POGLAVLJE 1

Uvod

OVA knjiga je osmišljena da vam pomogne u efiikasnom korišćenju programskog jezika Java i njegovih osnovnih biblioteka: `java.lang`, `java.util` i `java.io`, i potpaketa kao što su `java.util.concurrent` i `java.util.function`. Povremeno se objašnjavaju i druge biblioteke.

Knjiga se sastoji od devedeset tema, a svaka prenosi jedno pravilo. Pravila odražavaju prakse koje najbolji i najiskusniji programeri smatraju korisnim. Teme su neobavezno grupisane u jedanaest poglavlja, a svako se bavi nekim širim aspektom dizajna softvera. Knjiga nije osmišljena da se čita redom, od početka do kraja: svaka tema je manje-više samostalna. Teme se međusobno referenciraju, pa možete lako sami da trasirate svoj put kroz knjigu.

Mnoge nove mogućnosti dodate su platformi od objavljivanja prethodnog izdanja ove κnjige. Većina tema u ovoj knjizi koristi te mogućnosti na neki način. Sledeća tabela vam pokazuje gde ćete pronaći osnovno pominjanje najbitnijih karakteristika:

Karakteristika	**Teme**	**Verzija**
Lambde	Teme 42–44	Java 8
Tokovi	Teme 45–48	Java 8
Klase Optional	Tema 55	Java 8
Standardne metode u interfejsima	Tema 21	Java 8
`try`-sa-izvorima	Tema 9	Java 7
`@SafeVarargs`	Tema 32	Java 7
Moduli	Tema 15	Java 9

Većina tema je ilustrovana primerima programa. Glavna osobina ove knjige jeste to što sadrži primere koda koji dočaravaju mnoge projektne obrasce i idiome. Na mestima gde je to odgovarajuće, unakrsno su referencirani sa standardnim referentnim radom u ovoj oblasti [Gamma95].

Mnoge teme sadrže jedan ili više primera programa koji prikazuju neku praksu koju treba izbegavati. Takvi primeri, poznati i kao *antiobrasci,* jasno su označeni komentarom kao što je `// Nikada ne radite ovo!`. U svakom slučaju, tema objašnjava zbog čega je primer loš i predlaže drugačiji pristup.

Ova knjiga nije za početnike: ona pretpostavlja da ste već opušteni u radu sa Javom. Ukoliko niste, razmislite o nekom od mnogo dobrih uvodnih tekstova, kao što je *Java Precisely* [Sestoft16] Petera Sestofta. Iako je knjiga *Java efikasno* osmišljena da bude pristupačna svakome ko ima upotrebljivo poznavanje jezika, trebalo bi da podstakne na razmišljanje čak i napredne programere.

Većina pravila u ovoj knjizi izvedena je iz nekoliko osnovnih principa. Jasnoća i jednostavnost su od najveće važnosti. Korisnik komponente nikada ne treba da bude iznenađen njenim ponašanjem. Komponente bi trebalo da budu najmanje moguće, ali ne manje od toga. (U ovoj knjizi se pojam *komponenta* odnosi na bilo koji softverski element koji se može iznova koristiti, od pojedinačnih metoda, do složenih radnih okruženja koja se sastoje od više paketa.) Kod treba da bude višekratan, a ne da se kopira. Zavisnosti između komponenata treba da budu minimalne. Greške treba otkrivati što je pre moguće nakon što su napravljene, idealno bi bilo u trenutku kompajliranja.

Premda pravila iz ove knjige ne važe u sto posto situacija, ona karakterišu najbolje prakse u programiranju u najvećoj većini slučajeva. Ne treba da im robujete, ali ih kršite samo povremeno i sa dobrim razlogom. Ovladavanje umetnošću programiranja, kao i većinom drugih disciplina, sastoji se prvo od učenja pravila, a potom od učenja kada ta pravila treba prekršiti.

Ova knjiga većim delom ne govori o performansama. Ona se bavi pisanjem programa koji su jasni, ispravni, upotrebljivi, izdržljivi, fleksibilni i održivi. Ako uspete u tome, obično je relativno jednostavno postići performanse koje su vam potrebne (tema 67). Neke teme govore o problemima sa performansama, a neke od tih tema iznose i brojeve. Te brojeve, pre kojih stoji fraza „Na mom računaru”, trebalo bi smatrati u najboljem slučaju približnim.

Ako vam to išta vredi, moj računar je kućne izrade, ostareli Intel Core i7-4770K 3.5GHz sa četiri jezgra, 16 gigabajta radne memorije DDR3-1866 CL9, koristim verziju OpenJDK-a Azulov Zulu 9.0.0.15, u Microsoft Windowsu 7 Professional SP1 (64-bitni).

Kada se govori o osobinama programskog jezika Java i njegovih biblioteka, ponekad je potrebno navesti konkretnu verziju. Radi praktičnosti, ova knjiga koristi skraćene nazive umesto zvaničnih naziva verzija. Naredna tabela prikazuje nazive verzija i njihove skraćene nazive:

Zvaničan naziv verzije	Skraćeni naziv
JDK 1.0.*x*	Java 1.0
JDK 1.1.*x*	Java 1.1
Java 2 Platform, Standard Edition, v1.2	Java 2
Java 2 Platform, Standard Edition, v1.3	Java 3
Java 2 Platform, Standard Edition, v1.4	Java 4
Java 2 Platform, Standard Edition, v5.0	Java 5
Java Platform, Standard Edition 6	Java 6
Java Platform, Standard Edition 7	Java 7
Java Platform, Standard Edition 8	Java 8
Java Platform, Standard Edition 9	Java 9

Primeri su prilično kompletni, ali prednost je data njihovoj čitljivosti, a ne dovršenosti. U njima se koriste klase iz paketa `java.util` i `java.io`. Da biste kompajlirali primere, možda ćete morati da dodate jednu ili više bitnih deklaracija ili neki sličan šablonski kod. Veb lokacija ove knjige, `http://joshbloch.com/effectivejava`, sadrži proširene verzije svih primera, i njih možete kompajlirati i izvršiti.

Knjiga većim delom koristi tehničke pojmove kako su definisani u specifikaciji jezika Java, *The Java Language Specification, Java SE 8 Edition* [JLS]. Nekoliko pojmova treba posebno pomenuti. Jezik podržava četiri tipa podataka: *interfejse* (uključujući *anotacije*), *klase* (uključujući *nabrojive tipove*), *nizove* i *osnovne tipove*. Prva tri su poznata kao *referentni tipovi* (eng. *reference types*). Instance klasa i nizovi su *objekti*; osnovne vrednosti nisu. *Članovi* klase su njena *polja*, *metode*, *klase članice* i *interfejsi članovi*. *Potpis* metode čine njeno ime i tipovi njenih formalnih parametara; potpis *ne* uključuje tip rezultata metode.

U ovoj knjizi nekoliko pojmova se koristi drugačije nego u specifikaciji jezika Java. Za razliku od specifikacije, knjiga koristi *nasleđivanje* (eng. *inheritance*) kao sinonim za *pravljenje potklasa* (eng. *subclassing*). Umesto da se pojam nasleđivanje koristi za interfejse, ova knjiga prosto govori da klasa *implementira* interfejs ili da jedan interfejs *proširuje* drugi. Da bi se opisao nivo pristupa koji važi kada nijedan nije zadat, knjiga koristi tradicionalno *privatno u paketu* (eng. *package-private*) umesto tehnički ispravnog *pristup za paket* (eng. *package access*) [JLS, 6.6.1].

Ova knjiga koristi nekoliko tehničkih pojmova koji nisu definisani u specifikaciji jezika Java. Pojam *izvezeni API*, ili prosto *API*, odnosi se na klase, interfejse,

konstruktore, članove i serijalizovane obrasce kojima programer pristupa klasi, interfejsu ili paketu. (Pojam *API*, skraćenica za *application programming interface* – intefejs za programiranje aplikacija, koristi se umesto pojma *interfejs*, koji bi inače trebalo da ima prednost, da bi se izbeglo mešanje sa jezičkom konstrukcijom tog imena.) Programer koji piše program koji koristi API naziva se *korisnikom* API-ja. Klasa čija implementacija koristi API jeste *klijent* API-ja.

Klase, interfejsi, konstruktori, članovi i serijalizovani obrasci nazivaju se zajedničkim imenom *elementi API-ja*. Izvezeni API se sastoji od elemenata API-ja kojima se može pristupiti izvan paketa koji definiše API. To su elementi API-ja koje bilo koji klijent može koristiti i za koje autor API-ja obezbeđuje podršku. Nije slučajnost da su to elementi za koje i alat Javadoc generiše dokumentaciju u svom standardnom režimu rada. Slobodno rečeno, izvezeni API nekog paketa sastoji se od javnih i zaštićenih članova i konstruktora svake javne klase ili interfejsa u tom paketu.

U Javi 9, *sistem modula* je dodat platformi. Ako biblioteka koristi sistem modula, njen izvezeni API je unija izvezenih API-ja svih paketa koji se izvoze deklaracijom modula biblioteke.