ABAP OO
+ Test framework

João Vitor de Camargo, Suzana Machado, SAP
October, 2019

Day 2

§ Exceptions

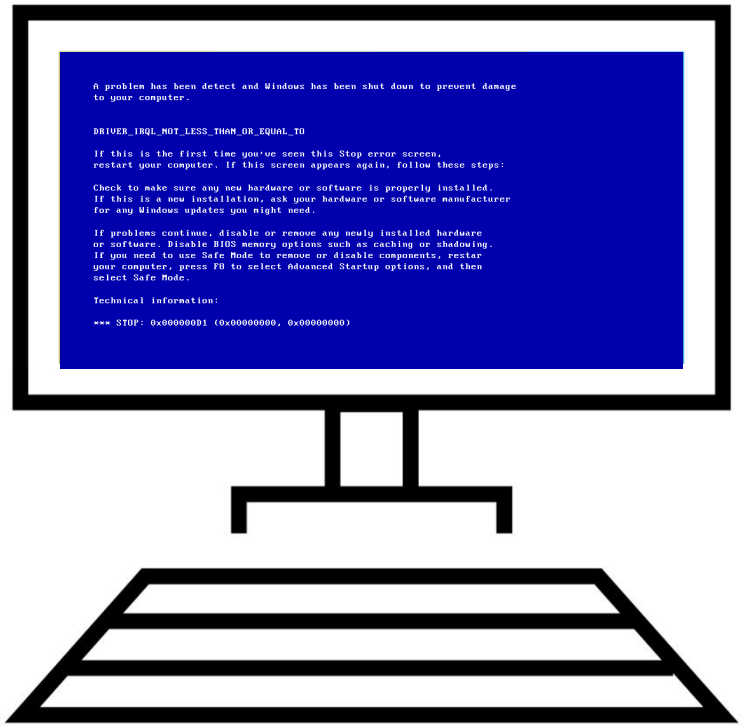
- Types of exceptions in ABAP
- When to use each type

§ Events

- What are they
- How to use

§ Tests

- Test Classes, Test Doubles and Test Helpers
- Best practices


A problem has been detect and Windows has been shut down to prevent damage
to your computer.
DRIVER_IRQL_NOT_LESS_THAN_OR_EQUAL_TO
If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow these steps:
Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.
If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restar
your computer, press F8 to select Advanced Startup options, and then
select Safe Mode.
Technical information:
*** STOP: 0x000000D1 (0x00000000, 0x00000000)

Exceptions

Why exceptions?

Category	ABAP programming error
Runtime Errors	SYNTAX_ERROR
ABAP Program	Z_ABAP_UNCHECKED_EXCEP_JVC_1
Application Component	Not assigned
Date and Time	14.10.2019 19:12:42

Short Text

Syntax error in program "Z_ABAP_UNCHECKED_EXCEP_JVC_1 ".

What happened?

Error in the ABAP Application Program

The current ABAP program "SAPLALDB" had to be terminated because it has come across a statement that unfortunately cannot be executed.

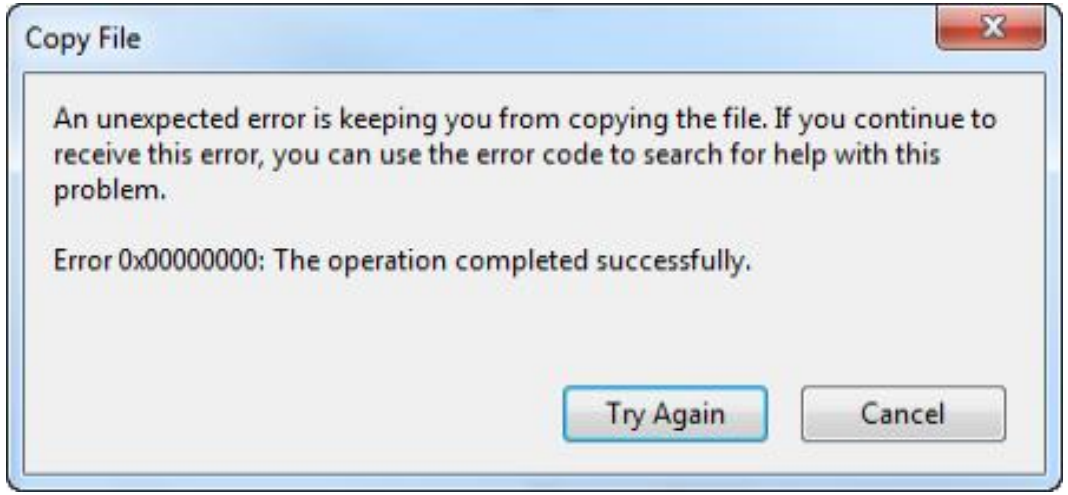
The following syntax error occurred in program "Z_ABAP_UNCHECKED_EXCEP_JVC_1 " in include "Z_ABAP_UNCHECKED_EXCEP_JVC_1 " in line 10:
"The class "LCX_UNCHECKED_EXCEPTION" was not derived from either "CX_ST" "ATIC_CHECK" or "CX_DYNAMIC_CHECK"."
" "
" "

The include has been created and last changed by:
Created by: "CAMARGOJ "
Last changed by: "CAMARGOJ "
Error in the ABAP Application Program

The current ABAP program "SAPLALDB" had to be terminated because it has come across a statement that unfortunately cannot be executed.

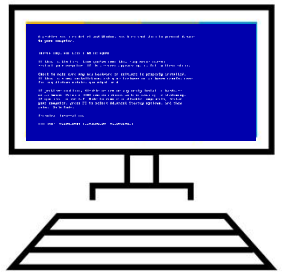
Error analysis

The following syntax error was found in the program
Z_ABAP_UNCHECKED_EXCEP_JVC_1 :
"The class "LCX_UNCHECKED_EXCEPTION" was not derived from either "CX_ST" "ATIC_CHECK" or "CX_DYNAMIC_CHECK"."
" "
" "



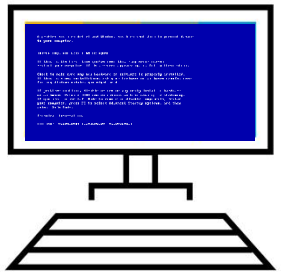
Checked Exceptions

- Extends from **CX_STATIC_CHECK**
- Needs to be explicitly declared in the interface
- Needs to be explicitly be handled or thrown up



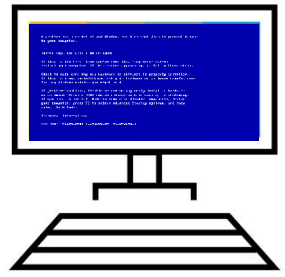
Unchecked Exceptions

- Extends from **CX_NO_CHECK**
- Can not be declared in the interface
  - You can not obligate the code to handle a unchecked exception
- Do not need to be explicitly handled or thrown up
  - It is automatically thrown up in the stack
  - Still needs to be handled if there is a chance of happening



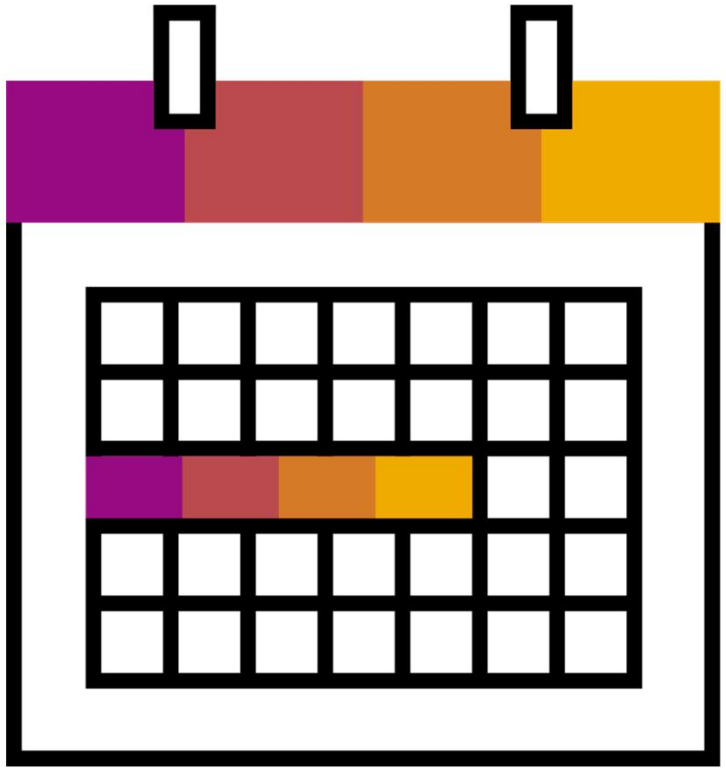
Dynamic Exceptions

- Extends from **CX_DYNAMIC_CHECK**
- Can be declared in the interface (not obligatory)
- Do not need to be explicitly handled
  - But needs to be explicitly thrown up
- **Classic example:** division by 0



HANDS ON

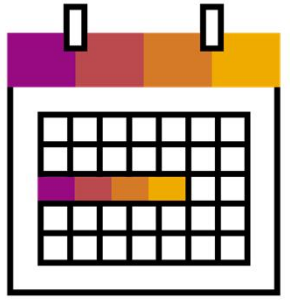
Part 3



Events

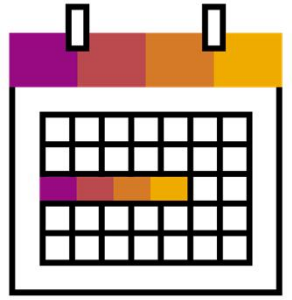
Events

- **Functions that can be triggered**
  - Events can receive parameters, but not return one
- An event must have a **handler**
  - The handler will listen the event
  - When the event is raised, the handler will be called to perform an action



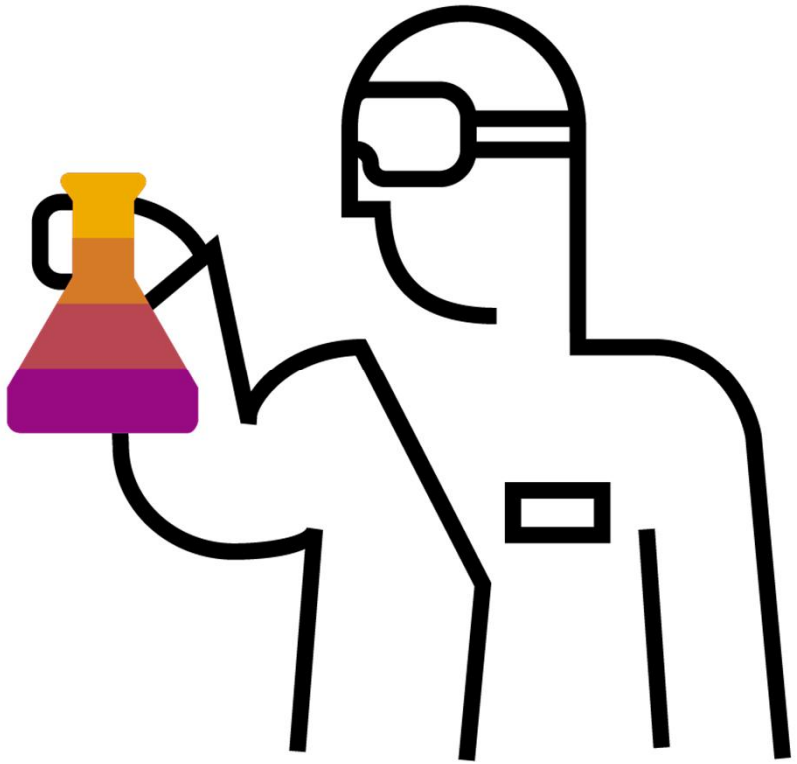
Events

- A handler must be **activated**
  - And deactivated if necessary
- Every event also receives an implicit parameter
  - The **sender**: the one who raised the event

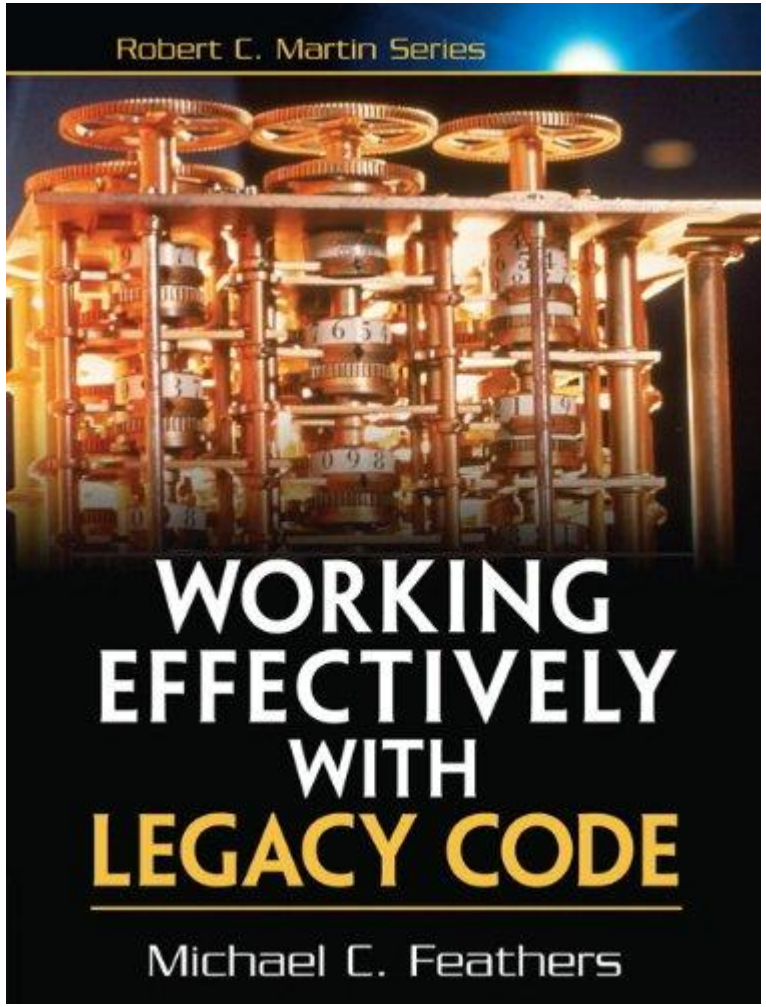


HANDS ON

Part 4



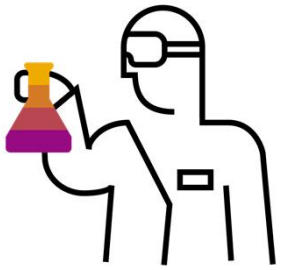
Tests



“Legacy code is code without tests”

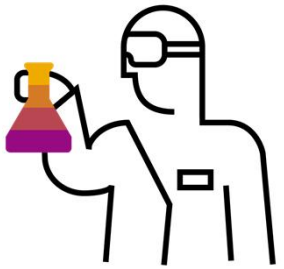
Dependencies

- **Dependencies** should be **isolated**
  - I don’t want my tests to write in productive tables, for example
  - Easy to accomplish with single responsibility + composition
- If you’re having too much work to mock dependencies
  - Then probably there is some shared responsibility
  - Or your tests are not so unitary



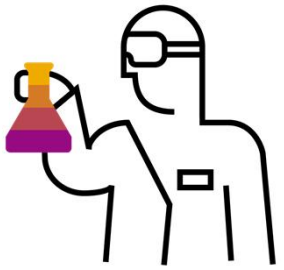
Testing

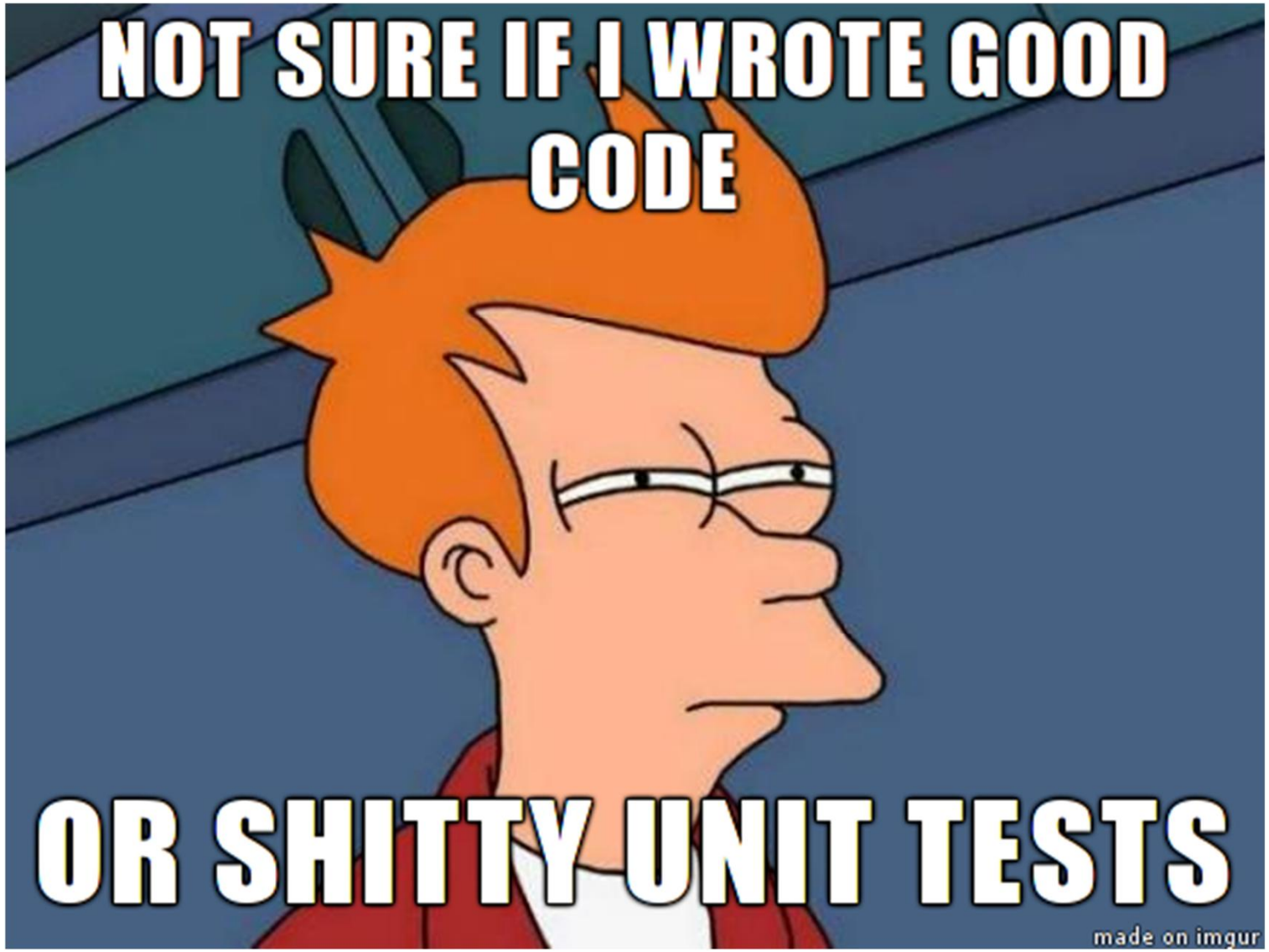
- **Test Class** (LTC)
  - Class with executable tests
  - All methods should be private
- **Friends**
  - A test class can be friends of a productive class
  - In this case, the test class can access private or protected data



Testing

- **Test Double** (LTD)
  - Mocks a dependency (like a writer or a reader, for example)
- **Test Helper** (LTH)
  - Used to help testing a class
  - Example: a class that raises events





Thank you.

Contact information:

João Vitor de Camargo
Developer and Quality Engineer at HCM North America
joao.vitor.camargo@sap.com

Suzana Machado
Developer and Scrum Architect at HCM Spain
suzana.machado@sap.com