DEO

# Jezik Java

# POGLAVLJE 1

# Istorija i evolucija Jave

Ako želite potpuno da razumete Javu, morate da razumete 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 sa novim koncepcijama neophodnim za njenu jedinstvenu misiju. Dok ćemo se u narednim poglavljima ove knjige baviti praktičnim aspektima Jave – njenom sintaksom, njenim 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. Poboljšanje računarskih jezika i njihovo razvijanje nastaju iz dva osnovna razloga:

- Da bi se jezici prilagodili izmenjenom okruženju i novoj primeni.
- Da bi se u jezike ugradila poboljšanja i novosti na polju 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 od objektno orijentisanih osobina 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 koji su se tokom poslednjih nekoliko decenija odvijali na polju programskih jezika. 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 modernog programiranja: C

Pojava jezika C potresla 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 pri pravljenju sistemskih programa. Kao što možda i sami znate, kada se projektuje računarski jezik, često se prave kompromisi slični sledećim:

- lakoća korišćenja u odnosu na moć
- mogućnost pogrešne primene u odnosu na efikasnost
- strogost primene pravila u odnosu na proširivost.

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 namenjenih primenama u nauci, on nije sasvim dobar 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 što prvi računarski jezici, kao što su BASIC, COBOL i FORTRAN, nisu projektovani na osnovu strukturnih principa. 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" – masu prepletenih skokova i uslovnih grananja koje gotovo niko nije mogao da razume. Iako su jezici kao što je Pascal strukturirani, njima efikasnost nije bila osnovni cilj, a nedostajale su im i određene osobine koje bi ih učinile primenljivim za širok opseg različitih programa. (Konkretno, ako uzmemo u obzir standardne dijalekte Pascala koji su u to vreme postojali, Pascal nije bio jezik koji bi 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, 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 programskim jezicima.

Jezik C, koji je stvorio 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 Richards. BCPL je uticao na jezik, zvan B, koji je stvorio 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 Brian Kernighan i Dennis Ritchie opisali u knjizi *Programski jezik C* (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 modernog 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 sa 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 ovu osobinu.

## C++: sledeći korak

Krajem sedamdesetih i početkom osamdesetih godina dvadesetog veka, C je postao dominantan programski jezik, a i danas se široko 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 sledeće.

Pristup programiranju se dramatično izmenio od onog doba kada su se pojavili računari. Na primer, na prvim računarima je bilo potrebno da se sve binarne mašinske instrukcije ručno unesu 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 korišćeni jezik bio je, naravno, FORTRAN. Mada je on bio značajan korak u programiranju, teško da je podsticao pravljenje 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, i on ima ograničenja u savlađivanju složenih situacija. Kada program naraste na negde između 25.000 i 100.000 programskih redova, on postaje tako složen da je teško pojmiti njegovu celinu. C++ savlađuje ovu prepreku i omogućava programerima da veće programe shvate i njima upravljaju.

C++ je 1979. stvorio Bjarne Stroustrup dok je radio u Bellovim laboratorijama u Murray Hillu, New Jersey. Stroustrup je novi jezik prvobitno nazvao „C sa klasama“. Godine 1983. ovo ime 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, World Wide Web i Internet poprimiće svoj sadašnji oblik, a ovaj događaj će začeti novu revoluciju na polju programiranja.

## Nastanak Jave

Javu su prvobitno, 1991. godine, koncipirali James Gosling, Patrick Naughton, Chris Warth, Ed Frank i 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. Bill Joy, Arthur van Hoff, Jonathan Payne, 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 u odnosu 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 ovakvih uređaja koriste različiti tipovi procesora. Problem sa jezicima C i C++ (kao i sa 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 u tome 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 su počeli 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. Ovaj činilac bio je, naravno, World Wide Web. Da se Web nije uobličio približno u isto vreme kada i Java, ovaj programski jezik – upotrebljiv ali složen – možda bi se koristio samo za programiranje kućne elektronike. Međutim, sa pojavom World Wide Weba, Java se probila u prvi plan računarskih jezika, jer su i za Web bili potrebni prenosivi programi.

Većina programera vrlo rano nauči da su prenosivi programi, iako poželjni, prilično neostvarljivi. Iako je potraga za efikasnim, prenosivim (platformski nezavisnim) programima stara gotovo koliko i samo programiranje, potisnuli su je u drugi plan preči problemi.

Štaviše, pošto se većina korisnika računara svrstala 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 Weba, 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 različite platforme, 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 pravljenju koda za Internet. U stvari, isti problem zbog koga je Java prvobitno smišljena, pojavio se i na Internetu – samo u velikom obimu. To je prouzrokovalo da se fokus primene Jave premesti sa kućne elektronike na programiranje za Internet. Dakle, premda je želja za ostvarenjem 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, Java sa jezicima C i C++ ima zajedničke i one osobine koje su ova dva jezika učinili 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 okruženje Interneta, 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 sa jezikom C++ su znatne i, ako ste programirali na jeziku C++, osećate se kao „kod kuće" i sa 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 na polju 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 utelovljuje i nov način na koji ljudi pristupaju pisanju programa. Konkretno, Java je poboljšala i unapredila koncepciju objektne orijentisanosti koju je koristio i C++. Otuda Java nije jezik koji postoji sam za sebe, već predstavlja sadašnji izraz jednog procesa koji je započeo pre mnogo godina. Sama ova činjenica dovoljna je da Javi obezbedi mesto u istoriji računarskih jezika. 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 bio 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 svoje tehnologije .NET Framework. Oba jezika imaju istu opštu sintaksu, podržavaju distribuirano programiranje i koriste isti model objekta. 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 računarskim jezicima i način njihove upotrebe.

## Zašto je Java važna za Internet

Internet je pomogao da se Java probije u prve redove programiranja, a Java je, za uzvrat, jako uticala na Internet. Objašnjenje je sasvim jednostavno: Java širi skup objekata koji se slobodno mogu kretati kiber-prostorom. Na mreži između servera i vašeg ličnog računara kreću se dve veoma široke kategorije objekata: pasivni podaci i dinamički, aktivni programi. Na primer, kada čitate poruku elektronske pošte, vi gledate pasivne podatke. Čak i kada sa mreže preuzmete neki program, i njegov kôd je pasivan sve dok ga ne pokrenete. Međutim, postoji i druga vrsta objekata koja se može preneti: dinamički program koji se automatski izvršava. Takav program je aktivni agens u klijentskom računaru, pa ipak, aktivira se sa servera. Na primer, server može da obezbedi program za ispravno prikazivanje podataka koje šalje klijentu.

Iako su dinamički programi koji rade preko mreže poželjni, oni donose i ozbiljne probleme iz domena bezbednosti i prenosivosti. Pre nego što se pojavila Java, kiber-prostor je praktično bio zatvoren za polovinu žitelja koji u njemu sada obitavaju. Videćete da se Java bavi pomenutim problemima i da je na taj način otvorila vrata novom obliku programa: apletu.

### Java apleti

*Aplet* je posebna vrsta Java programa namenjena distribuciji preko Interneta i automatskom izvršavanju u čitačima Weba (engl. *Web browsers*) koji podržavaju Javu. Štaviše, aplet se s mreže preuzima na zahtev, baš kao slika, zvučna ili video sekvenca. Osnovna razlika je u tome što je aplet *inteligentan program*, a ne samo animacija ili multimedijalna datoteka. Drugim rečima, aplet je program koji može da reaguje na akciju korisnika i da se dinamički menja – a ne samo da večito ponavlja istu animaciju ili zvučnu sekvencu.

Ma koliko apleti bili uzbudljivi, oni bi ostali samo pusta želja kada Java ne bi mogla da se uhvati u koštac sa dva osnovna problema koji uz njih idu: bezbednošću i prenosivošću. Pre nego što nastavimo, definišimo najpre značenje ova dva izraza kada je u pitanju Internet.

### Bezbednost

Verovatno znate da, svaki put kada sa mreže preuzimate neki „normalan" program, rizikujete da računar zarazite virusom. Pre pojave Jave, većina korisnika nije baš često preuzimala izvršne datoteke, a oni koji su to činili, pre izvršavanja su proveravali da li su zaraženi virusima. I pored toga, većina korisnika je stalno brinula da ne zarazi svoj sistem. Osim virusa, postoje i druge vrste zlonamernih programa kojih se treba čuvati. Takvi programi mogu da prikupljaju lične podatke, kao što je broj kreditne kartice, stanje tekućeg računa

ili lozinka, tako što pretražuju sadržaj lokalnog sistema datoteka na vašem računaru. Java otklanja oba ova rizika stvarajući tzv. „zaštitnu barijeru“ (engl. *firewall*) između mrežne aplikacije i vašeg računara.

Kada koristite čitač Weba koji podržava Javu, možete bezbedno da preuzimate Java aplete bez bojazni od virusa ili zlih namera. Java obezbeđuje ovu zaštitu tako što program ograničava na okruženje za izvršavanje Java programa, ne dozvoljavajući mu pristup drugim delovima računara. (Uskoro ćete saznati kako to radi.) Mogućnost preuzimanja apleta bez straha od štete ili bojazni da će biti prekršena bezbednosna pravila, za mnoge predstavlja najveću novinu koju je Java donela.

### Prenosivost

Kao što smo pomenuli, u svetu se koristi više vrsta računara i operativnih sistema – a mnogi takvi računari su priključeni na Internet. Za programe koji treba da se dinamički preuzimaju na različite vrste platformi koje su povezane sa Internetom, mora postojati mogućnost generisanja prenosivog izvršnog koda. Ubrzo ćete saznati da isti mehanizam kojim se ostvaruje bezbednost izvršavanja apleta omogućuje i njihovu prenosivost. Javino rešenje za ova dva problema zaista je i elegantno i efikasno.

## 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*) je visokooptimizovan skup instrukcija koji u trenutku izvršavanja programa tumači (interpretira) Javin izvršni sistem, poznat kao *Javina virtuelna mašina* (Java Virtual Machine, JVM). U suštini, JVM je *interpretator bajt koda.* Ovo će vas možda iznenaditi, pošto je zbog postizanja boljih performansi većina savremenih jezika namenjena direktnom prevođenju u izvršni kôd, a ne interpretiranju. Međutim, činjenica da Java programe izvršava JVM, pomaže da se reše glavni problemi s preuzimanjem programa sa Interneta. Evo i zašto.

Prevođenje Java programa u bajt kôd omogućuje mnogo lakše izvršavanje u različitim okruženjima. Objašnjenje je jednostavno: za različite platforme potrebno je 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 koja postoje 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 prethodno direktno preveden u izvršni kôd. U Javi ova razlika u vremenu izvršavanja nije tako velika. Pošto je bajt kôd u velikoj meri optimizovan, njegovo korišćenje omogućuje Javinoj virtuelnoj mašini da programe izvršava mnogo brže nego što biste očekivali.

Mada je Java izvorno namenjena interpretiranju, sa 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 vrši različita proveravanja koja se mogu izvoditi samo u trenutku izvršavanja. Zbog toga Java prevodi kôd po potrebi, tokom izvršavanja. Sem toga, ne prevodi se sav 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 izvršavanja. Bezbednost i prenosivost programa ne smanjuju se kada se na bajt kôd primeni ovakvo dinamičko prevođenje, jer izvršnim okruženjem i dalje upravlja JVM.

## Pojmovi koji su oblikovali Javu

Priča o istoriji Jave ne može da bude potpuna ako se ne osvrnemo i na izraze koji se u njoj koriste. 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, daje prikaz osnovnih pravaca razmišljanja:

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

Dva od ovih pojmova, bezbedno i prenosivo, već smo objasnili. Pogledajmo šta podrazumevaju 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++, tada će vam prelazak na Javu biti sasvim bezbolan. 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 čistunske maksime „sve je objekat“ i pragmatičnog pravila „ne smetaj mi“. Model objekta u Javi jednostavan je i lako se proširuje, dok prosti tipovi, kao što su celi brojevi, nisu ni obuhvaćeni objektima jer su tako mnogo efikasniji.

## Robusno

Web 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). 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 ove probleme jer dodeljuje i oslobađa memoriju umesto vas. (U stvari, memorija se oslobađa potpuno automatski, pošto Java obezbeđuje čišćenje sistema od nekorišćenih objekata.) U tradicionalnom okruženju izuzeci često nastaju u situacijama kao što su deljenje 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 dizajnera Jave bilo je stvaranje prenosivog koda koji će trajno raditi. Kada danas napišete program koji radi, niko ne može 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 dizajneri 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 i zauvek". Ovaj cilj je najvećim delom i postignut.

### Interpretirano i visokoefikasno

Kao što je ranije objašnjeno, Java omogućuje pravljenje programa za više platformi prevodeći izvorni kôd u međuproizvod, zvan bajt kôd. Ovakav 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 izvršavanje procedura* (Remote Method Invocation, RMI). Dakle, program može da zahteva izvršavanje procedura koje se nalaze na bilo kojoj mrežnoj adresi.

### Dinamično

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

Sledeće glavno izdanje Jave bilo je verzija 1.3 proizvoda J2SE. Ova verzija predstavlja 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 sistem U/I. Izmenjene su i kolekcije i klase za umrežavanje. Sem toga, posvuda je učinjeno više malih izmena. Uprkos značajnom broju novih karakteristika, verzija 1.4 zadržala je gotovo stoprocentnu kompatibilnost izvornog koda s prethodnim verzijama.

Poslednje izdanje Jave nosi oznaku J2SE 5. Radi se o revoluciji!

## Revolucija zvana J2SE 5

U životnom ciklusu Jave, izdanje J2SE 5 predstavlja fundamentalan događaj. 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 (opšti) tipovi (engl. *generics*)
- metapodaci (engl. *metadata*)
- automatsko pakovanje (engl. *autoboxing*) i automatsko raspakivanje (engl. *auto-unboxing*)
- nabrojani tipovi (engl. *enumerations*)
- poboljšana petlja **for** u stilu for – each
- argumenti promenljive dužine (vararg)
- uvoz statičnih članova (engl. *static import*)
- formatiran U/I (engl. *formatted I/O*)
- skup klasa za konkurentno programiranje (engl. *concurrency utilities*)
- nadogradnje za API

To nije spisak minornih izmena niti parcijalnih nadogradnji. Svaka stavka spiska predstavlja značajan dodatak jeziku Java. Jedan deo stavki, kao što su generički tipovi, poboljšana petlja **for** i argumenti promenljive dužine, uvode nove elemente sintakse. Druge stavke, poput automatskog pakovanja i raspakivanja, menjaju semantiku jezika. Metapodaci dodaju programiranju novu dimenziju. U svakom slučaju, uticaj ovih 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 se tekući proizvod zove J2SE 5, a razvojni komplet – JDK 5. Da bi održao doslednost, Sun je odlučio da 1.5 upotrebljava kao svoj interni broj verzije. Dakle, 5 je eksterni, a 1.5 interni broj verzije.

---

***NAPOMENA*** *Pošto Sun koristi 1.5 kao interni broj verzije, kada zapitate prevodioca koji je njegov broj verzije, odgovoriće „1.5" umesto „5". I mrežna dokumentacija koju isporučuje Sun upotrebljava 1.5 kao oznaku mogućnosti dodatih u J2SE 5. Po pravilu, kad god vidite „1.5", to prosto znači „5".*

## 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 bezbednosti i prenosivosti. 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. Svet Jave nikada nije dugo stajao u mestu.

Izdanjem J2SE 5 stvorena je nova Java, prilagođena potrebama programskog okruženja koje se stalno menja. Java ponovo predvodi razvoj programskih jezika.