

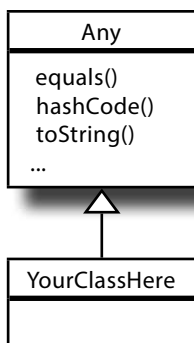
klase za podatke

Rad sa podacima

7

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*). Ovdje 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 predstavimo 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.*

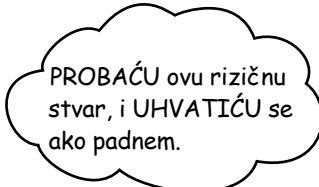
Otkrićete kako da pravite *bezbedne pozive*, i kako Kotlinov operator **Elvis** sprečava *vaše potresanje*. A kada završimo sa vrednostima null, saznaćete kako da **generišete i hvatate izuzetke** kao profesionalac.



E, baš vam hvala.



Ovo je operator Elvis.



PROBAĆU ovu rizičnu stvar, i UHVATIĆU se ako padnem.



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

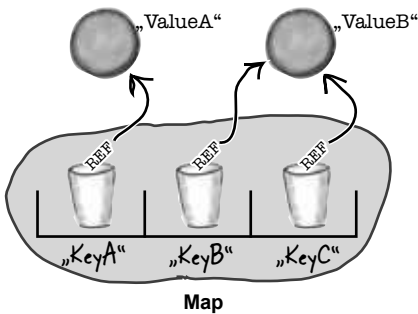
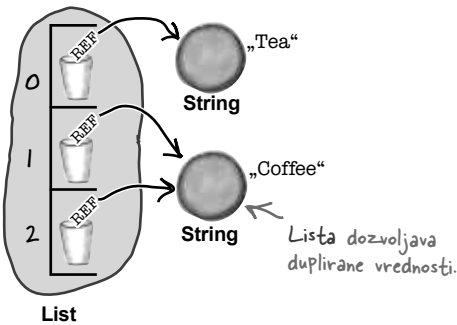
Organizujte se



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...

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 redefinisavanje 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



Mapa dozvoljava duplikate vrednosti, ali ne i duplikate ključeva.

generički tipovi

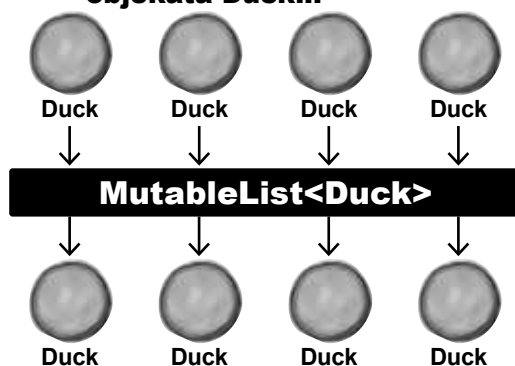
10

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, interfejsa 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.

Vet<T: Pet>
treat(t: T)



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, DogRetailer i 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

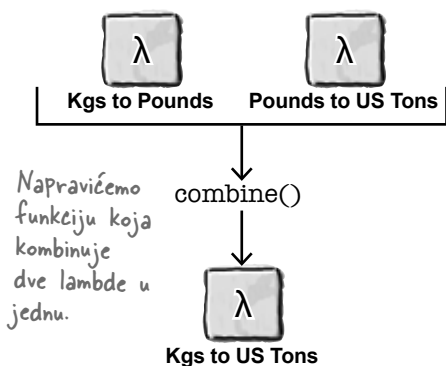
11

Tretiranje koda kao podataka

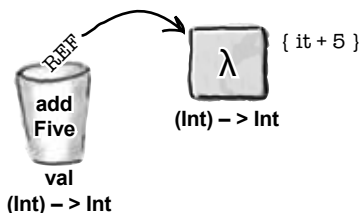
Želite da pišete još moćniji i fleksibilniji kôd?

Ako želite, potrebne su vam **lambde**. *Lambda* – ili *lambda izraz* – blok je koda koji možete prosledivati kao objekat. U ovom poglavlju ćete otkriti **kako da definišete lambda**, **kako da je dodelite promenljivoj**, a potom **izvršite njen kôd**. Učičete 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 lambda za tip promenljive	333
Napravite projekat Lambdas	334
Lambdu možete proslediti funkciji	339
Pozovite lambda 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 lambda	347
Napišite funkciju koja i prima i vraća lambda	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



Ja uzimam dva parametra tipa Int, x i y. Sabiram ih i vraćam rezultat.



ugrađene funkcije višeg reda

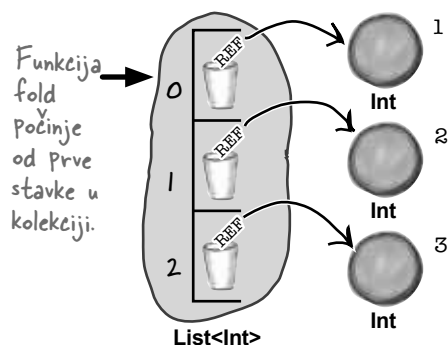
12

Oснаžite svoj kôd

Kotlin ima izobilje ugrađenih funkcija višeg reda.

U ovom poglavlju ćemo vam predstaviti neke od najkorisnijih. Upoznaćete fleksibilnu **porodicu filtera** 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**.

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



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

korutine

Paralelno izvršavanje koda

Neke zadatke je bolje obaviti u pozadini.

Ako hoćete da očitete 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 saznacete tajnu kako da razgovarate sa Bobom, dok istovremeno slušate Suzi.



Bam! Bam! Bam! Bam! Bam! Bam!
Tiš! Tiš!

↑
Ovaj put, timpani i činele udaraju paralelno.

testiranje

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*.



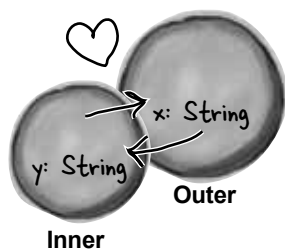
ostaci



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. Deklaracije i izrazi objekata	426
7. Proširenja	429
8. Return, break i continue	430
9. Još zabave sa funkcijama	432
10. Interoperabilnost	434