Poglavlje 1

Uvod u Javu

- JAVA KAO PROGRAMSKA PLATFORMA
- JAVINI SLOGANI IZ „BELOG PAPIRA"
- JAVA I INTERNET
- KRATAK ISTORIJAT JAVE
- UOBIČAJENE POGREŠNE PREDSTAVE O JAVI

Prvo pojavljivanje Jave izazvalo je ogromno uzbuđenje, ne samo u računarskim časopisima, već i u najznačajnijim medijima, kao što su *New York Times*, *Washington Post* i *Business Week*. Treba istaći da je to bio prvi i jedini programski jezik o kojem je emitovana 10-minutna reportaža na nacionalnom radiju (National Public Radio). Samo za proizvode koji nastaju korišćenjem ovog *specifičnog* računarskog jezika, bio je uspostavljen akcionarski fond vrednosti 100000000 $. Vrlo je zabavno prisetiti se tih divnih dana, pa ćemo vam u ovom poglavlju prikazati kratak istorijat jezika Java.

Java kao programska platforma

U prvom izdanju ove knjige, o jeziku Java morali smo da napišemo sledeće:

„Kada je u pitanju računarski jezik Java, u svemu se preteruje: Java je sigurno *dobar* programski jezik. Nema sumnje da je to jedan od boljih jezika koji je na raspolaganju ozbiljnim programerima. Smatramo da bi *potencijalno* to mogao postati odličan programski jezik, ali je za to već verovatno suviše kasno. Kada neki jezik jednom stupi na scenu, suočava se sa surovom realnošću kompatibilnosti sa postojećim kodom."

Zbog ovog pasusa naš urednik je dobio silne prekore od jedne jako važne osobe u kompaniji Sun, čije ime nećemo spominjati. Ali, kasnije se uvidelo da su naše prognoze bile tačne. Java ima veliki broj dobrih jezičkih karakteristika – pregledaćemo ih kasnije u ovom poglavlju. Ima i neke nedostatke, a novije dopune ovog jezika nisu tako elegantne kao prvobitne, zbog surove realnosti potrebe za kompatibilnošću.

Ali, kao što smo već rekli u prvom izdanju, Java nikad nije bila samo jezik. Postoji mnogo programskih jezika, od kojih su neki suviše razmetljivi. Jezik Java je čitava *platforma*, sa ogromnom bibliotekom koja sadrži kôd koji se može ponovo upotrebiti i sa izvršnim okruženjem koje obezbeđuje servise kao što su bezbednost, prenosivost kroz operativne sisteme i automatsko sakupljanje otpada.

Kao programer, želećete jezik sa prijatnom sintaksom i razumljivom semantikom (tj. ne C++). Java to ispunjava, kao i desetine drugih finih jezika. Neki jezici obezbeđuju prenosivost, sakupljanje otpada i slično, ali ne i pristojnu biblioteku, primoravajući vas da uspostavljate

sopstvenu, ukoliko želite lepu grafiku, pristup mreži, ili bazi podataka. Dakle, Java poseduje sve – dobar jezik, visokokvalitetno izvršno okruženje i ogromnu biblioteku. Ova kombinacija je razlog zašto je Java neodoljiv izbor za tako veliki broj programera.

„Beli papir" – slogani jezika Java

Autori jezika Java napisali su uticajni „Beli papir" u kome su objasnili svoje projektne ciljeve i dostignuća. Takođe su objavili kraći rezime koji je organizovan u skladu sa sledećih 11 slogana:

Jednostavan	Prenosiv
Objektno orijentisan	Interpretiran
Distribuiran	Visoke performanse
Robustan	Višenitan
Bezbedan	Dinamičan
Arhitektonska neutralnost	

U ovom odeljku:

- sumiraćemo, preko izvoda iz „Belog papira", šta Java projektanti kažu o svakom sloganu; i
- reći šta mi mislimo o svakom od njih, na osnovu naših iskustava sa aktuelnom verzijom jezika Java.

NAPOMENA: U trenutku dok ovo pišemo, „Beli papir" se može pronaći na lokaciji `http://java.sun.com/ocs/whitw/langenv/`. Rezime sa 11 slogana nalazi se na `ftp:javasoft.com/docs/papers/java-overview.ps`.

Jednostavan

> *Želeli smo da sačinimo sistem u kome bi se lako programiralo, bez mnogo ezoteričnih vežbanja, i koji koristi moć današnje standardne prakse. I pored toga što smatramo da je C++ nepodesan, jezik Java smo projektovali da bude što je moguće više blizak jeziku C++ kako bismo taj sistem učinili razumljivijim. Java ne sadrži mnoge retko korišćene, teško razumljive i zbunjujuće karakteristike jezika C++, koje na osnovu našeg iskustva donose mnogo više briga nego koristi.*

Zaista, sintaksa jezika Java je prečišćena verzija sintakse za C++. Nema potrebe za datotekama zaglavlja, aritmetikom pokazivača (čak ni za sintaksom pokazivača), strukturama, unijama, preopterećivanjem operatora, virtuelnim osnovnim klasama itd. (Pročitajte napomene za C++koje su umetnute tokom teksta da biste saznali više o razlikama između jezika Java i C++.) Međutim, projektanti nisu pokušavali da poprave sve nezgrapne karakteristike jezika C++. Na primer, sintaksa iskaza `switch` ostaje nepromenjena u jeziku Java. Ukoliko znate C++, prelazak na Java sintaksu vam neće predstavljati teškoću.

Ukoliko ste navikli na vizuelno programsko okruženje (kao što je Visual Basic), Java vam neće izgledati jednostavno. Postoji mnogo čudne sintakse (mada ne treba mnogo da se uhvati priključak). Što je još važnije, u jeziku Java morate mnogo više da programirate. Lepota Visual Basica je u tome što njegovo vizuelno projektno okruženje gotovo automatski obezbeđuje veliki deo infrastrukture neke aplikacije. U jeziku Java, odgovarajuće funkcionalnosti moraju se ručno programirati, što obično predstavlja dobar deo koda. Međutim, postoje nezavisna razvojna okruženja, koja obezbeđuju razvoj programa u stilu *prevuci i pusti*.

Sledeći aspekt jednostavnosti jeste biti mali. Jedan od ciljeva jezika Java jeste da omogući konstrukciju softvera koji može da se izvršava na usamljenim i nevelikim mašinama. Veličina osnovnog interpretera i podrške klasa je negde oko 40 kilobajta; dodatne osnovne standardne biblioteke i podrška za niti (suštinski samosadržavajući mikrokernel) povećava to za dodatnih 175 kilobajta.

To je veliko dostignuće. Međutim, imajte na umu da su biblioteke grafičkog korisničkog interfejsa (GUI) mnogo veće.

Objektno orijentisan

Jednostavno rečeno, objektno orijentisano projektovanje predstavlja tehniku programiranja fokusiranu na podatke (objekte) i na interfejse ka tim objektima. Ako bismo pravili analogiju sa stolarskim zanatom, objektno orijentisani stolar bio bi uglavnom usmeren na stolicu koju pravi, a manje na alate koje pri tome koristi; neobjektno orijentisani stolar prvenstveno bi razmišljao o svom alatu. Objektno orijentisane sposobnosti jezika Java su suštinski iste one kao kod jezika C++.

Objektna orijentacija je dokazala svoju vrednost u poslednjih 30 godina i nepojmljivo je da je moderni programski jezici ne koriste. Zaista, objektno orijentisane karakteristike jezika Java uporedive su sa onima za C++. Glavne razlike između jezika Java i C++ predstavljaju višestruko nasleđivanje, koje je kod Jave zamenjeno jednostavnijim konceptom interfejsa, kao i Javin model metaklasa. Mehanizam refleksije (pročitajte poglavlje 5) i karakteristika serijalizacije objekata (pročitajte poglavlje 12), u velikoj meri olakšavaju implementaciju trajnih objekta i GUI graditelja, koji mogu da integrišu nove komponente.

NAPOMENA: Ukoliko nemate iskustva sa objektno orijentisanim programskim jezicima, pročitajte pažljivo poglavlja od 4 do 6. U njima je objašnjeno šta je objektno programiranje i zbog čega je ono korisnije za programiranje sofisticiranih projekata od tradicionalnih proceduralno orijentisanih jezika kao što su C ili Basic.

Distribuiran

Java poseduje proširivu biblioteku rutina za kopiranje sa TCP/IP protokolima, kao što su HTTP i FTP. Java aplikacije mogu da pristupaju objektima preko mreže i preko URL-a, sa podjednakom lakoćom kao kada pristupaju lokalnom sistemu datoteka.

Ustanovili smo da su mrežne sposobnosti jezika Java ujedno i snažne i jednostavne za korišćenje. Svako ko je pokušao da se bavi programiranjem na internetu koristeći neki drugi jezik, uživaće u tome koliko Java pojednostavljuje zamorne zadatke kao što je otvaranje logičkog priključka. (Umrežavanje se obrađuje u delu 2 ove knjige.) Mehanizam udaljenog pozivanja metoda (*Remote Method Invocation*, RMI) omogućava komunikaciju između distribuiranih objekata (to se takođe obrađuje u delu 2).

Danas postoji i odvojena arhitektura, Java 2 Enterprise Edition (J2EE), koja podržava veoma širok spektar distribuiranih aplikacija.

Robustan

Jezik Java je namenjen za pisanje programa koji moraju biti pouzdani na mnogo načina. Java se u velikoj meri ističe u ranoj proveri mogućih problema, kasnijoj dinamičkoj proveri (tokom izvršavanja) i eliminaciji situacija koje su sklone generisanju greške... Najveća razlika između jezika Java i C/C++ jeste u tome što Java ima model pokazivača, koji eliminiše mogućnost upisivanja preko postojećeg sadržaja memorije i kvarenje podataka.

Ova karakteristika je takođe veoma korisna. Java kompajler otkriva mnoge probleme koji bi se, u drugim jezicima, pokazali samo pri izvršavanju. S druge strane, ova karakteristika jezika Java obradovaće svakog ko je, usled greške u pokazivaču, proveo sate u traženju oštećenih podataka u memoriji.

Ukoliko ste koristili jezik kao što je Visual Basic, koji eksplicitno ne koristi pokazivače, verovatno se pitate zašto je ovo toliko važno. C programeri nisu te sreće. Njima su potrebni pokazivači za pristup stringovima, nizovima, objektima, pa čak i datotekama. U Visual Basicu ne koristite pokazivače ni za jedan od ovih entiteta, niti treba da brinete o alociranju memorije za njih. S druge strane, mnoge strukture podataka su teške za implementaciju u jezicima koji ne koriste pokazivače. Java pruža ono najbolje iz oba sveta. Nisu vam potrebni pokazivači za svakodnevne konstrukcije kao što su stringovi i nizovi. Ukoliko vam je potrebna, snagu pokazivača imate na raspolaganju, na primer, za povezane liste. I uvek ste potpuno sigurni, jer ne možete pristupiti lošem pokazivaču, napraviti greške pri alociranju memorije ili morati da se štitite od curenja memorije.

Bezbedan

> *Java je namenjena korišćenju u mrežnim/distribuiranim okruženjima. Prema tome, bezbednost je prilično istaknuta. Java omogućava konstrukciju sistema koji su zaštićeni od virusa i zlonamerne modifikacije.*

U prvom izdanju rekli smo: „Dakle, nikad ne treba reći, nikad više" i izgleda da smo bili u pravu. Nedugo po isporuci prve verzije Java Development Kit, grupa bezbednosnih eksperata sa Univerziteta Prinston pronašla je jedva primetne greške u bezbednosnim karakteristikama Java 1.0. Sun Microsystems je podstakao istraživanja bezbednosti jezika Java, učinivši dostupnim za javnost specifikaciju i implementaciju virtuelne mašine i bezbednosnih biblioteka. Brzo su popravili sve poznate greške po pitanju bezbednosti. U svakom slučaju, veoma je teško nadmudriti bezbednosne mehanizme Jave. Dosad pronađene greške bile su tehničke prirode i malobrojne.

Od samog početka, jezik Java je projektovan da potpuno onemogući određene vrste napada, kao što su:

- prekoračenje izvršnog steka – uobičajeni napad crva i virusa;
- kvarenje memorije izvan njenog prostora za obradu;
- čitanje ili upisivanje datoteka bez dozvole.

Jedan broj bezbednosnih karakteristika dodat je u međuvremenu u jezik Java. Od verzije 1.1, Java sadrži pojam digitalno potpisanih klasa (pročitajte deo 2). Sa potpisanim klasama možete biti sigurni ko ih je pisao. Kada verujete autoru određene klase, možete odlučiti da takvoj klasi obezbedite veće privilegije na svojoj mašini.

NAPOMENA: Konkurentski Microsoftov mehanizam isporuke koda, koji se zasniva na ActiveX tehnologiji, po pitanju bezbednosti se oslanja jedino na digitalne potpise. Očigledno, to nije dovoljno – kao što može da potvrdi svaki korisnik Microsoftovih proizvoda, programi poznatih proizvođača padaju čineći štetu. Java poseduje mnogo snažniji bezbednosni model od ActiveX, pošto kontroliše aplikaciju dok se izvršava i zaustavlja je po potrebi, pre nego što napravi pustoš.

Arhitektonska neutralnost

> *Kompajler stvara objekat-datoteku, čiji je format neutralan što se tiče arhitekture – kompajlirani kôd se može izvršavati na mnogim procesorima, pod pretpostavkom prisustva Java*

izvršnog sistema. Java kompajler ostvaruje ovo tako što generiše tzv. bajtkôd instrukcije, koje nemaju nikakve veze sa arhitekturom korišćenog računara. Umesto toga, one su projektovane da se podjednako lako interpretiraju na svakoj mašini, kao i da se lako i u letu prevode u nativni mašinski kôd.

To nije nova ideja. Pre više od 20 godina, Niklaus Wirt, u svojoj originalnoj implementaciji Pascala, i kod UCSD Pascal sistema koristili su potpuno istu tehniku.

Naravno, interpretiranje bajtkoda je neminovno sporije od izvršavanja mašinskih instrukcija pri punoj brzini, pa se postavlja pitanje koliko je to dobra ideja. Međutim, virtuelne mašine imaju opciju za prevođenje najčešće izvršavanih sekvenci bajtkoda u mašinski kôd, postupak pod nazivom kompilacija *tačno na vreme*. Ova strategija je dokazala svoju efikasnost, tako da se čak i Microsoftova .NET platforma oslanja na virtuelnu mašinu.

Virtuelna mašina ima i druge prednosti. Ona podiže bezbednost pošto može da proverava ponašanje sekvenci sa instrukcijama. Neki programi čak u letu stvaraju bajtkôd, čime dinamički unapređuju sposobnosti programa koji se izvršava.

Prenosiv

Za razliku od C i C++, u ovoj specifikaciji ne postoje aspekti koji su zavisni od implementacije. Veličine primitivnih tipova podataka su određene, kao i njihovo ponašanje u aritmetici.

Na primer, `int` u jeziku Java uvek je 32-bitna celobrojna vrednost. U C/C++, `int` može da znači 16-bitna celobrojna vrednost, 32-bitna celobrojna vrednost ili bilo koja druga vrednost koju odredi proizvođač kompajlera. Jedino ograničenje je u tome da tip `int` mora imati najmanje bajtova kao `short int` i da ne može imati više bajtova od `long int`. Postavljanjem fiksne veličine za tipove brojeva, otklanja se veći deo glavobolje za prenosivost. Binarni podaci se čuvaju i prenose u fiksnom formatu, čime se eliminiše zbrka oko redosleda bajtova. Stringovi se čuvaju u standardnom Unikod formatu.

Biblioteke koje su deo ovog sistema definišu prenosive interfejse. Na primer, postoji apstraktna Window klasa i njene implementacije za UNIX, Windows i Macintosh.

Kao što svi koji su to nekada pokušali znaju, pisanje programa koji dobro izgleda na Windowsu, Macintoshu i na 10 vrsta UNIX-a zahteva napor herojskih razmera. Jezik Java 1.0 je ostvario taj napor obezbeđujući jednostavan komplet alata koji mapira uobičajene elemente korisničkog interfejsa na većem broju platformi. Nažalost, rezultat toga bila je biblioteka koja, uz silan rad, može da pruži rezultate koji su na različitim sistemima jedva prihvatljivi. (I obično su se pojavljivale različite greške u grafičkim implementacijama na različitim platformama.) Ali to je bio početak. Postoje brojne aplikacije kod kojih je prenosivost mnogo važnija od uglađenosti korisničkog interfejsa i koje su koristile rane verzije jezika Java. Do danas, komplet alata za korisnički interfejs je u potpunosti preuređen i više se ne oslanja na korisnički interfejs matičnog računara. Rezultat je mnogo konzistentniji i, po našem mišljenju, atraktivniji nego u ranim verzijama Jave.

Interpretiran

Java interpreter može direktno da izvršava Java bajtkôd na bilo kojoj mašini, na kojoj se taj interpreter postavi. Budući da je linkovanje postepen i lakši postupak, sam razvoj može biti brži i istraživački.

Postepeno linkovanje ima svoje prednosti, ali je njegov doprinos razvoju zapravo preuveličan. U svakom slučaju, ustanovili smo da su razvojni alati Jave prilično spori. Ukoliko ste navikli na brzinu klasičnog Microsoft Visual C++ okruženja, verovatno ćete biti razočarani performansama razvojnih okruženja Jave. (Međutim, aktuelna verzija paketa Visual Studio nije tako

munjevita kao klasična okruženja. Bez obzira na jezik u kome programirate, od svog poslodavca tražite da vam obezbedi brži računar da biste koristili najnovija razvojna okruženja.)

Visoke performanse

Mada su performanse prevedenog bajtkoda obično više nego prikladne, postoje situacije kada su potrebne više performanse. Bajtkôd može da se prevede u letu (tokom izvršenja) u mašinski jezik, za određeni CPU na kome se izvršava aplikacija.

Ukoliko za izvršenje bajtkoda koristite interpreter, „visoke performanse" ne bi bio termin koji bismo koristili. Na mnogim platformama, međutim, postoji takođe drugi oblik kompilacije, JIT (engl. *just-in-time*, tačno-na-vreme) kompajleri. Oni rade tako što jedanput kompiliraju bajtkôd u nativni kôd, keširaju te rezultate, a zatim ih pozivaju u slučaju potrebe. Ovakav pristup neverovatno ubrzava uobičajeno korišćen kôd pošto se prevođenje radi samo jednom. Iako i dalje neznatno sporiji od pravog kompajlera osnovnog koda, JIT kompajler može da pruži desetostuko, pa čak i dvadesetostruko ubrzanje nekih programa, a biće i gotovo uvek znatno brži od interpretera. Ova tehnologija se neprestano unapređuje, tako da se eventualno mogu pojaviti rezultati koji nisu ni uporedivi sa tradicionalnim sistemima za kompilaciju. Na primer, kompajler tačno-na-vreme može da nadgleda koji kôd se često izvršava i da po pitanju brzine optimizira upravo taj kôd.

Višenitan

Prednosti višenitne obrade su bolji interaktivni odzivi i ponašanje u realnom vremenu.

Ukoliko ste ikada pokušali da se bavite višenitnom obradom u nekom drugom jeziku, bićete prijatno iznenađeni kako je to jednostavno u jeziku Java. Niti kod Jave takođe mogu da koriste prednosti višeprocesorskih sistema ukoliko to čini i osnovni operativni sistem. Sa druge strane, na većini platformi implementacija niti se veoma razlikuje i, u tom smislu, jezik Java se ne trudi da bude nezavisan od platforme. Samo kôd koji vrši višenitno pozivanje ostaje isti na svim mašinama; Java prebacuje višenitnu implementaciju na operativni sistem na koji se oslanja, ili na biblioteku niti. (Niti se obrađuju u delu 2.) Na kraju, lakoća višenitnog programiranja predstavlja jedan od glavnih razloga privlačnosti Jave za programiranje na strani servera.

Dinamičan

Java je dinamičniji jezik od C ili C++, na mnoge načine. On je projektovan tako da se prilagođava okruženju koje evoluira. Biblioteke mogu slobodno da dodaju nove metode i promenljive primerka, bez ikakvih uticaja na klijente. U Javi je pronalaženje informacija u vremenu izvršavanja potpuno direktno.

Ovo je važna karakteristika u situacijama u kojima kôd treba da se doda u program dok se on izvršava. Pravi primer predstavlja kôd učitan sa interneta da se izvršava u pretraživaču. U verziji Java 1.0, utvrđivanje izvršnog tipa informacije bilo je sve samo ne potpuno otvoreno, dok aktuelne verzije pružaju programeru potpun uvid u strukturu i ponašanje njegovih objekata. Ovo je izuzetno korisno za sisteme koji treba da analiziraju objekte u toku izvršenja, kao što su Java graditelji grafičkog interfejsa, inteligentni alati za otklanjanje grešaka, priključive komponente i objektne baze podataka.

NAPOMENA: Microsoft je realizovao proizvod pod nazivom J++, koji pripada istoj familiji kao Java. J++ se takođe izvršava na virtuelnoj mašini, koja je kompatibilna sa virtuelnom mašinom jezika Java, za izvršavanje Java bajtkoda, ali postoje bitne razlike pri povezivanju sa spoljašnjim kodom. Osnovna jezička sintaksa je gotovo identična kao kod Jave. Međutim, Microsoft je dodao jezičke konstrukcije sumnjive upotrebljivosti, osim pri povezivanju sa Windows API intrefejsom. Pored toga što

dele zajedničku sintaksu, osnovne biblioteke ova dva jezika (stringovi, pomoćni programi, višenitna obrada, matematika itd.) u suštini su identične. Međutim, biblioteke za slike, korisničke interfejse i pristup udaljenim objektima, sasvim su različite. U ovom trenutku, Microsoft više ne podržava J++, već umesto toga uvodi nov jezik pod nazivom C#. On takođe ima mnogo sličnosti sa jezikom Java, ali koristi potpuno drugačiju virtuelnu mašinu. Postoji čak i J# za migraciju J++ aplikacija na virtuelnu mašinu koju koristi C#. U ovoj knjizi ne obrađujemo J++, C# ili J#.

Java i internet

Ovde je ideja jednostavna: korisnici će učitavati Java bajtkodove sa interneta i izvršavati ih na svojim mašinama. Java programi, koji rade na veb stranama, nazivaju se *apleti*. Da biste koristili neki aplet, potreban vam je veb pretraživač na kojem je omogućena Java. Budući da Sun licencira izvorni Java kôd i insistira na tome da neće biti promena u jeziku i u osnovnoj strukturi biblioteka, neki Java aplet će se izvršavati na svakom pretraživaču za koji važi da ima omogućenu Javu. Nažalost, stvarnost je drugačija. Različite verzije Netscape i Internet Explorer, izvršavaju različite verzije Jave, od kojih su neke ozbiljno zastarele. Ova manjkavost jako otežava razvoj apleta koji mogu da koriste prednosti najaktuelnijih verzija Jave. Da bi razrešio ovaj problem, Sun je razvio *Java dopunski modul* (engl. *Java Plug-in*), alat koji za Netscape i Internet Explorer čini dostupnim najnovije izvršno okruženje jezika Java (pročitajte poglavlje 10).

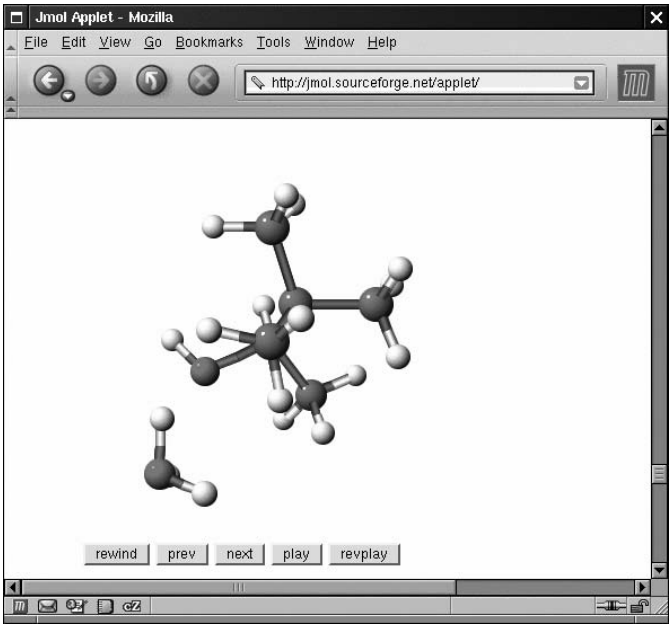
Kada korisnik učitava neki aplet, to u velikoj meri radi kao ugrađivanje slike u veb stranicu. Taj aplet postaje deo stranice, a tekst teče po prostoru koji se koristi za aplet. Poenta je u tome da je ta slika *živa*. Ona reaguje na komande korisnika, menja svoj izgled i šalje podatke između računara koji predstavlja određeni aplet i računara koji ga koristi.

Na slici 1-1 prikazan je dobar primer za dinamičku veb stranicu – aplet za pregled molekula – koji sprovodi sofisticirane proračune. Pomoću miša, možete rotirati i zumirati svaki molekul da biste bolje razumeli njegovu strukturu. Ova vrsta direktne manipulacije ne može se ostvariti sa statičkim veb stranicama, ali apleti i to omogućavaju. (Ovaj aplet možete pronaći na lokaciji `http://jmol.sourceforge.net`.)

Apleti mogu da se koriste za dodavanje dugmadi i ulaznih polja na veb stranicu. Ali učitavanje tih apleta preko telefonske linije je veoma sporo, a slične stvari možete postići koristeći dinamičke HTML i HTML obrasce, kao i skript jezik kao što je JavaScript. Naravno, rani apleti su bili korišćeni za animacije: poznati globusi koji se okreću, crtani junaci, pokretni tekstovi itd. Ali animirani GIF-ovi mogu da postignu isto. Prema tome, apleti su neophodni samo za obogaćenu interakciju, ali ne i za opšti veb dizajn.

Apleti na internet veb stranicama nisu ostvarili veliki uspeh, što je posledica nekompatibilnosti pretraživača i nepodesnog učitavanja koda apleta kroz spore veze. Na *intranetima* situacija je sasvim drugačija. Obično nema problema sa propusnim opsegom, tako da vreme učitavanja apleta ne dolazi u pitanje. Na intranetu je takođe moguće kontrolisati koji pretraživač se koristi, ili da se konzistentno koristi Java dopunski modul. Službenici ne mogu da pogreše lokaciju, ili da pogrešno konfigurišu programe koji se isporučuju preko veba pri svakom korišćenju, a sistem administrator nema potrebe da šeta naokolo i ažurira kôd na klijentskim mašinama.

Mnoge korporacije su programe koji se koriste za proveru inventara, planiranje odmora, isplatu dnevnica za službena putovanja itd., upakovale u aplete koji koriste pretraživače kao platformu za isporuku.



Slika 1-1 Aplet Jmol

U vreme pisanja ove knjige, značaj se ponovo prebacio sa programiranja usredsređenog na klijente na *programiranje-na-strani-servera*. Naročito zato što aplikativni serveri mogu da koriste sposobnosti nadgledanja virtuelne mašine Java za sprovođenje automatske raspodele opterećenja, povezivanje sa bazom podataka, sinhronizaciju objekata, bezbedno isključivanje i ponovno uključivanje, kao i druge operacije koje su neophodne za skalabilne serverske aplikacije, a koje se veoma teško mogu pravilno implementirati. Prema tome, programeri aplikacija bi trebalo da kupe i koriste, a ne da prave iz početka ove složene mehanizme. Time se njihova produktivnost povećava – jer mogu da se okrenu oblastima za koje su kompetentni, poslovnoj logici svojih programa, a ne „peglanju" performansi servera.

Kratak istorijat jezika Java

U ovom odeljku prikazan je kratak istorijat jezika Java. On se zasniva na različitim štampanim izvorima (od kojih je najznačajniji intervju sa kreatorima jezika Java, objavljen u onlajn magazinu *Sun-World* iz jula 1995. god.).

Početak jezika Java vezuje se za 1991. god., kada je grupa inženjera kompanije Sun, koju su predvodili Patrik Naughton i (jedan od rukovodilaca i svestrano usmeren računarski čarobnjak) James Gosling, odlučila da projektuje mali računarski jezik koji bi se mogao koristiti na korisničkim uređajima kao što su daljinski upravljači za kablovsku televiziju. Pošto ovakvi uređaji nemaju previše memorije, trebalo je da jezik bude mali i da generiše vrlo čvrst kôd. Takođe, pošto različiti proizvođači mogu da se opredele za različite centralne procesorske jedinice (CPU), bilo je važno da taj jezik ne bude vezan ni za jednu posebnu arhitekturu. Sam projekat imao je radni naziv „Green".

Potrebe za malim, čvrstim kodom koji je neutralan što se tiče platforme, dovela je ovaj tim do rehabilitacije modela kojeg su neki implementatori Pascala isprobavali u ranom periodu PC tehnologije. Niklaus Wirth, pronalazač Pascala, pionir je dizajna prenosivog jezika koji

generiše međukôd za neku hipotetičku mašinu. (Te hipotetičke mašine se obično nazivaju virtuelne mašine – otuda i termin Java virtuelna mašina ili JVM.) Ovaj međukôd bi zatim mogao da se koristi na bilo kojoj mašini koja ima odgovarajući interpreter. Inženjeri projekta Green takođe su koristili virtuelnu mašinu, čime su rešili glavni problem.

Međutim, kako su zaposleni u kompaniji Sun bili vezani za UNIX, oni su svoj jezik zasnovali na jeziku C++, a ne na Pascalu. Nadalje, oni su svoj jezik pravili kao objektno orijentisan, a ne kao proceduralni jezik. Ali, kao što Gosling kaže u intervjuu: „Sve vreme, jezik je bio alat, ne cilj". Gosling je odlučio da novi jezik nazove „Oak" (hrast), (pretpostavljam zato što mu se dopadao izgled hrastovog drveta koje se vidi kroz prozor njegove kancelarije). U kompaniji Sun su kasnije uvideli da već postoji računarski jezik pod tim imenom, pa su naziv promenili u Java. Ispostavilo se da je to bio inspirativan izbor.

Projekat Green je 1992. god. izbacio svoj prvi proizvod, pod nazivom „*7". Bio je to izuzetno inteligentan daljinski upravljač. (Imao je snagu SPARC stanice, u kutiji 20 cm × 15 cm × 15 cm.) Nažalost, niko u kompaniji nije bio zainteresovan za proizvodnju, pa je tim morao da pronađe druge načine da plasira svoju tehnologiju. Međutim, nijedna kompanija za proizvodnju standardne elektronske opreme nije bila zainteresovana. Grupa je zatim dala ponudu za projekat upravljača za kablovsku TV koji bi opsluživao nove kablovske servise, kao što je video snimak po zahtevu. Nisu dobili ugovor. (Interesantno je da je kompaniju koja je potpisala ugovor predvodio isti onaj Jim Clark koji je pokrenuo Netscape – kompaniju koja je veoma zaslužna za uspeh jezika Java.)

Projekat Green (pod novim nazivom „First Person.Inc.") proveo je celu 1993. i polovinu 1994. godine tražeći nekoga ko bi otkupio njihovu tehnologiju – bez uspeha. (Patrik Naughton, jedan od osnivača grupe i čovek koji je uradio najveći deo marketinga, tvrdi da je sakupio 300000 milja u avionskim letovima u pokušajima da proda ovu tehnologiju.) Projekat First Person raspušten je 1994. godine.

Dok se sve ovo dešavalo u kompaniji Sun, deo interneta nazvan veb rastao je neslućenom brzinom. Ključnu stvar za veb predstavlja pretraživač, koji prevodi stranu sa hipertekstom na ekran. 1994., većina ljudi koristila je Mosaic, nekomercijalni veb pretraživač koji je potekao iz superračunarskog centra na Univerzitetu Illinois, tokom 1993. god. (U stvaranju Mosaica delimično je učestvovao Marc Andreesen, za 6,85$ na sat, kao diplomac na istraživačko – radnom projektu. Slavu i sreću je pronašao kao jedan od osnivača i šef tehnologije u Netscapeu.)

U intervjuu za *SunWorld*, Gosling kaže da su sredinom 1994. godine projektanti jezika shvatili sledeće:„Mogli bismo napraviti stvarno dobar pretraživač. To je bila jedna od nekoliko stvari u glavnom toku klijent/server aplikacija, kojoj su bile potrebne neke od čudnih stvari koje smo razvili: arhitektonska neutralnost, rad u realnom vremenu, pouzdanost, bezbednost – elementi koji nisu preterano značajni u svetu radnih stanica. I tako, napravili smo pretraživač."

Aktuelni pretraživač napravili su Patrik Naughton i Jonatthan Payne i on je evoluirao u pretraživač HotJava. Ovaj pretraživač napisan je u jeziku Java da bi pokazao svu snagu ovog jezika. Ali projektanti su imali na umu i snagu nečega što je danas poznato kao apleti, tako da su omogućili pretraživaču da izvršava kôd unutar veb stranica. Ovaj „tehnološki dokaz" prikazan je na skupu SunWorld '95, 23.05.1995. god. i podstakao „ludilo" za jezikom Java koje traje i do danas.

Sun je objavio prvu verziju jezika Java, početkom 1996. godine. Ljudi su brzo shvatili da Java 1.0 neće ubrzati razvoj ozbiljnih aplikacija. Naravno, mogli ste da koristite Java 1.0 za izradu apleta za pokretni tekst, koji je pomerao tekst prema slučajnom izboru po slici. Ali u toj verziji niste mogli ni da ga *štampate*. Iskreno, jezik Java 1.0 u prvom trenutku nije bio sasvim

spreman. Njegov naslednik, u verziji 1.1, popunio je najuočljivije nedostatke, znatno unapredio sposobnost refleksije i dodao novi model događaja za programiranje GUI. Međutim, i dalje je bio prilično ograničen.

Najveća novost na konferenciji JavaOne 1998. godine bila je najava izlaska Java 1.2, u kojem su nezgrapni GUI i grafički alati, zamenjeni sofisticiranim verzijama sa podešavanjem veličina, čime je ovaj jezik prišao obećanju „Napiši jednom, izvršavaj bilo gde", mnogo bliže od svojih prethodnika. Tri dana(!) po njegovom objavljivanju u decembru 1998. godine, marketinško odeljenje kompanije Sun promenilo mu je naziv u zavodljivo *Java2 Standard Edition Softtware Development Kit Version 1.2*.

Osim „Standard Edition" (standardnog izdanja), uvedena su još dva izdanja: „Micro Edition" za uređaje kao što su mobilni telefoni, PDA-ovi itd. i „Enterprise Edition" za obradu na strani servera. Ova knjiga odnosi se na standardno izdanje.

Verzije 1.3 i 1.4 standardnog izdanja predstavljaju postepena poboljšanja u odnosu na inicijalno izdanje Java 2, sa uvek rastućom standardnom bibliotekom, unapređenim performansama i, naravno, nekoliko otklonjenih grešaka. Tokom tog perioda, stišala su se mnoga početna preterivanja o Java apletima i aplikacijama na strani klijenata, ali se zato jezik Java nametnuo kao platforma za aplikacije na strani servera.

Verzija 5.0 je prva posle verzije 1.1, koja je na značajan način ažurirala *jezik* Java. (Ova verzija je prvobitno označena sa 1.5, ali je broj verzije skočio na 5.0 na konferenciji JavaOne 2004.) Posle mnogih godina istraživanja, dodati su generički tipovi (koji se grubo mogu uporediti sa C++ šablonima STL) – izazov je bio u dodavanju ove karakteristike bez potrebe da se menja virtuelna mašina. Još nekoliko korisnih karakteristika jezika inspirisao je C#: novu sintaksu for ciklusa koja prolazi kroz sve elemente kolekcije, samoraspakivanje i metapodaci. Promene jezika uvek predstavljaju izvor problema sa kompatibilnošću, ali su neke od ovih novih jezičkih karakteristika veoma zavodljive, pa verujemo da će ih programeri oberučke prihvatiti. U tabeli 1-1, prikazan je razvoj *jezika* Java.

Tabela 1-1: Razvoj jezika Java

Verzija	Nove karakteristike jezika
1.0	Sam jezik
1.1	Unutrašnje klase
1.2	Nema
1.3	Nema
1.4	Tvrđenja
5.0	Generičke klase, novi for ciklus, promenljivi parametri, metapodaci, enumeracije, statički uvoz

U tabeli 1-2 prikazuje se rast *biblioteke* Java tokom godina. Kao što možete videti, veličina programskog interfejsa aplikacije (API) izuzetno se povećala.

Tabela 1-2: Rast Java Standard Edition API

Verzija	Broj klasa i interfejsa
1.0	211
1.1	477
1.2	1524
1.3	1840
1.4	2723
5.0	3270

Uobičajene pogrešne predstave o jeziku Java

Ovo poglavlje ćemo zaključiti listom nekih uobičajenih pogrešnih predstava o jeziku Java, zajedno sa komentarima.

Java je proširenje HTML.

Java je programski jezik; HTML predstavlja način za opisivanje veb stranice. Oni nemaju ništa zajedničko, osim što postoje HTML proširenja za postavljanje Java apleta na neku veb stranicu.

Koristim XML tako da mi Java ne treba.

Java je programski jezik; XML predstavlja način za opisivanje podataka. XML podatke možete obrađivati bilo kojim programskim jezikom, ali Java API sadrži odličnu podršku za XML obradu. Osim toga, mnogi važni XML alati nezavisnih proizvođača implementirani su u jezik Java. Da biste o ovome saznali više, obratite pažnju na deo 2.

Java je jednostavan programski jezik za učenje.

Nijedan programski jezik, koji je snažan kao Java, nije jednostavan. Uvek morate da pravite razliku između toga koliko je lako pisati programčiće i koliko je teško baviti se ozbiljnim poslom. Takođe, uzmite u obzir da se samo u četiri poglavlja ove knjige objašnjava jezik Java. Preostala poglavlja ove knjige pokazuju načine da taj jezik uposlite, koristeći Java *biblioteke*. Ove biblioteke sadrže hiljade klasa i interfejsa i na desetine hiljada funkcija. Srećom, nije potrebno da ih znate sve pojedinačno, ali i treba da znate mnogo toga kako biste za nešto ozbiljnije koristili jezik Java.

Java će postati univerzalni programski jezik za sve platforme.

Teorijski, ovo je moguće. Da se to dogodi sigurno je i želja svih proizvođača softvera, osim Microsofta. Štaviše, mnoge aplikacije koje savršeno dobro rade na stonim računarima ne moraju da rade dobro na drugim uređajima ili unutar pretraživača. Takođe, te aplikacije su pisane da koriste prednosti brzine procesora i nativne biblioteke korisničkog interfejsa, i svakako su isprobane na svim važnijim platformama. Među tim aplikacijama su programi za obradu teksta, programi za uređivanje fotografija i veb pretraživači. Po pravilu, pisani su u jezicima C ili C++, i ne vidimo nikakvu korist za krajnjeg korisnika u tome da ih sada ponovo piše u jeziku Java.

Java je samo još jedan programski jezik.

Java je lep programski jezik; većina programera mu daje prednost u odnosu na C, C++ ili C#. Ali postojalo je na stotine lepih programskih jezika koji nikada nisu ostvarili široku popularnost, dok su jezici sa očiglednim nedostacima, kao što su C++ i Visual Basic, bili izuzetno uspešni.

Zašto? Uspeh programskog jezika mnogo više zavisi od upotrebljivosti *sistema za podršku* koji ga okružuje, nego od elegancije njegove sintakse. Postoje li korisne, prikladne i standardne biblioteke za karakteristike koje treba da implementirate? Postoje li proizvođači softverskih alata koji sačinjavaju kvalitetna programska i okruženja za otklanjanje grešaka? Hoće li se taj jezik i skup alata integrisati sa preostalom računarskom infrastrukturom? Jezik Java je uspešan, jer njegova biblioteka klasa omogućava da na lak način radite nešto što je ranije bilo teško, kao što je umrežavanje i višenitna obrada. Činjenica da Java smanjuje greške pokazivača predstavlja jednu vrstu premije, tako da programeri mogu biti produktivniji ako koriste Javu. Ali ovi faktori nisu razlog za uspeh Jave.

Sada, kada je dostupan C#, jezik Java je zastareo.

C# je preuzeo mnoge dobre ideje iz Jave, kao što su čist programski jezik, virtuelna mašina i sakupljanje otpadaka. Ali iz ko zna kakvih razloga, C# je neke dobre stvari propustio, pre svega bezbednost i nezavisnost od platforme. Sa naše tačke gledišta, najveća prednost C# jeste njegovo odlično razvojno okruženje. Ukoliko ste vezani za Windows, C# ima mnogo smisla. Ali ako sudite prema oglasima za posao, Java je i dalje izbor većine programera.

Java je u privatnom vlasništvu i zato je treba izbegavati.

Ovo je nešto o čemu svako treba sam da odluči. Postoje trenuci kada smo frustrirani nekim aspektima jezika Java i kad želimo mogućnost da konkurentski tim obezbedi „open source" rešenje. Ali situacija nije tako jednostavna.

Iako Sun ima konačnu kontrolu nad jezikom Java, kroz „Java Community Process" uključeno je i mnogo drugih kompanija u razvoj revizija jezika i projektovanje novih biblioteka. Izvorni kôd za virtuelnu mašinu i biblioteke je slobodan i dostupan, ali samo za inspekciju, ne za modifikacije i redistribuciju.

Ako potražite druge „open source" jezike, nije sigurno da ćete zaključiti da su oni napravili bolji posao. Najpopularniji su tri „P" u grupi „LAMP" (Linux, Apache, MySQL i Perl/PHP/Python). Ovi jezici imaju svoje prednosti, ali takođe pate od poremećaja usled promene verzija, ograničenih biblioteka i nestašice razvojnih alata.

Sa druge strane spektra nalaze se C++ i C#, koji su standardizovani u nezavisnim komitetima za standardizaciju. Slažemo se da je ovakav postupak transparentniji nego što je to Java Community Process. Međutim, rezultati nisu bili tako upotrebljivi kako biste mogli pomisliti. Nije dovoljno da standardizujete samo jezik i većinu osnovnih biblioteka. Kod realnog programiranja, brzo prevazilazite rukovanje stringovima, kolekcijama i datotekama. U slučaju C#, Microsoft je zvanično potvrdio da će većinu biblioteka zadržati izvan postupka standardizacije.

Moguće je da budućnost Jave leži u „open source" procesu. Ali u ovom trenutku, Sun je uverio veliki broj ljudi da je odgovoran za upravljanje razvojem jezika Java.

Java se interpretira, tako da je suviše spor za ozbiljne aplikacije.

U ranom periodu jezika Java, on jeste bio interpretiran. Danas, osim na mikroplatformama kao što su mobilni telefoni, Java virtuelna mašina koristi kompajler tačno-na-vreme. Najvažniji elementi vašeg programa izvršavaće se jednako brzo u jeziku Java, kao što bi to bio slučaj u jeziku C++.

Java ima dodatno opterećenje u odnosu na C++, koje nema nikakve veze sa virtuelnom mašinom. Sakupljanje otpadaka je nešto sporije od ručnog upravljanja memorijom i Java programi zahtevaju više memorije od C++ programa slične funkcionalnosti. Pokretanje programa može biti sporo, posebno kod veoma velikih programa. Java GUI su sporiji od svojih nativnih suparnika, jer su oslikani u maniru koji je nezavisan od platforme.

Ljudi su se godinama žalili da je jezik Java sporiji od C++. Međutim, računari su danas mnogo brži nego u vreme kada su se pojavile ove pritužbe. Spor Java program će se i danas izvršavati mnogo bolje nego što se taj, zadivljujuće brzi C++ program izvršavao pre nekoliko godina. U sadašnjem trenutku, pritužbe zvuče kao „kiselo grožđe", a neki nezadovoljnici sada počinju da prebacuju kako je korisnički interfejs jezika Java zapravo ružan, a ne spor.

Svi Java programi se izvršavaju unutar veb stranice.

Svi Java *apleti*, izvršavaju se unutar veb pretraživača. To je definicija za aplet – Java program koji se izvršava unutar pretraživača. Ali sasvim je moguće, i prilično korisno, pisati izdvojene

Java programe koji se izvršavaju nezavisno od veb pretraživača. Takvi programi (koji se obično nazivaju *aplikacije*) su u potpunosti prenosivi. Samo preuzmite kôd i izvršite ga na nekoj drugoj mašini. Budući da je jezik Java pogodniji i manje sklon greškama od običnog jezika C++, on predstavlja dobar izbor za pisanje programa. To je još bolji izbor kada se uzmu u obzir i alati za povezivanje sa bazom podataka, kao što je Java Database Connectivity (pročitajte deo 2). Java je svakako očigledan izbor za prvi jezik u kome se uči programiranje.

Većina programa u ovoj knjizi su samostalni (engl. *stand-alone*) programi. Naravno da apleti predstavljaju zabavni deo. Ali samostalni Java programi su važniji i mnogo korisniji u praksi.

Java programi predstavljaju veliki sigurnosni rizik.

U ranim danima Jave, postojali su neki široko publikovani izveštaji o otkazima u Java bezbednosnom sistemu. Većina otkaza odnosila se na implementaciju Jave u određeni pretraživač. Istraživači su shvatili kao izazov da pokušaju da pronađu pukotine u oklopu Jave i da nadvladaju snagu i složenost bezbednosnog modela apleta. Tehnički nedostaci koje su oni pronalazili brzo su otklanjani, a koliko je nama poznato, nijedan od aktuelnih sistema nije bio ugrožen. Da biste ovo sagledali, uzmite u obzir doslovno milione napada virusa na izvršne datoteke Windowsa i makroe Worda, koji dovode do velikih neuspeha, uz neverovatno odsustvo kritike zbog slabosti napadnutih platformi. Takođe, ActiveX mehanizam Internet Explorera može da predstavlja plodno tlo za zloupotrebe. Ali, jednostavno, način da se on zaobiđe je toliko očigledan da se bezmalo niko nije ni trudio da to objavljuje.

Neki sistem administratori su čak deaktivirali Javu u pretraživačima preduzeća, ali i dalje dopuštaju svojim korisnicima da učitavaju izvršne datoteke, ActiveX kontrole i Word dokumente. To je prilično smešno – trenutno, rizik da vas napadnu neprijateljski Java apleti je možda uporediv sa rizikom nesreće pri letu avionom; rizik od infekcije pri otvaranju Word dokumenta uporediv je sa rizikom da poginete kada peške prelazite preko frekventnog autoputa.

JavaScript je pojednostavljena verzija Jave.

Jezik skriptovanja JavaScript, koji se može koristiti unutar veb strana, nastao je u kompaniji Netscape i prvobitno se zvao LiveScript. JavaScript poseduje sintaksu koja podseća na Javu, ali zapravo ne postoje nikakve relacije (osim, naravno, u nazivu). Podskup JavaScript je standardizovan kao ECMA-262, ali proširenja potrebna za realan rad nisu standardizovana tako da pisanje JavaScript koda koji se izvršava i u Netscapeu i u Internet Exploreru, može biti veoma frustrirajuće.

Sa jezikom Java, mogu svoj računar da zamenim internet uređajem *vrednim 500$.*

Kada se pojavio jezik Java, padale su različite opklade šta će se događati. Od prvog izdanja ove knjige, verovali smo da je apsurdno očekivati da će korisnici kućnih računara odustati od snažnih i udobnih stonih varijanti i opredeliti se za ograničene mašine bez lokalnog diska. Došli smo do zaključka da mrežni računar sa Java podlogom predstavlja pouzdanu opciju za „nulti nivo administracije" i smanjenje troškova kod posedovanja poslovnih računara, ali čak se ni to ne događa u većoj meri.

S druge strane, jezik Java se naširoko distribuira u mobilnoj telefoniji. Moramo priznati da još nismo sreli takvu Java aplikaciju za mobilne telefone za koju korisnici smatraju da je moraju imati, ali posao prodaje igrica i programa za zaštitu ekrana zauzeo je svoje mesto na mnogim tržištima.

SAVET: Da biste pronašli odgovore na uobičajena pitanja o Javi, pročitajte neki od Javinih FAQ listova na vebu: `http://www.apl.jhu.edu/~hall/java/FAQs-and-Tutorias.html`.