


Diese Leseprobe haben Sie beim  
 [edv-buchversand.de](http://edv-buchversand.de) heruntergeladen.  
Das Buch können Sie online in unserem  
Shop bestellen.

[Hier zum Shop](#)

## Auf einen Blick

1	Einführung .....	23
2	Die Programmiersprache ABAP .....	35
3	Klassen und Interfaces .....	59
4	Methoden .....	113
5	Namen .....	155
6	Variablen und Literale .....	169
7	Interne Tabellen .....	197
8	Kontrollfluss .....	211
9	Kommentare .....	235
10	Formatierung .....	247
11	Fehlerbehandlung .....	265
12	Unit Tests .....	293
13	Pakete .....	345
14	Wie Sie Clean ABAP umsetzen .....	369

# Inhalt

Einleitung .....	17
<b>1 Einführung</b> .....	<b>23</b>
<b>1.1 Was ist Clean ABAP?</b> .....	<b>23</b>
1.1.1 Was ist Lesbarkeit? .....	24
1.1.2 Was ist die Geschichte hinter Clean ABAP? .....	25
<b>1.2 Wie kann ich mit Clean ABAP anfangen?</b> .....	<b>26</b>
<b>1.3 Wie gehe ich mit Legacy-Code um?</b> .....	<b>28</b>
<b>1.4 Wie kann ich Code automatisch prüfen?</b> .....	<b>30</b>
<b>1.5 Wie steht Clean ABAP im Verhältnis zu anderen Programmierleitfäden?</b> .....	<b>32</b>
<b>1.6 Wie kann ich mich in der Clean-ABAP-Community engagieren?</b> .....	<b>33</b>
<b>1.7 Zusammenfassung</b> .....	<b>34</b>
<b>2 Die Programmiersprache ABAP</b> .....	<b>35</b>
<b>2.1 Legacy-Code</b> .....	<b>35</b>
<b>2.2 Performance</b> .....	<b>38</b>
2.2.1 Clean Code vs. Performance .....	38
2.2.2 Sauber starten und nur in zwingenden Fällen abweichen .....	39
2.2.3 Messen, statt nur vermuten .....	40
<b>2.3 Objektorientierte vs. prozedurale Programmierung</b> .....	<b>41</b>
2.3.1 Das Programmierparadigma von ABAP .....	41
2.3.2 Der Unterschied zwischen Funktionsgruppen und Klassen .....	42
2.3.3 Besonderheiten der ABAP-Objektorientierung .....	46
2.3.4 Wenn Sie keine Wahl haben .....	48
<b>2.4 Funktionale vs. prozedurale Sprachkonstrukte</b> .....	<b>49</b>
<b>2.5 Obsolete Sprachelemente</b> .....	<b>52</b>

<b>2.6 Entwurfsmuster</b> .....	54
2.6.1 Zu viele Singletons .....	54
2.6.2 Zu viele Muster gemischt .....	56
<b>2.7 Zusammenfassung</b> .....	56
<b>3 Klassen und Interfaces</b> .....	59
<hr/>	
<b>3.1 Objektorientierung</b> .....	59
3.1.1 Interfaces .....	61
3.1.2 Klassen und Objekte .....	67
3.1.3 Zustand .....	88
<b>3.2 Geltungsbereich und Sichtbarkeit</b> .....	90
3.2.1 Globaler und lokaler Geltungsbereich .....	90
3.2.2 Sichtbarkeit .....	95
<b>3.3 Konstruktoren</b> .....	99
3.3.1 Geltungsbereich und Sichtbarkeit .....	99
3.3.2 Dependency Injection .....	101
3.3.3 Statische Erzeugungsmethoden .....	102
3.3.4 Erzeugungsmuster .....	105
3.3.5 Instanziierung .....	109
<b>3.4 Zusammenfassung</b> .....	111
<b>4 Methoden</b> .....	113
<hr/>	
<b>4.1 Objektorientierte Programmierung</b> .....	113
4.1.1 Statische Methoden und Instanzmethoden .....	113
4.1.2 Öffentliche Instanzmethoden .....	117
4.1.3 Redefinition einer Methode .....	119
<b>4.2 Parameter</b> .....	121
4.2.1 Wie viele Parameter sind zu viele? .....	122
4.2.2 Optionale Eingabeparameter .....	123
4.2.3 Bevorzugte Eingabeparameter .....	124
4.2.4 Boolesche Eingabeparameter .....	125
4.2.5 EXPORTING-Parameter .....	127
4.2.6 RETURNING-Parameter .....	129

4.2.7 CHANGING-Parameter .....	131
4.2.8 Übergabe per Wert und Übergabe per Referenz .....	133
<b>4.3 Methodeninhalt</b> .....	137
4.3.1 Eine Sache tun .....	137
4.3.2 Eine Abstraktionsstufe absteigen .....	139
4.3.3 Methoden klein halten .....	142
4.3.4 Früh scheitern .....	144
4.3.5 CHECK oder RETURN .....	146
<b>4.4 Methoden aufrufen</b> .....	148
4.4.1 Eingabeparameter übergeben .....	148
4.4.2 Ausgabeparameter empfangen .....	149
4.4.3 Das Konstrukt CALL METHOD .....	150
4.4.4 Optionale Parameternamen .....	151
4.4.5 Selbstreferenzen .....	152
<b>4.5 Zusammenfassung</b> .....	153
<b>5 Namen</b> .....	155
<hr/>	
<b>5.1 Gute Namen</b> .....	155
5.1.1 Aussagekräftige Namen .....	156
5.1.2 Domänenbegriffe .....	157
5.1.3 Plural oder Singular? .....	158
5.1.4 Abkürzungen .....	158
5.1.5 Klassen und Methoden benennen .....	159
5.1.6 Rauschwörter .....	160
5.1.7 Konsistente Begriffe .....	160
5.1.8 Entwurfsmuster im Code .....	161
5.1.9 Ungarische Notation und andere Präfixe .....	161
<b>5.2 Eigenheiten von ABAP</b> .....	162
5.2.1 Objektgruppen mit Namenskollisionen .....	162
5.2.2 Snake Case oder Camel Case? .....	162
<b>5.3 Affixe: Präfixe, Suffixe und Infixe</b> .....	163
5.3.1 Notwendige Affixe .....	164
5.3.2 Nützliche Affixe .....	165
5.3.3 Unnütze Affixe .....	166
<b>5.4 Mit Legacy-Code umgehen</b> .....	166
<b>5.5 Zusammenfassung</b> .....	167

<b>6 Variablen und Literale</b>	169
<b>6.1 Variablen</b>	170
6.1.1 Variablen deklarieren	170
6.1.2 Kontrollfluss und Gültigkeitsbereich	173
6.1.3 Verkettete Deklarationen	174
6.1.4 Schleifenvariablen	175
<b>6.2 Konstanten</b>	177
6.2.1 Konstanten richtig nutzen	177
6.2.2 Aufzählungsklassen	179
6.2.3 Konstanten gruppieren	184
<b>6.3 Zeichenketten</b>	186
6.3.1 String-Literale	186
6.3.2 Strings zusammenbauen	187
<b>6.4 Boolesche Ausdrücke</b>	188
6.4.1 Wann benutze ich Boolesche Ausdrücke?	188
6.4.2 Boolesche Funktion XSDBOOL für Inline-Deklarationen	190
<b>6.5 Reguläre Ausdrücke</b>	191
6.5.1 Einfache reguläre Ausdrücke	191
6.5.2 Grundlegende Prüfungen	192
6.5.3 Komplexe reguläre Ausdrücke	193
<b>6.6 Das Schlüsselwort REDUCE</b>	193
<b>6.7 Zusammenfassung</b>	196
<b>7 Interne Tabellen</b>	197
<b>7.1 Die richtige Tabellenart verwenden</b>	198
7.1.1 Standardtabellen	198
7.1.2 Sortierte Tabellen	199
7.1.3 Hash-Tabellen	199
<b>7.2 DEFAULT KEY vermeiden</b>	200
<b>7.3 Zeilen hinzufügen mit INSERT INTO TABLE und APPEND TO</b>	201
7.3.1 Die Anweisung APPEND	201
7.3.2 Die Anweisung INSERT	202

<b>7.4 Prüfen, ob eine Tabelle eine bestimmte Zeile enthält</b>	202
7.4.1 Die Anweisung READ TABLE	202
7.4.2 Die Anweisung LOOP AT	203
7.4.3 Die Anweisung LINE_EXISTS	203
<b>7.5 Tabelleninhalte abfragen</b>	204
7.5.1 Die Anweisung LOOP AT	204
7.5.2 Die Anweisung READ TABLE	205
<b>7.6 Die Anweisung LOOP AT ... WHERE ... und verschachtelte IF-Anweisungen</b>	206
7.6.1 Verschachteltes IF	206
7.6.2 Die Anweisung LOOP AT ... WHERE ...	207
<b>7.7 Unnötige Tabellenabfragen identifizieren</b>	207
<b>7.8 Tabellenzeilen blockweise und Zeile für Zeile bearbeiten</b>	208
<b>7.9 DESCRIBE TABLE und die Funktion LINES</b>	209
7.9.1 Die Anweisung DESCRIBE TABLE	209
7.9.2 Die Anweisung LINES	209
<b>7.10 Zusammenfassung</b>	210
<b>8 Kontrollfluss</b>	211
<b>8.1 Das Schlüsselwort IF</b>	212
8.1.1 IF-Ausführungspfade	212
8.1.2 IF-Anweisungen kritisch hinterfragen	215
<b>8.2 Schachtelungstiefe</b>	217
<b>8.3 Bedingungen</b>	218
8.3.1 Bedingungen positiv formulieren	219
8.3.2 IS NOT oder NOT IS	222
8.3.3 Komplexe Bedingungen	223
<b>8.4 Das Schlüsselwort CASE</b>	225
8.4.1 CASE oder IF	225
8.4.2 CASE oder SWITCH	227
8.4.3 Wiederholte CASE- oder SWITCH-Anweisungen	228
<b>8.5 Die Anweisung DO 1 TIMES</b>	229
8.5.1 Pseudoschleifen für den Kontrollfluss	230
8.5.2 Refactoring	231
<b>8.6 Zusammenfassung</b>	233

<b>9</b>	<b>Kommentare</b>	235
9.1	Präziser Code benötigt keine Kommentare	235
9.2	Kommentare richtig platzieren und verwenden	238
9.3	Kommentare, die Sie vermeiden sollten	239
9.4	FIXME-, TODO- und XXX-Kommentare	242
9.5	Spezielle Kommentare: ABAP Doc, Pragmas und Pseudokommentare	244
9.6	Zusammenfassung	245
<b>10</b>	<b>Formatierung</b>	247
10.1	Einen konsistenten Stil verfolgen	248
10.2	Den Code fürs Lesen optimieren	249
10.3	Der Pretty Printer	250
10.4	Wie viele Anweisungen pro Zeile?	252
10.5	Zeilenlänge	252
10.6	Code straffen	254
10.7	Leerzeilen	255
10.8	Zuweisungen ausrichten	256
10.9	Variablendeklarationen ausrichten	257
10.10	Wohin mit den Klammern?	257
10.11	Methodenparameter formatieren	258
10.11.1	Aufrufe mit einem Parameter	258
10.11.2	Zeilenumbrüche bei mehreren Parametern	259
10.11.3	Position der Parameter	259
10.11.4	Parameter einrücken	260
10.11.5	Vertikale Ausrichtung von Parameterwerten	261
10.11.6	Umbrüche und Einrückungen in Methodenaufrufen	261
10.11.7	Inline-Deklarationen einrücken	262
10.12	Zusammenfassung	263

<b>11</b>	<b>Fehlerbehandlung</b>	265
11.1	Nachrichten	265
11.2	Rückgabewerte	269
11.2.1	Ausnahmen oder Rückgabewerte	270
11.2.2	Umgang mit Fehlern	272
11.3	Ausnahmen	274
11.3.1	Ausnahmen für Fehlerfälle	274
11.3.2	Klassenbasierte Ausnahmen	275
11.3.3	Ausnahmen auf Basis der Klasse CX_STATIC_CHECK	280
11.3.4	Ausnahmen auf Basis der Klasse CX_NO_CHECK	281
11.3.5	Ausnahmen auf Basis der Klasse CX_DYNAMIC_EXCEPTION	282
11.4	Ausnahmen auslösen und behandeln	284
11.4.1	Ausnahmen-Oberklassen auswählen	284
11.4.2	Ausnahmen auslösen	286
11.4.3	Ausnahmen behandeln	288
11.4.4	Wann sollte ein Dump ausgelöst werden?	290
11.5	Zusammenfassung	291
<b>12</b>	<b>Unit Tests</b>	293
12.1	Testklassen	294
12.1.1	Eigenschaften von Testklassen	295
12.1.2	Gültigkeitsbereich von Testklassen	298
12.1.3	Testhelferklassen	300
12.1.4	Tests ausführen	301
12.2	Testmethoden	304
12.2.1	Methoden für den Testaufbau	304
12.2.2	Der Given-When-Then-Stil	308
12.3	Die getestete Klasse	309
12.4	Namen von Testklassen und -methoden	310
12.5	Assertions	313
12.5.1	Effiziente Assertions schreiben	314
12.5.2	Erwartete Ausnahmen prüfen	318
12.5.3	Unerwartete statische Ausnahmen	319
12.5.4	Eigene Assertions	321
12.5.5	Constraints	323

<b>12.6 Test-Doubles</b> .....	325
12.6.1 Dependency Inversion und Test-Doubles .....	326
12.6.2 ABAP Test Double Framework .....	331
12.6.3 Weitere Testwerkzeuge .....	335
<b>12.7 Test-Seams</b> .....	337
<b>12.8 Konzepte zum Umgang mit Unit Tests</b> .....	339
12.8.1 Testgetriebene Entwicklung .....	339
12.8.2 Eigenschaften sauberer Tests .....	341
12.8.3 Testabdeckung .....	342
<b>12.9 Zusammenfassung</b> .....	343
<b>13 Pakete</b> .....	345
<hr/>	
<b>13.1 Allgemeine Paketkonzepte</b> .....	345
13.1.1 Anwendungsfälle .....	346
13.1.2 Wiederverwendungsebenen .....	346
13.1.3 Kohäsion .....	347
<b>13.2 Paketkonzept in ABAP</b> .....	347
13.2.1 Pakettypen .....	348
13.2.2 Gekapselte Pakete .....	349
13.2.3 Paketschnittstellen .....	349
13.2.4 Best Practices .....	352
<b>13.3 Optionen für das Paketdesign</b> .....	353
13.3.1 Anwendungsbasierte Hierarchie .....	354
13.3.2 Schichtenbasierte Hierarchie .....	355
13.3.3 Aufteilung der Hierarchien nach Übersetzungsrelevanz .....	356
<b>13.4 Paketprüfungen</b> .....	358
13.4.1 Was sind Paketprüfungen? .....	359
13.4.2 Manuelle Durchführung von Paketprüfungen .....	361
13.4.3 Automatisierte Ausführung von Paketprüfungen .....	362
13.4.4 Behebung von Paketprüfungsfehlern .....	363
13.4.5 Best Practices .....	366
<b>13.5 Konsequenzen einer mangelhaften oder fehlenden Paketstrategie</b> .....	367
<b>13.6 Zusammenfassung</b> .....	368

<b>14 Wie Sie Clean ABAP umsetzen</b> .....	369
<hr/>	
<b>14.1 Gemeinsames Verständnis der Teammitglieder</b> .....	370
14.1.1 Kollektive Code Ownership .....	370
14.1.2 Clean Code Developer Initiative .....	372
<b>14.2 Den Broken-Window-Effekt angehen</b> .....	374
14.2.1 Statische Code-Prüfung .....	376
14.2.2 Metriken .....	377
14.2.3 Code-Abdeckung .....	377
<b>14.3 Code-Reviews und Lernen</b> .....	378
14.3.1 Code-Review-Präfix .....	378
14.3.2 Styleguide .....	378
14.3.3 Sichtbar machen .....	379
14.3.4 Feedback-Kultur .....	379
<b>14.4 Clean Code Advisor</b> .....	382
<b>14.5 Lerntechniken</b> .....	382
14.5.1 Kata .....	383
14.5.2 Dojo .....	384
14.5.3 Code-Retreat .....	384
14.5.4 Fellowship .....	385
14.5.5 Pair Programming .....	385
14.5.6 Mob Programming .....	386
14.5.7 Gewohnheiten .....	386
<b>14.6 Continuous Learning in funktionsübergreifenden Teams</b> .....	387
14.6.1 Profil eines Teammitglieds .....	388
14.6.2 Funktionsübergreifende Teams .....	389
14.6.3 Multiplikatoren im Team .....	389
14.6.4 Community of Practice .....	390
<b>14.7 Zusammenfassung</b> .....	390
Das Autorenteam .....	391
Index .....	393