DEO I

Jezik Java

POGLAVLJE 1
Istorija i evolucija Jave

POGLAVLJE 2
Pregled jezika Java

POGLAVLJE 3
Tipovi podataka, promenljive i nizovi

POGLAVLJE 4
Operatori

POGLAVLJE 5
Upravljačke naredbe

POGLAVLJE 6
Uvod u klase

POGLAVLJE 7
Metode i klase izbliza

POGLAVLJE 8
Nasleđivanje

POGLAVLJE 9
Paketi i interfejsi

POGLAVLJE 10
Obrada izuzetaka

POGLAVLJE 11
Višenitno programiranje

POGLAVLJE 12
Nabrojani tipovi, automatsko pakovanje i metapodaci (anotacije)

POGLAVLJE 13
U/I, apleti i druge teme

POGLAVLJE 14
Generički tipovi

POGLAVLJE 1

Istorija i evolucija Jave

Ako želite da potpuno razumete Javu, morate shvatiti razloge koji su uticali na njeno stvaranje, sile koje su je uobličile i elemente koje je dobila u nasleđe. Slično uspešnim programskim jezicima koji su joj prethodili, i Java je mešavina najboljih elemenata svog bogatog nasleđa, kombinovanih s novim koncepcijama neophodnim za njenu jedinstvenu misiju. Dok ćemo se u narednim poglavljima ove knjige baviti praktičnim aspektima Jave – njenom sintaksom, bibliotekama i primenama – u ovom poglavlju ćete saznati kako i zašto je Java nastala, šta je čini tako važnom i kako se menjala tokom godina.

Iako je postala nerazdvojivi deo mrežnog okruženja interneta, treba istaći da je Java na prvom mestu programski jezik. Programski jezici se usavršavaju i razvijaju iz dva osnovna razloga:

- da bi se prilagodili izmenjenom okruženju i novoj primeni
- da bi se u njih ugradila poboljšanja i novosti iz oblasti programiranja.

Kao što ćete se i sami uveriti, na razvoj Jave uticala su oba ova elementa, u približno jednakoj meri.

Poreklo Jave

Java je srodnik jezika C++, koji je direktan potomak jezika C. Veći deo svojih osobina Java je nasledila od ova dva jezika. Iz jezika C Java je preuzela sintaksu. Mnoge objektno orijentisane osobine Jave nastale su pod uticajem jezika C++. U stvari, više karakteristika koje definišu Javu potiču od njenih prethodnika ili su nastale kao odgovor na njih. Štaviše, nastanak Jave ima duboke korene u procesima poboljšavanja i prilagođavanja programskih jezika tokom poslednjih nekoliko decenija. Iz pomenutih razloga, u ovom odeljku ćemo prikazati redosled događaja i uticaja koji su doveli do nastanka Jave. Kao što ćete videti, svaka inovacija programskog jezika nastaje iz potrebe da se reši određeni temeljni problem koji nije bio rešiv u ranijim programskim jezicima. U tom pogledu ni Java nije izuzetak.

Rađanje savremenog programiranja: C

Pojava jezika C uzdrmala je računarski svet. Njegov uticaj ne treba potcenjivati, pošto je on iz osnova izmenio pristup programiranju i programerski način razmišljanja. Pojava jezika C direktan je rezultat potrebe za strukturiranim, efikasnim jezikom visokog nivoa koji može da zameni asemblerski kôd za pisanje sistemskih programa. Kao što možda i sami znate, kada se projektuje računarski jezik, često se prave kompromisi slični sledećim:

- lakoća korišćenja naspram moći
- mogućnost pogrešne primene naspram efikasnosti
- strogost primene pravila naspram proširivosti.

Pre C-a, programeri su obično morali da biraju jezik u kome je jedan skup osobina bio favorizovan u odnosu na drugi. Na primer, iako se FORTRAN može koristiti za pisanje prilično efikasnih programa za primenu u nauci, on nije najpogodniji za pisanje sistemskih programa. Zatim, premda se BASIC lako uči, on nije previše moćan, a nedostatak strukturnih elemenata čini ga nepogodnim za pisanje dugačkih programa. Asemblerom se mogu napraviti veoma efikasni programi, ali se on teško uči i teško efikasno koristi. Štaviše, otklanjanje grešaka u asemblerskom kodu može da bude veoma komplikovano.

Još je gore to što prvi računarski jezici, kao što su BASIC, COBOL i FORTRAN, nisu projektovani u skladu s principima strukturiranog programiranja. Umesto njih, oslanjali su se na naredbu GOTO kao na glavnu programsku kontrolu. Zbog toga su se programi pisani na ovim jezicima lako preobražavali u tzv. „špagete kôd" – masu prepletenih skokova i uslovnih grananja koje gotovo niko nije mogao da razume. Iako su jezici kao što je Pascal strukturirani, njihova efikasnost nije bila osnovni cilj, a nedostajale su im i određene osobine koje bi ih učinile primenljivim za širok opseg programa. (Konkretno, ako uzmemo u obzir standardne dijalekte Pascala koji su u to vreme postojali, Pascal nije dolazio u obzir za pisanje sistemskih programa.)

Dakle, pre nastanka C-a, nijedan jezik nije uspeo da razreši sukobljene principe koji su osujećivali ranije napore. Pa ipak, potreba za takvim jezikom bila je urgentna. Početkom sedamdesetih godina prošlog veka, kada je računarska revolucija počela da se zahuktava, potražnja za softverom uveliko je prevazilazila mogućnosti programera da ga isporuče. U akademskim krugovima žustro su se lomila koplja u pokušajima da se stvori bolji programski jezik. Međutim, a to je možda i važnije, počeo je da se oseća i drugi uticaj. Računarski hardver se toliko raširio da je dosegao „kritičnu masu". Računari više nisu čuvani iza strogo zaključanih vrata. Po prvi put su programeri dobili naizgled neograničen pristup svojim mašinama. To je donelo slobodu u eksperimentisanju i istovremeno omogućilo programerima da počnu da prave sopstvene alatke. U predvečerje rađanja C-a, računarska pozornica je bila spremna za kvalitativan skok u oblasti programskih jezika.

Jezik C, koji je stvorio Denis Riči (Dennis Ritchie) i ugradio ga na mašinu DEC PDP-11 koja je radila pod UNIX-om, bio je rezultat razvojnog procesa započetog na starijem jeziku, BCPL, koji je razvio Martin Ričards (Martin Richards). BCPL je uticao na jezik, zvan B, koji je stvorio Ken Tompson (Ken Thompson), a ovaj je zatim sedamdesetih godina doveo do razvoja C-a. Tokom više godina, de facto standard jezika C bio je onaj koji se koristio na operativnom sistemu UNIX, a koji su Brajan Kernigan (Brian Kernighan) i Denis Riči opisali u knjizi *The C Programming Language* (Prentice-Hall, 1978). Jezik C je i formalno standardizovan decembra 1989, kada je Američki nacionalni institut za standardizaciju (American National Standards Institute, ANSI) usvojio standard za jezik C.

Mnogi smatraju da nastanak jezika C označava početak savremenog doba u računarskim jezicima. On je uspešno pomirio sukobljene principe koji su toliko mučili njegove prethodnike. Tako je nastao moćan, efikasan, strukturiran jezik koji se srazmerno lako uči. On ima i drugi, gotovo neprimetan aspekt: C je programerski jezik. Pre pojave C-a, računarski jezici obično su nastajali kao rezultat akademskih eksperimenata ili pod uticajem birokratskih komiteta. C je u tom pogledu drugačiji. Njega su zamislili, izgradili i razvili pravi programeri, i zbog toga on odražava njihov pristup programiranju. C su izbrusili, isprobali i o njemu

neprestano razmišljali ljudi koji su ga stvarno i koristili. Rezultat je bio jezik koji se programerima sviđao. Zaista, C je ubrzo privukao mnoge sledbenike koji su za njim žudeli s gotovo verskim zanosom. Zbog toga je on među programerima veoma brzo i sigurno našao svoje mesto. Ukratko, C je jezik koji su programeri stvorili za programere. Kao što ćete kasnije videti, Java je nasledila tu osobinu.

C++: sledeći korak

Krajem sedamdesetih i početkom osamdesetih godina dvadesetog veka, C je postao dominantan programski jezik, a i danas se masovno koristi. Pošto je C bio uspešan i upotrebljiv programski jezik, možda se pitate zašto je postojala potreba i za nečim drugim. Odgovor je u reči *složenost*. Kroz čitavu istoriju programiranja, sve složeniji programi uvek su zahtevali sve bolje načine za rešavanje te složenosti. C++ je bio odgovor na takve zahteve. Da biste bolje razumeli zašto je hvatanje u koštac sa složenošću programa bilo osnovni razlog za nastanak jezika C++, razmotrite tekst koji sledi.

Pristup programiranju se dramatično izmenio od onog doba kada su se pojavili računari. Recimo, na prvim računarima su se sve binarne mašinske instrukcije morale ručno uneti sa konzole. To je moglo da se radi sve dok programski kôd nije bio duži od nekoliko stotina instrukcija. Kako su programi rasli, ovo je postalo nepraktično, pa je smišljen asembler, jezik koji je programerima omogućio da rade na većim, složenijim programima koristeći simbolične predstave mašinskih instrukcija. Kako su programi i dalje rasli, smišljani su jezici visokog nivoa koji su programerima stavljali na raspolaganje sve više alatki neophodnih za borbu sa složenošću programa.

Prvi široko korišćen jezik bio je, naravno, FORTRAN. Mada je on bio značajan korak u programiranju, teško da je podsticao pisanje jasnih i lako razumljivih programa. Šezdesete godine su iznedrile *strukturirano programiranje*. Slavu te metode proneli su jezici kao što je C. Strukturirani jezici su po prvi put omogućili programerima da prilično lako pišu umereno složene programe. Međutim, čak i uz metode strukturiranog programiranja, kada projekat dostigne određenu veličinu, njegova složenost u jednom trenutku prevaziđe mogućnosti programera. Početkom osamdesetih godina, u mnogim projektima princip strukturiranog programiranja nasilno je primenjivan preko svojih mogućnosti. Da bi se ovaj problem prevazišao, smišljen je nov način programiranja, tzv. *objektno orijentisano programiranje* (*OOP*). O objektno orijentisanom programiranju detaljnije ćemo govoriti kasnije u knjizi, pa se sada zadovoljimo samo kratkom definicijom: OOP je metodologija programiranja koja pomaže pri organizovanju složenih programa koristeći nasleđivanje, kapsuliranje i polimorfizam.

Sve u svemu, iako je C jedan od najboljih svetskih programskih jezika, ni on nije bez ograničenja u savladavanju složenih situacija. Kada program premaši izvesnu veličinu, postaje tako složen da je teško pojmiti njegovu celinu. Mada je teško odrediti tačnu veličinu kada se to događa jer to zavisi i od prirode programa i od programera, uvek je postojala granica iznad koje je postajalo nemoguće upravljati programom. Nove mogućnosti koje je doneo jezik C++ učinile da ta prepreka nestane, a programeri bolje shvataju složenije programe i lakše njima upravljaju.

C++ je 1979. stvorio Bjarne Stroustrup dok je radio u Belovim laboratorijama u Mari Hilu, Nju Džerzi. Stroustrup je novi jezik prvobitno nazvao „C sa klasama". Godine 1983. ime mu je promenjeno u C++. (U jeziku C, „++" označava operator uvećanja. Tako bi C++ značilo „poboljšani C". – prim. rec.) C++ proširuje jezik C dodajući mu objektno orijentisane osobine. Pošto je C++ izgrađen na temelju jezika C, on obuhvata sve njegove osobine, svojstva i prednosti. To je osnovni razlog uspeha jezika C++, koji nije načinjen s namerom da se stvori potpuno nov programski jezik, već da se poboljša jedan koji se već pokazao veoma uspešnim.

Sve je spremno za pojavu Jave

Krajem osamdesetih i početkom devedesetih godina, objektno programiranje na jeziku C++ uzelo je maha. I zaista, za trenutak je izgledalo da su programeri konačno pronašli savršen programski jezik. Pošto C++ objedinjuje visoku efikasnost i stilske elemente C-a sa objektno orijentisanim pristupom, mogao se koristiti za pravljenje široke lepeze programa. Međutim, baš kao toliko puta u prošlosti, ponovo je narasla potreba da se računarski jezici pomaknu za još jedan stepenik u svojoj evoluciji. Za nekoliko godina, internet i veb poprimiće svoj sadašnji oblik, a ovaj događaj će začeti novu revoluciju na polju programiranja.

Nastanak Jave

Javu su prvobitno, 1991. godine, koncipirali Džejms Gosling (James Gosling), Patrik Noton (Patrick Naughton), Kris Vort (Chris Warth), Ed Frenk (Ed Frank) i Majk Šeridan (Mike Sheridan), iz korporacije Sun Microsystems, Inc. Trebalo je 18 meseci rada da bi se došlo do prve radne verzije. Ovaj jezik je prvobitno dobio ime „Oak", ali je ono 1995. godine promenjeno u „Java". U periodu od prve realizacije Oaka s jeseni 1992. i zvaničnog najavljivanja Jave u proleće 1995, još mnogo osoba je doprinelo uobličavanju i razvijanju ovog jezika. Bil Džoj (Bill Joy), Artur van Hof (Arthur van Hoff), Džonatan Pejn (Jonathan Payne), Frenk Jelin (Frank Yellin) i Tim Lindholm bili su glavni saradnici na usavršavanju prvobitnog prototipa.

Donekle iznenađuje to što glavni pokretač za razvoj Jave nije bio internet! Osnovni motiv je bila potreba za jezikom nezavisnim od računarske platforme (tj. neutralnim prema arhitekturi), koji bi se mogao koristiti za pravljenje softvera namenjenog različitim elektronskim uređajima u domaćinstvu, kao što su mikrotalasne pećnice i daljinski upravljači. Možda i sami pogađate da se kao kontroleri takvih uređaja koriste različiti tipovi procesora. Problem sa jezicima C i C++ (kao i s većinom drugih jezika), leži u tome što se pri prevođenju u binarni oblik oni moraju usmeriti na određeni procesor. Iako se program pisan na jeziku C++ može prevesti za bilo koji procesor, za to je potreban potpun prevodilac namenjen konkretnom procesoru. Teškoća je to što su prevodioci skupi i sporo se prave. Očigledno je bilo potrebno lakše i jeftinije rešenje. U pokušaju da dođu do njega, Gosling i ostali počeli su da rade na prenosivom jeziku koji ne zavisi od platforme i čiji bi kôd mogao da se izvršava na različitim procesorima i u različitim okruženjima. Ovi napori su konačno doveli do rađanja Jave.

Otprilike u vreme kada su razrađivani detalji novog programskog jezika, pojavio se drugi, bez sumnje značajniji uticaj koji je odigrao ključnu ulogu u budućnosti Jave. Taj činilac bio je, naravno, World Wide Web. Da se veb nije uobličio približno u isto vreme kada i Java, taj programski jezik – upotrebljiv ali složen – možda bi se koristio samo za programiranje kućne elektronike. Međutim, pojavom veba, Java se probila u prvi plan računarskih jezika, jer su i za veb bili potrebni prenosivi programi.

Većina programera vrlo rano nauči da su prenosivi programi, iako poželjni, prilično neostvarljivi. Iako je potraga za efikasnim, prenosivim (nezavisnim od platforme) programima stara gotovo koliko i samo programiranje, preči problemi su je potisnuli u drugi plan. Štaviše, pošto se u to vreme većina korisnika računara svrstavala u tri konkurentska tabora: Intel, Macintosh i UNIX, najveći broj programera se zadržao unutar svojih utvrđenih granica i potreba za prenosivim kodom izgubila je svoj primarni značaj. Međutim, sa razvojem interneta i veba, stari problem prenosivosti ponovo se pojavio u punoj snazi. Na kraju krajeva, internet predstavlja raznolik, distribuiran univerzum ispunjen mnogim vrstama računara, operativnih sistema i procesora. Iako su na Internet priključene mnoge vrste platformi, korisnici bi želeli da na svima može da se izvršava isti program. Ono što je jednom bilo neugodan ali malo značajan problem, postalo je problem koji je pod hitno trebalo rešiti.

Godine 1993, članovima tima za razvoj Jave postalo je jasno da se problem prenosivosti, s kojim su se često sretali pri programiranju koda za kontrolere elektronskih uređaja, pojavljuje i pri pisanju koda za internet. U stvari, isti problem zbog koga je Java prvobitno smišljena, pojavio se i na internetu – samo u velikom obimu. Zbog toga je fokus primene Jave premešten s kućne elektronike na programiranje za internet. Dakle, premda je želja za dobijanjem programskog jezika koji ne zavisi od platforme bila početni motiv, za uspeh Jave na visokom nivou zaslužniji je internet.

Pomenuli smo ranije da Java većinu svojih osobina duguje jezicima C i C++. To nije slučajno. Autori Jave su dobro znali da će poznata sintaksa jezika C i prizvuk objektno orijentisanih osobina jezika C++ privući mnoge iskusne C/C++ programere. Osim očiglednih sličnosti, Javi i jezicima C i C++ zajedničke su i one osobine koje su ova dva jezika učinile uspešnim. Kao prvo, Javu su oblikovali, isprobavali i poboljšavali profesionalni programeri. To je jezik utemeljen na potrebama i iskustvu osoba koje su ga stvorile. Zbog toga je Java i jezik programera. Drugo, Java je harmoničan i logički dosledan jezik. Treće, izuzev ograničenja koje postavlja internet, Java omogućuje programeru potpunu kontrolu. Ako programirate na pravi način, to će se pokazati u vašem programu. A pokazaće se i ako ste loš programer. Ukratko, Java nije jezik koji štiti nevešte početnike, nego jezik za profesionalne programere.

Zbog sličnosti između Jave i jezika C++, nameće se misao da je Java samo „C++ u verziji za internet". Grdno ćete pogrešiti ako prihvatite tu ideju. Java se od jezika C++ znatno razlikuje – i praktično i po filozofiji. Mada je tačno da je Java nastala pod uticajem jezika C++, ona nije njegova poboljšana verzija. Na primer, Java nije ni uzlazno ni silazno kompatibilna sa jezikom C++. Naravno, sličnosti su znatne i, ako ste programirali na jeziku C++, osećate se kao „kod kuće" i s Javom. Još nešto: Java nije napravljena da zameni C++ već da reši određen skup problema. C++ je stvoren da reši drugačiji skup problema. Oba jezika će uporedo postojati još mnogo godina.

Kao što smo pomenuli na početku poglavlja, programski jezici se razvijaju iz dva razloga: da bi se prilagodili promenama okruženja i da bi prihvatili novine u oblasti programiranja. Promena okruženja koja je izazvala pojavu Jave bila je potreba za programima koji rade nezavisno od platforme, namenjenim za distribuciju širom interneta. Međutim, Java omogućava i nov način pristupa pisanju programa. Na primer, Java je poboljšala i unapredila koncepciju objektne orijentisanosti koju je koristio i C++, a donela je integrisanu podršku za višenitni rad i ugrađenu biblioteku koja omogućava jednostavan pristup internetu. Ali uzevši sve u obzir, Java nije postala tako izuzetan jezik zbog pojedinih mogućnosti koje pruža, nego zato što je kao celina bila savršen odgovor na pojavu novog i veoma distribuiranog računarskog sveta. Za programiranje na internetu, Java predstavlja ono što je bio C za sistemsko programiranje: revolucionarnu snagu koja je izmenila svet.

Veza sa jezikom C#

Doseg i moć Jave još uvek se osećaju u svetu razvoja računarskih jezika. Dobar deo njenih inovativnih karakteristika, konstrukcija i koncepata postao je obavezan sastavni deo svih novih jezika. Uspeh Jave je toliki da ga je nemoguće ignorisati.

Jezik C# možda je najvažniji primer Javinog uticaja, jer joj je veoma blizak. Microsoft je razvio C# kao podršku za svoju tehnologiju .NET Framework. Oba jezika imaju istu opštu sintaksu, podržavaju distribuirano programiranje i koriste isti model objekata. Naravno, između njih postoje i neke razlike, ali im je opšti izgled i utisak koji ostavljaju veoma sličan. „Ukrštanje" jezika Java i C# do sada je najjači dokaz da je Java promenila način razmišljanja o programskim jezicima i način njihove upotrebe.

Kako je Java promenila internet

Internet je pomogao da se Java probije u prve redove programiranja, a Java je, za uzvrat, jako uticala na internet. Osim opšteg pojednostavljivanja veb programiranja, Java je uvela novu vrstu programa koja radi u mrežnom okruženju i koja se zove aplet. Aplet je promenio način na koji korisnici vide sadržaj objavljen na internetu. Java je takođe donela rešenje za neke od najnezgodnijih problema u oblasti prenosivosti i bezbednosti softvera na internetu. Te elemente detaljnije ćemo razmotriti u narednim odeljcima.

Java apleti

Aplet je posebna vrsta Java programa namenjena distribuciji preko interneta i automatskom izvršavanju u čitačima veba (engl. *Web browsers*) koji podržavaju Javu. Aplet se preuzima s mreže, bez intervencije korisnika. Ako korisnik mišem pritisne hipervezu koja sadrži aplet, čitač veba automatski preuzima i izvršava aplet. Apleti su kratki programi koji se najčešće koriste za prikazivanje podataka koji stižu od servera, za obradu podataka koje unosi korisnik ili za izradu jednostavnih funkcija – kao što je kalkulator iznosa kamate – i izvršavaju se lokalno a ne na serveru. Ukratko, aplet omogućava da se deo funkcionalnosti premesti sa servera na klijenta.

Uvođenje apleta izmenilo je programiranje za internet zato što je proširilo svet objekata koji se slobodno kreću kiberprostorom. Postoje dve veoma opšte kategorije objekata koji se razmenjuju između servera i klijenta: pasivni podaci i dinamički, aktivni programi. Na primer, kada čitate poruke koje ste dobili e-poštom, pregledate pasivne podatke. Čak i kad preuzmete program, kôd programa je i dalje samo grupa pasivnih podataka dok ga ne izvršite. Za razliku od toga, aplet je dinamički izvršiv program. Uprkos tome što je takav program aktivan na klijentskom računaru, njegovim izvršavanjem upravlja server.

Koliko god da su korisni programi koji se dinamički izvršavaju u mrežnom okruženju, oni takođe donose ozbiljne probleme u oblastima bezbednosti i prenosivosti. Razume se, mora se obavezno sprečiti da program koji klijentski računar automatski preuzima i izvršava čini štetu. Osim toga, takav program treba da bude u stanju da se izvršava u različitim okruženjima i pod različitim operativnim sistemima.

Kao što ćete videti u detaljnijem razmatranju koje sledi, Java rešava navedene probleme efikasno i elegantno.

Bezbednost

Verovatno znate sledeće: kad god s mreže preuzimate neki „normalan“ program, preuzimate rizik, jer kôd koji preuzimate može sadržati virus, trojanac ili drugu vrstu štetnog koda. Suština problema leži u činjenici da zlonameran kôd uspeva da prouzrokuje štetu zato što neovlašćeno pristupa sistemskim resursima. Na primer, virus može da prikupi lične podatke, kao što su brojevi kreditnih kartica, stanja na bankovnim računima i lozinke za njih, tako što pretraži sadržaj sistema datoteka na lokalnom računaru. Da bi Java omogućila bezbedno preuzimanje i izvršavanje apleta na klijentskom računaru, bilo je neophodno sprečiti da aplet pokrene tu vrstu napada.

Java pruža tu zaštitu tako što aplet ograničava na Javino izvršno okruženje i ne dozvoljava mu da pristupa drugim delovima računara. (U nastavku teksta videćete kako se to postiže.) Mogućnost preuzimanja apleta uz puno poverenje da neće biti načinjena šteta i narušena bezbednost, mnogi smatraju najinovativnijim aspektom Jave.

Prenosivost

Prenosivost je veoma važan aspekt interneta jer na njemu postoje najrazličitije vrste računara i operativnih sistema. Ako dati Java program treba da radi na praktično svakom računaru na internetu, neophodno je omogućiti da se taj program izvršava na različitim sistemima. Na primer, u slučaju apleta, isti aplet treba da preuzima i izvršava široka lepeza procesora, operativnih sistema i čitača veba koji su povezani na internet. Pošto nije praktično da se za svaku vrstu računara piše i koristi različita verzija apleta, *isti* kôd mora da radi na *svim* računarima. Zbog toga je bio potreban način za generisanje prenosivog izvršnog koda. Kao što ćete uskoro videti, isti mehanizam koji pruža bezbednost omogućava i prenosivost.

Magija Jave: bajt kôd

Ono što Javi omogućuje da reši upravo pomenute probleme bezbednosti i prenosivosti jeste činjenica da prevodilac jezika Java ne generiše izvršni kôd već tzv. bajt kôd. *Bajt kôd* (engl. *bytecode*) jeste visokooptimizovan skup instrukcija koji u trenutku izvršavanja programa tumači (interpretira) Javin izvršni sistem, poznat kao *Javina virtuelna mašina* (Java Virtual Machine, JVM). U suštini, prvobitna JVM bila je zamišljena kao *interpretator bajt koda*. To će vas možda iznenaditi, pošto se zbog postizanja boljih performansi mnogi savremeni jezici direktno prevode u izvršni kôd. Međutim, činjenica da Java programe izvršava JVM, pomaže da se reše glavni problemi s preuzimanjem programa sa interneta. Evo i zašto.

Java programa preveden u bajt kôd mnogo se lakše izvršava u različitim okruženjima zato što je za različite platforme potrebno napraviti samo različite virtuelne mašine. Kada se u određeni sistem jednom ugradi odgovarajući paket za izvršavanje, na sistemu će moći da se pokrene svaki Java program. Iako se Javine virtuelne mašine razlikuju na različitim platformama, sve one razumeju isti Javin bajt kôd. Kada bi se Java programi direktno prevodili u izvršni kôd odgovarajućih mašina, onda bi morale postojati različite verzije programa za svaki procesor priključen na internet. To, naravno, nije pogodno rešenje. Iz rečenog proizlazi da je izvršavanje bajt koda pomoću JVM-a najjednostavniji način pravljenja uistinu prenosivih programa.

To što Java programe ne izvršava neposredno procesor, nego JVM, čini ih i bezbednijim. Pošto svime upravlja JVM, ova virtuelna mašina može potpuno da obuhvati program, onemogućavajući da sporednim efektima remeti okolni sistem. Kasnije ćete videti da je bezbednost još i povećana određenim ograničenjima u jeziku Java.

Kada se program prevodi u neki međuoblik i zatim interpretira pomoću virtuelne mašine, on se po pravilu izvršava sporije nego da je direktno preveden u izvršni kôd. U Javi ta razlika u vremenu izvršavanja nije tako velika. Pošto je bajt kôd u velikoj meri optimizovan, njegova upotreba omogućuje Javinoj virtuelnoj mašini da programe izvršava mnogo brže nego što biste očekivali.

Mada je Java prvobitno zamišljena kao interpretiran jezik, s tehničkog aspekta nema nikakvih prepreka da se – radi ubrzanja rada – bajt kôd odmah prevede u izvršni kôd određenog računara. Zato je firma Sun nedugo posle objavljivanja Jave ponudila i svoj JIT (Just In Time) prevodilac po imenu HotSpot. Kada je JIT deo JVM-a, odabrani delovi bajt koda prevode se u izvršni kôd tokom izvršavanja – deo po deo, po potrebi. Treba razumeti da nije moguće da se ceo Java program odmah prevede u izvršni kôd, jer Java obavlja različite provere koje se mogu izvoditi samo u trenutku izvršavanja. Zbog toga Java prevodi kôd po potrebi, tokom izvršavanja. Sem toga, ne prevodi se ceo bajt kôd, nego samo oni njegovi delovi koji će bitno ubrzati izvršavanje. Preostali deo koda se i dalje interpretira. Međutim, čak

i ovim načinom „prevođenja po potrebi“ još uvek se postižu bolje performanse pri izvršavanju koda. Bezbednost i prenosivost programa ne smanjuju se kada se na bajt kôd primeni ovakvo dinamičko prevođenje, jer izvršnim okruženjem i dalje upravlja JVM.

Servleti: Java na serverskoj strani

Koliko god da su apleti korisni, oni su ipak samo polovina jednačine klijent/server. Nedugo posle objavljivanja prvobitne verzije, postalo je očigledno da bi Java bila korisna i na serverskoj strani. Rezultat je servlet. Servlet je kratak program koji se izvršava na serveru. Kao što apleti dinamički proširuju funkcionalnost čitača veba, tako i servleti dinamički proširuju funkcionalnost veb servera. Uvođenjem servleta Java je premostila oba strane veze između klijenta i servera.

Servleti se koriste za generisanje dinamičkog sadržaja koji se zatim šalje klijentu. Recimo, na stranici veb prodavnice servlet može učitavati cenu određenog artikla iz baze podataka. Podatak o ceni zatim se koristi za dinamičko generisanje veb stranice koja se šalje čitaču. Mada je dinamički generisan sadržaj na raspolaganju pomoću mehanizama kao što je CGI (Common Gateway Interface), servlet pruža više prednosti, među kojima su i bolje performanse.

Pošto se servleti (kao i svi drugi Java programi) prevode u bajt kôd koji izvršava JVM, postiže se veoma visok stepen prenosivosti. To znači da se isti servlet može koristiti u velikom broju različitih serverskih okruženja. Jedina obaveza je da server podržava JVM i kontejner za servlet.

Pojmovi karakteristični za Javu

Nijedna priča o istoriji Jave nije potpuna ako se ne pomenu izrazi koji su karakteristični za nju. Mada su osnovni činioci koji su uticali na pojavu Jave prenosivost i bezbednost, postoje i drugi činioci koji su odigrali važnu ulogu u oblikovanju Jave. Sledeća lista pojmova, koju je sastavio autorski tim Jave, ukazuje na osnovne pravce razmišljanja:

- jednostavno
- bezbedno
- prenosivo
- objektno orijentisano
- robusno
- višenitno
- nezavisno od platforme
- interpretirano
- visokoefikasno
- distribuirano
- dinamičko

Dva pojma – bezbedno i prenosivo – već smo objasnili. Pogledajmo šta znače ostali.

Jednostavno

Java je koncipirana tako da programeri mogu lako da je nauče i efikasno koriste. Pod uslovom da imate određeno iskustvo u programiranju, neće vam biti teško da ovladate Javom. Ako već poznajete osnovne pojmove objektno orijentisanog programiranja, učenje Jave će vam biti još lakše. Najbolje je ako ste iskusan programer na jeziku C++ jer ćete veoma lako preći na Javu. Pošto je Java nasledila sintaksu jezika C i C++, kao i mnoge objektno orijentisane osobine jezika C++, učenje Jave većini programera ne stvara teškoće.

Objektno orijentisano

Iako je Java nastala pod uticajem svojih prethodnika, nije bilo predviđeno da njen izvorni kôd bude kompatibilan sa izvornim kodom bilo kog drugog jezika. Zato su autori Jave imali potpuno odrešene ruke. Jedan rezultat koji je odavde proizišao jeste čist, upotrebljiv i pragmatičan tretman objekata. Slobodno pozajmljujući principe iz mnogih originalnih objektnih softverskih okruženja koja su nastala tokom minulih decenija, Java je uspela da održi ravnotežu između čistunskog stava „sve je objekat“ i pragmatičnog pravila „ne smetaj mi“. Model objekta u Javi jednostavan je i lako se proširuje, dok se prosti tipovi, kao što su celi brojevi, ne obrađuju kao objekti jer je tako mnogo efikasnije.

Robusno

Veb okruženje sa više platformi postavlja izuzetne programske zahteve jer program mora da se pouzdano izvršava na različitim sistemima. Zbog toga je pri projektovanju Jave dat prioritet sposobnosti da se napravi robustan program. U cilju postizanja pouzdanosti rada programa, Java vas ograničava u nekoliko ključnih područja, terajući vas da programske greške ispravite u ranoj fazi. Istovremeno, Java vas oslobađa briga o mnogim najčešćim programskim greškama. Pošto je Java strogo tipiziran jezik, ona proverava kôd u trenutku njegovog prevođenja. Međutim, ona ga proverava i u trenutku izvršavanja. U stvari, mnoge neuhvatljive greške – one koje se pojavljuju u situacijama koje je teško simulirati – u Javi se ne mogu ni napraviti. Glavna prednost Jave je to što možete da predvidite kako će se ono što ste napisali ponašati u različitim uslovima.

Da biste bolje razumeli robusnost Jave, setite se dva glavna razloga otkazivanja programa: greške pri upravljanju memorijom i loša obrada izuzetaka (tj. grešaka koje nastaju tokom izvršavanja programa). U tradicionalnom programskom okruženju, upravljanje memorijom može da bude težak i zapetljan posao. Na primer, u jezicima C i C++ programer mora ručno da zauzme i oslobodi svu dinamički dodeljivanu memoriju. Ovo ponekad dovodi do problema, jer programeri zaboravljaju da oslobode prethodno dodeljenu memoriju ili, što je gore, pokušavaju da oslobode memoriju koju drugi deo njihovog koda još uvek koristi. Java otklanja te probleme jer dodeljuje i oslobađa memoriju umesto vas. (U stvari, memorija se oslobađa potpuno automatski, pošto Java obezbeđuje „sakupljanje smeća“, tj. uklanjanje iz sistema objekata koji se više ne koriste.) U tradicionalnom okruženju izuzeci često nastaju pri deljenju nulom ili kada se ne pronađe odgovarajuća datoteka, i oni se moraju razrešiti rogobatnom, teško čitljivom konstrukcijom. Java je i ovde od pomoći jer obezbeđuje objektno orijentisanu obradu izuzetaka. U dobro napisanom Java programu, sve greške koje nastanu u vreme izvršavanja mogu se obraditi istim tim programom i tome treba težiti.

Višenitno

Java je projektovana tako da izađe u susret realnim zahtevima pravljenja interaktivnih mrežnih programa. Da bi se to postiglo, Java podržava višenitno programiranje – varijantu programiranja koja omogućuje da vaš program istovremeno radi više stvari. Javin izvršni sistem ima elegantno i potpuno rešenje za sinhronizovanje više procesa, koje omogućuje da projektujete interaktivne sisteme što glatko rade. Javin lako primenljiv pristup istovremenom radu omogućava da više razmišljate o specifičnom ponašanju programa, umesto da se bavite podsistemom za višeprogramski rad.

Nezavisno od platforme

Osnovni problem projektanata Jave bio je kako obezbediti trajan i prenosiv kôd. U vreme stvaranja Jave, jedan od najozbiljnijih problema s kojim su se suočavali programeri bio je sledeći: ako su napisali program koji radi danas, niko nije mogao da garantuje da će on raditi i sutra, čak i na istoj mašini. Operativni sistemi se neprestano poboljšavaju, poboljšavaju se i procesori, a kada se sve kombinuje s promenama u osnovnim resursima sistema, može se dogoditi da program više ne radi kako treba. Zbog toga su projektanti Jave morali da donesu nekoliko teških odluka o samom jeziku i o Javinoj virtuelnoj mašini. Njihov slogan je bio: „Napiši jednom, izvršavaj bilo gde, bilo kada“. Taj cilj je najvećim delom i postignut.

Interpretirano i visokoefikasno

Kao što je ranije objašnjeno, Java omogućuje pravljenje programa za više platformi tako što prevodi izvorni kôd u međuproizvod, zvan bajt kôd. Takav kôd se može izvršavati na svakom sistemu koji ima Javinu virtuelnu mašinu. Najveći broj prethodnih pokušaja da se napravi kôd koji će raditi nezavisno od platforme ostvarivao je to na račun performansi. Kao što smo već pomenuli, Javin bajt kôd je pažljivo optimizovan tako da se po potrebi, pomoću JIT prevodioca, lako prevodi u mašinski kôd konkretnog računara, čime se postižu visoke performanse. Javini izvršni sistemi koji obezbeđuju ovakvo prevođenje njime nimalo ne umanjuju nezavisnost koda od platforme.

Distribuirano

Java je posebno namenjena distribuiranom okruženju interneta jer lako rukuje protokolima TCP/IP. U stvari, pristupanje resursu pomoću URL-a u osnovi se ne razlikuje od pristupanja datoteci. Java podržava i *daljinsko pozivanje metoda* (Remote Method Invocation, RMI). Dakle, program može da zahteva izvršavanje procedura koje se nalaze na bilo kojoj mrežnoj adresi.

Dinamičko

Java programi sadrže znatne količine podataka o tipu koji se koriste za proveravanje i razrešavanje pristupa objektima u trenutku izvršavanja. Time je omogućeno dinamičko povezivanje koda na pouzdan i efikasan način. To je od ključnog značaja za robusnost okruženja apleta, u kojem se delići bajt koda mogu dinamički ažurirati na sistemu koji izvršava program.

Evolucija Jave

Već je prvo izdanje Jave izazvalo revoluciju, ali njime nije prestalo doba brzih, inovativnih promena Jave. Za razliku od većine drugih softverskih sistema koji se poboljšavaju korak po korak, Java je nastavila da se intenzivno razvija. Ubrzo posle objavljivanja Jave 1.0, pojavila se

verzija 1.1. Osobine uvedene s Javom 1.1 mnogo su značajnije nego što bi se moglo pretpostaviti na osnovu broja verzije. U nju je dodato mnogo novih biblioteka, redefinisani su načini na koje apleti obrađuju događaje i iznova su konfigurisane mnoge osobine biblioteka iz verzije 1.0. U ovoj verziji takođe su u drugi plan (kao zastarele) potisnute mnoge osobine Jave 1.0. Na taj način, Java 1.1 je unela neke nove atribute u prvobitnu specifikaciju i istovremeno neke odbacila.

Sledeće veće poboljšanje Jave bila je verzija Java 2, gde brojka 2 označava drugu generaciju. Pojava Jave 2 predstavlja prelomni događaj koji obeležava početak „moderne faze" ovog jezika. Prvo izdanje Jave 2 nosilo je broj verzije 1.2. Možda vam je to čudno. Objašnjenje je to što se ovaj broj prvobitno odnosio na verziju Javinih biblioteka, a zatim je prihvaćen kao oznaka celog izdanja. Od izdavanja Jave 2, firma Sun je prepakovala Javu, nazvala je J2SE (Java 2 Platform Standard Edition) i brojeve verzija počela da primenjuje na taj proizvod.

Java 2 uvodi niz novih mogućnosti, među kojima su Swing i Collections Framework, poboljšanu Javinu virtuelnu mašinu i razne poboljšane programske alatke. Iz Jave 2 neki elementi su i odbačeni. To se najviše odnosi na klasu **Thread** u kojoj su metode **suspend**(), **resume**() i **stop**() označene kao zastarele.

Sledeće glavno izdanje Jave bila je verzija 1.3 proizvoda J2SE. To je prvo veliko poboljšanje prvobitne Jave 2. Njime je najvećim delom poboljšana postojeća funkcionalnost i istovremeno strože obezbeđeno razvojno okruženje. Programi pisani na verzijama 1.2 i 1.3 u načelu su na nivou izvornog koda međusobno kompatibilni. Iako verzija 1.3 sadrži manji broj izmena od prethodna tri izdanja, ona je svakako važna.

Novo obogaćenje Java je doživela u izdanju J2SE 1.4. Ono sadrži više važnih nadogradnji, poboljšanja i dodataka. Primera radi, dodati su nova ključna reč **assert**, ulančani izuzeci (engl. *chained exceptions*) i kanalski U/I. Izmenjene su i kolekcije i klase za umrežavanje. Uz to, posvuda je učinjeno više malih izmena. Uprkos značajnom broju novih karakteristika, verzija 1.4 zadržala je gotovo stoprocentnu kompatibilnost izvornog koda s prethodnim verzijama.

Sledeće izdanje Jave bilo je J2SE 5 i donelo je revolucionarne promene. Za razliku od većine prethodnih nadogradnji Jave, koje su donosile važna, ali nesuštinska poboljšanja, J2SE 5 iz osnova proširuje doseg, moć i oblast važenja tog jezika. Od pojavljivanja Jave pre deset godina nije bilo tako važnog, niti toliko željno očekivanog izdanja Jave.

Da biste shvatili veličinu izmena koje je J2SE 5 donela Javi, razmotrite sledeći spisak njenih glavnih novih mogućnosti:

- generički tipovi (engl. *generics*)
- anotacije (engl. *annotations*)
- metapodaci (engl. *metadata*)
- automatsko pakovanje (engl. *autoboxing*) i automatsko raspakivanje (engl. *auto-unboxing*)
- nabrojani tipovi (engl. *enumerations*)
- poboljšana petlja **for** u stilu for–each
- metode s promenljivim brojem argumenata (vararg)
- uvoz statičkih članova (engl. *static import*)
- U/I operacije s formatiranim podacima (engl. *formatted I/O*)
- skup klasa za konkurentno programiranje (engl. *concurrency utilities*)
- nadogradnje za API

To nije spisak minornih izmena niti delimičnih nadogradnji. Svaka stavka spiska predstavljala je značajan dodatak jeziku Java. Jedan deo stavki, kao što su generički tipovi, poboljšana petlja **for** i metode s promenljivim brojem argumenata, uvode nove elemente sintakse. Druge stavke, poput automatskog pakovanja i raspakivanja, menjaju semantiku jezika. Metapodaci dodaju programiranju novu dimenziju. U svakom slučaju, uticaj tiih dodataka prevazilazi njihove neposredne efekte. Oni menjaju sam karakter Jave.

Važnost novih mogućnosti odražava se u broju verzije „5". Bilo bi normalno da je sledeći broj verzije za Javu bio 1.5. Međutim, izmene i nove mogućnosti toliko su značajne da prelazak sa 1.4 na 1.5 naprosto nije mogao da izrazi veličinu promene. Sun je povećao broj verzije na 5 da bi naglasio važnost događaja, pa je ta verzija nazvana J2SE 5, a razvojni komplet – JDK 5. Međutim, da bi se održala doslednost, Sun je odlučio da 1.5 upotrebljava kao svoj interni broj verzije, koji se takođe naziva broj *razvojne verzije*. Dakle, 5 je eksterni, a 1.5 interni broj verzije. Broj „5" u JSE 5 nazvan je *broj proizvodne verzije*.

Sledeće izdanje Jave nazvano je Java SE 6. Kompanija je još jednom odlučla da izmeni ime platforme Java. Prvo, obratite pažnju na to da je izostavljeno „2". Platforma je dobila ime *Java SE*, a puno zvanično ime proizvoda bilo je *Java Platform, Standard Edition 6*. Java razvojni komplet dobio je ime JDK 6. Kao i u verziji J2SE 5, broj 6 u naslovu Java SE 6 predstavlja broj verzije proizvoda. Interni razvojni broj je 1.6.

Java SE 6 je nadograđena na osnovicu J2SE 5, koju poboljšava i proširuje. U verziji SE 6, samom jeziku nisu dodate važnije maogućnosti, ali su proširene biblioteke API funkcija, uneto je nekoliko novih paketa i poboljšanja izvršnog okruženja. Tokom njenog dugačkog životnog ciklusa (prema Java terminologiji) uvedeno je više izmena. Verzija Java SE 6 uglavnom je poslužila da još više učvrsti napredak koji je donela verzija J2SE 5.

Java SE 7

Najnovije izdanje Jave nazvano je Java SE 7, razvojni komplet se zove JDK 7 a interni broj verzije je 1.7. Java SE 7 je prvo važnije izdanje Jave nakon što je kompaniju Sun Microsystems preuzeo Oracle (postupak je započet aprila 2009. godine a završen januara 2010. godine). Java SE 7 ima mnoge nove mogućnosti, među kojima su i značajna proširenja jezika i biblioteka API funkcija. Dodate su i nadogradnje Java izvršnog okruženja radi podrške za jezike koji nisu Java, ali su za Java programere najzanimljivije dopune samog jezika i pratećih biblioteka.

Nove mogućnosti jezika razvijene su u okviru projekta *Coin*. Svrha projekta Coin bila je da identifikuje brojne manje izmene koje bi bile ugrađene u JDK 7. Mada se te nove mogućnosti opisuju kao „manje značajne", one ipak prilično značajno utiču na kôd na koji se odnose. U stvari, za mnoge programere te izmene će svakako biti najvažnije nove mogućnosti koje uvodi Java SE 7. Sledi lista novih mogućnosti jezika:

- Naredbom **switch** odsad može da upravlja i tip **String**.
- Binarni celobrojni literali.
- Znak za podvlaku u numeričkim literalima.
- Proširena naredba **try**, nazvana *try-with-resources*, podržava automatsko upravljanje resursima. (Na primer, tokovi se sada mogu zatvarati automatski kada više nisu potrebni.)
- Pogađanje tipa (pomoću operatora *diamond*) pri konstruisanju generičke instance.

- Poboljšana obrada izuzetaka kada dva ili više njih treba da presretne ista naredba catch (višestruki catch) i bolja provera tipa u izuzecima koji se ponovo izazivaju.
- Mada nije u pitanju izmena sintakse, poboljšana su upozorenja koja prevodilac generiše za neke metode s promenljivim brojem argumenata, a imate i veći stepen kontrole nad tim upozorenjima.

Kao što vidite, uprkos tome što se mogućnosti koje je uveo projekat Coin ne smatraju tako značajnim izmenama jezika, koristi koje će one doneti biće znatno veće nego što bi se zaključilo po izrazu „manje značajan". Pre svega, naredba **try**-*with-resources* suštinski će izmeniti način pisanja koda koji radi s tokovima. Osim toga, mogućnost da tip **String** upravlja naredbom **switch** odavno je očekivano poboljšanje koje će u mnogim slučajevima pojednostaviti programski kôd.

Java SE 7 donosi i više dopuna Java API biblioteka. Dve najvažnije su proširenja NIO Frameworka i dodavanje Fork/Join Frameworka. NIO (koji se prvobitno zvao New I/O) dodat je Javi u verziji 1.4. Međutim, izmene koje su predložene za Javu SE 7 suštinski proširuju njegove mogućnosti. Te izmene su toliko značajne da se često koristi izraz NIO.2.

Biblioteka Fork/Join Framework pruža važnu podršku za paralelno programiranje. Paralelno programiranje je uobičajeno ime za tehnike koje omogućavaju efikasno korišćenje računara u koje je ugrađeno više od jednog procesora, uključujući i sisteme s više jezgara. Prednost okruženja s više jezgara jeste mogućnost značajnog poboljšavanja performansi programa. Biblioteka Fork/Join Framework omogućava paralelno programiranje tako što

- pojednostavljuje definisanje i pokretanje poslova koji se mogu izvršavati istovremeno.
- automatski radi s više procesora (kad ih ima).

To znači da pomoću biblioteke Fork/Join Framework lako možete praviti skalabilne aplikacije koje automatski koriste sve procesore koji su na raspolaganju u radnom okruženju aplikacije. Razume se, nisu baš svi algoritmi pogodni za paralelizovanje, ali se značajno može ubrzati izvršavanje onih koji jesu.

Ovo izdanje knjige dopunjeno je tako da obrađuje verziju Java SE 7. Dodati su opisi mnogih novih mogućnosti, dopuna i proširenja.

Kultura inovacija

Java je od svojih početaka bila u središtu kulture inovacija. Njeno prvo izdanje promenilo je definiciju programiranja za internet. Javina virtuelna mašina (JVM) i bajt kôd izmenili su način razmišljanja o problemima bezbednosti i prenosivosti koda. Aplet (a potom i servlet) oživeli su dotad statičan internet. Proces Javine zajednice (Java Community Process, JCP) promenio je način usvajanja novih ideja i njihovog prenošenja u jezik. Pošto se Java koristi za programiranje Android sistema, ona je sastavni deo revolucije koja je pokrenuta s pojavom pametnih telefona. Svet Jave nikada nije dugo stajao u mestu.

Java SE 7 je najnovije izdanje u Javinoj neprekidnoj i dinamičnoj istoriji.