

Tartalomjegyzék

1. fejezet • Bevezetés

2. fejezet • Objektumok létrehozása és megszüntetése

1. Konstruktor helyett használhatunk gyártófüggvényt 5
2. Sokparaméteres konstruktor esetén használjunk építőt 10
3. Alakítsuk az osztályt egykévé (singleton) privát konstruktorral
vagy felsorolás típussal 16
4. Példányosíthatatlanság privát konstruktorral 18
5. Ne gyártsunk felesleges objektumokat 19
6. Szüntessük meg az elavult objektumhivatkozásokat 22
7. Ne használjunk felszámolókat (finalizers) 25

3. fejezet • Általános objektum metódusok

8. Az equals felülírása az általános szerződés szerint 31
9. Az equals felülírásakor a hashCode-ot is írjuk felül 41
10. Minden esetben írjuk át a toString metódust 46
11. A clone körültekintő felülírása 48
12. Fontoljuk meg a Comparable implementálását 55

4. fejezet • Osztályok és interfészek

13. Minimalizáljuk az osztályok és tagjaik elérhetőségét 61
14. Publikus osztályokban a publikus mezők helyett használjunk
hozzáférő metódusokat 64
15. Csökkentsük minimálisra a változtathatóságot 66
16. A kompozíció jobb az öröklésnél 73
17. Tervezzük és dokumentáljuk az öröklést, vagy tiltsuk meg 78
18. Elvont osztályokkal szemben részesítsük előnyben az interfészeket 82
19. Interfészeket csak típusok definiálásához használjunk 86
20. A címkézett osztályokkal szemben preferáljuk az osztályhierarchiákat 88
21. A stratégiák ábrázolásához használjunk függvény objektumokat 91
22. A statikus tagosztályokat jobban kedveljük a nem statikusaknál 93

5. fejezet • Általános típusok

23. Új kódban ne használjunk nyers típusokat	97
24. Szüntessük meg az ellenőrizetlenséggel kapcsolatos figyelmeztetéseket	103
25. Használjunk listákat a tömbök helyett	105
26. Támogassuk az általános típusokat	110
27. Támogassuk az általános metódusokat	115
28. Használjunk kötött helyettesítő karaktereket az API rugalmasabbá tételéhez	119
29. Fontoljuk meg a típusbiztos heterogén tárolók használatát	126

6. fejezet • Felsorolások és annotációk

30. Használjunk felsorolt típust az int konstansok helyett	133
31. Használjunk példánymezőket a sorszámfüggvények helyett	143
32. Használjunk EnumSet-et a bitmezők helyett	144
33. Használjunk EnumMap-et a sorszámfüggvény helyett	146
34. A bővíthető felsorolásokat valósítsuk meg felületekkel	150
35. Az elnevezési minták helyett használjunk annotációkat	153
36. Használjuk következetesen az Override annotációt	159
37. Használjunk jelölőfelületeket a típusdefiniáláshoz	161

7. fejezet • Metódusok

38. Ellenőrizzük a paraméterek érvényességét	165
39. Készítsünk biztonsági másolatokat, ha szükséges	167
40. Tervezzük meg gondosan a metódusok paraméter-szignatúráját	172
41. Csak meggondoltan használjuk a túlterhelést	174
42. Megfontoltan használjuk a varargs lehetőségeit	179
43. Adjunk vissza üres tömböket vagy kollekciókat, ne null-t	183
44. Írjunk doc megjegyzéseket minden fontos API elemhez	185

8. fejezet • Általános programozás

45. Minimalizáljuk a helyi változók hatókörét	191
46. Használjunk for-each ciklust a hagyományos for ciklus helyett	194
47. Ismerjük meg és használjuk a programkönyvtárakat	196
48. Kerüljük a float és a double típusok használatát, ha pontos értékekre van szükségünk	199
49. Használjunk primitív típusokat a becsomagolt primitívek helyett	201
50. Kerüljük a karakterláncok használatát, ha van más, megfelelőbb típus	204
51. Vigyázzunk a karakterláncok összefűzésével együtt járó teljesítmény-csökkenésre	206
52. Hivatkozzunk az objektumokra a saját felületükkel	208
53. Használjunk felületeket a többszöröző (reflection) helyett	209
54. Használjuk bölcsen a natív metódusokat	212
55. Optimalizáljunk meggondoltan	213
56. Ragaszkodjunk az általánosan elfogadott elnevezési konvenciókhoz	216

9. fejezet • Kivételek

57. A kivételeket csak valóban kivételes helyzetekben használjuk	221
58. Használjunk ellenőrzött kivételeket a helyrehozható körülményekhez és futásidejű kivételeket a programozási hibákhoz	224
59. Kerüljük az ellenőrzött kivételek szükségtelen használatát	225
60. Ha csak lehet, használjuk a szabványos kivételeket	227
61. Az absztrakciónak megfelelő kivételt válasszunk	229
62. Minden metódus minden kivételét dokumentáljuk	231
63. Írjunk olyan információkat a részletes üzenetekbe, amelyek jól megragadják a hiba lényegét	233
64. Törekedjünk az elemi hibaszint megőrzésére	235
65. Ne hagyjunk figyelmen kívül kivételeket	237

10. fejezet • Konkurens feldolgozás

66. Szinkronizáljuk a megosztott, változtatható adatok elérését	239
67. Kerüljük a túlzott szinkronizációt	244
68. Használjunk végrehajtókat (executors) és feladatokat (tasks) a szálak helyett	249
69. Használjunk párhuzamosító segédeszközöket a wait és a notify helyett	251
70. Dokumentáljuk a szállbiztonságot	256
71. Megfontoltan használjuk a lusta inicializációt	259
72. Ne alapozzuk programunk működését a szálütemezőre	262
73. Kerüljük a szálcsoportokat	264

11. fejezet • Szerializálás

74. Megfontoltan végezzük a Serializable felület megvalósítását.	265
75. Fontoljuk meg az egyedi serializált forma használatát	270
76. Írjunk biztonságos readObject metódusokat	277
77. A példányvezérlést lehetőleg felsorolt típusal oldjuk meg, ne a readResolve-val	282
78. Fontoljuk meg a serializáló proxy használatát a serializált példányok helyett	286

Függelék

Az első és második kiadás egymásnak megfelelő alfejezetei	291
Irodalmi hivatkozások	293

Tárgymutató.	297
----------------------	-----