

Indice generale

Prefazione	XV
Introduzione	xxi
	Ringraziamenti.....	xxiii
Capitolo 1	Codice pulito	1
	Che il codice sia!.....	2
	Cattivo codice.....	3
	Il costo totale di possedere un vero groviglio di codice.....	4
	La grande riprogettazione cosmica.....	4
	Una questione di atteggiamento.....	5
	Il dilemma di base.....	6
	L'arte della pulizia del codice.....	6
	Che cosa si intende con “codice pulito”?.....	7
	Scuole di pensiero.....	13
	Gli autori.....	14
	La regola dei boy-scout.....	15
	Il “prequel” e i principi.....	16
	Conclusioni.....	16
	Bibliografia.....	16
Capitolo 2	Nomi significativi	17
	Introduzione.....	18
	Usate nomi “parlanti”.....	18
	Evitate la disinformazione.....	19
	Adottate distinzioni sensate.....	20
	Usate nomi pronunciabili.....	22
	Usate nomi ricercabili.....	23
	Evitate le codifiche.....	23
	Notazione ungherese.....	24

Prefissi per i membri.....	24
Interfacce e implementazioni	25
Evitate le mappe mentali.....	25
Nomi di classi.....	25
Nomi di metodi.....	26
Non fate i “simpatici”	26
Una parola, un concetto	27
Non siate fuorvianti.....	27
Usate nomi tratti dal dominio della soluzione.....	28
Usate nomi tratti dal dominio del problema	28
Aggiungete un contesto significativo	28
Non aggiungete contesti inesistenti	30
Conclusioni	30

Capitolo 3 **Funzioni33**

Che sia piccola!.....	36
Blocchi e indentazione	37
Che faccia una cosa sola.....	37
Sezioni all’interno delle funzioni.....	38
Un livello di astrazione per funzione	39
Leggete il codice da cima a fondo: la regola dei passi	39
Istruzioni switch	39
Usate nomi descrittivi.....	41
Argomenti di funzione.....	42
Forme monadiche comuni.....	43
Flag usati come argomenti	43
Funzioni diadiche	44
Triadi	44
Oggetti usati come argomenti.....	45
Liste di argomenti.....	45
Verbi e parole chiave.....	45
Niente effetti collaterali.....	46
Argomenti di output.....	47
Separate i comandi dalle richieste.....	47
Scegliete le eccezioni invece di restituire codici di errore.....	48
Estraete i blocchi try/catch	49
La gestione degli errori è “una cosa”	49
Il magnetone per dipendenze Error.java	50
Non ripetetevi (il principio DRY)	50
Programmazione strutturata	51
Come scrivere le funzioni in questo modo?.....	51
Conclusioni	52
SetupTeardownIncluder	52
Bibliografia	54

Capitolo 4	Commenti.....	55
	I commenti non bastano a migliorare il codice cattivo	57
	Spiegatevi nel codice.....	57
	Buoni commenti.....	57
	Commenti legali.....	57
	Commenti informativi.....	58
	Descrizione dell'intento.....	58
	Chiarimenti.....	59
	Avvertenze	60
	Commenti TODO	61
	Amplificazione	61
	Javadoc nelle API pubbliche	62
	Cattivi commenti.....	62
	Pensieri	62
	Commenti ridondanti.....	63
	Commenti fuorvianti.....	65
	Commenti obbligati	65
	Commenti a "log"	66
	Puro "rumore".....	66
	Assoluto "rumore".....	68
	Non usate un commento al posto di una funzione o una variabile	69
	Contrassegni di posizione	69
	Commenti per le parentesi graffe chiuse.....	69
	Attribuzioni.....	70
	Codice commentato	70
	Commenti HTML	71
	Informazioni fuori posizione	72
	Eccesso di informazioni.....	72
	Scarso legame con il codice	73
	Intestazioni di funzioni	73
	Javadoc nel codice non pubblico	73
	Esempio	73
	Bibliografia	77
Capitolo 5	Formattazione	79
	Lo scopo della formattazione	80
	Formattazione verticale.....	80
	La metafora della rivista	81
	Spaziatura verticale fra i concetti	82
	Densità verticale	83
	Distanza verticale.....	84
	Ordinamento verticale.....	89
	Formattazione orizzontale.....	89
	Spaziatura e densità orizzontale.....	90
	Allineamento orizzontale.....	91

	Indentazione.....	92
	Livelli fittizi	94
	Le regole del team	94
	Le regole di formattazione di Uncle Bob	95
Capitolo 6	Oggetti e strutture	99
	Astrazione dei dati	100
	Asimmetria dei dati/oggetti	101
	La Legge di Demetra	103
	Relitti ferroviari	104
	Ibridi.....	105
	Nascondere la struttura	105
	Data Transfer Object	106
	Active Record	107
	Conclusioni	107
	Bibliografia	108
Capitolo 7	Gestione degli errori.....	109
	Usate eccezioni al posto dei codici di return	110
	Scrivere prima l'istruzione try-catch-finally.....	111
	Usate eccezioni non controllate	113
	Fornite un contesto con le eccezioni.....	114
	Definite le classi per le eccezioni in termini di esigenze del chiamante	114
	Definite il flusso "normale".....	116
	Non restituite null	117
	Non passate null	118
	Conclusioni	119
	Bibliografia	119
Capitolo 8	Delimitazioni	121
	Usare codice esterno.....	122
	Esplorazione delle delimitazioni	124
	Imparare a usare log4j	124
	I learning test sono più che gratis.....	126
	Usare codice che non esiste ancora.....	126
	Delimitazioni chiare.....	127
	Bibliografia	128
Capitolo 9	Unit test	129
	Le tre leggi dello sviluppo TDD (Test-Driven Development).....	130
	Curate la pulizia dei test.....	131
	I test garantiscono le "...bilità".....	132
	Test "puliti".....	132

	Un linguaggio per test che sia specifico del dominio	135
	Un doppio standard	135
	Una sola richiesta per test.....	138
	Un unico concetto per ogni test	138
	F.I.R.S.T.	140
	Conclusioni	140
	Bibliografia	141
Capitolo 10	Classi	143
	Organizzazione delle classi	144
	Incapsulazione	144
	Le classi dovrebbero essere piccole!	144
	Il principio SRP (Single Responsibility Principle).....	146
	Coesione	148
	Curando la coesione si generano tante piccole classi.....	149
	Organizzare gli interventi di modifica	154
	Isolamento dalle modifiche	157
	Bibliografia	158
Capitolo 11	Sistemi	159
	Come edificereste una città?.....	160
	Separate la realizzazione dall'uso di un sistema.....	160
	Separazione di main.....	161
	Factory.....	162
	Dependency Injection	163
	Estensione di scala.....	163
	Confusione di ambiti	166
	Proxy Java	167
	Framework AOP puri Java	168
	AspectJ	172
	Sottoporre a test l'architettura del sistema.....	172
	Ottimizzazione delle decisioni.....	173
	Usate gli standard con cura, solo se dimostrano il proprio valore	173
	I sistemi richiedono l'impiego di linguaggi specifici del dominio	174
	Conclusioni	174
	Bibliografia	174
Capitolo 12	Simple Design	177
	Come far emergere un codice pulito.....	178
	Regola 1 di Simple Design – Passa tutti i test	178
	Regole 2-4 di Simple Design – Refactoring	178
	Niente duplicazione.....	179
	Espressività.....	181

Minimizzare classi e metodi	182
Conclusioni	183
Bibliografia	183

Capitolo 13 Concorrenza185

A che cosa serve la concorrenza?	186
Miti e fraintendimenti	187
Sfide	188
Concorrenza: principi di difesa.....	188
Il principio SRP (Single Responsibility Principle).....	188
Corollario: limitate il livello di visibilità (scope) dei dati.....	189
Corollario: usate copie dei dati.....	189
Corollario: i thread dovrebbero essere il più possibile indipendenti	190
Studiate la vostra libreria	190
Collection thread-safe	190
Studiate i modelli di esecuzione	191
Produttori-Consumatori.....	191
Lettori-Scrittori.....	191
La cena dei filosofi.....	192
Attenzione alle dipendenze fra metodi sincronizzati.....	192
Riducete al minimo le sezioni sincronizzate	193
Scrivere il codice di chiusura corretto è difficile	193
Collaudo del codice di un thread	194
Trattate i fallimenti occasionali come possibili problemi dei thread	194
Curate innanzitutto il funzionamento del codice mono-thread	195
Rendete versatile il codice a thread.....	195
Rendete configurabile il codice a thread	195
Provate a generare più thread delle CPU disponibili.....	195
Eseguite i test su piattaforme differenti.....	196
Mettete alla prova il vostro codice per provare a forzare i fallimenti.....	196
Test manuali	196
Test automatici	197
Conclusioni	198
Bibliografia	199

Capitolo 14 Raffinamento progressivo201

Implementazione di Args	202
Come ci sono arrivato?.....	208
Args: “la brutta”.....	209
E così mi sono fermato qui.....	221
L’incrementalismo.....	221

	Argomenti String	223
	Conclusioni	262
Capitolo 15	JUnit.....	263
	Il framework JUnit	264
	Conclusioni	277
Capitolo 16	Refactoring di SerialDate	279
	Innanzitutto, facciamola funzionare	280
	Ora raffiniamola.....	282
	Conclusioni	295
	Bibliografia	295
Capitolo 17	Avvertenze ed euristiche	297
	Commenti.....	298
	C1: Informazioni inappropriate.....	298
	C2: Commenti obsoleti	298
	C3: Commenti ridondanti	298
	C4: Commenti mal scritti	299
	C5: Codice in commento	299
	Ambiente.....	299
	E1: Build che richiedono più di un passo	299
	E2: Test che richiedono più di un passo	300
	Funzioni.....	300
	F1: Troppi argomenti	300
	F2: Argomenti di output	300
	F3: Flag usati come argomenti.....	300
	F4: Funzioni “morte”.....	300
	Generali.....	300
	G1: Più linguaggi in un file di codice sorgente	300
	G2: Non viene implementato un comportamento naturale.....	301
	G3: Comportamento errato alle delimitazioni	301
	G4: Disattivazione delle “sicure”	301
	G5: Duplicazioni	302
	G6: Codice posto al livello di astrazione errato	302
	G7: Classi base che dipendono dalle loro derivate.....	303
	G8: Eccesso di informazioni.....	304
	G9: Il “dead code”	304
	G10: Separazione verticale	304
	G11: Incoerenza	304
	G12: Congestione.....	305
	G13: Accoppiamento artificioso	305
	G14: Una questione di invidia (“feature envy”)	305
	G15: Argomenti “a selettore”.....	306
	G16: Offuscamento dello scopo	307

G17: Responsabilità mal collocate	307
G18: Modificatore static inappropriato	308
G19: Usate variabili descrittive	308
G20: Il nome di una funzione deve essere comunicativo	309
G21: Comprendere l'algoritmo	309
G22: Rendere fisiche le dipendenze logiche	310
G23: Meglio il polimorfismo di un if/else o di uno switch/case	311
G24: Seguite convenzioni standard	311
G25: Al posto dei "numeri magici", usate costanti "parlanti" ...	312
G26: Siate precisi	313
G27: Il principio "structure over convention"	313
G28: Incapsulate i costrutti condizionali	313
G29: Evitate i negativi nei costrutti condizionali	314
G30: Le funzioni dovrebbero fare una cosa sola	314
G31: Accoppiamenti temporali nascosti	315
G32: Non siate arbitrari	315
G33: Incapsulate le condizioni di delimitazione	316
G34: Le funzioni devono discendere un solo livello di astrazione	316
G35: Mantenete ai livelli più elevati i dati configurabili	318
G36: Evitate la navigazione transitiva	319
Java	319
J1: Evitate i lunghi elenchi di import; usate i caratteri jolly	319
J2: Non ereditate le costanti	320
J3: Costanti o enum?	321
Nomi	322
N1: Scegliete nomi descrittivi	322
N2: Scegliete nomi collocati al corretto livello di astrazione ...	323
N3: Usate la nomenclatura standard, se possibile	324
N4: Nomi non ambigui	324
N5: Usate nomi lunghi per grandi campi di visibilità (scope) ..	325
N6: Evitate le codifiche	325
N7: I nomi dovrebbero descrivere anche gli effetti collaterali ..	325
Test	326
T1: Test insufficienti	326
T2: Usate uno strumento di copertura!	326
T3: Non saltate i test che considerate banali	326
T4: Un test ignorato è un problema di ambiguità	326
T5: Verificate le condizioni di delimitazione	327
T6: Applicate test esaustivi in prossimità dei bug	327
T7: Considerate quando i problemi sembrano far pensare a uno schema	327
T8: I risultati dei test di copertura possono essere rivelatori ...	327
T9: I test dovrebbero essere veloci	327
Conclusioni	327
Bibliografia	328

Appendice A	Concorrenza II	329
	Un esempio client/server	329
	Il server	329
	Aggiunta del threading	331
	Osservazioni sul server.....	331
	Conclusioni.....	333
	Possibili percorsi di esecuzione.....	333
	Numero di percorsi	334
	Approfondimento	335
	Conclusioni.....	338
	Studiare la libreria.....	338
	Il framework Executor.....	338
	Soluzioni senza bloccaggi.....	339
	Classi problematiche quanto ai thread	340
	Le dipendenze fra i metodi possono pregiudicare il funzionamento del codice concorrente	341
	Tollerare i problemi	342
	Bloccaggio basato sul client.....	342
	Bloccaggio basato sul server	344
	Migliorare la produttività (throughput).....	345
	Calcolo del throughput mono-thread.....	346
	Calcolo del throughput multi-thread.....	347
	Deadlock	347
	Esclusione reciproca (mutual exclusion)	348
	Blocco e attesa (lock & wait).....	348
	Nessuna prelazione (no preemption)	348
	Attesa circolare (circular wait)	349
	Risolvere l'esclusione reciproca.....	349
	Risolvere il lock & wait.....	349
	Risolvere il problema del no preemption	350
	Risolvere il problema circular wait.....	350
	Collaudo del codice multi-thread.....	351
	Strumenti per il test del codice multi-thread.....	353
	Conclusioni	354
	Tutorial: esempi di codice	354
	Client/server senza threading.....	354
	Client/server con threading	358
Appendice B	org.jfree.date.SerialDate	359
Appendice C	Indice delle euristiche	423
Epilogo	425
Indice analitico	427