POGLAVLJE 1

Uvod u Javu

U ovom poglavlju:

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 desetominutna reportaža na nacionalnom radiju (*National Public Radio*). Samo za proizvode koji nastaju korišćenjem ovog *specifičnog* računarskog jezika, bio je uspostavljen akcionarski fond u vrednosti od 100 miliona dolara . Vrlo je zabavno prisetiti se tog uzbudljivog vremena, pa ćemo vam u ovom poglavlju prikazati kratak istorijat jezika Java.

1.1 Java kao programska platforma

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

„Ako sagledamo Javu kao kompjuterski jezik, tolika euforija je ipak malo preterana: Java je sigurno *dobar* programski jezik. Nema sumnje da je to jedan od boljih jezika koji je na raspolaganju ozbiljnim programerima. Smatramo da je Java imala potencijal da postane 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 Microsystems*, čije ime nećemo spominjati. Ali, kasnije se uvidelo da su

naše prognoze bile tačne. Java ima veliki broj dobrih karakteristika – pomenućemo ih detaljnije i naknadno u ovom poglavlju. Ima i neke nedostatke, a novije dopune ovog jezika nisu tako elegantne kao prvobitne, kako bi se očuvala kompatibilnost sa postojećim kodom.

Međutim kao što smo već rekli u prvom izdanju, Java nikad nije bila samo jezik. Postoji mnogo programskih jezika, od kojih neki privlače i previše pažnje. Programski jezik Java predstavlja čitavu *platformu* sa ogromnom bibliotekom koja sadrži kôd koji se može iskoristiti i sa izvršnim okruženjem koje omogućava bezbednost, izvršavanje istog koda na više operativnih sistema i automatsko oslobađanje upotrebljene memorije.[1]

Ukoliko se bavite programiranjem, želećete jezik sa prijatnom sintaksom i razumljivom semantikom (tj. ne kao jezik C++). Java to ispunjava, kao i desetine drugih jezika. Neki jezici obezbeđuju: izvršavanje koda na više operativnih sistema, automatsko oslobađanje upotrebljene memorije i slične mogućnosti, ali ne i pristojnu biblioteku, primoravajući vas da se snađete, ukoliko želite dopadljivu 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.

1.2 Karakteristike jezika Java

Autori jezika Java napisali su zvaničnu dokumentaciju tzv. „Belu knjigu " u kojoj su objasnili svoje projektne ciljeve i dostignuća. Takođe su objavili kraći rezime čija je stuktura organizovana u skladu sa sledećih 11 karakteristika programskog jezika Java:

1. Jednostavan
2. Objektno orijentisan
3. Distribuiran
4. Robustan
5. Bezbedan
6. Ima neutralnu arhitekturu
7. Prenosiv
8. Treba da se interpretira
9. Visokih performansi
10. Višenitan
11. Dinamičan

1 Garbage collection

U ovom odeljku:

- sumiraćemo na osnovu izvoda iz zvanične dokumentacije, šta Java projektanti kažu o svakoj njenoj karakteristici;
- reći ćemo šta mislimo o svakoj karakteristici, na osnovu naših iskustava sa aktuelnom verzijom jezika Java.

NAPOMENA: U trenutku dok ovo pišemo, zvanična dokumentacija se može pronaći na sajtu kompanije Oracle www.oracle.com/technetwork/java/langenv-140151.html. Rezime sa 11 karakteristika nalazi se na linku http://labs.oracle.com/features/tenyears/volcd/papers/7Gosling.pdf.

1.2.1 Jednostavan

> *Želeli smo da napravimo sistem u kome bi se lako programiralo, bez potrebe za komplikovanim uhodavanjem i koji koristi postojeći način razmišljanja. I pored toga što smatramo da je C++ neprikladan, jezik Java smo projektovali da bude što je moguće sličniji jeziku C++ kako bismo Javu učinili razumljivijom (za postojeće C++ programere). Java ne sadrži mnoge retko korišćene, teško razumljive i zbunjujuće karakteristike koje ima jezik C++i koje na osnovu našeg iskustva donose mnogo više problema nego koristi.*

Zaista, sintaksa jezika Java je unapređena verzija sintakse za C++. Nema potrebe za datotekama zaglavlja, pokazivačkom aritmetikom (čak ni za pokazivačkom sintaksom), 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. Java ima jako čudnu sintaksu (mada ne treba mnogo privikavanja). Što je još važnije, u jeziku Java morate mnogo više da programirate. Lepota Visual Basic-a je u tome što njegovo vizuelno projektno okruženje gotovo automatski obezbeđuje veliki deo infrastrukture neke aplikacije. U jeziku Java, morate programirati ručno odgovarajuće funkcionalnosti, što obično predstavlja dobar deo koda. Međutim, postoje nezavisna razvojna okruženja, koja obezbeđuju razvoj programa po principu *prevuci i pusti (drag and drop)*.

> *Sledeći aspekt jednostavnosti jeste mali izvršni fajl. Jedan od ciljeva jezika Java jeste da omogući konstrukciju softvera koji može da se izvršava samostalno na mašinama slabijih performansi. Veličina osnovnog interpretera i dodatnih klasa iznosi oko 40 kilobajta; osnovne standardne biblioteke i podrška za niti (suštinski mikrokernel) povećava veličinu za dodatnih 175 kilobajta.*

To je bilo veliko dostignuće za to vreme. Međutim, imajte na umu da je biblioteka sada prilično veća. Danas odvojeno postoji *Java Micro Edition* unutar koje se nalazi manja biblioteka, što je veoma pogodno za uređaje koji se ugrađuju unutar drugih mašina.

1.2.2 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; „stolar koji nije ,,objektno orijentisan'' 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 40 godina i nezamislivo je da je neki moderni programski jezik ne koristi. Zaista, objektno orijentisane karakteristike jezika Java uporedive su sa onima za C++. Glavne razlike između jezika Java i C++ ogledaju se u višestrukom nasleđivanju, koje je kod Jave zamenjeno jednostavnijim konceptom interfejsa, kao i modelom metaklasa (pročitajte poglavlje 5).

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.

1.2.3 Distribuiran

> *Java poseduje iscrpnu biblioteku rutina za rad 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 Java ima velike mrežne mogućnosti koje su pritom 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 poput otvaranja mrežne konekcije (socket-a). (Umrežavanje se obrađuje u knjizi II.) Daljinsko pozivanje metoda (*Remote Method Invocation,* RMI) omogućava komunikaciju između distribuiranih objekata (to se takođe obrađuje u delu 2).

1.2.4 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 u kojima lako dolazi do pojave grešaka... Najveća razlika između jezika Java i C/C++ jeste u tome što Java ima takav model pokazivača, koji eliminiše mogućnost upisivanja preko postojećeg sadržaja memorije i oštećenja podataka.

Ova karakteristika je takođe veoma korisna. Java kompajler otkriva mnoge probleme koji bi se u drugim jezicima pokazali samo pri izvršavanju. Sa druge strane, ova karakteristika jezika Java obradovaće svakog ko je, usled greške sa pokazivačima, proveo sate u traženju uzroka 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. Programeri za C nisu te sreće. Njima su potrebni pokazivači za pristup stringovima, nizovima, objektima, pa čak i datotekama. U Visual basic-u se ne koriste pokazivači ni za jedan od ovih entiteta, pa nije potrebno da brinete o alociranju memorije za njih. Sa 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 stvari kao što su stringovi i nizovi. Ukoliko vam je snaga pokazivača potrebna, imate je na raspolaganju, na primer, za povezane liste. Uvek ste potpuno sigurni, jer ne možete pristupiti lošem pokazivaču, napraviti greške pri alociranju memorije i ne morate da se štitite od curenja memorije[2].

1.2.5 Bezbedan

Java je namenjena korišćenju u mrežnim/distribuiranim okruženjima. Prema tome,mnogo truda je uloženo u bezbednost. Java omogućava konstrukciju sistema koji su zaštićeni od virusa i zlonamerne modifikacije.

U prvom izdanju rekli smo: „Dakle, nikad ne treba reći nikad " i bili smo u pravu. Nedugo po isporuci prve verzije *Java Development Kit*, grupa bezbednosnih eksperata sa Univerziteta Prinston pronašla je jedva primetne greške u bezbednosnim karakteristikama Jave 1.0. Kompanija Sun Microsystems je pokrenula istraživanja bezbednosti jezika Java tako što je učinila dostupnom za javnost specifikaciju i implementaciju virtuelne mašine i bezbednosnih biblioteka. Na taj način, brzo su popravili sve poznate greške po pitanju bezbednosti. U svakom slučaju, veoma je teško nadmudriti bezbednosne mehanizme Jave. Dosad pronađene greške bile su vrlo specifične i malobrojne.

2 Memory leaking = pojava da se memorija ne oslobađa nakon upotrebe

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

- prekoračenje izvršnog steka – uobičajeni napad crva i virusa;
- pristup memoriji izvan dela dodeljenog procesu;
- čitanje ili upisivanje datoteka bez dozvole.

Tokom vremena u Javu su dodati brojni bezbednosni mehanizmi. Od verzije 1.1, Java sadrži mogućnost digitalnog potpisivanja klasa (pročitajte drugi deo). 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: Majkrosoft koristi drugačiji sistem isporuke programa koji je baziran na tehnologiji *ActiveX* i oslanja se jedino na digitalne potpise. Očigledno, to nije dovoljno – kao što može da potvrdi svaki korisnik Majkrosoftovih proizvoda,čak i programi poznatih proizvođača nisu u potpunosti stabilni i mogu napraviti štetu. Java poseduje mnogo snažniji bezbednosni model od *ActiveX* pošto kontroliše aplikaciju dok se izvršava i zaustavlja je po potrebi, pre nego što izazove nepopravljive posledice.

1.2.6 Ima neutralnu arhitekturu

Kompajler stvara objektnu datoteku, čiji je format nezavisan od operativnog sistema na kome se pokreće– kompajlirani kôd se može izvršavati na mnogim procesorima, pod pretpostavkom prisustva izvršnog sistema Java. 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 odgovarajući mašinski kôd.

To nije nova ideja. Pre više od 40 godina, Niklaus Wirt, u svojoj originalnoj implementaciji Paskala, kao i UCSD Paskal sistem 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, i taj postupak se naziva *kompajliranje u letu (just-in-time)*. Ova strategija je dokazala svoju efikasnost, tako da se čak i Majkrosoftova .NET platforma oslanja na virtuelnu mašinu.

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

1.2.7 Prenosiv

Za razliku od C i C++, u ovoj specifikaciji ne postoje aspekti koji su zavisni od implementacije. Veličine primitivnih tipova podataka su fiksne, 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 problema oko prenosivosti. 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 sistema definišu interfejse koji su prenosivi. Na primer, postoji apstraktna Window klasa i njene implementacije za UNIX, Windows i Macintosh.

Kao što svi koji su nekada pokušali da pišu programe znaju, pisanje programa koji dobro izgleda na Windows-u, Macintosh-u 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 iskoristile prednosti ranih verzija jezika Java. Do danas, komplet alata za korisnički interfejs je u potpunosti napisan ispočetka i više se ne oslanja na postojeći korisnički interfejs računara. Rezultat je mnogo konzistentniji i, po našem mišljenju, atraktivniji nego u ranim verzijama Jave.

1.2.8 Treba da se interpretira

Isti Java bajtkôd može da se izvršava na svakom kompjuteru za koji postoji Java interpreter. Budući da je linkovanje postepen i lakši postupak, sam razvoj može biti brži i istraživački proces.

Postepeno linkovanje ima svoje prednosti, ali je njegov doprinos razvoju zapravo preuveličan. U početku, ustanovljeno je da su razvojni alati Jave prilično spori. Danas, bajtkodove prevodimo u mašinski kôd kompajliranjem u letu.

1.2.9 Visokih performansi

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

U početku se mnogi korisnici Jave nisu slagali sa tvrdnjom da su performanse „više nego dovoljne". Međutim, danas kompajliranje u letu postiže toliko dobre performanse da postaje konkurentno i sa tradicionalnim načinom kompajliranja i u većini slučajeva su performanse čak i bolje zato što se ovako dobija više informacija na raspolaganju. Na primer, kompajler u letu može da nadgleda koji se kôd često izvršava i da optimizuje upravo taj kôd. Postoji i sofisticiranija optimizacija, a to znači uklanjanje poziva metoda ili umetanje metoda na mesto poziva. Kompajler u letu tačno zna koje su klase učitane. Stoga, on može koristiti umetanje koje se zasniva na trenutno učitanoj kolekciji klasa, kada se određena metoda nikada ne zamenjuje[3] i takvu optimizaciju je moguće kasnije poništiti ukoliko je to potrebno.

1.2.10 Višenitan

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

Ukoliko ste ikada pokušali programiranje sa više niti 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 operativni sistem podržava takvu proceduru. 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 ili na biblioteku niti. Na kraju, lakoća višenitnog programiranja predstavlja jedan od glavnih razloga privlačnosti Jave za programiranje na strani servera.

1.2.11 Dinamičan

Java je dinamičniji jezik od C ili C++ iz više razloga. On je projektovan tako da se prilagođava okruženju koje se stalno unapređuje. Biblioteke mogu slobodno da dodaju nove metode i polja, bez ikakvih uticaja na klijente. U Javi je prilično jednostavno pronalaženje informacija prilikom izvršavanja programa.

Ovo je važna karakteristika u situacijama u kojima kôd treba da se doda u programu tokom izvršavanja. Pravi primer predstavlja kôd učitan sa interneta koji se izvršava u pretraživaču. U verziji Java 1.0, utvrđivanje izvršnog tipa informacije bilo je sve samo ne lako, 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 alati za pravljenje korisničkog interfejsa (GUI builder), inteligentni alati za otklanjanje grešaka[4], priključive komponente i objektne baze podataka.

3 Method override
4 Smart debugger

NAPOMENA: Veoma brzo nakon početnog uspeha Jave, Microsoft je napravio proizvod pod nazivom J++ sa programskim jezikom i virtuelnom mašinom koji su bili skoro identični kao i kod Jave. Microsoft više ne podržava J++, već umesto toga uvodi novi jezik pod nazivom C#. On takođe ima mnogo sličnosti sa jezikom Java, ali koristi drugačiju virtuelnu mašinu. Postoji čak i J# za migraciju J++ aplikacija za virtuelnu mašinu koju koristi C#. U ovoj knjizi nećemo se baviti J++, C# ili J# jezicima.

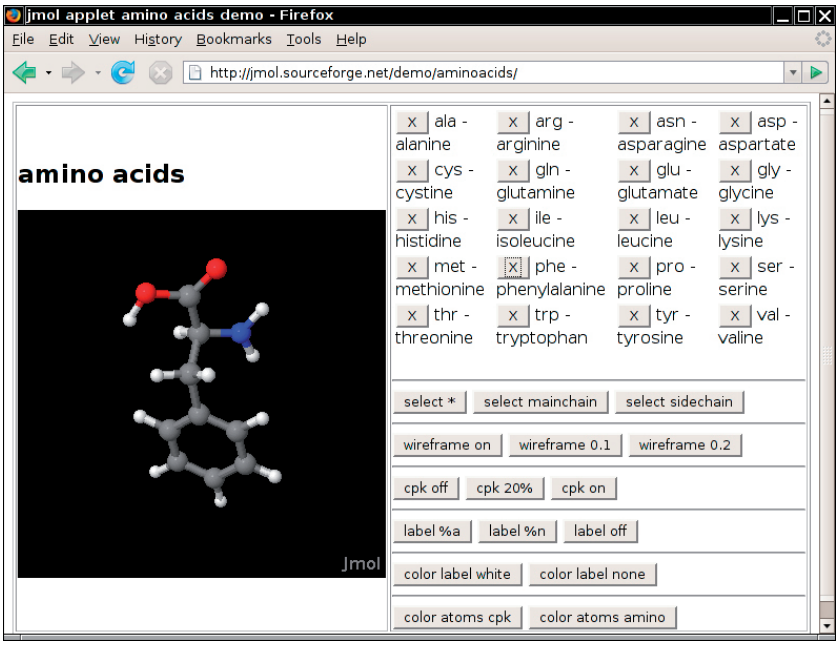
1.3 Java apleti i internet

Ovde je ideja jednostavna: korisnici će učitavati Java bajtkodove sa interneta i izvršavati ih na svojim mašinama. Java programi koji rade na internet prezentacijama nazivaju se *apleti*. Da biste koristili neki aplet, potreban vam je internet pretraživač na kojem je omogućena Java, tako da ne morate da instalirate nikakav softver. Budući da Oracle licencira izvorni Java kôd i insistira na tome da neće biti promena u jeziku i u osnovnoj strukturi biblioteka, Java aplet će se izvršavati na svakom pretraživaču koji ima podršku za Javu. Uvek kada posetite stranicu koja sadrži aplet, dobijate i najnoviju verziju programa. Zahvaljujući bezbednosnim merama virtuelne mašine, ne morate da brinete o napadima od strane malicioznog koda.

Kada korisnik učitava neki aplet, on u velikoj meri radi kao ugrađivanje slike u internet stranu. Taj aplet postaje deo stranice, a tekst se prostire oko prostora 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 isporučuje određeni aplet i računara koji ga pokreće.

Na slici 1-1 prikazan je dobar primer za dinamičku internet stranicu na kojoj su prikazani komplikovani proračuni. Aplet Jmol prikazuje molekularnu strukturu. Pomoću miša, možete rotirati i zumirati svaki molekul da biste bolje razumeli njegovu strukturu. Ova vrsta direktne manipulacije ne može se ostvariti statičkim internet stranicama, ali apleti i to omogućavaju. (Ovaj aplet možete pronaći na lokaciji `http://jmol.sourceforge.net`.)

Kada su se prvi put pojavili, apleti su izazvali veliko oduševljenje. Mnogi su ubeđeni da je privlačnost apleta doprinela velikom interesovanju za Javu. Ipak, početna euforija se uskoro pretvorila u veliku frustraciju. Različite verzije Netscape-a i Internet Explorer-a koristile su i različite verzije Jave, od kojih su neke čak bile i prilično zastarele, tako da je bilo izuzetno teško razviti aplete koji bi iskoristili pogodnosti najnovije Java verzije. Danas većina veb stranica u pretraživačima za potrebe dinamičkih efekata koristi JavaScript ili Flash. Sa druge strane, Java je postala najpopularniji jezik za razvoj serverskih aplikacija koje kreiraju internet stranice i izvršavaju logički deo aplikacije.



Slika 1-1 Aplet Jmol

1.4 Kratak istorijat jezika Java

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

Početak jezika Java vezuje se za 1991. godinu kada je grupa inženjera kompanije Sun, koju su predvodili Patrik Naughton i James Gosling (jedan od rukovodilaca i odličan i svestran računarski stručnjak), odlučila da projektuje mali računarski jezik koji bi mogao da se koristi 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 efikasan kôd. Takođe, pošto različiti proizvođači mogu da se opredele za različite procesore (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, efikasnim kodom koji je nezavisan od platforme na kojoj se izvršava, dovela je ovaj tim do ponovne upotrebe modela koji su koristile neke implementirane verzije Paskala u ranom periodu PC tehnologije. Niklaus Wirth, pronalazač Paskala, osmislio je dizajn 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 koristili UNIX, oni su svoj jezik zasnovali na jeziku C++, a ne na Paskalu. 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, a 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 pravi izbor.

Projekat Green je 1992. godine izbacio svoj prvi proizvod, pod nazivom „*7". Bio je to izuzetno inteligentan daljinski upravljač. (Imao je snagu SPARC stanice, u kutiji 15 cm x 10 cm x 10 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-po-zahtevu[5]. 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.)

Stručnjaci na projektu Green (pod novim nazivom „First Person.Inc.") proveli su 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 300,000 milja u avionskim letovima u pokušajima da proda ovu tehnologiju.) Projekat First Person napušten je 1994. godine.

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

5 video-on-demand

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: neutralnost arhitekture, 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. godine 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 verzija Java 1.0 nije podesna za razvoj ozbiljnih aplikacija. Naravno, mogli ste da koristite Java 1.0 za izradu apleta, recimo za pomeranje teksta prema slučajnom izboru po ekranu. Ali, u toj verziji niste mogli ni da štampate. Iskreno, jezik Java 1.0 u prvom trenutku nije bio sasvim spreman za širu upotrebu. Njegov naslednik, u verziji 1.1, popravio je najočiglednije nedostatke, znatno unapredio sposobnost refleksije i dodao novi model događaja za GUI programiranje. Međutim, i dalje je bio prilično ograničen.

Najveća novost na konferenciji JavaOne 1998. godine bila je najava izlaska Java 1.2 u kojem su nedorađeni GUI i grafički alati zamenjeni unapređenim verzijama kod kojih je bilo moguće podešavati veličinu elemenata. Time se ovaj jezik približio 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 ime: *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 uređaji itd. i „Enterprise Edition" za obradu na serverskoj strani. Ova knjiga je pisana za standardno izdanje.

Verzije 1.3 i 1.4 standardnog izdanja predstavljaju postepena poboljšanja u odnosu na prvobitno izdanje Java 2, sa uvek rastućom standardnom bibliotekom, unapređenim performansama i, naravno, dosta 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 značajno poboljšala *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) – izazov je bio u dodavanju ove mogućnosti bez potrebe da se izmeni virtuelna mašina. Još nekoliko korisnih karakteristika je inspirisano jezikom C#: nova sintaksa for ciklusa koja prolazi kroz sve elemente kolekcije, samoraspakivanje i metapodaci.

Tabela 1-1: Razvoj jezika Java

Verzija	Godina	Nove karakteristike jezika	Broj klasa i interfejsa
1.0	1996	Osnovna verzija jezika	211
1.1	1997	Unutrašnje klase	477
1.2	1998	Nema	1524
1.3	2000	Nema	1840
1.4	2002	Testiranje koda pomoću assert	2723
5.0	2004	Generičke klase, novi for ciklus, lista parametara promenljive dužine, metapodaci, enumeracije, statički uvoz	3279
6	2006	Nema	3779
7	2011	switch sa stringovima, diamond operator, zapis konstanti u binarnom sistemu, poboljšanja u pogledu rukovanja izuzecima	4024

Version 6 (without the .0 suffix) was released at the end of 2006. Again, there are no language changes but additional performance improvements and library enhancements.

As datacenters increasingly relied on commodity hardware instead of specialized servers, Sun Microsystems fell on hard times and was purchased by Oracle in 2009. Development of Java stalled for a long time. In 2011, Oracle released a new version with simple enhancements as Java 7, and decided to defer more ambitious changes to Java 8, which is expected in 2013.

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

1.5 Uobičajene zablude o jeziku Java

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

Java je proširenje HTML.

Java je programski jezik; HTML predstavlja način za opisivanje strukture veb stranice. Oni nemaju ništa zajedničko, osim što postoje HTML oznake za postavljanje Java apleta na neku internet 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 obradu XML podataka. Osim toga, mnogi važni XML alati nezavisnih proizvođača implementirani su u jezik Java. Da biste o ovome saznali više, obratite pažnju na Knjigu II.

Java je jednostavan programski jezik za učenje.

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

Java će postati univerzalni programski jezik za sve platforme.

Teorijski, ovo je moguće. Sigurno je to i želja svih proizvođača softvera, osim Microsofta. Međutim, mnoge aplikacije koje savršeno dobro rade na desktop računarima ne moraju da rade dobro na drugim uređajima ili unutar pretraživača. Takođe, te aplikacije su pisane da koriste prednosti brzine procesora i postojeće biblioteke korisničkog interfejsa, i svakako su isprobane na svim važnijim platformama. Među tim aplikacijama su programi za obradu teksta, programi za uređivanje fotografija i internet pretraživači. Po pravilu, pisani su u jezicima C ili C++, i ne vidimo nikakvu korist za krajnjeg korisnika u tome da se ponovo pišu 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 sa mogućnostima koje možete da iskoristite? Postoje li proizvođači softverskih alata koji proizvode kvalitetna programska okruženja i alate 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 komplikovane stvari uradite na lak način, poput umrežavanja i višenitne obrade. Java smanjuje greške u radu sa pokazivačima što je velika prednost, tako da programeri mogu biti produktivniji ako koriste Javu. Ali sve to nije presudno za uspeh Jave.

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

C# je preuzeo mnoge dobre ideje od Jave, kao što su jednostavan programski jezik, virtuelna mašina i automatsko uklanjanje upotrebljene memorije. Ali iz ko zna kakvih razloga, C# je neke dobre stvari izostavio, pre svega bezbednost i nezavisnost od platforme. Ukoliko ste vezani za Windows, C# ima mnogo smisla. Ali ako sudite prema broju oglasa za posao, Java je i dalje izbor većine programera.

Java je u privatnom vlasništvu i zato je treba izbegavati.

Kada je Java nastala, kompanija Sun je omogućila besplatne licence distributerima i krajnjim korisnicima. Iako je Sun imao potpunu kontrolu nad jezikom Java, uključeno je i mnogo drugih kompanija u razvoj novih verzija jezika i projektovanje novih biblioteka. Izvorni kôd za virtuelnu mašinu i biblioteke je oduvek bio dostupan javnosti, ali samo za pregled, ne i za modifikaciju i redistribuciju. Java je bila „closed source[6], koji se pristojno ponaša".

2007. godine dolazi do dramatične promene. Te godine kompanija Sun je objavila da će buduće verzije Jave biti dostupne pod GPL licencom (General Public License), što je zapravo ista licenca koju koristi Linux. Oracle je veoma posvećen tome da Java ostane open source. Ali postoji jedna sitnica, a to su patenti. Svako može dobiti pravo na patent radi korišćenja i modifikovanja Jave, koji podleže GPL-u, ali samo na desktop i serverskim platformama. Ako želite da koristite Javu u ugrađenim sistemima, potrebna vam je druga licenca, i verovatno ćete morati da to i platite. Ipak, ovi patenti će nestati u narednoj deceniji i tada će Java biti potpuno besplatna.

Java se interpretira, tako da je suviše spora 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 kompajliranje u letu. Kritični elementi vašeg programa izvršavaće se jednako brzo u jeziku Java, kao što bi to bio slučaj u jeziku C++, a u nekim slučajevima čak i brže.

Java ipak ima dodatno opterećenje u odnosu na C++. Vreme koje je potrebno za pokretanje virtuelne mašine je veoma sporo, a Java GUI je sporiji od standardnog GUI-a ugrađenog u OS, jer je iscrtan na način koji je nezavisan od platforme.

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

Svi Java programi se izvršavaju unutar internet stranice.

Svi Java *apleti*, izvršavaju se unutar internet pretraživača. To je definicija za aplet – Java program koji se izvršava unutar pretraživača. Većina Java programa su samo-

6 Termin koji označava da izvorni kôd nije moguće modifikovati ili upotrebljavati u drugim projektima

stalne aplikacije koje se pokreću izvan pretraživača. Zapravo, veliki broj Java programa izvršava se na web serverima i kreira kod za web stranice.

Java programi predstavljaju veliki sigurnosni rizik.

U ranim danima Jave, postojali su neki široko objavljeni izveštaji o propustima u Java bezbednosnom sistemu. Većina propusta odnosila se na implementaciju Jave u određenom pretraživaču. 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 uzrokuju veliku štetu, 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, što predstavlja daleko veći rizik. Čak i nakon 15 godina od svog nastanka, Java je daleko bezbednija od svih alternativa.

JavaScript je pojednostavljena verzija Jave.

Skript jezik JavaScript, koji se može koristiti unutar internet strana, nastao je u kompaniji Netscape i prvobitno se zvao LiveScript. JavaScript poseduje sintaksu koja podseća na Javu, ali zapravo ne postoji veza (osim, naravno, u nazivu). Podskup jezika JavaScript je standardizovan kao ECMA-262. JavaScript je daleko više integrisan u pretraživačima nego što su to Java apleti. Pored toga, program JavaScript ima mogućnost da modifikuje dokument koji se prikazuje, dok aplet može samo da kontroliše izgled na ograničenom prostoru.

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

Kada se pojavio jezik Java, padale su različite opklade da li će se ovo događati. Iako su kompanije proizvodile prototipove mrežnih računara koji su podržavali Javu, korisnici nisu blil spremni da se odreknu snažnog desktop računara kako bi ga zamenili ograničenom mašinom bez lokalnog diska. Došli smo do zaključka da mrežni računar sa Java podrškom predstavlja pogodnu opciju za ideju „računar bez administracije" i smanjenje troškova u poslovnom svetu, ali čak se ni to ne događa u većoj meri.

Sadašnja generacija tablet računara ne koristi Java programski jezik[7].

7 Recenzent prevoda se ne slaže sa autorom pošto je većina tableta danas bazirana na operativnom sistemu Android za koji se aplikacije programiraju u Javi. Autor je verovatno želeo da kaže da se Java ME ne koristi koliko je to ranije bio slučaj.