Poglavlja

Uvod xxi
1 Početak: *Da se bućnemo* 1
2 Osnovni tipovi i promenljive: *Budite promenljivi* 31
3 Funkcije: *Izlazak iz funkcije main* 59
4 Klase i objekti: *Ponešto o klasama* 91
5 Potklase i natklase: *Korišćenje nasleđivanja* 121
6 Apstraktne klase i interfejsi: *Ozbiljan polimorfizam* 155
7 Klase za podatke: *Rad sa podacima* 191
8 Null i izuzeci: *Bezbedni i sigurni* 219
9 Kolekcije: *Organizujte se* 251
10 Generički tipovi: *Znajte šta je in, a šta je out* 289
11 Lambde i funkcije višeg reda: *Tretiranje koda kao podataka* 325
12 Ugrađene funkcije višeg reda: *Osnažite svoj kod* 363
i Korutine: *Paralelno izvršavanje koda* 397
ii Testiranje: *Neka vaš kod bude odgovoran* 409
iii Ostaci: *Najbitnijih deset stvari (koje nismo obradili)* 415

Sadržaj

Uvod

Vaš mozak o Kotlinu. I evo *vas*, pokušavate da nešto *naučite*, a vaš *mozak* vam čini uslugu starajući se da vam ništa od naučenog ne *ostane u glavi*. Vaš mozak misli: „Bolje da ostavim mesta za bitnije stvari, kao što su divlje životinje koje treba izbegavati i procenu da li je snoubording bez odeće loša zamisao." Pa kako ćete prevariti svoj mozak da misli da vam život zavisi od ovladavanja programiranjem u Kotlinu?

Kome je knjiga namenjena? xxii
Znamo šta mislite xxiii
Znamo šta vaš *mozak* misli xxiii
Metakognicija: razmišljanje o razmišljanju xxv
Evo šta smo MI uradili: xxvi
Pročitaj me xxviii
Recenzentski tim xxx
Zahvalnice xxxi

početak

Da se bućnemo

1

Kotlin podiže talase.

Od svog prvog izdanja, Kotlin je zadivio programere svojom ***pristupačnom sintaksom, sažetošću, fleksibilnošću i snagom***. U ovoj knjizi, naučićemo vas kako da **gradite aplikacije u Kotlinu**, a počećemo tako što ćete napraviti jednostavnu aplikaciju i pokrenuti je. Usput ćete upoznati deo osnovne Kotlinove sintakse, kao što su *naredbe, petlje* i *uslovno grananje*. Vaše putovanje je upravo počelo...

Mogućnost da birate za koju ćete platformu kompajlirati kôd znači da Kotlinov kôd može da se pokrene na serverima, u oblaku, u čitačima veba, na mobilnim uređajima itd.

Dobrodošli u Kotlingrad 2

Kotlin možete koristiti gotovo svuda 3

Šta ćemo raditi u ovom poglavlju 4

Instalirajte IntelliJ IDEA (Community Edition) 7

Napravimo osnovnu aplikaciju 8

Upravo ste napravili svoj prvi Kotlinov projekat 11

Dodajte projektu novu Kotlinovu datoteku 12

Anatomija funkcije main 13

Dodajte funkciju main datoteci App.kt 14

Probna vožnja 15

Šta možete da kažete u funkciji main? 16

Prođi opet i opet i opet... 17

Upetljani primer 18

Uslovno grananje 19

Korišćenje if za vraćanje vrednosti 20

Ažurirajte funkciju main 21

Korišćenje Kotlinovog interaktivnog komandnog okruženja 23

U REPL možete uneti višeredne odlomke koda 24

Kutija sa Kotlinovim alatima 30

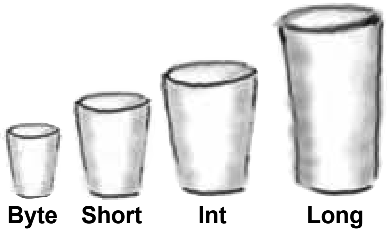
osnovni tipovi i promenljive

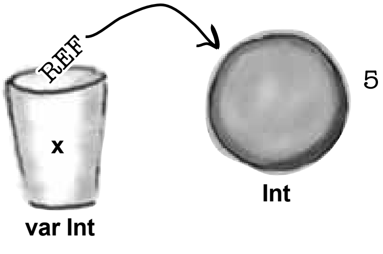
2

Budite promenljivi

Sav kôd zavisi od jedne stvari – promenljivih.

U ovom poglavlju ćemo zaviriti pod haubu i pokazati ***kako funkcionišu Kotlinove promenljive***. Otkrićete Kotlinove **osnovne tipove**, kao što su *Ints*, *Floats* i *Booleans*, i naučiti kako Kotlinov kompajler može **inteligentno da izvede tip promenljive iz vrednosti koja joj je data**. Naučićete kako se koriste **šabloni znakovnih nizova** za konstruisanje složenih znakovnih nizova pomoću vrlo malo koda, i kako se prave **nizovi** sa više vrednosti. Najzad, otkrićete *zašto su objekti toliko bitni za život u Kotlingradu*.





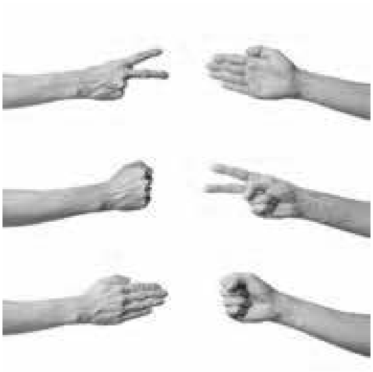
Kodu su potrebne promenljive 32
Šta se dešava kada deklarišete promenljivu 33
Promenljiva čuva referencu do objekta 34
Kotlinovi osnovni tipovi 35
Kako se izričito deklariše tip promenljive 37
Koristite odgovarajuću vrednost za tip promenljive 38
Dodeljivanje vrednosti drugoj promenljivoj 39
Moramo da konvertujemo vrednost 40
Šta se dešava kada konvertujete vrednost 41
Čuvajte se prelivanja 42
Čuvanje više vrednosti u nizu 45
Napravite aplikaciju Phrase-O-Matic 46
Dodajte kôd datoteci PhraseOMatic.kt 47
Kompajler izvodi tip niza iz njegovih vrednosti 49
var znači da promenljiva može da pokazuje drugi niz 50
val znači da promenljiva pokazuje isti niz zauvek... 51
Pomešane reference 54
Kutija sa Kotlinovim alatima 58

Funkcije

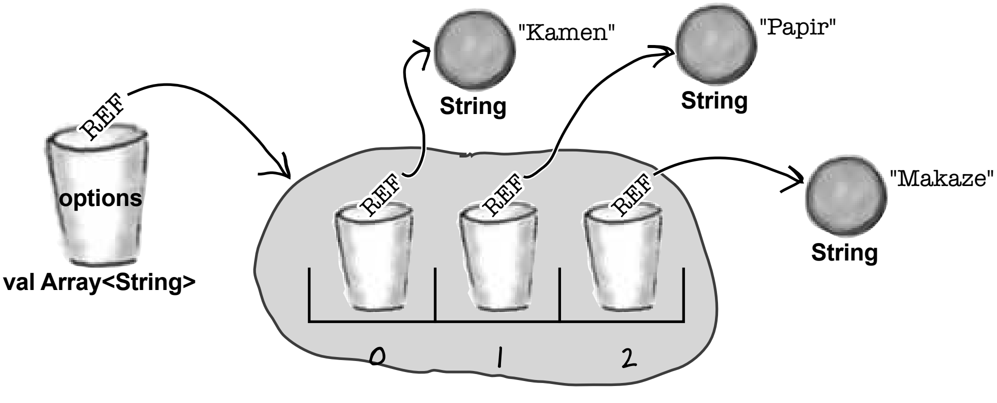
3

Izlazak iz funkcije main

Vreme je da odmaknemo malo dalje i učimo o funkcijama. Dosad se sav kôd koji ste pisali nalazio unutar funkcije *main* u aplikaciji. Ali ako želite da pišete kôd koji je **bolje organizovan** i **lakše se održava**, treba da znate ***kako da podelite kôd u odvojene funkcije***. U ovom poglavlju ćete naučiti *kako se pišu funkcije* i *komunicira* sa aplikacijom tako što ćete napraviti igru. Otkrićete kako se pišu kompaktne ***funkcije sa jednim izrazom***. Usput ćete saznati kako da ***prolazite više puta kroz opsege i kolekcije*** koristeći moćnu petlju *for*.



Napravite igru: Kamen, papir, makaze 60
Dizajn igre visokog nivoa 61
Igra treba da izabere opciju 63
Kako se pišu funkcije 64
Funkciji možete proslediti više stvari 65
Od funkcije možete dobiti nešto nazad 66
Funkcije sa telom od jednog izraza 67
Dodajte funkciju getGameChoice datoteci Game.kt 68
Funkcija getUserChoice 75
Kako rade petlje for 76
Zatražite izbor opcije od korisnika 78
Procena ispravnosti korisnikovog unosa 81
Dodajte funkciju getUserChoice datoteci Game.kt 83
Dodajte printResult datoteci Game.kt 87
Kutija sa Kotlinovim alatima 89



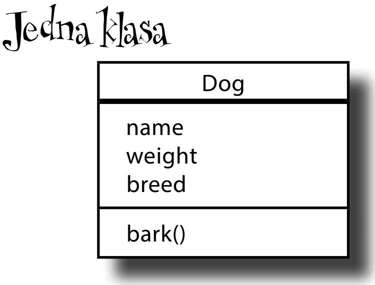
klase i objekti

4

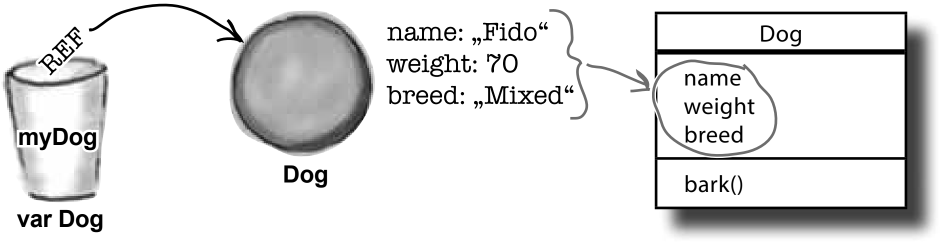
Ponešto o klasama

Vreme je da odemo dalje od Kotlinovih osnovnih tipova.

Pre ili kasnije, poželećete da upotrebite nešto *više* od Kotlinovih osnovnih tipova vrednosti. I tu na scenu stupaju ***klase***. Klase su *šabloni* koji vam omogućavaju da **pravite sopstvene tipove objekata** i da definišete njihova svojstva i funkcije. Ovde ćete naučiti **kako da oblikujete i definišete klase**, i kako da ih koristite da biste **pravili nove tipove objekata**. Upoznaćete ***konstruktore***, ***blokove za inicijalizaciju***, ***metode za očitavanje*** i ***metode za upisivanje***, i otkrićete kako se oni mogu koristiti za zaštitu svojstava. Najzad, naučićete kako je **sakrivanje podataka ugrađeno u *sav* Kotlinov kôd**, štedeći vam vreme, trud i kucanje na tastaturi.



Tipovi objekata definisani pomoću klasa 92
Kako da dizajnirate sopstvene klase 93
Definišimo klasu Dog 94
Kako se pravi objekat Dog 95
Kako se pristupa svojstvima i funkcijama 96
Napravite aplikaciju Songs 97
Čudo nastanka objekta 98
Kako se prave objekti 99
Iza scene: pozivanje konstruktora Dog 100
Detaljnije o svojstvima 105
Fleksibilna inicijalizacija svojstava 106
Kako se koriste blokovi za inicijalizaciju 107
MORATE inicijalizovati svojstva 108
Kako se proverava ispravnost vrednosti svojstava? 111
Kako napisati namensku metodu za očitavanje vrednosti 112
Kako napisati namensku metodu za zadavanje vrednosti 113
Kompletan kôd za projekat Dogs 115
Kutija sa Kotlinovim alatima 120



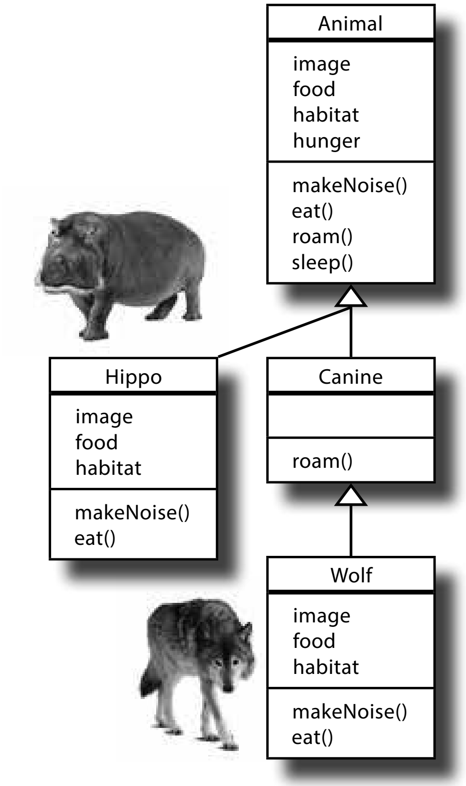
potklase i natklase

5

Korišćenje nasleđivanja

Da li ste ikada pomislili da bi neki tip objekta bio savršen samo još ako biste mogli da promenite nekoliko stvari?

Pa, to je jedna od prednosti **nasleđivanja**. Ovde ćete naučiti kako da pravite **potklase**, i nasleđujete svojstva i funkcije **natklasa**. Otkrićete ***kako da redefinišete funkcije i svojstva*** da bi se vaše klase ponašale onako kako *vi* želite, i saznaćete kada je to svrsishodno (a kada nije). Najzad, videćete kako nasleđivanje pomaže da **izbegnete dupliranje koda**, i kako da povećate fleksibilnost pomoću **polimorfizma**.



Nasleđivanje pomaže da izbegnete dupliranje koda 122
Šta ćemo da radimo 123
Osmisli strukturu nasleđivanja klasa životinja 124
Koristite nasleđivanje da biste izbegli dupliranje koda u potklasama 125
Šta potklase treba da pregaze? 126
Neke životinje možemo grupisati 127
Dodajte klase Canine i Feline 128
Koristite IS-A da biste proverili hijerarhiju klasa 129
Test IS-A radi svuda u stablu nasleđivanja 130
Napravićemo neke Kotlinove životinje 133
Deklarišite natklasu i njena svojstva i funkcije kao otvorene 134
Kako potklasa nasleđuje od natklase 135
Kako se (i kada) redefinišu svojstva 136
Redefinisanje svojstava omogućava da uradite više od dodeljivanja početnih vrednosti 137
Kako se redefinišu funkcije 138
Redefinisana funkcija ili svojstvo ostaju open... 139
Dodajte klasu Hippo projektu Animals 140
Dodajte klase Canine i Wolf 143
Koja funkcija je pozvana? 144
Kada pozovete funkciju za promenljivu, verzija objekta će reagovati 146
Supertip možete koristiti za parametre i za povratnu vrednost funkcije 147
Ažurirani kôd Animals 148
Kutija sa Kotlinovim alatima 153

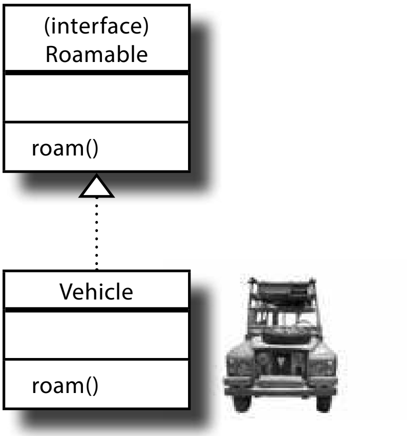
apstraktne klase i interfejsi

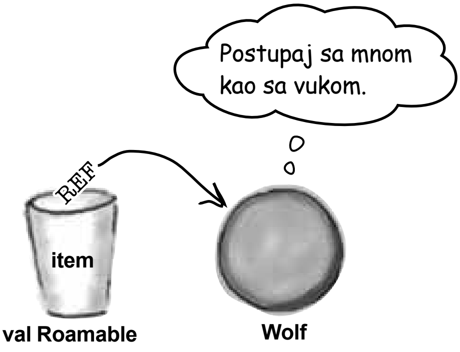
6

Ozbiljan polimorfizam

Hijerarhija nasleđivanja natklasa samo je početak.

Ako želite da ***u potpunosti iskoristite polimorfizam***, treba da projektujete koristeći **apstraktne klase** i **interfejse**. U ovom poglavlju ćete otkriti kako se pomoću apstraktnih klasa zadaje koje klase u hijerarhiji ***mogu, a koje ne mogu biti instancirane***. Videćete kako one mogu primorati potklase da ***obezbede sopstvene implementacije***. Saznaćete kako se interfejsi koriste za ***zajednička ponašanja nezavisnih klasa***. A usput, do tančina ćete upoznati ***is***, ***as*** i ***when***.





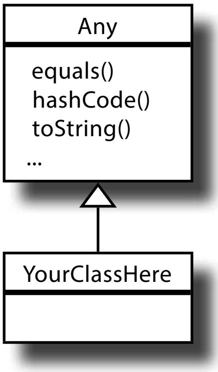
Nova poseta hijerarhiji klase Animal 156
Neke klase ne bi trebalo instancirati 157
Apstraktno ili konkretno? 158
Apstraktna klasa može imati apstraktna svojstva i funkcije 159
Klasa Animal ima dve apstraktne funkcije 160
Kako se implementira apstraktna klasa 162
MORATE implementirati sva apstraktna svojstva i funkcije 163
Ažurirajmo projekat Animals 164
Nezavisne klase mogu imati zajedničko ponašanje 169
Interfejs omogućava da definišete zajedničko ponašanje IZVAN hijerarhije natklase 170
Definišimo interfejs Roamable 171
Kako se definišu svojstva interfejsa 172
Deklarišite da klasa implementira interfejs... 173
Kako se implementira više interfejsa 174
Kako ćete znati da li da napravite klasu, potklasu, apstraktnu klasu ili interfejs? 175
Ažurirajte projekat Animals 176
Interfejsi omogućavaju polimorfizam 181
Gde se koristi operator is 182
Koristite when da poredite promenljivu i mnogo opcija 183
Operator is obično obavlja pametnu konverziju 184
Koristite as da obavite izričitu konverziju 185
Ažurirajte projekat Animals 186
Kutija sa Kotlinovim alatima 189

klase za podatke

7 Rad sa podacima

Niko ne želi da troši život izmišljajući toplu vodu.

Većina aplikacija sadrži klase čija je glavna namena *skladištenje podataka*. Da bi vam olakšali programerski život, tvorci Kotlina su smislili koncept **klase data tj. klase za podatke** (engl. *data class*). Ovde ćete naučiti kako klase za podatke omogućavaju da pišete kôd koji je ***čistiji i koncizniji*** nego što ste i sanjali da je moguće. Istražićete njihove **pomoćne funkcije** (engl. *utility functions*) i otkriti kako da ***raspakujete objekat za podatke i dobijete njegove sastavne delove***. Usput ćete saznati kako ***unapred zadate vrednosti parametara*** mogu učiniti kôd fleksibilnijim, a predstavićemo vam i **Any**, *majku svih natklasa*.



Objekti data se smatraju jednakim ako njihova svojstva čuvaju iste vrednosti.

== poziva funkciju equals	192
equals se nasleđuje od natklase Any	193
Zajedničko ponašanje definisano u Any	194
Možda bismo želeli da equals proverava jesu li dva objekta ekvivalentna	195
Klasa data omogućava da pravite objekte data	196
Klase za podatke redefinišu nasleđeno ponašanje	197
Kopirajte objekte data pomoću funkcije copy	198
Klase za podatke definišu funkcije componentN...	199
Napravite projekat Recipes	201
Pomešane poruke	203
Generisane funkcije koriste samo svojstva definisana u konstruktoru	205
Inicijalizovanje mnogo svojstava može dovesti do nezgrapnog koda	206
Kako se koriste unapred definisane vrednosti konstruktora	207
I funkcije mogu da koriste unapred definisane vrednosti	210
Preklapanje funkcije	211
Ažurirajmo projekat Recipes	212
Kutija sa Kotlinovim alatima	217

null i izuzeci

8

Bezbedni i sigurni

Svi žele da pišu bezbedan kôd.

Divna vest je da je Kotlin napravljen tako da mu bezbednost koda bude u srži. Na početku ćemo vam pokazati zašto Kotlinova upotreba **tipova koji prihvataju null** znači da će se *teško desiti da doživite NullPointerException tokom čitavog boravka u Kotlingradu*. Otkrićete kako da pravite *bezbedne pozive*, i kako Kotlinov operator **Elvis** sprečava *vaše potresanje*. A kada završimo sa vrednostima null, saznaćete kako da **generišete i hvatate izuzetke** kao profesionalac.

Ovo je operator Elvis.

Kako se uklanjaju reference do objekata iz promenljivih? 220

Uklonite referencu do objekta pomoću null 221

Tip koji prihvata null koristićete svuda gde i tip koji ne prihvata null 222

Kako se pravi niz tipova koji prihvataju null 223

Kako se pristupa funkcijama i svojstvima tipa koji prihvata null 224

Bezbedni pozivi 225

Bezbedne pozive možete ulančavati 226

Pomoću bezbednih poziva možete dodeljivati vrednosti... 228

Koristite let da bi se kôd izvršio ako vrednosti nisu null 231

Korišćenje let sa stavkama niza 232

Umesto korišćenja izraza if... 233

Operator !! namerno generiše NullPointerException 234

Napravite projekat Null Values 235

Izuzetak se generiše u izuzetnim okolnostima 239

Hvatanje izuzetaka pomoću try/catch 240

Koristite finally za stvari koje hoćete da uradite bez obzira na sve 241

Izuzetak je objekat tipa Exception 242

Izuzeci se mogu generisati izričito 244

try i throw su izrazi 245

Kutija sa Kotlinovim alatima 250

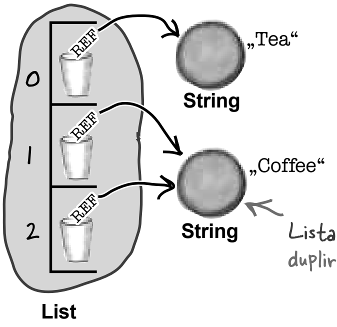
kolekcije

9

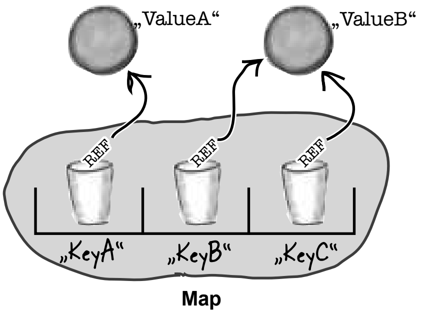
Organizujte se

Da li ste ikada poželeli nešto fleksibilnije od niza?

Kotlin dobijate sa obiljem korisnih **kolekcija** koje pružaju više fleksibilnosti i bolje kontrolisanje ***čuvanja grupa objekata i upravljanja njima***. Želite *listu čija se veličina može menjati da biste joj dodavali stavke*? Želite da *sortirate, mešate ili obrćete njen sadržaj*? Želite da *pronađete nešto po imenu*? Ili želite nešto što će automatski *iskoreniti duplikate,* a da vi ne mrdnete malim prstom? Ako želite bilo šta od tih stvari, i više od toga, nastavite da čitate. Sve to je ovde...



Lista dozvoljava duplirane vrednosti.



Mapa dozvoljava duplikate vrednosti, ali ne i duplikate ključeva.

Nizovi mogu biti korisni... 252
...ali ima stvari sa kojima niz ne može da se nosi 253
Ako ste u nedoumici, idite u biblioteku 254
List, Set i Map 255
Fantastične liste... 256
Napravite izmenjivu listu 257
Možete ukloniti vrednost... 258
Možete menjati redosled i praviti grupne izmene... 259
Napravite projekat Collections 260
Liste dozvoljavaju duple vrednosti 263
Kako se pravi skup 264
Kako skup traži duplikate 265
Heš kodovi i jednakost 266
Pravila za redefinisanje hashCode i equals 267
Kako se koristi izmenjivi skup 268
Ažurirajte projekat Collections 270
Vreme je za mape 276
Kako se koristi mapa 277
Napravite izmenjivu mapu 278
Možete uklanjati unose iz izmenjive mape 279
Mape i izmenjive mape možete kopirati 280
Kompletan kod projekta Collections 281
Kutija sa Kotlinovim alatima 287

generički tipovi

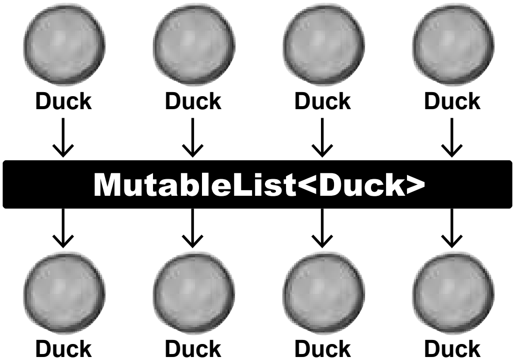
10

Znajte šta je in, a šta je out

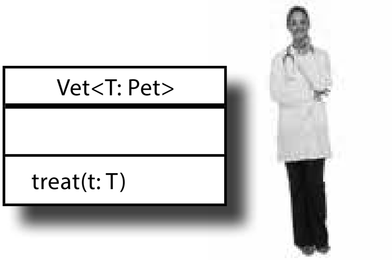
Svi vole dosledan kôd.

Jedan od načina za pisanje doslednog koda, manje podložnog problemima, jeste korišćenje ***generičkih tipova*** (engl. *generics*). U ovom poglavlju ćemo pogledati kako **Kotlinove klase kolekcija koriste generičke tipove** da bi vas sprečile da smestite Kupus u List<Galeb>. Otkrićete kada i kako treba da ***pišete sopstvene generičke klase, interfejse i funkcije***, i kako da **ograničite generički tip** na određeni supertip. Najzad, saznaćete ***kako se koriste kovarijanse i kontravarijanse,*** zahvaljujući kojima **VI** upravljate ponašanjem generičkog tipa.

SA generičkim tipovima, objekti ULAZE samo kao reference do objekata Duck...



...i IZLAZE kao reference tipa Duck.



Kolekcije koriste generičke tipove 290
Kako se izmenjiva lista definiše 291
Korišćenje parametara tipa sa izmenjivom listom 292
Šta sve možete sa generičkom klasom ili interfejsom 293
Evo šta ćemo uraditi 294
Napravite hijerarhiju klase Pet 295
Definišite klasu Contest 296
Dodajte svojstvo scores 297
Napravite funkciju getWinners 298
Napravite nekoliko objekata Contest 299
Napravite projekat Generics 301
Hijerarhija Retailer 305
Definišite interfejs Retailer 306
Možemo da napravimo objekte CatRetailer, DogRetaileri FishRetailer... 307
Upotrebite out za kovarijantan generički tip 308
Ažurirajte projekat Generics 309
Treba nam klasa Vet 313
Napravite objekte Vet 314
Upotrebite in za kontravarijantan generički tip 315
Generički tip može biti lokalno kontravarijantan 316
Ažurirajte projekat Generics 317
Kutija sa Kotlinovim alatima 324

lambde i funkcije višeg reda

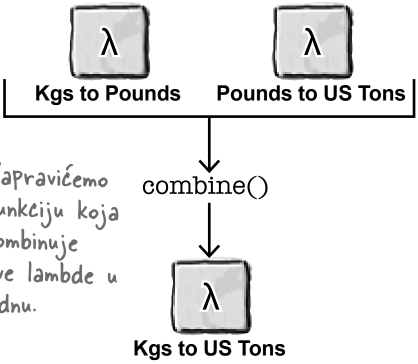
11

Tretiranje koda kao podataka

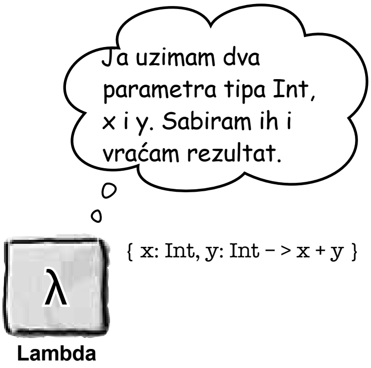
Želite da pišete još moćniji i fleksibilniji kôd?

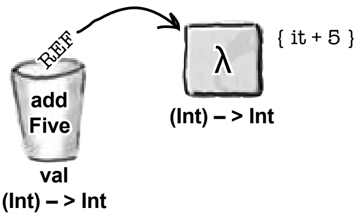
Ako želite, potrebne su vam **lambde**. *Lambda* – ili *lambda izraz* – blok je koda koji možete prosleđivati kao objekat. U ovom poglavlju ćete otkriti ***kako da definišete lambdu, kako da je dodelite promenljivoj***, a potom ***izvršite njen kôd***. Učićete o **tipovima funkcije**, i kako vam oni mogu pomoći da pišete **funkcije višeg reda** koje koriste lambde kao vrednosti parametara ili povratne vrednosti. Usput ćete saznati i kako mali ***sintaksni slatkiš može da vam zasladi programerski život***.

Predstavljamo lambde 326
Kako izgleda lambda 327
Lambdu možete dodeliti promenljivoj 328
Lambda izrazi imaju tip 331
Kompajler može da izvede tipove lambdinih parametara 332
Koristite odgovarajuću lambdu za tip promenljive 333
Napravite projekat Lambdas 334
Lambdu možete proslediti funkciji 339
Pozovite lambdu u telu funkcije 340
Šta se dešava kada pozovete funkciju 341
Lambdu možete da premestite IZVAN ()... 343
Ažurirajte projekat Lambdas 344
Funkcija može da vrati lambdu 347
Napišite funkciju koja i prima i vraća lambde 348
Kako se koristi funkcija combine 349
Koristite typealias da dodelite različito ime postojećem tipu 353
Ažurirajte projekat Lambdas 354
Kutija sa Kotlinovim alatima 361



Napravićemo funkciju koja kombinuje dve lambde u jednu.





ugrađene funkcije višeg reda

12

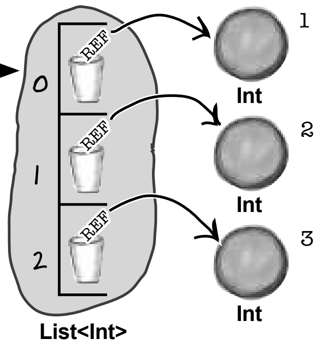
Osnažite svoj kôd

Kotlin ima izobilje ugrađenih funkcija višeg reda.

U ovom poglavlju ćemo vam predstaviti neke od najkorisnijih. Upoznaćete fleksibilnu ***porodicu filtara*** i otkriti kako vam oni mogu pomoći da smanjite kolekciju. Naučićete kako da ***transformišete kolekciju pomoću mape, kako da u petlji prolazite kroz njene stavke koristeći forEach*** i kako da ***grupišete stavke u kolekciji koristeći groupBy***. Upotrebićete i ***fold*** da biste obavili složena izračunavanja *koristeći samo jedan red koda*. Do kraja poglavlja moći ćete da pišete kôd koji je **moćniji nego što ste i sanjali da je moguće**.

Kotlin ima gomilu ugrađenih funkcija višeg reda 364
Funkcije min i max rade sa osnovnim tipovima 365
Detaljniji pogled na lambda parametar u minBy i maxBy 366
Funkcije sumBy i sumByDouble 367
Napravite projekat Groceries 368
Upoznajte funkciju filter 371
Koristite map da biste transformisali kolekciju 372
Šta se dešava kada se kôd pokrene 373
forEach radi kao petlja for 375
forEach nema povratnu vrednost 376
Ažurirajte projekat Groceries 377
Upotrebite groupBy da podelite kolekciju na grupe 381
groupBy možete koristiti u lancima poziva za funkcije 382
Kako se koristi funkcija fold 383
Iza scene: funkcija fold 384
Još primera za fold 386
Ažurirajte projekat Groceries 387
Kutija sa Kotlinovim alatima 394
Odlazak iz grada... 395

Ove stavke nemaju prirodni redosled. Da bismo pronašli najveću ili najmanju vrednost moramo zadati kriterijum kao što je unitPrice ili quantity.



Funkcija fold počinje od prve stavke u kolekciji.

korutine

i

Paralelno izvršavanje koda

Neke zadatke je bolje obaviti u pozadini.

Ako hoćete da očitate podatke sa sporog spoljnog servera, verovatno ne želite da se ostatak koda dosađuje čekajući da se taj posao završi. U takvim situacijama, **korutine će vam biti nove najbolje drugarice**. Korutine (engl. *coroutine*) omogućavaju da pišete kôd koji se ***izvršava asinhrono***. To znači *manje vremena u besposličarenju, bolje korisničko iskustvo*, a može i *učiniti aplikaciju skalabilnijom*. Nastavite da čitate i saznaćete tajnu kako da razgovarate sa Bobom, dok istovremeno slušate Suzi.

Ovaj put, timpani i činele udaraju paralelno.

testiranje

ii

Neka vaš kôd bude odgovoran

Svi znaju da dobar kôd treba da radi.

Ali svaka izmena koda donosi i rizik da ste uveli greške koje će ga sprečiti da radi kako bi trebalo. Zbog toga je *temeljno testiranje* izuzetno važno: ono znači da ćete saznati za probleme u kodu *pre nego što ga upotrebite u živom okruženju*. U ovom dodatku govorimo o bibliotekama ***JUnit*** i ***KotlinTest*** koje možete upotrebiti za **jedinično testiranje koda** tako da *uvek imate mrežu za spasavanje*.

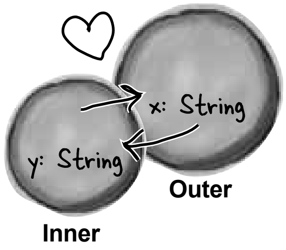
ostaci

iii

Najbitnijih deset stvari (koje nismo obradili)

Čak i nakon svega ovoga, ima još malo.

Ima još nekoliko stvari koje bi trebalo da znate. Ne bismo se osećali dobro ako bismo ih zanemarili, a zaista smo želeli da vam pružimo knjigu koju možete da podignete bez prethodnog treninga u lokalnoj teretani. Pre nego što odložite knjigu, **pročitajte i ove sitnice**.



Objekti Inner i Outer imaju posebnu vezu. Inner može da koristi promenljive iz Outer i obrnuto.

1. Paketi i uvoženje 416
2. Modifikatori vidljivosti 418
3. Klase enum 420
4. Zapečaćene klase 422
5. Ugnežđene i unutrašnje klase 424
6. Deklaracie i izrazi objekata 426
7. Proširenja 429
8. Return, break i continue 430
9. Još zabave sa funkcijama 432
10. Interoperabilnost 434