Clean ABAP in der Praxis

ABAPConf 2021

Michael Keller
educatedBytes GmbH

Vorstellung

Michael Keller

Jahrgang 1982

Diplom-Wirtschaftsinformatiker (FH)

Entwickler und Berater

Dozent und Blogger

Wenn ich diesen Vortrag halten müsste, wäre er ganz kurz:

„macht es einfach, Punkt“

Uwe F. aus der SAP Community

Clean ABAP

Vorstellung und Motivation

Ausgangssituation

„Einmal entwickelt, tausendfach gewartet.“

„Einmal geschrieben, tausendfach gelesen.“

Softwarealterung

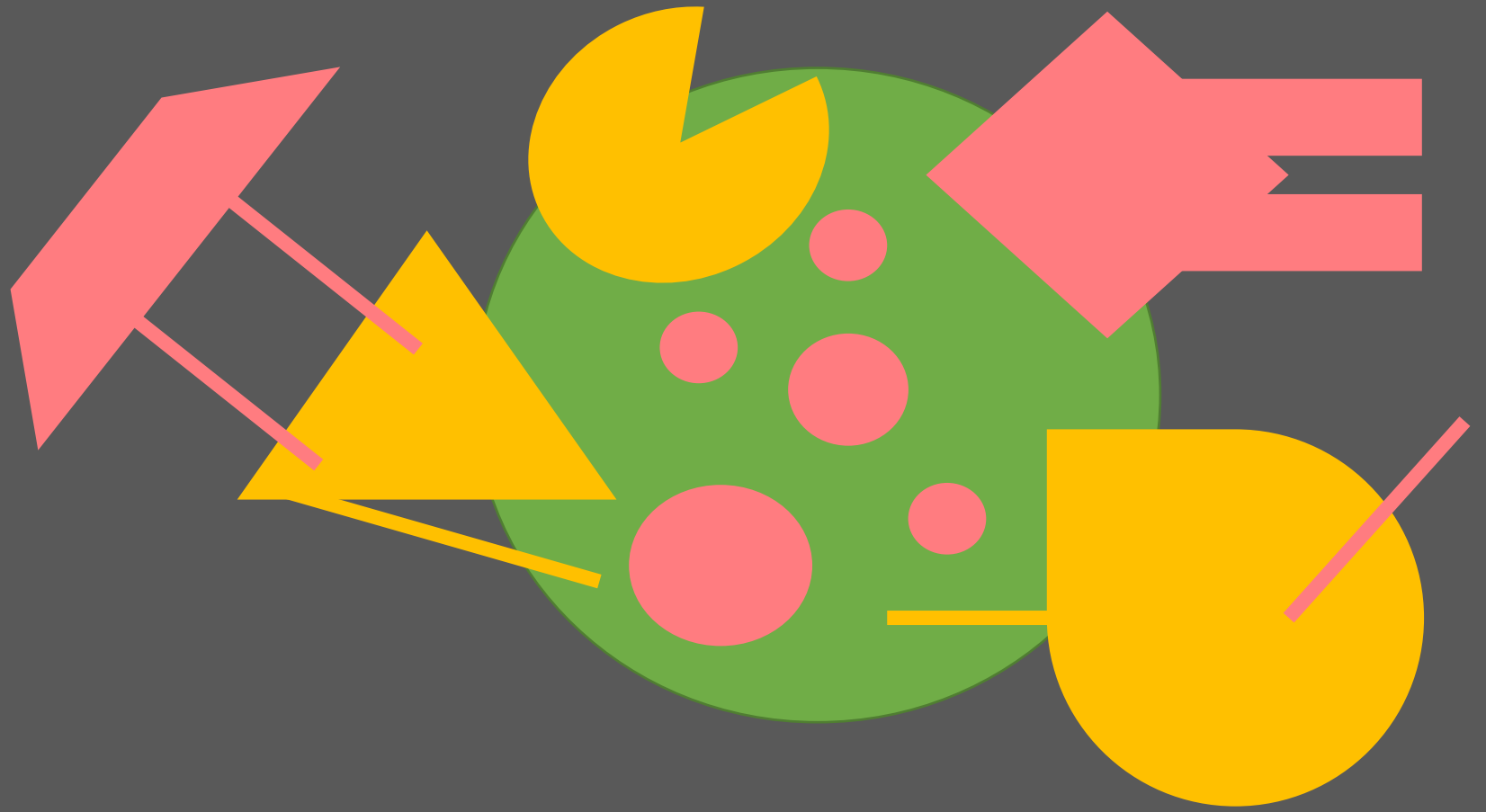
Softwareerosion

Legacy Code

Einige Kennzeichen

- historisch gewachsen und wichtig
- unterschiedliche Entwicklungsstile
- unterschiedlicher Zeitgeist
- unterschiedliche Lösungen für dieselbe Anforderung
- wiederverwendeter Quelltext
- fehlende, veraltete und unvollständige Dokumentation
- keine Ansprechpartner

Veränderung



Strenge, expressionistische Darstellung eines historisch gewachsenen MV45AFZZ-Includes
Vorlage unbekannt, vermutlich mehrere Entwickler in verschiedenen Schaffensphasen und IT-Epochen

Ein Lösungsbaustein

Clean Code

„Als „sauber“ bezeichnen Softwareentwickler in erster Linie Quellcode, aber auch Dokumente, Konzepte, Regeln und Verfahren, die intuitiv verständlich sind. Als intuitiv verständlich gilt alles, was mit wenig Aufwand und in kurzer Zeit richtig verstanden werden kann.“

Quelle: Wikipedia

Zentrale Elemente

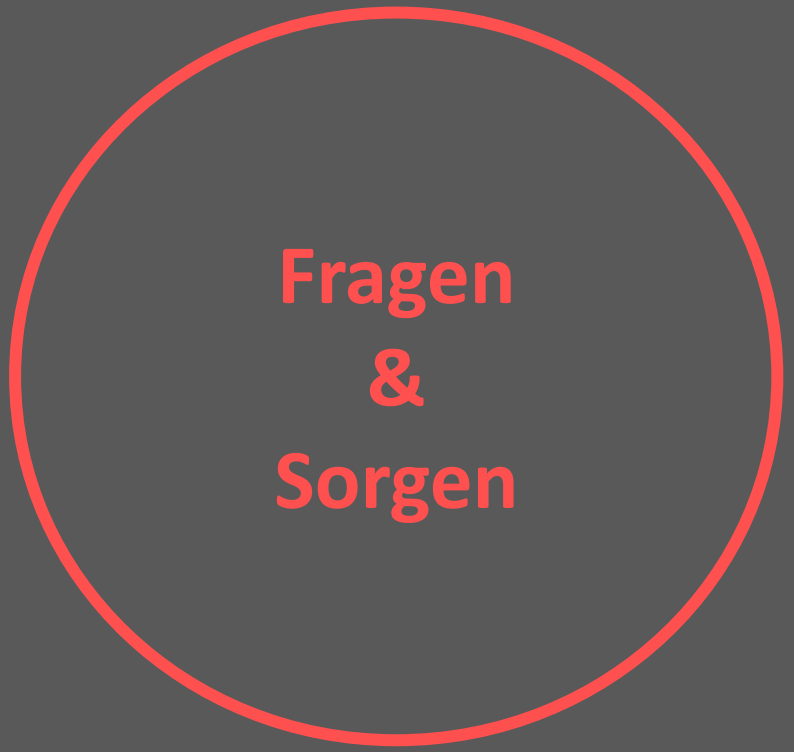
Wartbarkeit

Testbarkeit

Verständlichkeit

Lesbarkeit

Zentrale Themen



Weitere Lösungsbausteine

- ABAP Objects
- ABAP Development Tools for Eclipse
- dauerhaftes Lernen
- Erfahrung
- SAP Community, Twitter, LinkedIn
- Soft Skills

Lesbarkeit und Verständlichkeit

Einmal geschrieben, tausendfach gelesen.

Noch öfters darüber nachgedacht.

Formatierung

Fürs Lesen optimiert.

Ein irritierendes Angebot

Sehr gEEhrte Damen und Herren,

Es GEHT UM *iHr* Interesse

An **Unserem** *Product*

vOm 09. Dezember Zweitausendeinundzwanzig.

Wir Bieten **ihnen** Wie folgt **An**.

Entwicklungsumgebung

ABAP Formatter nutzen

nicht mehr als 120 Zeichen pro Zeile

Schritt für Schritt

```
IF sy-subrc = 0. ... ELSEIF sy-subrc = 1. ... ELSE. ... ENDIF.
```

eine Zeile, eine Anweisung

```
IF sy-subrc = 0.
  ...
ELSEIF sy-subrc = 1.
  ...
ELSE.
  ...
ENDIF.
```

Ausrichtung und Leerzeichen

```
DATA lv_maxrc        TYPE        abap_bool.
```

```
DATA lv_maxrc TYPE abap_bool.
```

unnötige Leerzeichen vermeiden

Leerzeilen

```
READ TABLE ...

IF sy-subrc = 0.

  ...

ENDIF.
```

```
READ TABLE ...
IF sy-subrc = 0.
  ...
ENDIF.
```

unnötige Leerzeilen vermeiden

Namen

Gut gewählt tragen sie zur Lesbarkeit und zum Verständnis bei.

Sprache

englische Begriffe

Substantive für Klassen und Interfaces

Verben für Methoden

```
METHODS besorge_bestellung ...
METHODS get_bestellung ...
```

```
METHODS get_purchase_order ...
METHODS get_pur_order ...
METHODS get_po ...
```

Begriffswahl

Geschäftsobjekte und technische Objekte beachten

Fachbereich

IT

Begriffsfindung

- Konzepte, Dokumentationen usw. aus Projekt
- Gespräche mit Anwendern und Kollegen
- SAP Onlinehilfe
- BAPI Explorer
- SAPterm
- Oberflächen von Transaktionen in DE bzw. EN
- Default-Komponentennamen von Datenelementen
- Google Übersetzer und LEO Wörterbuch

Namensgebung

erklärende Namen ohne zusätzliche technische Aspekte

```
CONSTANTS co_char1 TYPE char1 VALUE 'N'.
DATA ls_t001 TYPE STANDARD TABLE ...
CLASS zcl_t001 ...
METHODS a_miracle_happens ...
```

```
CONSTANTS co_answer_no TYPE char1 VALUE 'N'.
DATA ls_company_code TYPE STANDARD TABLE ...
CLASS zcl_company_codes ...
METHODS read_company_codes_from_db ...
```

Trennstellen und Abkürzungen

Wahl der Trennstelle und Abkürzung nach Verständlichkeit

```
METHODS getpofromdb ...
METHODS getpo_fromdb ...
METHODS get_p_o_from_d_b ...
```

```
METHODS get_po_from_db ...
METHODS get_pur_order_from_db ...
METHODS get_purchase_order_from_db ...
```

Kommentare

“ Weniger ist mehr! Und auffälliger.

Keine Strukturen

Struktur per Klassen und Methoden

```
“ Selektion
SELECT * FROM EKKO ...

“ Vorbereitung Ausgabe
CALL FUNCTION ’LVC_FIELDCATALOG_MERGE’ ...

“ Ausgabe
CALL FUNCTION ’REUSE_ALV_GRID_DISPLAY_LVC’ ...
```

Kein Mehrwert

ohne echten Mehrwert kein Kommentar

```
" lv_max_value kann nach Rücksprache mit Andreas
" niemals den Wert 99 überschreiten, was ich aber
" nicht so sehe. Egal, gleich ist Wochenende!
IF lv_counter < lv_max_value.
  ...
ENDIF.
```

Überflüssig

keine Quelltextdoppelung

warum, nicht was

```
**********************
* START-OF-SELECTION *
**********************
START-OF-SELECTION.
```

```
" an dieser Stelle erhöhen wir die Variable um 1
lv_counter = lv_counter + 1.
```

Falsch bzw. irreführend

gute Bezeichner brauchen kein erläuterndes Kommentar

```
“ für 16% Steuer rechnen
CONSTANTS c_t TYPE ... VALUE '0.19'.
```

Auskommentierter Quelltext

auskommentierter Quelltext wird entfernt

```
" lv_counter = lv_counter + 1.
lv_counter = lv_counter + 2.
```

Testbarkeit

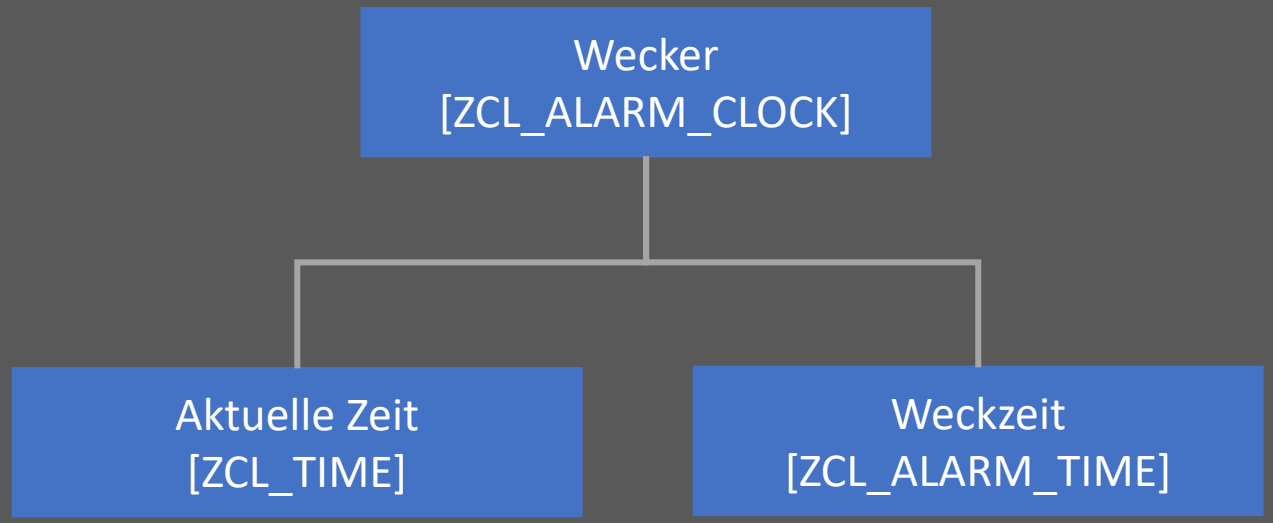
Mut zur Veränderung - mit Sicherheit.

Analoges Beispiel

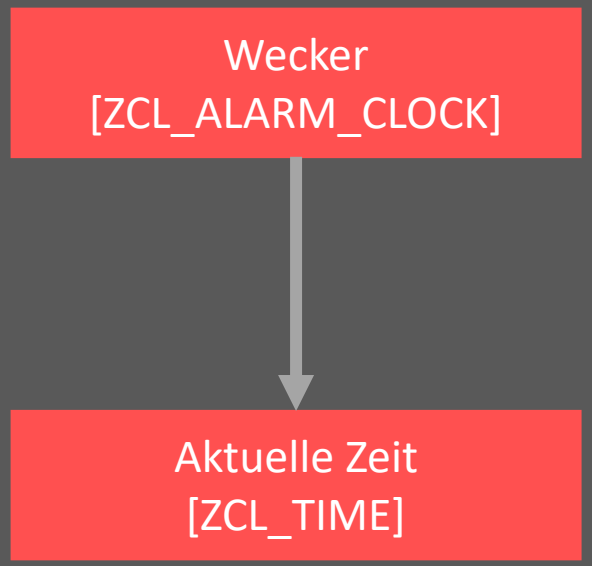
Digitales Beispiel

„Noch 5 Stunden bis zum Wecken…“

01:00 Uhr in der Nacht, Weckzeit ist 06:00 Uhr

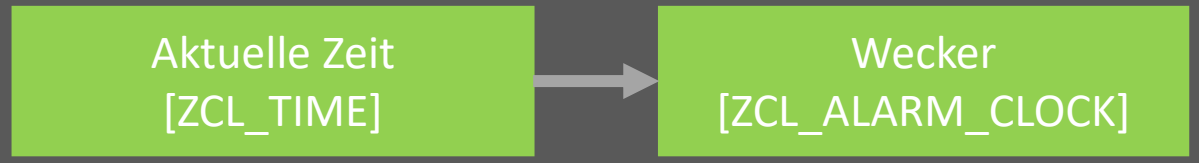


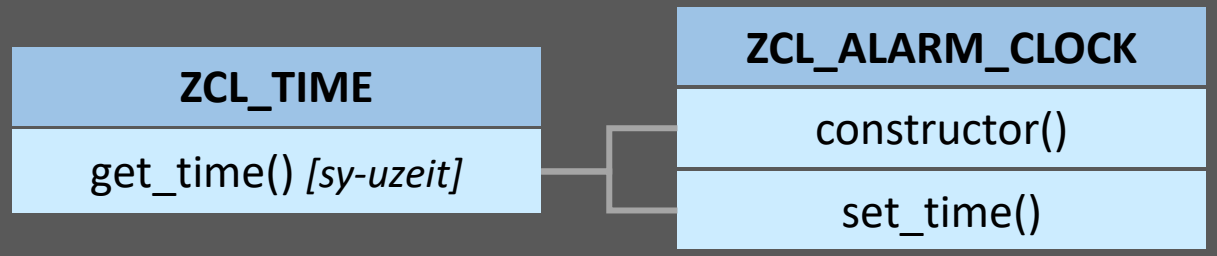
Nicht testbar




```
CLASS zcl_alarm_clock DEFINITION PUBLIC FINAL CREATE PRIVATE.
   PUBLIC SECTION.
      ...
   PRIVATE SECTION.
      DATA mo_time TYPE REF TO zcl_time.
      ...

  METHOD constructor.
    mo_time = new zcl_time( ).
  ENDMETHOD.
ENDCLASS.
```

Testbar






```
METHODS constructor
  IMPORTING
    io_time TYPE REF TO zcl_time.

METHODS set_time
  IMPORTING
    io_time TYPE REF TO zcl_time.
```

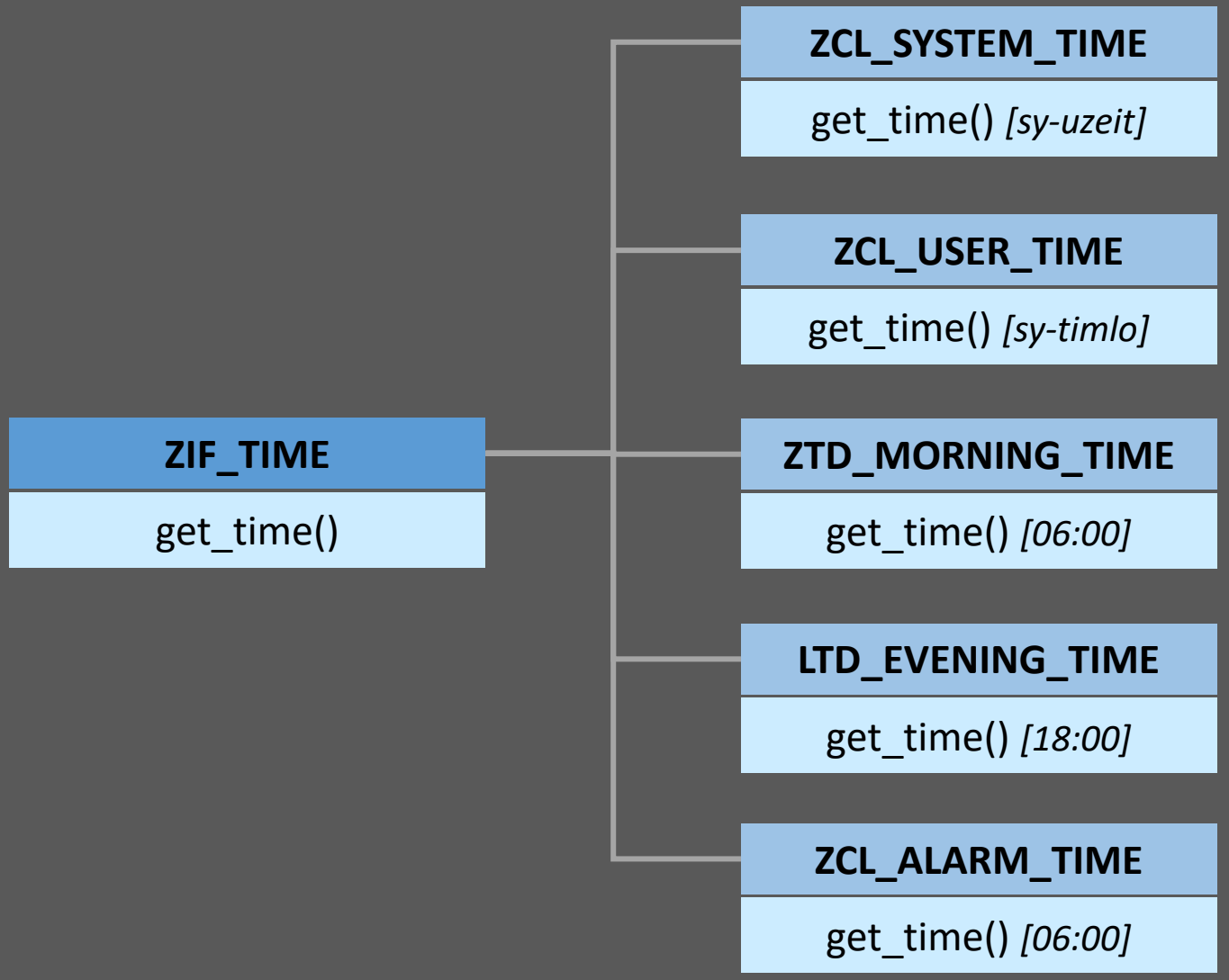
```
METHOD constructor.
  mo_time = io_time.
ENDMETHOD.
```

```
METHOD set_time.
  mo_time = io_time.
ENDMETHOD.
```

Interfaces

Helfen bei der Entkopplung.

Möglichkeiten durch Einsatz eines Interface

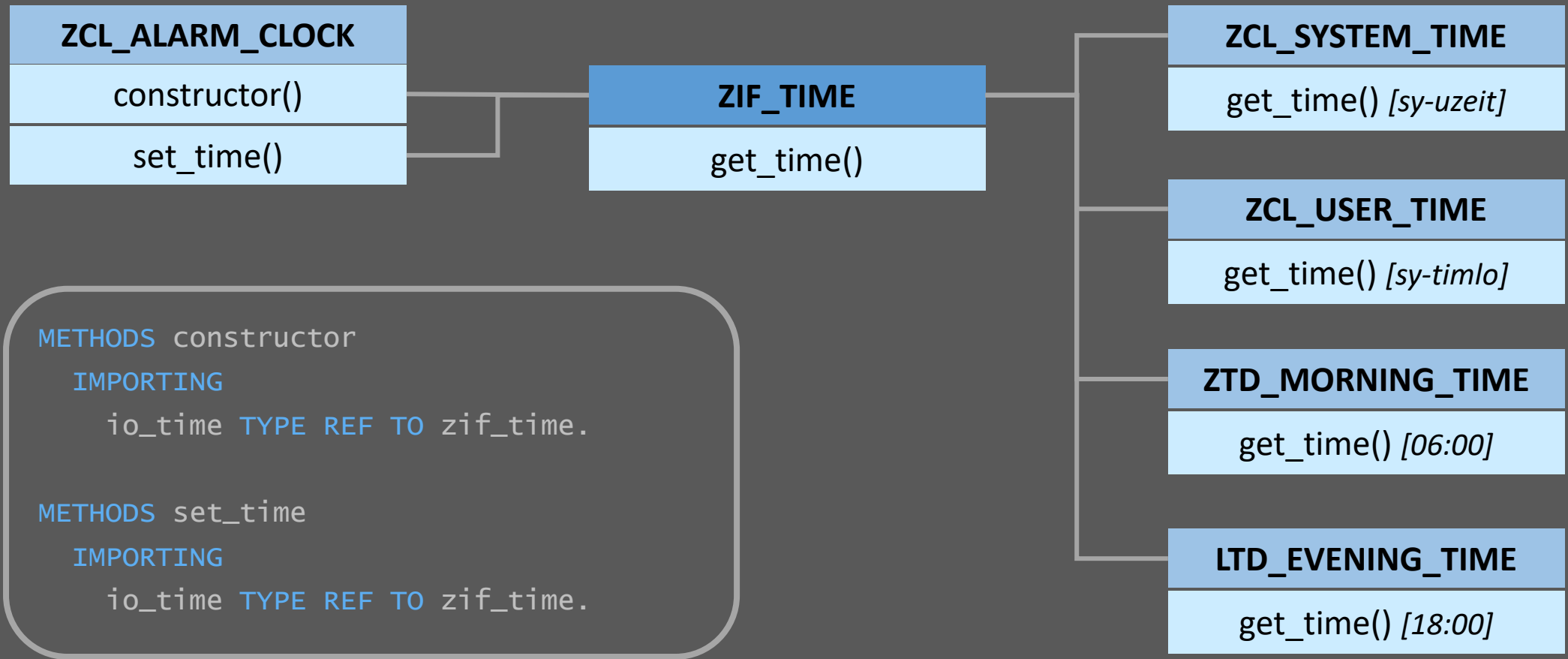


Möglichkeiten durch Einsatz eines Interface

```
INTERFACE zif_times PUBLIC.
  ...
  METHODS get_time RETURNING VALUE(rv_result) TYPE ...
  ...
ENDINTERFACE.
```

```
CLASS zcl_system_time DEFINITION PUBLIC FINAL CREATE PUBLIC.
   PUBLIC SECTION.
      INTERFACES zif_times.
   ...
   METHOD zif_times~get_time.
      rv_result = sy-uzeit.
   ENDMETHOD.
ENDCLASS.
```

Lose Kopplung zwischen Klassen via Interface




```
METHODS constructor
  IMPORTING
    io_time TYPE REF TO zif_time.

METHODS set_time
  IMPORTING
    io_time TYPE REF TO zif_time.
```

ABAP Unit

Testbarkeit nutzen.

Überblick

- öffentliche Methoden von Klassen testen
- automatisiert, wiederholt, massenhaft, schnell
- direkt aus den ABAP Development Tools
- lokale Testklassen mit Testmethoden zu globalen Klassen
- Tests werden also ebenfalls programmiert
- Gestaltung erfolgt nach dem „Given-When-Then“-Prinzip
- starke Bindung zwischen Tests und dem Ziel der Tests entsteht
- Tests unterstützen die Dokumentation

5 Stunden bis zum Wecken – mit Sicherheit

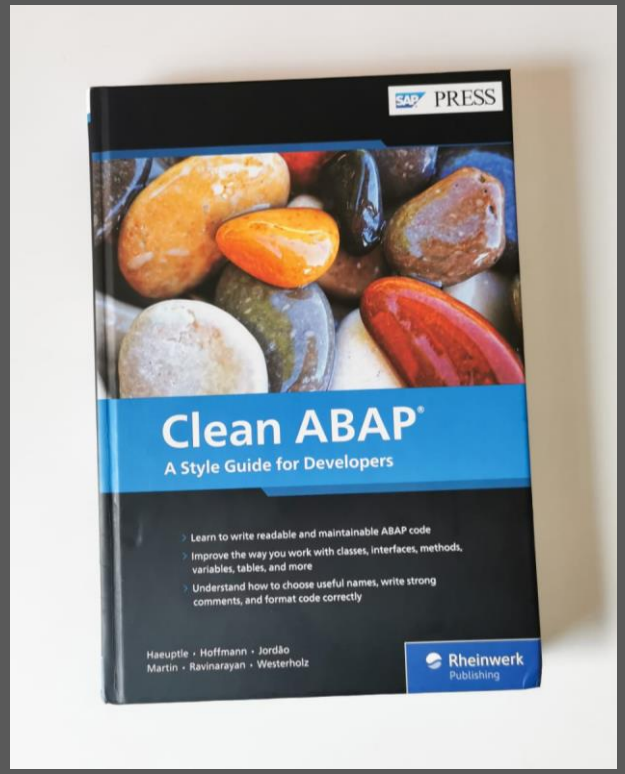
```
CLASS ltc_alarm_clock DEFINITION FOR TESTING...
   PRIVATE SECTION.
      METHODS calculate_1am_to_6am FOR TESTING.
ENDCLASS.
```

```
METHOD calculate_1am_to_6am.
   DATA(lo_td_morning_time_1am) = NEW ztd_morning_time( ).
   DATA(lo_alarm_clock) = NEW zcl_alarm_clock( lo_td_morning_time_1am ).

   DATA(lo_alarm_time) = NEW zcl_alarm_time( iv_alarm_time = '060000' ).
   lo_alarm_clock->set_alarm_time( lo_alarm_time ).

   DATA(lv_time_for_sleep) = lo_alarm_clock->calculate_time_until_alarm( ).

   cl_abap_unit_assert=>assert_equals( act = lv_time_for_sleep
                                       exp = '050000' ).
ENDCLASS.
```

Mit Clean ABAP beginnen

Jeder Tag ist ein guter Tag um damit zu beginnen.

https://github.com/SAP/styleguides/blob/main/clean-abap/CleanABAP.md



Vielen Dank!

Und weiterhin viel Spaß auf der ABAPConf 2021!