Dr. Edward Lavieri, Peter Verhas

Java 9

Napišite reaktivni, modularni, konkurentni i bezbedni kod

kompjuter biblioteka

Packt›

Dr. Edward Lavieri,
Peter Verhas

Java 9

kompjuter biblioteka

Packt›

Izdavač:

kompjuter biblioteka

Obalskih radnika 4a, Beograd

Tel: 011/2520272

e-mail: kombib@gmail.com

internet: www.kombib.rs

Urednik: Mihailo J. Šolajić

Za izdavača, direktor:
Mihailo J. Šolajić

Autor: Tarek Ziadé

Prevod: Slavica Prudkov

Lektura: Miloš Jevtović

Slog: Zvonko Aleksić

Znak Kompjuter biblioteke:
Miloš Milosavljević

Štampa: „Pekograf“, Zemun

Tiraž: 500

Godina izdanja: 2018.

Broj knjige: 500

Izdanje: Prvo

ISBN: 978-86-7310-523-9

Mastering Java 9
Dr. Edward Lavieri, Peter Verhas

ISBN 978-1-78646-873-4

O AUTORIMA

Dr Edward Lavieri je istaknuti dizajner igara i programer sa odličnim akademskim obrazovanjem. Doktorirao je kompjuterske nauke na Tehničkom univerzitetu u Koloradu, magistrirao je upravljanje informacionim sistemima na Univerzitetu Bowie State i upravljanje operacijama na Univerzitetu Arkanzas i stekao je diplomu učitelja na Univerzitetu Capella.

Organizuje kurseve kompjuterskih nauka od 2002. godine. Povukao se iz američke mornarice sa funkcije šefa komande, posle 25 godina službe. Kao osnivač i kreativni direktor studija za dizajn i razvoj softvera „three19“, konstantno dizajnira i razvija softvere. Koristi različite softvere i programerske alatke za razvoj igara. Velika je njegova strast prema razvoju sistema za adaptivno učenje, edukativnih igara i mobilnih aplikacija.

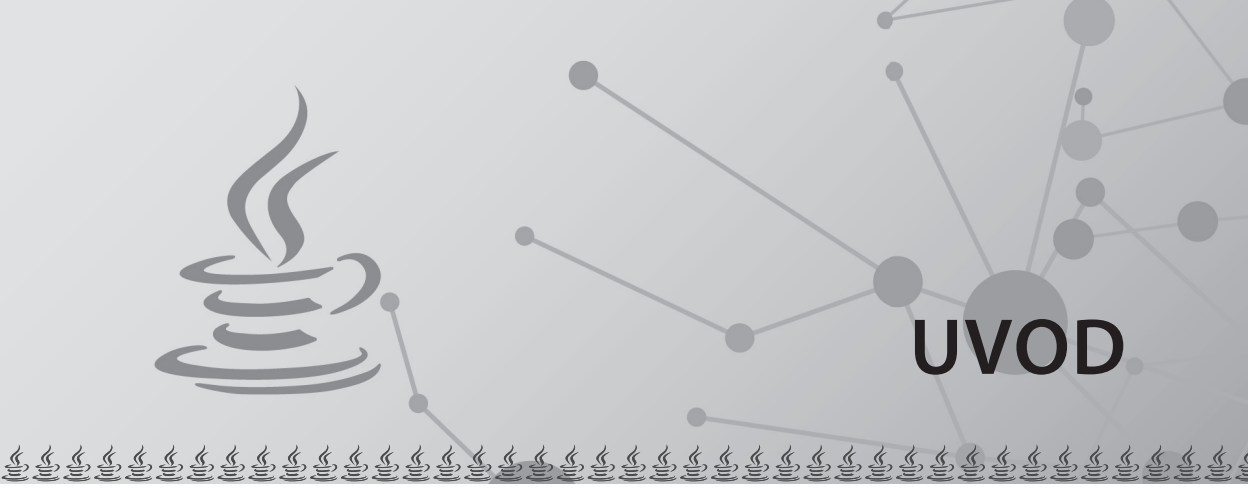
Edward Lavieri je autor knjiga *Adaptive Learning for Educational Game Design* (CreateSpace), *Getting Started with Unity 5* (Packt Publishing), *LiveCode Mobile Development HOTSHOT* (Packt Publishing), *LiveCode Mobile Development Cookbook* (Packt Publishing) i *Software Consulting: A Revolutionary Approach* (CreateSpace), a bio je i tehnički urednik knjige *Excel Formulas and Functions for Dummies* (Wiley Publishing). Takođe je organizovao kurseve kompjuterskih nauka, informacionih sistema i razvoja igara.

O RECENZENTU

Mandar Jog je stručni IT trener, sa više od 15 godina iskustva u obuci. On je stručnjak za tehnologije Java, J2EE i Android. Takođe poseduje SCJP i SCWCD sertifikate. Povremeno objavljuje tekstove na svom blogu u kojima čitaocima objašnjava kompleksne koncepte platformi Java i J2EE. Redovni je govornik na mnogim tehničkim seminarima i radionicama na fakultetima inženjerskih nauka.

Takođe je bio tehnički recenzent knjige *Modular Programming in Java 9* u izdanju „Packta“.

Mnogo se zahvaljujem supruzi Tejaswini - bila je moja inspiracija na ovom „putovanju“. Podjednako sam zahvalan sinu Ojasu - njegov osmeh me je uvek podsticao da nastavim dalje.



UVOD

Java 9 i njegove nove funkcije poboljšavaju Javu, jedan od jezika koji programeri najčešće koriste za izgradnju robusnih softverskih aplikacija. U Javi 9 je stavljen poseban naglasak na integraciju modularnosti pomoću alatke Jigsaw. Ova knjiga predstavlja jednostavno uputstvo za učenje programskog jezika Java 9.

Dati su pregled i objašnjenje novih funkcija koje su uvedene u jezik Java 9 i važnosti novih interfejsa API i poboljšanja. Neke od novih funkcija Jave 9 su inovativne, pa, ako ste iskusni programer, moći ćete da izradite ekonomičniju poslovnu aplikaciju kada naučite ove nove funkcije. Date su nove praktične smernice za primenu novostečenog znanja u vezi sa Javom 9 i dodatne informacije o budućem razvoju Java platforme. Ova knjiga će vam pomoći da povećate produktivnost, tako što ćete vašu aplikaciju učiniti bržom. Kada naučite najbolje tehnike koja se koriste u Javi, moći ćete da rešavate programerske probleme u vašoj organizaciji.

Nakon što pročitate ovu knjigu, ne samo da ćete znati važne koncepte Jave 9, već i važne aspekte programiranja pomoću ovog sjajnog programskog jezika.

ŠTA OBUHVATA OVA KNJIGA?

U Poglavlju 1, *Pregled Jave 9*, istražujemo najvažnije funkcije koje su uvedene u Javu 9, uključujući Project Jigsaw, Java Shell, sakupljač „smeća“ G1 i reaktivno programiranje. Ovo poglavlje sadrži uvod u ove funkcije kao pripremu za detaljnije razmatranje u narednim poglavljima.

U Poglavlju 2, *Otkrivanje Jave 9*, razmatramo nekoliko izmena Java platforme radi implementacije efikasnosti hip prostora, dodeljivanja memorije, poboljšanja procesa kompajliranja, testiranja tipa, anotacija, automatizovanih testova kompajlera tokom izvršenja i poboljšanog sakupljača „smeća“.

U Poglavlju 3, *Poboljšanja jezika Java 9*, naglasak je stavljen na izmene koje su izvršene u jeziku Java. Ove izmene utiču na promenljive hendlere, upozorenja o amortizaciji, poboljšanja u funkciji Project Coin u Javi 7 i obradu iskaza import.

U Poglavlju 4, *Izrada modularnih aplikacija u Javi 9*, razmatramo strukturu Java modula koji je određen pomoću funkcije Project Jigsaw i način na koji je ta funkcija implementirana

u Java platformu. U ovom poglavlju je dat i pregled važnih unutrašnjih promena Java platforme, koje se odnose na novi modularni sistem.

U Poglavlju 5, *Migriranje aplikacija u Javu 9*, istražujemo migraciju Java 8 aplikacija u Java 9 platformu. Razmatraćemo ručne i poluautomatizovane procese migracije.

U Poglavlju 6, *Eksperimentisanje sa Java Shellom*, razmatramo alatku komandne linije read-eval-print u Javi 9. Ovo poglavlje sadrži informacije o ovoj alatki, koncept read-eval-print, komande i opcije komandne linije koje se koriste pomoću JShella.

U Poglavlju 7, *Korišćenje novog podrazumevanog sakupljača „smeća" G1*, videćete detaljan pregled sakupljanja „smeća" i kako se upravlja njime u Javi 9.

U Poglavlju 8, *Merenje performanse aplikacija na mikro nivou pomoću JMH-a*, ispitujemo kako se pišu testovi performanse pomoću Java Microbenchmark Harnessa (JMH) - Java biblioteke alatki za pisanje testova za Java virtuelnu mašinu (JVM). Alatku Maven ćemo koristiti zajedno sa JMH-om da bismo vam prikazali moć merenja performansi na mikro nivou pomoću nove Java 9 platforme.

U Poglavlju 9, *Upotreba interfejsa ProcessHandle API*, dat je pregled nove klase interfejsa API koja omogućava upravljanje procesima operativnog sistema.

U Poglavlju 10, *Precizno praćenje steka*, predstavićemo novi interfejs API koji obezbeđuje efikasne funkcije za praćenje steka. Ovo poglavlje sadrži detaljne informacije o tome kako pristupiti podacima o praćenju steka.

U Poglavlju 11, *Nove alatke i poboljšanja postojećih*, biće reči o 16 poboljšanja Java Enhancement Proposals (JEP), koja su integrisana u Java 9 platformu. Ovi JEP-ovi sadrže veliki broj alatki i ažuriranja interfejsa API za lakši razvoj Jave pomoću boljih funkcija za optimizaciju Java aplikacija.

Poglavlje 12, *Poboljšanja konkurentnosti*, posvećeno je poboljšanjima konkurentnosti koja su uvedena u Java 9 platformu. Glavni naglasak je stavljen na podršku za reaktivno programiranje - poboljšanje konkurentnosti pomoću klase Flow interfejsa API. Takođe će biti razmatrana dodatna poboljšanja konkurentnosti koja su uvedena u Javu 9.

U Poglavlju 13, *Bezbednosna poboljšanja*, predstavićemo nekoliko malih izmena koje su izvršene u JDK-u u vezi sa bezbednošću. Bezbednosna poboljšanja koja su uvedena u Java 9 platformu omogućavaju iskusnim programerima da pišu i održavaju aplikacije koje su bezbednije nego ranije.

U Poglavlju 14, *Flegovi komandne linije*, istražujemo izmene flegova komandne linije u Javi 9. Koncepti koji su obuhvaćeni u ovom poglavlju sadrže JVM evidentiranje, kontrolu kompajlera, dijagnostičke komande, agent za profilisanje hip memorije, JHAT, validaciju argumenata flegova komandne linije i kompajliranje za starije verzije platforme.

U Poglavlju 15, *Najbolje tehnike u Javi 9*, naglasak je stavljen na korišćenje uslužnih programa koje obezbeđuje Java 9 platforma za umetanje UTF-8 datoteka svojstava, standarda Unicode 7.0.0, porta Linux/AArch64, višerezolucijskih slika i repozitorijuma sa podacima o zajedničkim lokalnim standardima.

Poglavlje 16, *Smernice za budući razvoj Java platforme*, sadrži pregled budućeg razvoja Java platforme izvan Jave 9. Posebno ćemo se osvrnuti na ono što je planirano za Javu 10 i dodatne izmene koje ćemo videti u budućnosti.

ŠTA VAM JE POTREBNO ZA OVU KNJIGU?

Da biste mogli da koristite ovu knjigu, potrebno je da poznajete makar osnove programskog jezika Java.

Takođe će vam biti potrebne sledeće softverske komponente:

- Java SE Development Kit 9 (JDK)

  http:/ / www. oracle. com/ technetwork/ java/ javase/ downloads/
- Integrisano razvojno okruženje (IDE) za pisanje koda. Evo predloga:
  - Eclipse

    https:/ / www. eclipse. org
  - IntelliJ

    https:/ / www. jetbrains. com/ idea/
  - NetBeans

    https:/ / netbeans. org

ZA KOGA JE OVA KNJIGA?

Ova knjiga je namenjena programerima koji se bave poslovnim programiranjem i Java programerima. Osnovno znanje jezika Java je neophodno.

KONVENCIJE

U ovoj knjizi pronaći ćete više različitih stilova za tekst koje smo upotrebili za različite vrste informacija. Evo nekih primera ovih stilova i objašnjenja njihovog značenja.

Reči koda u tekstu, nazivi tabela baze podataka, nazivi direktorijuma, nazivi fajlova, ekstenzije datoteka, nazivi putanja, kratki URL-ovi, korisnički unos i Twitter identifikatori su prikazani na sledeći način: „Ispod strukture podirektorijuma C:\chapter8-benchmark\src\main\java\com\packt nalazi se datoteka MyBenchmark.java.

Blok koda je postavljen na sledeći način:

```
public synchronized void protectedMethod()
{
. . .
}
```

Novi termini i važne reči su napisani masnim slovima.

Upozorenja ili važne napomene će biti prikazani u ovakvom okviru.

Saveti i trikovi prikazani su ovako.

POVRATNE INFORMACIJE OD ČITALACA

Povratne informacije od naših čitalaca su uvek dobrodošle. Obavestite nas šta mislite o ovoj knjizi – šta vam se dopalo ili šta vam se možda nije dopalo. Povratne informacije čitalaca su nam važne da bismo ubuduće kreirali naslove od kojih ćete dobiti maksimum.

Da biste nam poslali povratne informacije, jednostavno nam pošaljite e-mail na adresu feedback@packtpub.com i u naslovu poruke napišite naslov knjige.

Ako postoji tema za koju ste specijalizovani ili ste zainteresovani da pišete ili sarađujete na nekoj od knjiga, pogledajte vodič za autore na adresi www.packtpub.com/authors.

PREUZIMANJE PRIMERA KODA

Možete da preuzmete fajlove sa primerima koda prateći sledeće korake:

1. Posetite veb stranicu knjige „Naučite Java 9“:
   http://knjige.kombib.rs/naucite-java-9
2. Kliknite Preuzmite kod:

Kada su fajlovi preuzeti, ekstrahujte direktorijum, koristeći najnoviju verziju:

- WinRAR / 7-Zip za Windows
- Zipeg / iZip / UnRarX za Mac
- 7-Zip / PeaZip za Linux

ŠTAMPARSKE GREŠKE

Iako smo preduzeli sve mere da bismo obezbedili tačnost sadržaja, greške su moguće. Ako pronađete grešku u nekoj od naših knjiga (u tekstu ili u kodu), bili bismo zahvalni ako biste nam to prijavili. Na taj način možete da poštedite čitaoce od frustracije, a nama da pomognete da poboljšamo naredne verzije ove knjige. Ako pronađete neku štamparsku grešku, molimo vas da nas obavestite, tako što ćete posetiti stranicu http://ww w.packtpub.com/submit – errata, selektovati knjigu, kliknuti na link **Errata Submission Form** i uneti detalje o grešci koju ste pronašli. Kada je greška verifikovana, vaša prijava će biti prihvaćena i greška će biti aploudovana na naš veb sajt ili dodata u listu postojećih grešaka, pod odeljkom Errata za određeni naslov.

Da biste pogledali prethodno prijavljene greške, posetite stranicu https://www.packtpub.com/books/content/support i unesite naslov knjige u polje za pretragu. Tražena informacija će biti prikazana u odeljku **Errata.**

PIRATERIJA

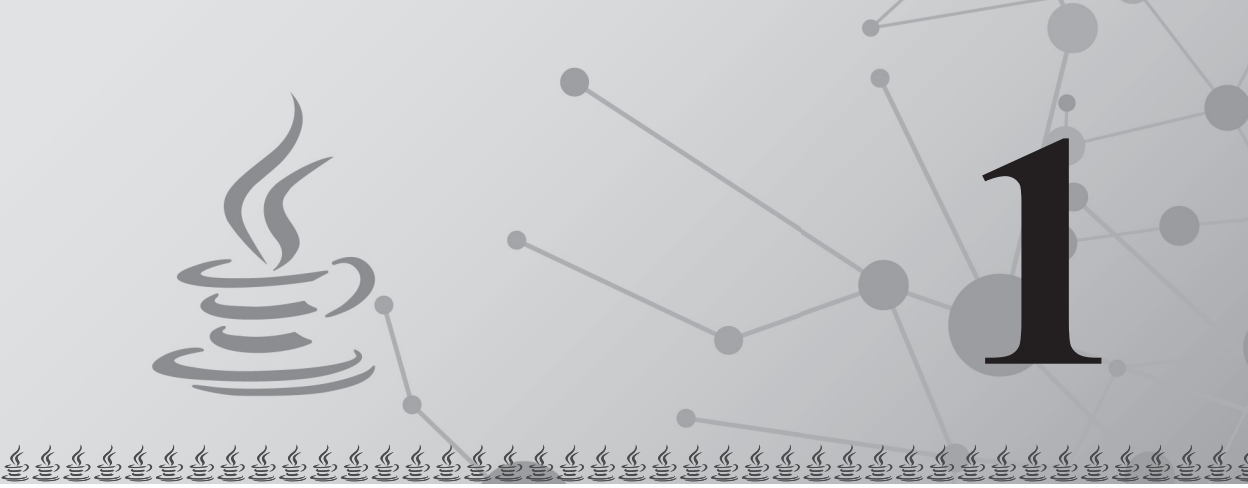
Piraterija autorskog materijala na Internetu je aktuelan problem na svim medijima. Mi u „Packtu“ zaštitu autorskih prava i licenci shvatamo veoma ozbiljno. Ako na Internetu pronađete ilegalnu kopiju naših knjiga, u bilo kojoj formi, molimo vas da nas o tome obavestite i da nam pošaljete adresu lokacije ili naziv veb sajta da bismo mogli da podnesemo odgovarajuću tužbu.

Kontaktirajte sa nama na adresi copyright@packtpub.com i informatori@kombib.rs i pošaljite nam link ka sumnjivom materijalu.

Unapred smo vam zahvalni na pomoći u zaštiti naših autora.

PITANJA

Ako imate bilo kakvih problema sa bilo kojim aspektom knjige, možete da kontaktirate sa nama na adresi questions@packtpub.com da biste nam poslali konkretna pitanja – učinićemo sve što je u našoj moći da rešimo svaki vaš problem.



1

Pregled Jave 9

Java je već u potpunosti izgrađena platforma još od prvog izdanja, koje se pojavilo pre više od dve decenije. Zahvaljujući sjajnoj zajednici programera i širokoj primeni u velikom broju industrijskih grana, platforma nastavlja da se razvija i „drži korak“ sa ostatkom sveta u pogledu performansi, bezbednosti i skalabilnosti. Naše „putovanje“ ćemo započeti istražujući najznačajnije funkcije koje su uvedene u Javu 9, najveće pokretače tih funkcija, ono što nas čeka u budućem razvoju platforme i neke od koncepata koji nisu uvedeni u ovom izdanju.

U ovom poglavlju ćemo razmatrati sledeće teme:

- Java 9 sa distance
- deljenje „monolita“
- eksperimentisanje sa Java Shellom
- preuzimanje kontrole nad spoljnim procesima
- poboljšanje performansi pomoću sakupljača „smeća“ G1
- merenje performansi pomoću JMH-a
- priprema za standard HTTP 2.0
- umetanje reaktivnog programiranja
- proširenje „liste želja“

JAVA SA DISTANCE

Možda se pitate zar Java 9 nije samo ažurirano izdanje sa nizom funkcija koje nisu uvedene u Javu 8. Izdanje Java 9 sadrži mnogo novih funkcija po kojima se razlikuje od drugih izdanja.

Modularizacija Java platforme (koja je razvijena u okviru projekta Jigsaw) predstavlja nesumnjivo najveće „delo“ koje je uspešno uvedeno u Javu 9. Projekat Jigsaw, prvobitno planiran za Javu 8, predstavlja jedan od glavnih razloga zbog kojeg je odlagano konačno izdanje Jave 9. U ovom projektu je, takođe, predstavljeno nekoliko značajnih izmena na Java platformi, zbog kojih se Java 9 smatra glavnim izdanjem. Ove funkcije ćemo detaljno razmatrati u narednim poglavljima.

JCP (Java Community Process) sadrži mehanizme za pretvaranje skupa predloga za funkcije (poznatijih i kao **Java Enhancement Proposals** ili **JEP-ovi**) u formalne specifikacije koje obezbeđuju osnove za proširenje platforme novim funkcijama. Pored predloga za poboljšanje Jave koji su povezani sa projektom Jigsaw, postoji dugačka lista drugih poboljšanja koja su uvedena u Javu 9. U ovoj knjizi ćemo razmatrati različite funkcije logičkih grupa koje se zasnivaju na odgovarajućim predlozima za poboljšanje, uključujući sledeće:

- **JavaShell** (koji se naziva i **JShell**) - interaktivno komandno okruženje za Java platformu
- novi prenosivi interfejsi API za upotrebu u procesima operativnog sistema
- sakupljač „smeća“ **Garbage-first (G1)** , koji je predstavljen u Javi 7, uveden u Javu 9 kao podrazumevani sakupljač „smeća“
- alatka **Java Microbenchmark Harness (JMH)**, koja može da se koristi za merenje performansi Java aplikacija, uvedena u okviru Java distribucije
- podrška za standarde HTTP 2.0 i WebSocket pomoću novog klijenta interfejsa API
- konkurentna poboljšanja, među kojima je i definicija klase `Flow` – ona opisuje interfejs za specifikacije Reactive Streams na Java platformi

Neki od prvobitnih predloga koji su prihvaćeni za Java 9 nisu uvedeni, pa su odloženi za kasnije, zajedno sa drugim zanimljivim funkcijama koje programeri mogu da očekuju u budućnosti.

Ako želite da eksperimentišete sa novim primerima i konceptima pre nego što pređete na sledeća poglavlja, možete da preuzmete JDK 9 za vaš sistem na adresi `http://www.oracle.com/`.

DELJENJE „MONOLITA“

Tokom godina uslužni programi Java platforme se razvijaju i povećavaju, pa ona postaje jedan veliki „monolit“. Da bi platforma postala pogodnija za ugrađene i mobilne uređaje, bilo je neophodno objaviti osnovna izdanja, kao što su Java CDC i Java ME. Međutim, ova izdanja se nisu pokazala kao dovoljno fleksibilna za savremene aplikacije sa različitim zahtevima kada je reč o funkcijama koje obezbeđuje JDK. Prema tome, modularni sistem je postao neophodan za umetanje ne samo modularizacije Java alatki (više od 5.000 Java klasa i 1.500 C ++ izvornih datoteka, sa više od 250.000 linija koda za Hotspot Runtime sistem), već i za obezbeđivanje mehanizma za kreiranje i upravljanje modularnim aplikacijama pomoću istog sistemskog modula koji se koristi u JDK-u. U Javi 8 je obezbeđen neposredni mehanizam koji omogućava aplikacijama da koriste samo podskup interfejsa API koji obezbeđuje JDK, a taj mehanizam je nazvan kompaktni profili (compact profiles). U stvari, kompaktni profili su, predstavljali osnovu za dalji posao koji je trebalo obaviti da bi bila prekinuta zavisnost između različitih komponenata JDK-a, koje su potrebne radi omogućavanja implementacija modularnog sistema u Javu.

Modularni sistem je razvijen u okviru projekta Jigsaw, na osnovu kojeg su formirani neki predlozi za poboljšanje Jave i ciljnog API-ja JSR (376). Uvedeno je mnogo štošta kao odgovor na zahteve za projekat Jigsaw, ali postoji veliki broj funkcija koje su predložene, a nisu uvedene u Javu 9. Osim toga, izvršena je kompletna rekonstrukcija JDK baze za kod i reorganizacija JDK slika koje se mogu distribuirati.

U zajednici programera je bilo mnogo polemike da li postojeće i izvorne Java modularne sisteme, kao što je OSGi, treba ažurirati u okviru JDK-a ili treba obezbediti kompletan novi modularni sistem. OSGi definiše aktivnosti prilikom izvršavanja, kao što su rezolucija modularne zavisnosti, instalacija, deinstalacija, pokretanje i zaustavljanje modula (koji se u OSGiju nazivaju paketi), prilagođeni učitavači klasa i tako dalje. Za razliku od OSGija, funkcija Project Jigsaw definiše modularni sistem prilikom kompajliranja aplikacija tokom kojeg se javlja zavisnost od rezolucije. Štaviše, instaliranjem i deinstaliranjem modula u okviru JDK-a eliminiše se potreba za eksplicitnim umetanjem zavisnosti tokom kompajliranja. Osim toga, učitavanje klasa modula je omogućeno pomoću postojeće hijerarhije učitavača klasa (pokretač-ekstenzija-sistemski učitavači klasa), iako je postojala mogućnost upotrebe prilagođenih učitavača klase modula, koja je prilično slična upotrebi učitavača klase modula za OSGi. Međutim, učitavači klase modula za OSGi nisu prihvaćeni. Java učitavanje klase ćemo detaljnije predstaviti kada budemo razmatrali modularni sistem u Javi.

Dodatne prednosti Java modularnog sistema su poboljšana bezbednost i poboljšane performanse. Zahvaljujući modularizaciji JDK-a i aplikacija u Jigsaw modulima, možemo da kreiramo dobro definisane granice između komponenata i odgovarajućih domena. Ove definisane granice se usklađuju sa bezbednosnom arhitekturom platforme i omogućavaju bolju iskorišćenost resursa. U dva opsežna poglavlja ćemo razmatrati sve prethodne koncepte i prihvatanje Jave 9, za koje je, takođe, potrebno određeno razumevanje mogućih pristupa za migraciju postojećih projekata u njoj.

EKSPERIMENTISANJE SA JAVA SHELLOM

Dugo vremena nije postojalo standardno komandno okruženje u programskom jeziku Java za eksperimentisanje sa novim jezičkim funkcijama ili bibliotekama ili za brzu izradu prototipa. Ako ste želeli da eksperimentišete, morali ste da napišete aplikaciju za testiranje pomoću osnovnog metoda, da je kompajlirate pomoću uslužnog programa `javac`, a zatim da je izvršite. Aplikaciju za testiranje ste mogli da napišete u komandnoj liniji ili pomoću Java IDE-a, ali to nije prikladno kao u interaktivnom komandnom okruženju.

Pokretanje komandnog okruženja u JDK 9 je jednostavno, kao i pokretanje sledeće komande (pod pretpostavkom da je direktorijum `bin` JDK 9 instalacije u trenutnoj putanji):

```
jshell
```

Možda vas pomalo zbunjuje činjenica da komandno okruženje nije uvedeno ranije u Java platformu, jer programski jezici, kao što su Python, Ruby i mnogi drugi, već sadrže interaktivno komandno okruženje u najnovijim verzijama. Međutim, uvođenje komandnog okruženja nije bilo na listi važnih funkcija za ranija Java izdanja, ali sada je to okruženje uvedeno u Javu i spremno je za upotrebu. Java komandno okruženje koristi JShell API, koji obezbeđuje funkcije za automatsko popunjavanje ili za izračunavanje izraza i isečaka koda. Jedno čitavo poglavlje je posvećeno razmatranju detalja Java komandnog okruženja da bi programeri mogli najbolje da ga iskoriste.

PREUZIMANJE KONTROLE NAD SPOLJNIM PROCESIMA

Kreiranje Java procesa i upravljanje ulazom/izlazom procesa pre izdanja JDK 9 su mogli da se izvrše na dva načina. Prvi način je korišćenje metoda `Runtime.getRuntime.exec()`, koji je omogućavao da izvršimo komandu u zasebnom OS procesu i da kreiramo `java.lang.Process` instancu koja je obezbeđivala određene operacije za upravljanje spoljnim procesom. Drugi način je korišćenje nove klase `java.lang.ProcessBuilder`, sa još nekim poboljšanjima koja se odnose na komunikaciju sa spoljnim procesom i na kreiranje instance `java.lang.Process` koja predstavlja spoljni proces. Oba mehanizma nisu fleksibilna, niti prenosiva, jer su komande koje su izvršavali spoljni procesi u velikoj meri zavisile od operativnog sistema (i bili su neophodni dodatni napori da bi bio izvršen prenos određenih operacija na više operativnih sistema). Jedno poglavlje je posvećeno novom interfejsu API za obradu procesa – u njemu ćete naučiti kreiranje i upravljanje spoljnim procesima na mnogo jednostavniji način.

POBOLJŠANJE PERFORMANSI POMOĆU SAKUPLJAČA „SMEĆA“ G1

Sakupljač „smeća“ G1 je već uveden u JDK 7 i sada je podrazumevano omogućen u izdanju JDK 9. Namenjen je za sisteme sa više jezgara za obradu i mnogo memorije. Koje su prednosti sakupljača „smeća“ G1 u odnosu na starije tipove sakupljača „smeća“, kako su postignuta ova poboljšanja, da li postoji potreba za ručnim podešavanjem i u kojim slučajevima i još nekoliko pitanja u vezi sa sakupljačem „smeća“ G1 razmotrićemo u posebnom poglavlju.

MERENJE PERFORMANSI POMOĆU JMH-A

U mnogim slučajevima može doći do smanjenja performansi u Java aplikacijama. Problem otežava nedostatak testova koji mogu obezbediti makar minimalnu garanciju da su ispunjeni zahtevi za performanse i da se performanse određenih funkcija neće pogoršavati tokom vremena. Merenje performansi Java aplikacija nije jednostavno, posebno zbog velikog broja optimizacija za kompajliranje i izvršenje, koje mogu da utiču na statistiku o performansama. Zbog toga, treba preduzeti dodatne mere, kao što su faze „zagrevanja“ i drugi trikovi, da bi bila obezbeđena preciznija merenja performansi. Java Microbenchmark Harness je radni okvir koji sadrži niz tehnika i odgovarajući interfejs API, koji se mogu koristiti za merenje performansi. JMH nije nova alatka, ali je uključena u distribuciju Java 9. Ako još niste dodali JMH u okvir sa alatkama, detaljno pročitajte poglavlje o upotrebi JMH-a za razvoj Java 9 aplikacija.

PRIPREMA ZA STANDARD HTTP 2.0

HTTP 2.0 je naslednik protokola HTTP 1.1. U ovoj novoj verziji protokola uklonjena su ograničenja i nedostaci iz prethodne verzije. HTTP 2.0 poboljšava performanse na više načina i obezbeđuje razne mogućnosti, kao što su multipleksiranje zahteva/odgovora u jednoj TCP vezi, „potisno" slanje odgovora na server, kontrola protoka, određivanje prioriteta zahteva i tako dalje.

Java obezbeđuje uslužni program `java.net.HttpURLConnection` koji može da se koristi za uspostavljanje nebezbedne HTTP 1.1 veze. Međutim, smatralo se da je interfejs API težak za održavanje i da u njemu nije moguće dodati podršku za HTTP 2.0, pa je predstavljen potpuno novi klijentski interfejs API za uspostavljanje veze pomoću protokola HTTP 2.0 ili protokola Web Socket. Novi klijent HTTP 2.0 i mogućnosti koje on obezbeđuje će biti razmatrani u jednom od narednih poglavlja.

UMETANJE REAKTIVNOG PROGRAMIRANJA

Reaktivno programiranje je paradigma koja se koristi za opis određenog obrasca za propagaciju promena u sistemu. Reaktivnost nije ugrađena u samoj Javi, već se reaktivni tokovi podataka mogu uspostaviti pomoću nezavisnih biblioteka, kao što su RxJava ili projekat Reactor (koji je deo okruženja Spring Framework). U JDK-u 9 se, takođe, reguliše potreba za interfejsom API koji pomaže u razvoju aplikacija visokog odziva zasnovanog na reaktivnim tokovima, koji obezbeđuju uslužni program `java.util.concurrent`. Klasa `Flow` i druge izmene koje su predstavljene u JDK-u 9 biće razmatrane u posebnom poglavlju.

PROŠIRENJE „LISTE ŽELJA"

Pored svih novih funkcija u JDK-u 9, u budućim izdanjima platforme očekuje se čitav skup novih funkcija, među kojima su i sledeće:

- **podrška za upotrebu primitivnih tipova za generičke parametre** - Ovo je jedna od funkcija koja je planirana za JDK 10 u okviru projekta Valhalla. Druga poboljšanja jezika, kao što su hendlovi vrednosti, već su deo izdanja Java 9 i biće predstavljena kasnije u ovoj knjizi.
- **apstrakcija generičkih tipova** - Ovo je jedan od najvažnijih delova projekta Valhalla, u kome treba da se obezbedi mogućnost očuvanja generičkih tipova prilikom izvršenja. Ciljevi u vezi sa očuvanjem generičkih tipova su sledeći:
    - Cilj je da se pomoću modula Modul Foreign Functional Interface uvede novi interfejs API za pozivanje izvornih funkcija i upravljanje njima. Pomoću ovog novog API-ja će biti uklonjeni neki nedostaci radnog okvira JNI, a

programerima aplikacija će posebno biti olakšano korišćenje ovog radnog okvira. Ovaj modul je razvijen u okviru projekta Panama u JDK ekosistemu.

- Novi „Money and Currency“ API (koji je razvijen u okviru API-ja JSR 354) prvobitno je planiran za Javu 9, ali je odložen.
- Novi lagani JSON API (koji je razvijen u okviru API-ja JSR 353) takođe je bio planiran za Javu 9, ali će biti uveden u Javu 10.

Ovo su samo neki od novih koncepata koji se mogu očekivati u narednim izdanjima JDK-a. Cilj je da se pomoću projekta Penrose premosti jaz između modularnog sistema u Javi i modularnog sistema OSGi i da se obezbede različite metodologije za interoperabilnost između dva sistema.

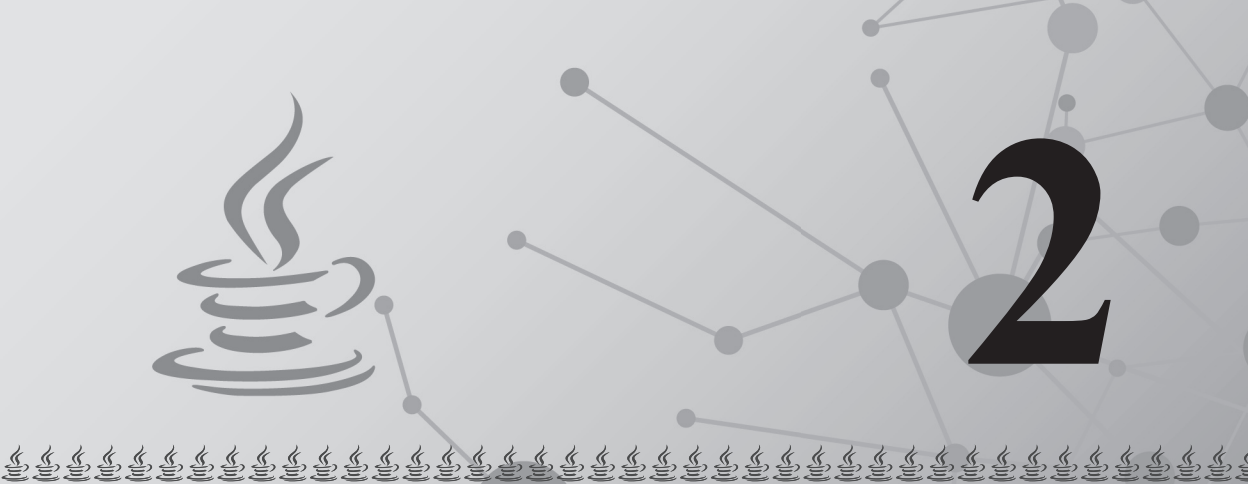
Graal VM je još jedan zanimljiv istraživački projekat koji je potencijalni „kandidat“ za sledeća izdanja Java platforme. Cilj je da se pomoću ovog projekta omoguće radne performanse Java dinamičkih jezika, kao što su JavaScript ili Ruby.

Sve ove koncepte ćemo detaljno razmatrati u poglavlju koje je posvećeno budućem razvoju JDK-a.

REZIME

U ovom kratkom uvodnom poglavlju otkrili smo mali „univerzum“ mogućnosti koje obezbeđuje JDK 9. Modularni sistem koji je predstavljen u ovom izdanju platforme je nesumnjivo osnova za razvoj Java aplikacija. Takođe smo otkrili da je u izdanju JDK 9 uvedeno i niz drugih značajnih funkcija i promena koje zaslužuju posebnu pažnju, pa će biti detaljno razmatrane u narednim poglavljima.

U sledećem poglavlju ćete videti 26 unutrašnjih izmena na Java platformi.



2

Otkrivanje Jave 9

Java 9 predstavlja glavno izdanje i sadrži veliki broj unutrašnjih izmena na Java platformi. Sve u svemu, ove izmene predstavljaju ogroman skup novih mogućnosti za Java programere. Neke od njih su zahtevali programeri, a za druge je inspiracija pronađena u poboljšanjima kompanije „Oracle“. U ovom poglavlju ćete videti 26 najvažnijih izmena - svaka je povezana sa poboljšanjima JDK Enchacement Proposal (JEP). JEP-ovi su indeksirani i smešteni na stranici `openjdk.java.net/jeps/0`, koju možete da posetite da biste pronašli dodatne informacije o svakom JEP-u.

JEP program je deo podrške kompanije „Oracle“ za otvoreni kod, otvorene inovacije i otvorene standarde. Postoje i drugi Java projekti, ali OpenJDK je jedini projekat koji je podržala ta kompanija.

U ovom poglavlju ćemo razmatrati izmene na Java platformi, koje imaju nekoliko impresivnih implikacija, uključujući:

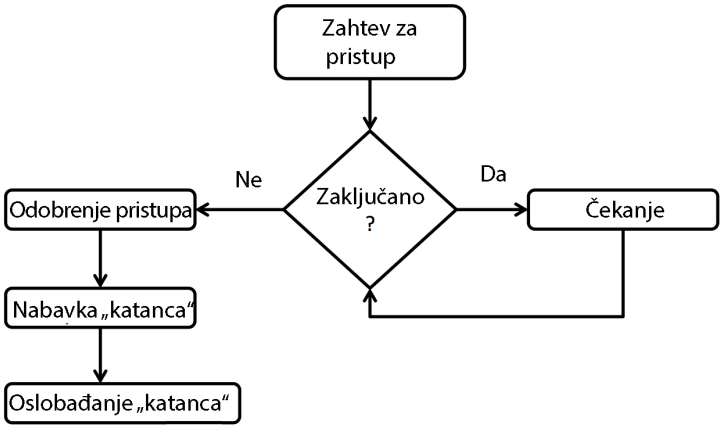
- efikasnost hip prostora
- dodeljivanje memorije
- poboljšanja procesa kompajliranja
- testiranje tipa
- anotacije
- automatizovana testiranja kompajlera prilikom izvršenja
- poboljšano sakupljanje „smeća“

POBOLJŠANO KONKURENTNO ZAKLJUČAVANJE (CONTENDED LOCKING) [JEP 143]

JVM koristi hip prostor za skladištenje klasa i objekata i dodeljuje mu memoriju kad god kreiramo objekat. Ovo olakšava sakupljanje „smeća“ u Javi, pa se oslobađa memorija koja je prethodno korišćena za skladištenje objekata na koje se više ne ukazuje. Java Stack memorija je malo drugačija i obično je mnogo manja od hip memorije.

JVM dobro upravlja oblastima podataka koje dele više programskih niti. Povezuje monitor sa svakim objektom i klasom, a ovi monitori imaju „katance“ (locks), koji su kontrolisani pomoću jedne niti svakom trenutku. Ti „katanci“, koje kontroliše JVM, obezbeđuju monitoru objekta kontrolnu nit.

Dakle, šta je konkurentno zaključavanje? Kada je nit u redu za objekat koji je trenutno zaključan, to znači da je ona „kandidat“ za zaključavanje. Na sledećem dijagramu je prikazano ovo konkurentno zaključavanje na visokom nivou:



Kao što možete da vidite na prethodnoj slici, ni jedna nit koja je u redu ne može da koristi zaključani objekat dok ne bude oslobođen.

Poboljšanja

Opšti cilj je da se pomoću poboljšanja JEP 143 povećaju ukupne performanse JVM upravljanja konkurentnim objektima u odnosu na zaključane Java monitore objekta. Poboljšanja za konkurentno zaključavanje su izvršena unutar JVM-a, pa programeri ne treba da preduzimaju dodatne radnje da bi ih iskoristili. Opšta poboljšanja se odnose na brže operacije, uključujući:

- brži ulaz u monitor
- brži izlaz iz monitora
- brže dobijanje obaveštenja

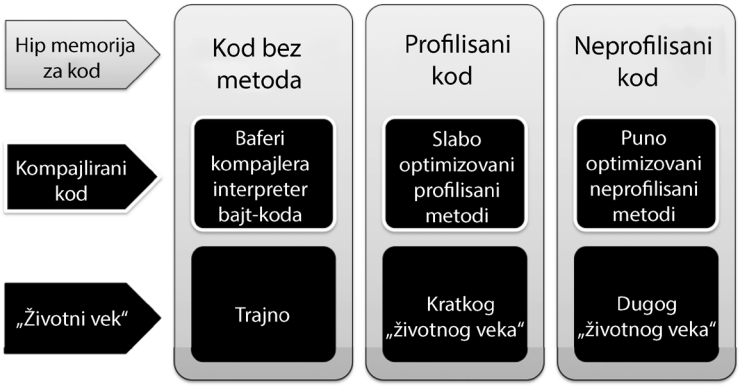
Obaveštenja predstavljaju operacije `notify()` i `notifyAll()`, koje se pozivaju kada je promenjen status zaključanog objekta. Testiranje ovog poboljšanja nije nešto što ćete lako izvršiti. Veća efikasnost na svim niovima je dobrodošla, pa će nam ovo poboljšanje biti od koristi, bez obzira na testiranje koje nije lako pratiti.

SEGMENTIRANA KEŠ MEMORIJA ZA KOD (CODE CACHE) [JEP 197]

Završena je nova verzija segmentirane keš memorije za kod JEP (197) i obezbeđeno je brže i efikasnije izvršavanje. Suština ove promene je podela keš memorije za kod na tri različita segmenta – bez metoda, profilisani i neprofilisani.

Keš memorija za kod je oblast memorije u kojoj Java virtuelna mašina skladišti generisani izvorni kod.

Svaki od segmenata keš memorije za kod koji smo prethodno pomenuli će sadržati određeni tip kompajliranog koda. Kao što možete da vidite na sledećem dijagramu, hip oblasti za kod su segmentirane prema tipu kompajliranog koda:



Dodeljivanje memorije

Hip memorija koja sadrži kod bez metoda je namenjena JVM internom kodu i sastoji se od 3 MB fiksnog bloka memorije. Ostatak keš memorije za kod je podjednako dodeljen profilisanim i neprofilisanim segmentima koda.

Sledeća komanda se može koristiti za definisanje veličine hip memorije koda za kompajlirani kod bez metoda:

```
-XX:NonMethodCodeCodeHeapSize
```

Sledeća komanda se može koristiti za definisanje veličine hip memorije koda za profilisane kompajlirane metode:

```
-XX:ProfiledCodeHeapSize
```

Sledeća komanda se može koristiti za definisanje veličine hip memorije koda za neprofilisane kompajlirane metode:

```
-XX:NonProfiledCodeHeapSize
```

Ova Java 9 funkcija svakako poboljšava efikasnost Java aplikacija. Takođe utiče na druge procese koji koriste keš memoriju.

ALATKA SMART JAVA COMPILATION - FAZA DVA [JEP 199]

Predlog JDK Enhancement Proposal 199 treba da poboljša proces kompajliranja koda. Svi Java programeri će upoznati alatku **javac** za kompajliranje izvornog koda u bajt-kod, koji JVM koristi za pokretanje Java programa. Alatka **Smart Java Compilation**, koja se naziva i Smart Javac i **sjavac**, dodaje „*pametni*“ omotač oko procesa javac. Ipak, osnovno poboljšanje u alatki sjavac se odnosi samo na kompajliranje obaveznog koda. Obavezni kod je u ovom slučaju kod koji je izmenjen od poslednjeg ciklusa kompajliranja.

Ovo poboljšanje možda neće biti od koristi programerima koji izrađuju samo male projekte. Međutim, razmotrite ogromno povećanje efikasnosti kada stalno morate ponovo da kompajlirate kod za srednje i velike projekte. Vreme koje će programeri uštedeti je dovoljan razlog da poboljšanje JEP 199 bude prihvaćeno.

Da li će ovo poboljšanje promeniti način kompajliranja koda? Verovatno neće, bar ne još uvek. Javac će ostati podrazumevani kompajler. Iako alatka sjavac obezbeđuje efikasnost prilikom inkrementnog pravljenja izvršne verzije programa, kompanija „Oracle“ smatra da ona nije dovoljno stabilna da bi postala deo standardnog kompajliranja.

Više informacija o alatki smar javac wrapper možete da pročitate na adresi: http://cr.openjdk.java.net/~briangoetz/JDK-8030245/webrev/src/share/classes/com/sun/tools/sjavac/Main.java-.html

RAZREŠAVANJE LINT I DOCLINT UPOZORENJA [JEP 212]

Nemojte da brinete ako ne poznajete Lint ili Doclint u Javi. Kao što možete da zaključite iz naslova odeljka, oni su izvori koji generišu upozorenja za javac. Sada ćemo pogledati ova upozorenja:

- **Lint** analizira bajt-kod i izvorni kod za javac. Namenjen je da identifikuje bezbednosne propuste u kodu koji se analizira. Takođe obezbeđuje uvid u skalabilnost i probleme u vezi sa zaključavanjem niti. Lint sadrži još mnogo korisnih funkcija, a opšta namena mu je da programerima uštedi vreme.

Više informacija o Lintu možete da pročitate na adresi: https://en.wikipedia.org/wiki/Lint_(software).

- **Doclint** je sličan Lintu i specifičan je za javadoc. Lint i Doclint generišu greške i upozorenja tokom procesa kompajliranja. U poboljšanju JEP 212 naglasak je stavljen na rezoluciju ovih upozorenja. Kada koristite opšte biblioteke, ne bi trebalo da se pojavljuje upozorenje. Da bi bio rešen problem sa pojavljivanjem upozorenja, nastao je JEP 212, koji je razrešen i implementiran u Javu 9.

Kompletnu listu Lint i Doclint upozorenja možete da pregledate na sajtu JDK Bug System https://bugs.openjdk.java.net.

Slojevita atribucija (tiered attribution) za javac [JEP 215]

JEP 215 predstavlja impresivan poduhvat za pojednostavljenje šeme za proveru tipa kompajlera javac. Prvo ćemo videti kako provera tipa funkcioniše u Javi 8, a onda ćemo istražiti promene u Javi 9.

U Javi 8 se provera tipa polinoma izvršava pomoću alatke za **spekulativnu atribuciju** (speculative attribution).

Spekulativna atribucija je metod provere tipa u okviru javac procesa kompajliranja. Ova provera tipa značajno troši dodatne resurse.

Spekulativna atribucija za proveru tipa je precizna, ali nije efikasna. Ove provere obuhvataju poziciju argumenta i eksponencijalno su sporije kada se provera tipa obavlja tokom izvršavanja rekurzije, polimorfizma, ugnežđenih petlji i lambda izraza. Dakle, cilj je da se pomoću poboljšanja JEP 215 promeni šema za proveru tipa, radi bržeg dobijanja rezultata. Rezultati koji su dobijeni pomoću spekulativne atribucije su tačni, ali se ne generišu brzo.

U novom pristupu, koji je uveden u Javu 9, koristi se alatka za slojevitu atribuciju. Ona implementira slojeviti pristup za proveru tipa izraza argumenata za sve pozive metoda. Takođe je omogućeno redefinisanje metoda. Da bi ova šema mogla da funkcioniše, kreirani su novi strukturalni tipovi za svaki od navedenih tipova argumenata metoda:

- lambda izrazi
- polinomi
- regularni pozivi metoda
- reference metoda
- izrazi za kreiranje instanci romboida (diamond)

Izmene kompajlera javac u poboljšanju JEP 215 su složenije od izmena koje su navedene u ovom odeljku. Programeri neće imati neke direktne koristi od njih, osim efikasnijeg kompajlera javac i uštede vremena.

ALATKA ANNOTATIONS PIPELINE 2.0 [JEP 217]

Java anotacije se odnose na posebnu vrstu metapodataka koji se nalazi unutar Java izvornih datoteka. Kompajler javac ne uklanja ove datoteke, pa mogu biti dostupne JVM-u tokom izvršenja.

Anotacije su slične referencama JavaDocs, jer počinju simbolom @. Postoje tri tipa anotacija:

- Najosnovniji oblik anotacije je **marker**. Ovo su samostalne anotacije sa jednom komponentom koja se naziva animation. Evo primera:

```
@thisIsAMarkerAnnotation
public double computeSometing(double x, double y)
{
  // do something and return a double
}
```

- Drugi tip je anotacija koja sadrži *jednu vrednost* ili deo podataka. Kao što možete da vidite u sledećem kodu, anotaciju koja počinje simbolom @ prate zagrade koje sadrže podatke:

```
@thisIsAMarkerAnnotation (data="compute x and y
 coordinates")
public double computeSometing(double x, double y)
{
  // do something and return a double
}
```

  Kodiranje tipa anotacije jedne vrednosti možete da izvršite i tako što ćete izostaviti komponentu `data=`, kao što je prikazano u sledećem kodu:

```
@thisIsAMarkerAnnotation ("compute x and y coordinates")
public double computeSometing(double x, double y)
{
  // do something and return a double
}
```

- Treći tip je anotacija koja sadrži *više komponenata podataka*. Kada koristite ovaj tip anotacije, ne možete da izostavite komponentu `data=`:

```
@thisIsAMarkerAnnotation (data="compute x and y
 coordinates", purpose="determine intersecting point")
 public double computeSometing(double x, double y)
 {
   // do something and return a double
 }
```

Dakle, šta se promenilo u Javi 9? Da bismo odgovorili na ovo pitanje, treba da se prisetimo nekih izmena uvedenih u Javu 8 koje su uticale na Java anotacije:

- lambda izrazi
- ponovljene anotacije
- Java anotacije

Ove promene u vezi sa Javom 8 uticale su na Java anotacije, ali nisu uticale na način kako ih kompajler javac obrađuje. Postojala su neka kodirana rešenja koja su omogućavala da javac upravlja novim anotacijama, ali ona nisu bila efikasna. Štaviše, ovu vrstu kodiranja (kodiranih zaobilaznih rešenja) je teško održavati.

Prema tome, u JEP-u 217 naglasak je stavljen na refaktorisanje javac protočne obrade anotacija. Ono je izvršeno unutar kompajlera javac, pa programerima nije vidljivo.

NOVA ŠEMA STRINGA VERZIJE (VERSION-STRING) [JEP 223]

Pre pojave Jave 9 brojevi izdanja nisu pratili industrijsku standardnu verziju **Semantic Versioning.** Na primer, za vreme pisanja ove knjige postojala su tri najnovija JDK izdanja:

- JDK 8 verzija 131
- JDK 8 verzija 121
- JDK 8 verzija 112

Standard Semantic Versioning koristi šemu Major.Minor.Patch (0.0.0):

- **Major** označava nove izmene u interfejsu API koje nisu kompatibilne sa starijim verzijama.
- **Minor** označava dodavanje funkcija koje su kompatibilne sa starijim verzijama.
- **Patch** se odnosi na ispravke grešaka ili manje izmene koje su kompatibilne sa starijim verzijama.

Kompanija „Oracle“ je prihvatila standard Semantic Versioning za Javu 9 i novija izdanja. Šema **major-minor-security** će se koristiti za prva tri elementa brojeva verzije Jave:

- **Major** - glavno izdanje koje se sastoji od značajnog novog skupa funkcija
- **Minor** - revizije i ispravke grešaka koje su kompatibilne sa starijim verzijama
- **Security** - ispravke koje su važne za poboljšanje bezbednosti

Možda vam na osnovu opisa JEP-a 223 šema za kreiranje izgleda jednostavna. Naprotiv, razvijen je niz veoma detaljnih pravila i tehnika za upravljanje brojevima budućih verzija. Da biste uvideli ovu složenost, pogledajte sledeći primer:

```
1.9.0._32.b19
```

AUTOMATSKO GENERISANJE TESTIRANJA KOMPAJLERA TOKOM IZVRŠENJA [JEP 233]

Java je, verovatno, programski jezik koji se najčešće koristi i koji je smešten na više različitih platformi, što otežava problem pokretanja testiranja ciljnog kompajlera na efikasan način. Svrha JEP-a 233 je kreiranje alatki koje bi mogle da automatizuju testiranja kompajlera tokom izvršenja.

Alatka koja je kreirana prvo generiše nasumični skup Java izvornog koda i/ili bajt-koda. Generisani kod će imati tri važne karakteristike nakon što ih ta alatka generiše:

- Biće sintaktički ispravan.
- Biće semantički ispravan.

Koristiće slučajnu početnu vrednost koja omogućava ponovnu upotrebu istog nasumično generisanog koda.

Kod koji je nasumično generisan će biti automatski sačuvan u sledećem direktorijumu:

```
hotspot/test/testlibrary/jit-tester
```

Ovi testovi će biti uskladišteni za ponovnu upotrebu. Mogu da se izvrše iz direktorijuma j-treg ili datoteke alatke „makefile“. Ponovno izvršavanje sačuvanih testova može da bude korisno prilikom testiranja stabilnosti vašeg sistema.

TESTIRANJE ATRIBUTA DATOTEKE KLASE KOJE JE GENERISAO JAVAC [JEP 235]

Nedostatak kreiranja testova ili nedovoljna mogućnost kreiranja testova za atribute datoteke klase bili su podsticaj da se izradi JEP 235. Cilj je bio da se omogući da javac pravilno i u potpunosti kreira atribute datoteke klase. Prema tome, iako neki atributi nisu upotrebljeni u datoteci klase, sve datoteke klase treba generisati sa kompletnim skupom atributa. Takođe mora postojati testiranje da se utvrdi da li su datoteke klase kreirane pravilno u odnosu na atribute datoteke.

Pre Jave 9 nije postojao metod za testiranje atributa datoteke klase. Pokretanje klase i testiranje koda radi očekivanih rezultata je bio metod koji je najčešće korišćen za testiranje javac generisanih datoteka klasa. Ovom tehnikom se nije mogla proveriti valjanost atributa datoteke.

Postoje tri kategorije atributa datoteke klase - atributi koje koristi JVM, opcioni atributi i atributi koje ne koristi JVM.

Atributi koje koristi JVM su sledeći:

- `BootstrapMethods`
- `Code`
- `ConstantValue`
- `Exceptions`
- `StackMapTable`

Ovo su opcioni atributi:

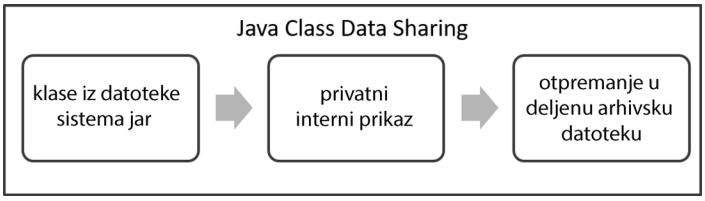
- `Deprecated`
- `LineNumberTable`
- `LocalVariableTable`
- `LocalVariableTypeTable`
- `SourceDebugExtension`
- `SourceFile`

Atributi koje ne koristi JVM su sledeći:

- `AnnotationDefault`
- `EnclosingMethod`
- `InnerClasses`
- `MethodParameters`
- `RuntimeInvisibleAnnotations`
- `RuntimeInvisibleParameterAnnotations`
- `RuntimeInvisibleTypeAnnotations`
- `RuntimeVisibleAnnotations`
- `RuntimeVisibleParameterAnnotations`
- `RuntimeVisibleTypeAnnotations`
- `Signature`
- `Synthetic`

SKLADIŠTENJE INTERNIH STRINGOVA U CDS ARHIVU [JEP 250]

Skladištenje stringova i pristupanje njima u arhivi **Class Data Sharing (CDS)** nije efikasno - oduzima previše vremena i troši memoriju. Na sledećem dijagramu je prikazano kako Java skladišti interne stringove u CDS arhivu:



Neefikasnost potiče iz trenutne šeme skladištenja. Stavke `CONSTANT_String`, koje se nalaze u skladištu Constant Pool, prikazuju se pomoću kodnog rasporeda UTF-8, posebno kada alatka **Class Data Sharing** otprema klase u deljene arhivske datoteke.

UTF-8 je 8-bitni standardni kodni raspored promenljive dužine.

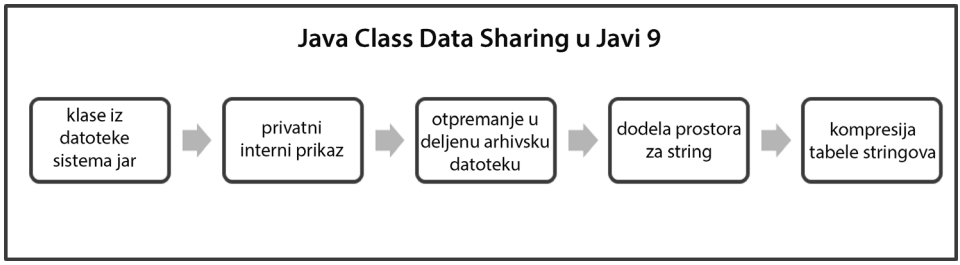
Problem

Kada se koristi aktuelni kodni raspored UTF-8, stringovi moraju biti konvertovani u objekte stringa - instance klase `java.lang.String`. Ova konverzija se odvija na zahtev, pa može usporiti sistem i nepotrebno trošiti memoriju. Vreme obrade je izuzetno kratko, ali se ne može zanemariti potrošnja memorije. Za svaki znak u internom stringu potrebno je najmanje tri bajta memorije.

Problem u vezi sa kodnim rasporedom UTF-8 je u tome što uskladišteni nizovi nisu dostupni svim JVM procesima.

Rešenje

CDS arhive sada dodeljuju određeni prostor hip memoriji za stringove:



Prostor za stringove se mapira pomoću tabele zajedničkih stringova (Shared String Table), heš tabele i deduplikacije.

Deduplikacija je tehnika kompresije podataka koja eliminiše duplikate informacija iz arhive.

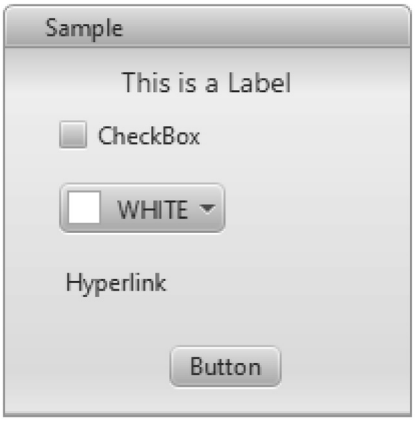
PRIPREMA KONTROLA KORISNIČKOG INTERFEJSA JAVAFX I CSS API-JA ZA MODULARIZACIJU [JEP 253]

JavaFX je skup paketa koji omogućavaju dizajniranje i razvoj multimedijalnog grafičkog korisničkog interfejsa. JavaFX aplikacije obezbeđuju programerima odličan interfejs API za kreiranje konzistentnog interfejsa za aplikacije. Stilovi **Cascading Style Sheets (CSS)** se mogu koristiti za prilagođavanje interfejsa. Jedna od odličnih karakteristika JavaFX-a je što je omogućeno da se zadaci programiranja i dizajniranja interfejsa lako odvajaju.

Pregled JavaFX-a

Postoji odlična alatka za vizuelno skriptovanje Scene Builder, koja omogućava da kreirate grafičke korisničke interfejse pomoću tehnike „prevuci i otpusti“ i postavki svojstava. Scene Builder generiše potrebne FXML datoteke koje koristi **integrisano razvojno okruženje (IDE)**, kao što je NetBeans.

bEvo primera korisničkog interfejsa koji je kreiran pomoću alatke Scene Builder:



A ovo je FXML datoteka koja je kreirana pomoću alatke Scene Builder:

```
        <?xml version="1.0" encoding="UTF-8"?>

        <?import java.lang.*?>
        <?import java.util.*?>
        <?import javafx.scene.control.*?>
        <?import javafx.scene.layout.*?>
        <?import javafx.scene.paint.*?>
        <?import javafx.scene.text.*?>

        <AnchorPane id="AnchorPane" maxHeight="-Infinity"
         maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0"
xmlns:fx="http://javafx.com/fxml/1"
xmlns="http://javafx.com/javafx/2.2">
          <children>
            <TitledPane animated="false" collapsible="false"
             layoutX="108.0" layoutY="49.0" text="Sample">
            <content>
              <AnchorPane id="Content" minHeight="0.0" minWidth="0.0"
               prefHeight="180.0" prefWidth="200.0">
              <children>
                <CheckBox layoutX="26.0" layoutY="33.0"
                 mnemonicParsing="false" prefWidth="94.0"
                 text="CheckBox" />
                <ColorPicker layoutX="26.0" layoutY="65.0" />
                <Hyperlink layoutX="26.0" layoutY="103.0"
                 text="Hyperlink" />
```

```
        <Label alignment="CENTER" layoutX="14.0" layoutY="5.0"
         prefWidth="172.0" text="This is a Label"
         textAlignment="CENTER">
         <font>
           <Font size="14.0" />
         </font>
        </Label>
        <Button layoutX="81.0" layoutY="146.0"
         mnemonicParsing="false" text="Button" />
      </children>
      </AnchorPane>
    </content>
    </TitledPane>
  </children>
</AnchorPane>
```

Implikacije za Javu 9

JavaFX kontrole i CSS funkcije pre Jave 9 bile su dostupne programerima samo povezivanjem sa unutrašnjim API-jima. Modularizacijom Jave 9 interni API-ji su postali nedostupni. Stoga je izrađen JEP 253 da bi bili definisani javni, umesto internih API-ja.

Ovo je bio veći poduhvat nego što se čini. Evo nekoliko radnji koje su preduzete u okviru ovog JEP-a:

- premeštanje JavaFX maski (skins) kontrole iz internog API-ja u javni:

  (`javafx.scene.skin`)
- obezbeđivanje konzistentnosti API-ja
- generisanje detaljnog javadoca
- Sledeće klase su premeštene iz internih paketa u javni paket `javafx.scene.control.skin`:

AccordionSkin	ButtonBarSkin	ButtonSkin	CellSkinBase
CheckBoxSkin	ChoiceBoxSkin	ColorPickerSkin	ComboBoxBaseSkin
ComboBoxListViewSkin	ComboBoxPopupControl	ContextMenuSkin	DateCellSkin
DatePickerSkin	HyperLinkSkin	LabelSkin	LabeledSkinBase
ListCellSkin	ListViewSkin	MenuBarSkin	MenuButtonSkin
MenuButtonSkinbase	NestedTableColumHeader	PaginationSkin	ProgressBarSkin
ProgressIndicatorSkin	RadioButtonSkin	ScrollBarSkin	ScrollPaneSkin
SeparatorSkin	SliderSkin	SpinnerSkin	SplitMenuButtonSkin

SplitPaneSkin	TabPaneSkin	TableCellSkin	TableCellSkinBase
TableColumnHeader	TableHeaderRow	TableHeaderSkin	TableRowSkinBase
TableViewSkin	TableViewSkinBase	TextAreaSkin	TextFieldSkin
TextInputControlSkin	TitledPaneSkin	ToggleButtonSkin	TooBarSkin
TooltipSkin	TreeCellSkin	TreeTableCellSkin	TreeTableRowSkin
TreeTableViewSkin	TreeViewSkin	VirtualContainerBase	VirtualFlow

Javni paket `javafx.css` sada ima dodatne klase:

- `CascadingStyle.java:public class CascadingStyle implements`
- `Comparable<CascadingStyle>`
- `CompoundSelector.java:final public class CompoundSelector extends Selector`
- `CssError.java:public class CssError`
- `Declaration.java:final public class Declaration`
- `Rule.java:final public class Rule`
- `Selector.java:abstract public class Selector`
- `SimpleSelector.java:final public class SimpleSelector extends Selector`
- `Size.java:final public class Size`
- `Style.java:final public class Style`
- `Stylesheet.java:public class Stylesheet`
- `CssParser.java:final public class CssParser`

KOMPAKTNI STRINGOVI [JEP 254]

Tip podataka stringa je važan deo skoro svake Java aplikacije. Iako je cilj da se pomoću JEP-a 254 kreiraju stringovi koji ne zauzimaju mnogo prostora, kreiranju ovakvih stringova se pristupilo oprezno da ne bi došlo do negativnog uticaja na postojeće performanse i kompatibilnost.

Status pre Jave 9

Pre Jave 9 podaci stringa su skladišteni kao niz znakova. Za svaki znak je bilo potrebno 16 bitova. Utvrđeno je da je za većinu objekata String potrebno samo 8 bitova ili 1 bajt, zato što što se najveći broj stringova sastoji od znakova Latin-1.

ISO Latin-1 Character Set je jednobajtni skup kodiranja znakova.

Novo u Javi 9

U Javi 9 stringovi su sada interno prikazani pomoću niza bajtova i polja flegova za reference kodiranja.

SPAJANJE IZABRANIH AŽURIRANJA XERCES 2.11.0 SA JAXP-OM [JEP 255]

Xerces je bioblioteka koja se koristi za raščlanjavanje XML-a u Javi. Ova biblioteka je nadgrađena na verziju 2.11.0 krajem 2010. godine, pa je cilj bio da se pomoću JEP-a 255 ažurira JAXP da bi promene bile integrisane u Xerces 2.11.0.

JAXP je Java API za obradu XML-a.

Pre pojave Jave 9 najnovije ažuriranje JDK-a u vezi sa XML procesom se zasnivalo na raščlanjavaču Xerces 2.7.1. U JDK-u 7 su izvršene dodatne izmene koje se zasnivaju na Xercesu 2.10.0. JEP 255 je još jedno poboljšanje JAXP-a koje se zasniva na Xercesu 2.11.0.

Xerces 2.11.0 podržava sledeće standarde:

- XML 1.0, četvrto izdanje
- Namespaces in XML 1.0, drugo izdanje
- XML 1.1, drugo izdanje
- Namespaces in XML 1.1, drugo izdanje
- XML Inclusions 1.0, drugo izdanje

Document Object Model (DOM):

- Level 3
  - Core
  - Load & save
- Level 2
  - Core
  - Events
- Traversal & Range
- Element Traversal, prvo izdanje
- Simple API for XML 2.0.2
- **Java APIs for XML Processing (JAXP)** 1.4
- Streaming API for XML 1.0
- XML Schema 1.0
- XML Schema 1.1
- XML Schema Definition Language

JDK je ažuriran radi dodavanja sledećih Xerces 2.11.0 kategorija:

- Catalog Resolver
- Datatypes
- Document Object Model Level 3
- XML Schema Validation
- XPointer

Javni API za JAXP nije izmenjen u Javi 9.

NADGRADNJA JAVAFX/MEDIA NA NOVIJU VERZIJU GSTREAMER [JEP 257]

JavaFX se koristi za kreiranje desktop i veb aplikacija. On je kreiran da bi zamenio Swing standardnu GUI biblioteku Jave. Klasa `Media`, `javafx.scene.media.Media` koristi se za instanciranje objekta koji predstavlja multimedijalni resurs. JavaFX/Media ukazuje na sledeću klasu:

```
public final class Media extends java.lang.Object
```

Ova klasa obezbeđuje referentne podatke multimedijalnom resursu. Paket `javax.scene.media` omogućava programerima da ugrade multimediju u JavaFX aplikacije. JavaFX/Media koristi GStreamer protočnu obradu.

GStreamer je radni okvir za obradu multimedije. Može da se koristi za izradu sistema u kojima se koristi multimedija nekoliko različitih formata, a nakon obrade se ti formati izvoze u izabrane formate.

Cilj je bio da se pomoću JEP-a 257 nadgradi JavaFX/Media na najnoviju verziju Gstreamera, radi garancije stabilnosti, performansi i bezbednosti.

HarfBuzz Font Layout Engine [JEP 258]

Layout Engine je pre Jave 9 korišćen za pojednostavljenje složenih fontova, posebno za vizuelizaciju fontova koji ne pripadaju uobičajenim Latin fontovima. U Javi je korišćen jedinstveni klijentski interfejs, koji se naziva i ICU, kao alatka za vizuelizaciju teksta. ICU Layout Engine je amortizovan, pa je u Javi 9 zamenjen mehanizmom HarfBuzz Font Layout Engine.

HarfBuzz je **OpenType** mehanizam za vizuelizaciju teksta. Ovaj tip mehanizma za vizuelizaciju obezbeđuje skript usklađen sa kodom da bi bilo olakšano raspoređivanje teksta po želji.

OpenType je HTML specifikacija za format fonta.

Inicijator promene mehanizma ICU Layout Engine u mehanizam HarfBuzz Font Layout Engine bila je kompanija IBM, koja je odlučila da prekine podršku mehanizmu ICU Layout Engine. Prema tome, JDK je ažuriran tako da sadrži mehanizam HarfBuzz Font Layout Engine.

HIDPI GRAFIKA U WINDOWSU I LINUXU [JEP 263]

U JEP-u 263 je stavljen naglasak na obezbeđivanje jasnog prikaza komponenata na ekranu u odnosu na gustinu piksela ekrana. Sledeći termini su važni za ovaj JEP (predstavljeni su zajedno sa opisnim informacijama koje su navedene u nastavku):

- **DPI-aware aplikacija** - aplikacija koja može da detektuje i skalira slike za određenu gustinu piksela ekrana
- **DPI-unaware aplikacija** - aplikacija koja ne detektuje, niti skalira slike za određenu gustinu piksela ekrana
- **HiDPI grafika** - grafika visoke rezolucije u tačkama po inču
- **Retina ekran** - Termin retina je osmislila kompanija „Apple“ da bi označila ekrane sa gustinom piksela od najmanje 300 piksela po inču.

Za prikazivanje grafike korisniku, tj. slika i grafičkih komponenata korisničkog interfejsa, performanse su, obično, od ključnog značaja. Prikazivanje slika u visokom kvalitetu može da bude donekle problematično. DPI se razlikuje, u zavisnosti od računarskih monitora. Postoje tri osnovna pristupa za razvoj aplikacija za ekrane:

- Izradite aplikacije, bez obzira na moguće različite dimenzije ekrana. Drugim rečima, kreirajte DPI-unaware aplikaciju.
- Izradite DPI-aware aplikaciju koja selektivno koristi unapred prikazane veličine slike za određeni prikaz.
- Izradite DPI-aware aplikaciju koja pravilno smanjuje ili povećava slike da bi ih prilagodila određenom ekranu na kome se izvršava aplikacija.

Jasno je da su prva dva pristupa problematična zbog različitih razloga. Za prvi pristup korisničko iskustvo nije potrebno. Naravno, ako se aplikacija kreira za veoma specifičan ekran na kome se ne očekuje promena gustine piksela, ovaj pristup bi mogao biti održiv.

Za drugi pristup je potrebno obaviti mnogo posla kada je reč o dizajniranju i razvoju da bi slike bile programski kreirane i implementirane u skladu sa očekivanom gustinom piksela ekrana. Potrebno je obaviti mnogo posla, a veličina aplikacije će biti nepotrebno povećana i neće biti prilagođena novoj različitoj gustini piksela.

Treći pristup je kreiranje DPI-aware aplikacije sa efikasnim funkcijama skaliranja. Pokazalo se da ovaj pristup funkcioniše dobro na Mac retina ekranima.

Pre pojave Jave 9 automatsko skaliranje i dimenzioniranje već su bili implementirani u Javu za operativni sistem Mac OS X. Ova funkcija je dodata u Javu 9 za operativne sisteme Windows i Linux.

MARLIN MODUL ZA VIZUELIZACIJU GRAFIKE [JEP 265]

Pomoću JEP-a 265 je Pisces za vizuelizaciju grafike zamenjen Marlin modulom za vizuelizaciju grafike u Java 2D API-ju. Ovaj API se koristi za crtanje 2D grafike i animacija.

Cilj je bio da se Pisces zameni modulom za vizuelizaciju koji je mnogo efikasniji i ne smanjuje kvalitet grafike. Taj cilj je ostvaren u Javi 9. Predviđena posredna korist ovog JEP-a 265 je dodavanje API-ja koji je dostupan za programere. Ranije je povezivanje sa interfejsima AWT i Java 2D obavljano interno.

UNICODE 8.0.0 [JEP 267]

Unicode 8.0.0 je objavljen 17. juna 2015. godine. U JEP-u 267 je stavljen naglasak na ažuriranje relevantnih API-ja za podršku standardu Unicode 8.0.0.

Novo u standardu Unicode 8.0.0

U standardu Unicode 8.0.0 je dodato skoro 8.000 znakova. Ovo su važni delovi tog izdanja:

- Ahom skript za tai ahom jezik (Indija)
- Arwi skript za tamilski jezik (arapski)
- znakovi prilagođeni čeroki jeziku
- CJK ujednačeni ideogrami
- simboli emotikona i simboli modifikatora za nijansu boje kože
- znak za gruzijsku valutu lari
- lk jezik (Uganda)
- kulango jezik (Obala Slonovače)

Ažurirane klase u Javi 9

Da bi u potpunosti bile usklađene sa novim standardom Unicode, neke Java klase su ažurirane.

Sledeće navedene klase su ažurirane za Javu 9 u skladu sa novim standardom Unicode:

- `java.awt.font.NumericShaper`
- `java.lang.Character`
- `java.lang.String`
- `java.text.Bidi`
- `java.text.BreakIterator`
- `java.text.Normalizer`

REZERVISANE OBLASTI STEKA ZA KRITIČNE SEKCIJE [JEP 270]

Cilj je bio da se pomoću JEP-a 270 reši problem prelivanja steka tokom izvršavanja kritičnih sekcija. Rešenje je nađeno tako što je rezervisan dodatni prostor za stek niti.

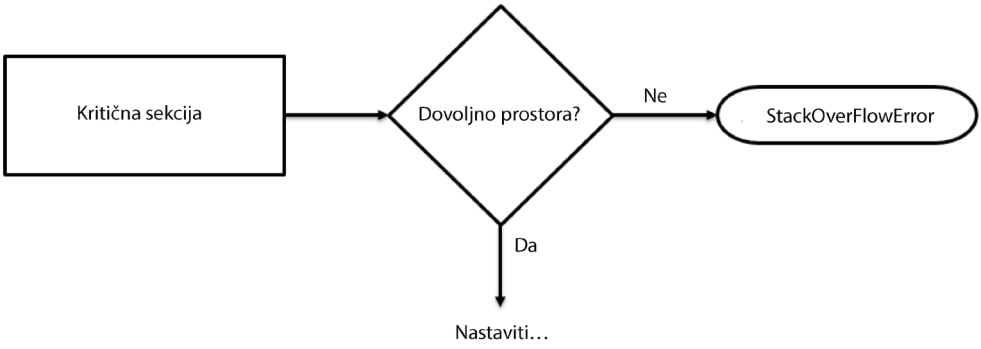
Situacija pre Jave 9

JVM generiše izuzetak `StackOverflowError` kada je potrebno da izvrši izračunavanje podataka u niti koja nema dovoljno prostora, a pri tom nema dozvolu za alokaciju dodatnog prostora. Ovo je asinhroni izuzetak. JVM takođe može sinhrono da generiše izuzetak `StackOverflowError` kada se pokrene neki metod.

Kada se pokrene neki metod, koristi se interni proces za prijavljivanje greške Stack Overflow. Iako je trenutna šema sasvim dobra za prijavljivanje greške, nema dovoljno prostora za pokretanje aplikacije radi jednostavnog oporavka od greške. Ovo je za programere i korisnike veliki problem. Ako je greška `StackOverflowError` generisana tokom važne računske operacije, podaci mogu da budu oštećeni, čime se izazivaju dodatni problemi.

Iako nije jedini uzrok ovih problema, „effected“ status „katanca“ iz klase `ReentrantLock` je čest uzrok neželjenih ishoda. Ovi problemi su bili očigledni u Javi 7, jer je klasa `ReentrantLock` implementirana pomoću koda `ConcurrentHasMap`. Taj kod je modifikovan za Javu 8, ali problemi su se i dalje javljali prilikom svake implementacije klase `ReentrantLock`. Slično se događalo i kada nije korišćena klasa `ReentrantLock`.

Na sledećem dijagramu je dat opšti pregled problema `StackOverflowError`:



U sledećem poglavlju ćete videti kako je ovaj problem rešen u Javi 9.

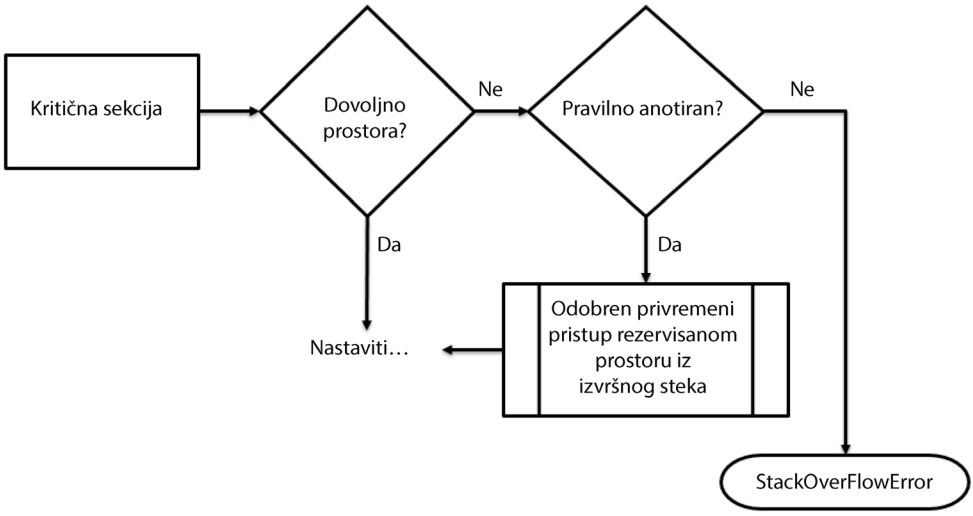
Novo u Javi 9

Zahvaljujući izmenama u JEP-u 270 za Javu 9, kritičnim sekcijama će automatski biti dodeljen dodatni prostor, tako da se mogu izvršiti do kraja, pa se neće pojaviti greška `StackOverflowError`. Ova izmena se zasniva na potrebi za dodatnim prostorom. Da bi bila omogućena ta funkcija, izvršene su potrebne izmene u JVM-u.

U stvari, JVM odlaže grešku `StakOverflowError` ili makar pokušava da je odloži, dok se kritične sekcije ne izvrše. Da bismo iskoristili ovu novu šemu, metodi moraju biti označeni sledećom anotacijom:

```
jdk.internal.vm.annotation.ReservedStackAccess
```

Kada metodi sadrže ovu anotaciju i uslov `StackOverflowError` postoji, biće odobren privremeni pristup rezervisanoj memoriji. Novi proces na visokom nivou apstrakcije je predstavljen na sledeći način:



DINAMIČKO POVEZIVANJE OBJEKTNIH MODELA KOJI SU DEFINISANI JEZIKOM [JEP 276]

Java interoperabilnost je poboljšana pomoću JEP-a 276. Izvršene su potrebne izmene JDK-da bi tokom izvršenja bili omogućeni programi za povezivanje iz više jezika, radi koegzistencije u jednoj JVM instanci. Ove izmene se odnose na operacije na visokom nivou, kao što se i moglo očekivati. Primer važne operacije na visokom nivou je čitanje ili pisanje svojstva pomoću elemenata, kao što su čitači (accessors) i menjači (mutators).

Operacije na visokom nivou se odnose na objekte nepoznatih tipova, koji se mogu pokrenuti pomoću instrukcije `INVOKEDYNAMIC`. Evo primera pozivanja svojstva objekta kada tip objekta nije poznat tokom kompajliranja:

```
INVOKEDYNAMIC "dyn:getProp:age"
```

Dokaz koncepta

Nashorn je lagano JavaScript Runtime okruženje sa visokim peformansma, koje omogućava ugrađivanje JavaScripta u Java aplikacije. On je kreiran za Javu 8 i zamenio je prethodni JavaScript mehanizam za pisanje skriptova, koji je bio zasnovan na Mozilla Rhinou. Nashorn obezbeđuje vezu između operacija na visokom nivou na bilo kom objektu nepoznatog tipa, kao što je `obj.something`, pri čemu se dobija sledeće:

```
INVOKEDYNAMIC "dyn.getProp.something"
```

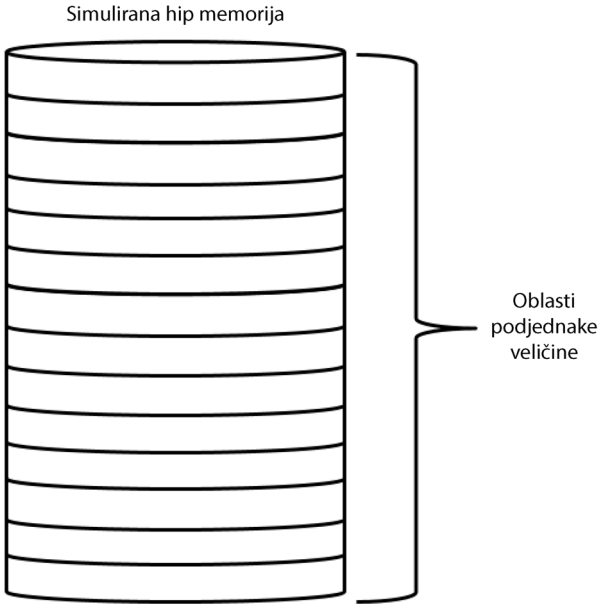
Dinamičko povezivanje se pokreće i obezbeđuje odgovarajuću implementaciju kada je to moguće.

DODATNI TESTOVI ZA OGROMNE OBJEKTE U SAKUPLJAČU „SMEĆA“ G1 [JEP 278]

Jedna od funkcija koja je dugo zastupljena na Java platformi je pozadinsko sakupljanje „smeća“. Cilj je da se pomoću JEP-a 278 kreiraju dodatni WhiteBox testovi za ogromne objekte kao funkcija sakupljača „smeća“ G1.

WhiteBox testiranje je API koji se koristi za pretraživanje JVM detalja. WhiteBox API za testiranje je predstavljen u Javi 7 i nadgrađen u Javu 8 i Javu 9.

Sakupljač „smeća“ G1 funkcioniše i dalje dobro u drugim verzijama Jave, jer su one još uvek upotrebljive, ali bilo je prostora za neka poboljšanja efikasnosti. Način na koji je G1 funkcionisao u starijim izdanjima Jave se zasnivao na deljenju hip memorije na nekoliko oblasti podjednake veličine, kao što je prikazano.



Problem u sakupljaču „smeća“ G1 je bio način na koji se upravljalo ogromnim objektima.

Ogromni objekat u kontekstu sakupljanja „smeća“ predstavlja svaki objekat koji zauzima više oblasti u hip memoriji.

Problem u vezi sa ogromnim objektima je u tome što, ukoliko oni zauzmu bilo koji deo oblasti u hip memoriji, preostali prostor ne može biti dodeljen drugim objektima. U Javi 9 je WhiteBox API proširen sa četiri tipa novih metoda:

- metodi koji su namenjeni da blokiraju kompletno sakupljanje „smeća“ i da iniciraju konkurentno označavanje
- metodi koji mogu da pristupe pojedinačnim oblastima hip memorije sakupljača „smeća“ G1 (pristup ovim oblastima se zasniva na čitanju atributa, kao kod pristupa trenutnom stanju oblasti)

- metodi koji direktno pristupaju internim promenljivim sakupljača „smeća“ G1
- metodi koji mogu da utvrde da li se ogromni objekti nalaze u hip memoriji (ako se nalaze u njoj, utvrđuju u kojoj su oblasti se nalaze)

POBOLJŠANJE OTKLANJANJA GREŠAKA U TESTU [JEP 279]

Programerima koji obavljaju mnogo testiranja biće od koristi da pročitaju predlog JEP 279. U Javu 9 je dodata funkcija koja automatski prikuplja informacije radi podrške otklanjanju grešaka u testu i uklanjanju prekoračenja vremena. Prikupljanje lako dostupnih dijagnostičkih podataka tokom testiranja namenjeno je da programerima i inženjerima obezbedi tačniju evidenciju i druge rezultate.

Postoje dve osnovne vrste informacija u okviru testiranja - informacije o okruženju i informacije o procesu.

Informacije o okruženju

Prilikom pokretanja testova informacije o okruženju za testiranje mogu biti važne za otklanjanje grešaka. Ove informacije obuhvataju sledeće:

- opterećenja procesora
- prostor na disku
- opterećenja ulaza/izlaza
- memorijski prostor
- otvorene datoteke
- otvorene sockete
- procese koji se izvršavaju
- sistemske događaje
- sistemske poruke

Informacije o Java procesu

Postoje i informacije dostupne tokom procesa testiranja koje su direktno povezane sa Java procesima. Ove informacije obuhvataju:

- C stekove
- status radne memorije računara
- minidump datoteke
- statistiku o hip memoriji
- Java stekove

Da biste saznali više o ovom konceptu, pročitajte informacije o JDK alatki Regression Test Harness (jtreg).

OPTIMIZACIJA NADOVEZIVANJA STRINGOVA [JEP 280]

JEP 280 predstavlja zanimljivo poboljšanje za Java platformu. Pre pojave Jave 9 kompajler javac je prevodio nadovezivanje stringova u lanac `StringBuilder :: append`. Ovo je neoptimalna metodologija prevođenja, za koju je često bilo potrebno podešavanje veličine StringBuildera unapred.

Ovim poboljšanjem je izmenjena sekvenca bajt-koda za nadovezivanje stringova, koju je generisao javac, da bi mogli da se koriste pozivi `INVOKEDYNAMIC`. Svrha poboljšanja je da se poveća optimizacija i podrže buduće optimizacije, bez potrebe za reformisanjem javac bajt-koda.

Više informacija o pozivima INVOKEDYNAMIC potražite u JEP-u 276.

Upotreba `INVOKEDYNAMIC` poziva u klasi `java.lang.invoke.StringConcatFactory` omogućava da koristimo metodologiju sličnu lambda izrazima, umesto da koristimo StringBuilderov proces „korak po korak“. Ovo obezbeđuje efikasniju obradu nadovezivanja stringova.

HOTSPOT C++ RADNI OKVIR ZA JEDINIČNO TESTIRANJE [JEP 281]

HotSpot je naziv JVM-a. Ovo Java poboljšanje je namenjeno za podršku razvoju C++ jediničnih testova za JVM. Ovde je prikazana delimična lista poboljšanja (nisu navedena po prioritetu):

- testiranje komandne linije
- kreiranje odgovarajuće dokumentacije
- debagiranje odredišta kompajlera
- elastičnost radnog okvira
- podrška za IDE
- individualno i izolovano jedinično testiranje
- individualni rezultati testova
- integrisanje sa postojećom infrastrukturom
- podrška za interne testove
- pozitivno i negativno testiranje
- kratko vreme izvršavanja testa
- podrška za sve JDK 9 platforme
- odredišta kompajliranja testova
- isključivanje testova
- grupisanje testova
- testiranje za koje je potrebna inicijalizacija JVM-a
- testovi koji dele prostor sa izvornim kodom
- testovi za kod koji je zavisan od platforme
- pisanje i izvršavanje jediničnog testiranja (za klase i metode)

Poboljšanje JEP 281 je dokaz sve veće proširenosti.

OMOGUĆAVANJE ALATKE GTK 3 U LINUXU [JEP 283]

GTK+ je zvanično poznat kao GIMP okvir sa alatkama - predstavlja alatke koje se koriste za kreiranje **grafičkog korisničkog interfejsa (GUI)**. Alatka se sastoji od vidžeta koji su

dostupni pomoću API-ja. Svrha JEP-a 283 je da obezbedi podršku za GTK 2 i GTK 3 u Linuxu prilikom razvijanja Java aplikacija sa grafičkim komponentama. Implementacija podržava Java aplikacije koje koriste pristupe JavaFX, AWT i Swing.

Možemo da kreiramo Java grafičke aplikacije pomoću pristupa JavaFX, AWT i Swing. Ovo je tabela koja sadrži kratak pregled ova tri pristupa, koja se odnose na alatku GTK pre pojave Jave 9:

PRISTUP	NAPOMENA
JavaFX	Koristi dinamičku GTK funkciju lookup. Komunicira sa AWT-om i Swingom pomoću JFXPanela. Koristi AWT funkciju štampanja.
AWT	Koristi dinamičku GTK funkciju lookup.
Swing	Koristi dinamičku GTK funkciju lookup.

Koje promene su neophodne za implementiranje ovog JEP-a? U pristupu JavaFX izmenjene su tri važne stavke:

- Dodato je automatizovano testiranje za GTK 2 i GTK 3.
- Dodata je funkcija za GTK 2 dinamičko učitavanje.
- Dodata je podrška za GTK 3.

U pristupu AWT i Swing implementirane su sledeće izmene:

- Dodato je automatizovano testiranje za GTK 2 i GTK 3.
- `AwtRobot` je premešten u GTK 3.
- Ažuriran je dijaloški okvir `FileChooserDialog` za GTK 3.
- Dodate su funkcije za GTK 3 dinamičko učitavanje.
- Modifikovan je Swing GTK LnF za GTK 3 podršku.

Swing GTK LnF je skraćenica za „Swing GTK look and feel".

NOVI HOTSPOT BUILD SYSTEM [JEP 284]

Java platforma koja je korišćena pre Jave 9 je predstavljala sistem sa duplim kodom, otkazivanjima i drugim nedostacima. Sistem je ponovo izrađen za Javu 9 na osnovu infra radnog okvira (u ovom kontekstu infra je skraćenica za infrastrukturu). Opšti cilj je bio da se pomoću JEP-a 284 build system nadgradi na pojednostavljeni build system. Specifični ciljevi obuhvataju:

- iskorišćavanje postojećeg sistema
- održivi kod
- smanjenje dupliranja koda
- pojednostavljenje
- podršku budućim poboljšanjima

Više o „Oracleovom“ radnom okviru infrastrukture možete da pročitate na sajtu: `http://www.oracle.com/technetwork/oem/frmwrk-infra-496656.Html`

REZIME

U ovom poglavlju su prikazane neke nove impresivne funkcije Java platforme, sa posebnim naglaskom na kompajler javac, JDK biblioteke i različite pakete za testiranje. Poboljšanja upravljanja memorijom, uključujući efikasnost hip prostora, dodelu memorije i poboljšano prikupljanje „smeća“, predstavljaju novi moćan skup poboljšanja Java platforme. Izmene u vezi sa procesom kompajliranja koje su povećale efikasnost su, takođe, razmatrane u ovom poglavlju. Osim toga, predstavili smo važna poboljšanja, kao što je proces kompajliranja, testiranja tipa, anotacije i automatizovane testove kompajlera tokom izvršenja.

U sledećem poglavlju ćete videti nekoliko manjih poboljšanja Java jezika koja su uvedena u Javu 9.