

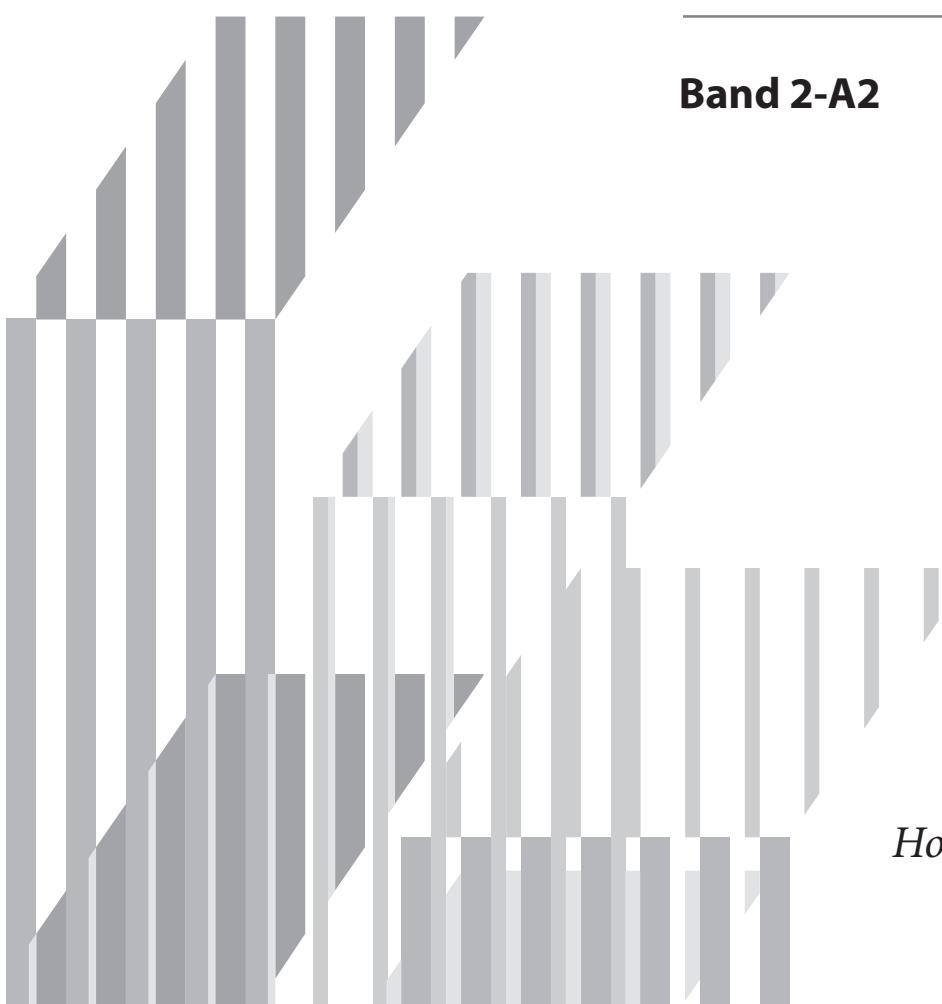
MODERNE STRUKTURIERTE PROGRAMMIERUNG

Objektorientiert und algorithmisch
ENTWERFEN | IMPLEMENTIEREN | TESTEN

Band 2-A2

Ranggruppen
Link-Listen

Praxis



Horst van Bremen

Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;
detaillierte bibliografische Daten sind im Internet über <http://dnb.dnb.de> abrufbar.

© 2025 Dipl.-Ing. Horst van Bremen

Gestaltung	Katharina Schmitt
Titel- und Einbandgrafik	Monika Sylvia Gomm † 1971
Englisch-Korrektorat	David Roseveare
Verlag	HMG®Verlags-GmbH, Neue Strasse 74, 32657 Lemgo Mail-Adresse: office@hmg-verlag.de Website: www.hmg-verlag.de
Druck	Libri Plureos GmbH, Friedensallee 273, 22763 Hamburg
Version / Datum	Version 1.1.3 vom 14.2.2025
Literaturverzeichnis	siehe Kapitel „11.1 Literaturverzeichnis“ auf Seite 411 ff.
Rechtliches	siehe Kapitel „11.2 Rechtliche Hinweise“ auf Seite 413 ff.
ISBN des Paperback-Buchs	978-3-910566-32-3
ISBN der Paperback-Buchreihe	978-3-910566-00-2

Über den Autor



Horst van Bremen
Diplom 1972 an der TU Darmstadt – Elektrotechnik und Informatik
39 Berufsjahre in der IT, davon 18 als Freiberufler

IT-Systemarchitekt
Datenbankexperte

Programmiererfahrung seit 1969
Strukturierte Programmierung nach Jackson seit 1974
Freier Autor seit 2011

Vorwort zur A2-Ausgabe

Profis, die ihre Programme mit JSP entwerfen, betrachten die algorithmischen Implementierungen als mäßig spannende Umsetzungsarbeiten, deren Hauptziel darin besteht, durch geschickte Modularisierung den Code übersichtlich und möglicherweise wiederverwendbar zu machen. Da eine JSP-Programmstruktur mit Operationen keinerlei Vorgaben für den formalen Programmaufbau macht, sind Erfahrung und Geschick für die Qualität des Ergebnisses maßgeblich. Diagramme der Modul-Hierarchien sind dabei hilfreich. Inhaltlich ist zu diesem Zeitpunkt ja alles bereits bei der Modellierung entschieden worden, so dass es über weite Strecken nur noch darum geht, den Entwurf getreulich in Code umzusetzen.

(Wenn Sie, geschätzte Leserinnen und Leser, aus dem vorhergehenden Abschnitt eine gewisse Langeweile herausgelesen haben, so liegen Sie damit richtig. JSP/MSP machen das Leben von Programmier(inne)n leichter, aber auch weniger spannend.)

Zugegeben, ein solches Kaliber wie EvaluateSportsDS in C erfordert schon deutlich mehr Energie als ein Brot-und-Butter-Programm, aber nach Zerlegung der Programmstruktur in ihre Komponenten stand vor allem die höchstmögliche Sorgfalt bei der Implementierung im Vordergrund. Der Einbau hinreichend vieler Trace-Anweisungen, der zunächst als lästiger Zusatzaufwand empfunden wird, garantiert in der Testphase die nötige Transparenz, um die Korrektheit der Programmaktionen verifizieren zu können. Diese Mühe sollte man sich also auf jeden Fall machen, aber dabei auch daran denken, dass dieser Code abschaltbar sein muss.

Ganz anders sieht es bei objektorientierten Implementierungen aus. Der Band 1-A2 demonstriert im Kapitel „13 OO-Vorbereitungen für EvaluateSportsDS“ auf Seite 359 ff, welch weiter Weg von der Programmstruktur mit Operationen zur fertigen Planung der OO-Klassen zurückzulegen ist. Das könnten Leser(innen) durchaus als ein deutliches Manko der hier vorgestellten MSP-Lösungen werten, aber was wäre die Alternative? Gewiss lassen sich diese Lösungswege nicht auf alle möglichen Aufgabenstellungen der objektorientierten Programmierung übertragen, aber für einen guten Teil davon, vor allem den mit Aspekten der Sequentialität, wird hier eine – hoffentlich attraktive – Alternative zum freien Erfindertum vorgelegt.

Die nun folgenden Kapitel zeigen ausführlich, wie die anspruchsvollen Aufgaben, welche der Internationale Schulsportwettbewerb stellt, auf der Basis von JSP/MSP in die Praxis umgesetzt werden können. Für mich, den Autor, war dabei besonders bemerkenswert, wie harmonisch die OO-Implementierungen sich aus dem Entwurf ergaben. Es war nicht nötig, irgendwelche Tricks anzuwenden oder gar, bildlich gesprochen, dem Wolf in den Rachen zu greifen, um das Rotkäppchen wieder herauszuholen, sondern alles ergab sich folgerichtig. Davon profitierte beispielsweise die inkrementelle Implementierung von MailSportResults. Insofern steht zu hoffen, dass das viele Papier ausschließlich mit sinnvollen Anregungen für Leser(innen) aller Qualifikationsstufen oberhalb von A1 bedruckt worden ist...

Lemgo, den 14.2.2025

Übersichtsverzeichnis Band 2-A2

7	EvaluateSportsDS in C	141
8	File Services in Java	205
9	EvaluateSportsDS in Java	247
10	MailSportResults in Java	317
11	Anhang	411



Inhaltsverzeichnis Band 2-A2

7 EvaluateSportsDS in C	141
7.1 Ergänzte Programmstrukturen	141
7.1.1 Top-Programmstruktur	141
7.1.2 Process R-School Group	142
7.1.3 Process Gender Group / Process Age Group	143
7.1.4 Gewinner-Listen-Auswertungen	144
7.1.5 Build Linked List	145
7.1.6 Compress Linked List	146
7.1.7 Add / Amend 1st Value Entries	147
7.1.8 Add Later Entry	148
7.2 Aufruf-Hierarchie	149
7.3 Moment mal!	150
7.3.1 Symbole und Bezeichnungen der JSP-Methode	150
7.3.2 Vergleich zu verwandten Symbolsprachen	150
7.3.3 Michael A. Jackson und seine „Codierknechte“	151
7.3.4 Grafisches	152
7.4 Entstehung der Aufrufhierarchie	152
7.5 C-Code von EvaluateSportsDS	155
7.5.1 Globale Deklarationen	155
7.5.2 main()	162
7.5.3 processCountryGroup()	165
7.5.4 processRSchoolGroup()	167
7.5.5 processGenderGroup()	173
7.5.6 processAgeGroup()	174
7.5.7 readSportsDS()	175
7.5.8 readSchoolDS()	178
7.5.9 buildLinkedList()	180
7.5.10 initializeLinkedList()	182
7.5.11 addPrefix_LL_Entry()	183
7.5.12 addOther_LL_Entry()	185
7.5.13 adjustChainEndIndex()	188
7.5.14 compressLinkedList()	189
7.5.15 displayPupilWinners()	192
7.5.16 completePupilSpoRes()	195
7.5.17 displaySchoolWinners()	196
7.5.18 completeSchoolSpoRes()	199
7.5.19 displayLinkedList()	200
7.5.20 itoa() und substr()	202
8 File Services in Java	205
8.1 Klasse NameAndCount_AC	205
8.2 Klasse BR_RefNameAndCount	206
8.3 Klasse BW_RefNameAndCount	207
8.4 Klasse FS_Code	208
8.5 Klasse Trace	209

Band 2-A2

8.6 Klasse FileRegister	210
8.6.1 initReaders()	212
8.6.2 initWriters()	212
8.6.3 register()	213
8.6.4 getXXX() methods	216
8.6.5 handleResults()	218
8.6.6 inclnRecCount()	219
8.6.7 incOutRecCount()	219
8.6.8 setXXX() methods	219
8.6.9 shutdown()	220
8.7 Klasse InFileUtil	225
8.7.1 open()	226
8.7.2 read()	229
8.7.3 close()	230
8.7.4 checkErrorLimitExceeded()	231
8.7.5 getXXX() methods	231
8.7.6 handleException()	232
8.7.7 setXXX() methods	233
8.7.8 shutdown()	234
8.8 Klasse OutFileUtil	236
8.8.1 open()	236
8.8.2 write()	239
8.8.3 close()	240
8.8.4 checkErrorLimitExceeded()	241
8.8.5 getXXX() methods	241
8.8.6 handleException()	242
8.8.7 setXXX() methods	243
8.8.8 shutdown()	243
9 EvaluateSportsDS in Java	247
9.1 Einleitung	247
9.2 Process Control-Klassen / Hilfsklassen	247
9.2.1 Klasse EvaluateSportsDS	247
9.2.2 Klasse StreamCode	252
9.2.3 Klasse Trace	253
9.2.4 Klasse Const	254
9.2.5 Klasse CountryGrpHandler	257
9.2.6 Klasse R_SchoolGrpHandler	259
9.2.7 Klasse GSA_Entry	268
9.2.8 Klasse GenderGrpHandler	269
9.2.9 Klasse AgeGrpHandler	271
9.2.10 Klasse SpCoRK_Util	273
9.3 Rangschlüsselklassen	278
9.3.1 Klasse PupilRank_1_AC	278
9.3.2 Klasse PupilRank_1	279
9.3.3 Klasse PupilRank_1_2_AC	281
9.3.4 Klasse PupilRank_1_2	283
9.3.5 Klasse PupilRank_1_3_AC	285
9.3.6 Klasse PupilRank_1_3	286
9.3.7 Klasse PupilRank_1_4_AC	287

9.3.8 Klasse PupilRank_1_4	289
9.4 Link-Listen-Klassen	290
9.4.1 Klasse Contestant_AC	290
9.4.2 Klasse Pupil	291
9.4.3 Klasse School	294
9.4.4 Klasse Contest_LL	296
9.5 Restliche Klassen	310
9.5.1 Hilfsklasse BuildStat	310
9.5.2 Klasse Conversion	312
10 MailSportResults in Java	317
10.1 Politisch unkorrekte Vorrede	317
10.2 Erste Schritte	318
10.2.1 Klasse MailSportResults beginnen	318
10.2.2 Klasse StreamCode	321
10.2.3 Klasse Trace	321
10.2.4 Klasse Const	322
10.2.5 Klasse ResultD_RK_Util	326
10.2.6 Klasse ResultD	330
10.2.7 Klasse SpCo_Rank_1_AC	331
10.2.8 Klasse SpCo_Rank_1	332
10.2.9 Klasse SpCo_Rank_1_2_AC	333
10.2.10 Klasse SpCo_Rank_1_2	335
10.2.11 Erster Test	336
10.3 Nationale und internationale Schul-Listen (1)	337
10.3.1 Klasse SchoolResD	340
10.3.2 Klasse S_L_Entry	342
10.4 Restliche einfache Klassen	346
10.4.1 Klasse L_S_T_E	346
10.4.2 Klasse SpoResD	351
10.4.3 Klasse MimeMessageUtil	354
10.4.4 Bestandsaufnahme / weiteres Vorgehen	357
10.5 Modellierungs-Einschub	359
10.5.1 Vorläufige Klassenliste	359
10.5.2 Zuordnung von Klassen zur Programmstruktur	359
10.5.3 Klassendiagramm mit Operationen / Kontrollen	365
10.6 Nationale und internationale Schul-Listen (2)	369
10.6.1 Klasse SchoolList (1)	369
10.6.2 Klasse ResultGrpHandler	371
10.6.3 Zweiter Test	373
10.6.4 Klasse SchoolList (2)	378
10.6.5 Klasse CountryGrpHandler (1)	380
10.6.6 Dritter Test	382
10.7 E-Mails mit Schulergebnissen erzeugen	384
10.7.1 Klasse PupilResultHandler	384
10.7.2 Klasse SchoolGrpHandler (1)	386
10.7.3 Klasse CountryGrpHandler (2)	391
10.7.4 Vierter Test	391
10.8 Fertigstellung	397
10.8.1 Klasse SchoolList (3)	397

Band 2-A2

10.8.2 Klasse SchoolGrpHandler (2)	401
10.8.3 main() ergänzen	402
10.8.4 Fünfter und letzter Test	402
10.9 Ergänzungen des Klassendiagramms	404
10.10 Vorläufiger Abschied vom Schulsportwettbewerb	407
10.11 Manöverkritik	407
11 Anhang	411
11.1 Literaturverzeichnis	411
11.2 Rechtliche Hinweise	413
11.2.1 Geschützte Bezeichnungen	413
11.2.2 Haftungsausschluss	417
11.2.3 Zitate	417
11.2.4 Bild- und Grafiknachweis	418
11.2.5 Programme	418
11.3 Versionsgeschichte zu 1.1.3	418