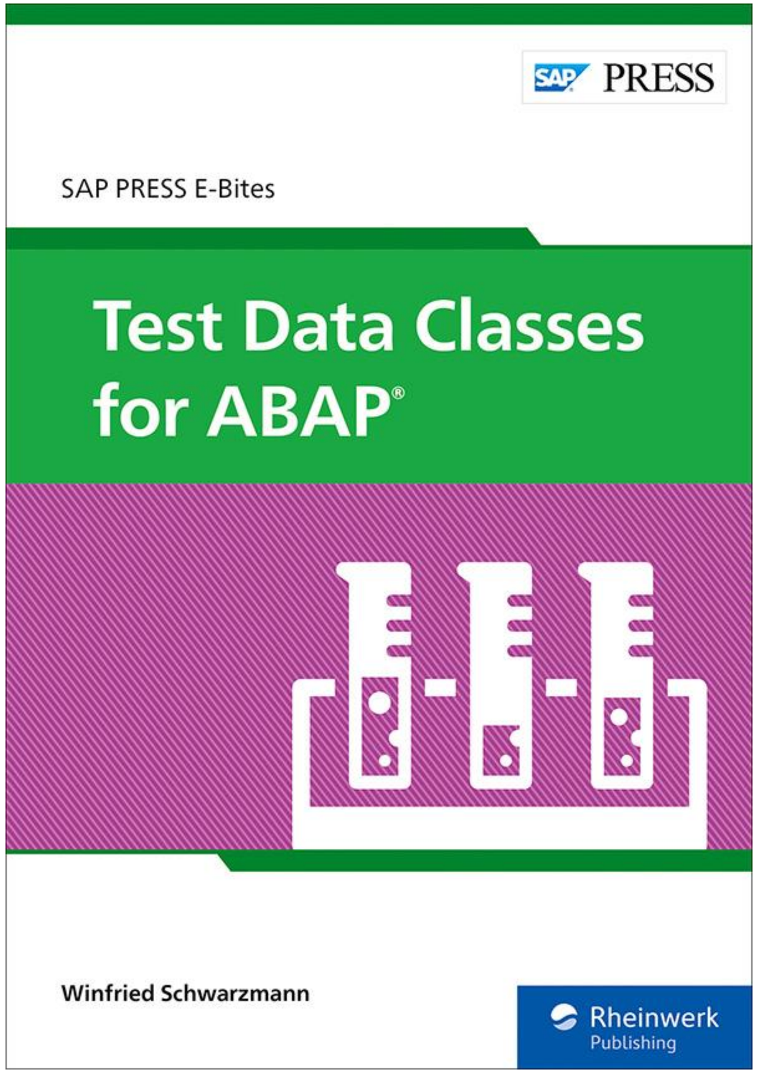
Part I: Theory

- Classes
- Test Classes
- Packages
- BAdIs

Part II: Training

Exercises with solutions for individuals and teams

Test Data Classes



E-Book
SAP Press
2021

Content:

Part I: Theory

- Designing and implementing test data classes

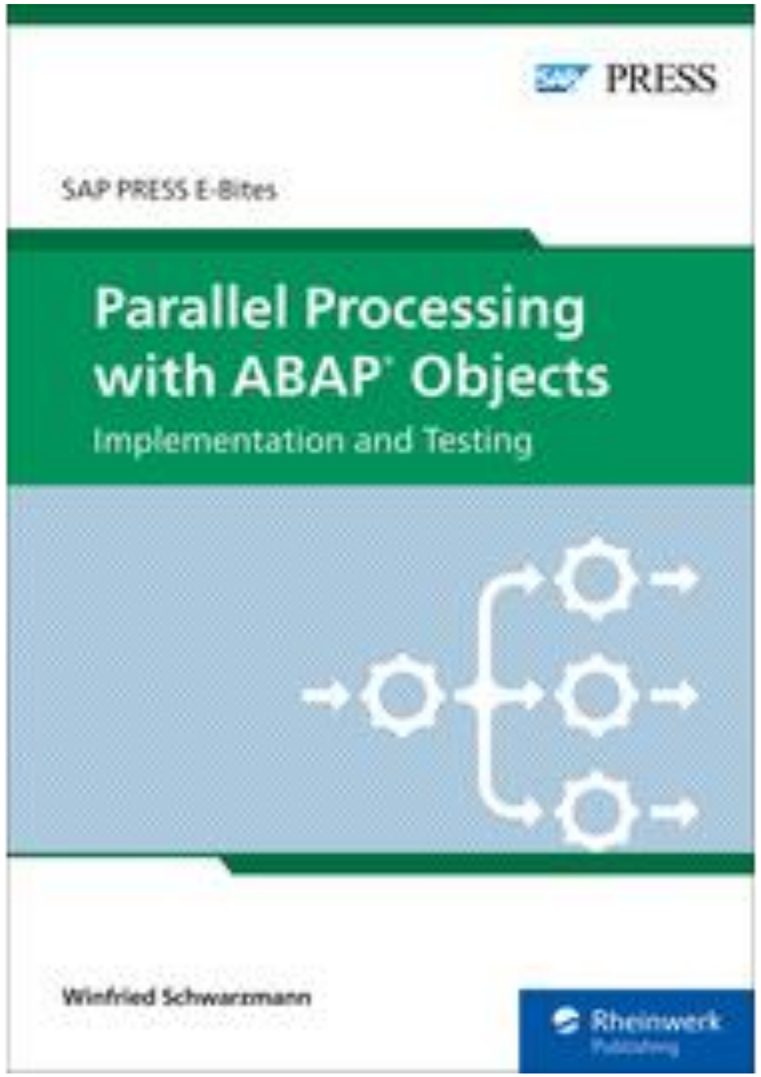
- Using test data classes for the entire test pyramid

- Using test message classes for verifying error handling

Part II: Training

Exercises with solutions
for individuals and teams

Parallel Processing



E-Book
SAP Press
2022

Content:

Implementing and testing parallel processes

Inheriting test data classes from productive data classes

Implementing, refactoring, and enhancing package design

Winfried Schwarzmann

SAP SE

THE BEST RUN SAP®

Follow us

www.sap.com/contactsap

THE BEST RUN SAP