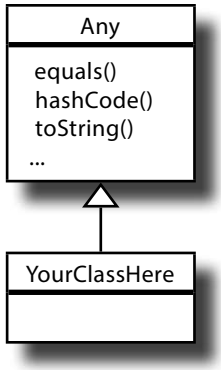
klase za podatke

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.

null i izuzeci

8 Bezbedni i sigurni

Svi žele da pišu bezbedan kôd.

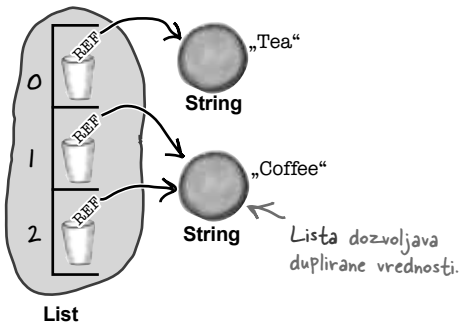
Divna vest je da je Kotlin napravljen tako da mu bezbednost koda bude u srži. Na početku ćemo vam pokazati zašto Kotlinova upotreba **tipova koji prihvataju null** znači da će se *teško desiti da doživite NullPointerException tokom čitavog boravka u Kotlingradu*. Otkrićete kako da pravite *bezbedne pozive*,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

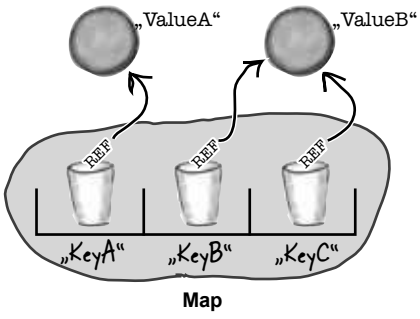
kolekcije

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...



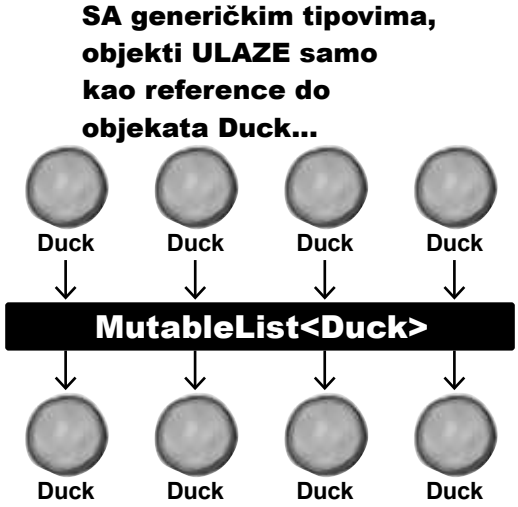


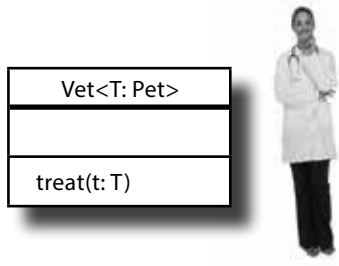
generički tipovi

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.



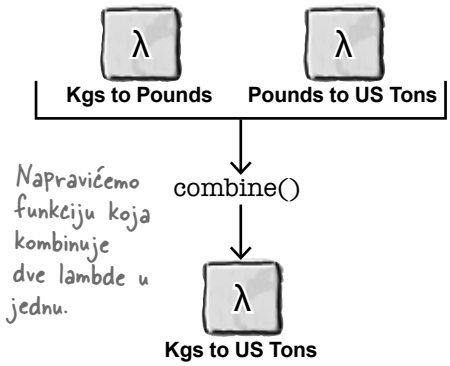


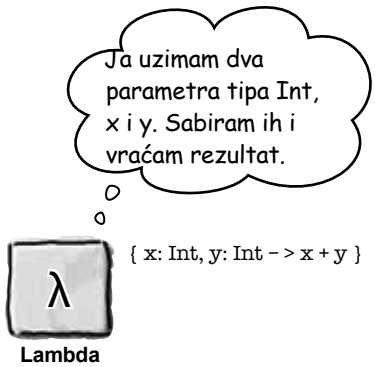
lambde i funkcije višeg reda

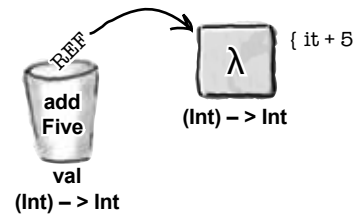
11 Tretiranje koda kao podataka

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***.





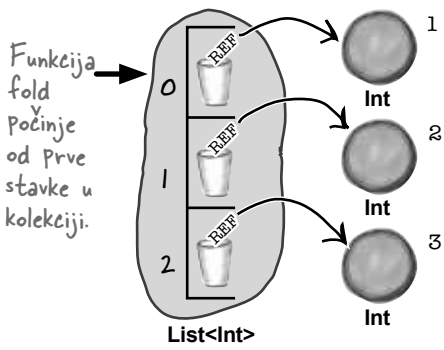


ugrađene funkcije višeg reda

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**.



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.

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

Ovaj put, timpani i činele udaraju paralelno.

testiranje

ii

Neka vaš kôd bude odgovoran

Svi znaju da dobar kôd treba da radi.

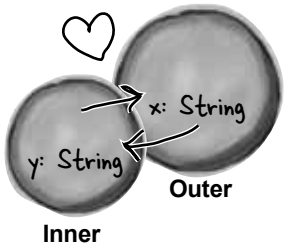
Ali svaka izmena koda donosi i rizik da ste uveli greške koje će ga sprečiti da radi kako bi trebalo. Zbog toga je *temeljno testiranje* izuzetno važno: ono znači da ćete saznati za probleme u kodu *pre nego što ga upotrebite u živom okruženju*. U ovom dodatku govorimo o bibliotekama ***JUnit*** i ***KotlinTest***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

ostaci

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.