Univerzitet „Vitez“ Travnik

Fakultet Poslovne informatike
Studij I ciklusa; I godina
Smijer: Poslovna informatika

- Programski jezik Java-

SEMINARSKI RAD

Travnik 20. 05. 2015

Univerzitet „Vitez“ Travnik

Fakultet Poslovne informatike
Studij I ciklusa; I godina
Smijer: Poslovna informatika

- Programski jezik Java-

SEMINARSKI RAD

IZJAVA: Ja, Amel Nebić, student Sveučilišta/Univerziteta „Vitez“ Travnik, Indeks broj: 033-14/DPI odgovorno i uz moralnu i akademsku odgovornost izjavljujem da sam ovaj rad izradio potpuno samostalno uz korištenje citirane literature i pomoć profesora odnosno asistenta.

STUDENT: Amel Nebić
PROFESOR: Doc. dr Siniša Minić
ASISTENT: : mr. sc. Mahir Zajmović
PREDMET: Principi programiranja

Potpis studenta:_____

SADRŽAJ

UVOD

Seminarski rad posvećen je programskom jeziku Java. Kao što je i sam naslov, ovdje ćemo govoriti o samom programskom jeziku zvanom Java. Dakle, biti će pojašnjena sintaksa jezika. Skup tačno određenih pravila koje programeri moraju znati i kojih se moraju pridržavati kako bi računalo napravilo ono što i želimo. Da bi razumjeli ovaj seminarski rad moramo razumijeti razloge koji su uticali na stvaranje programskog jezika Java i znati razmišljati kao programer. Ovaj seminarski rad započet ćemo sa nastankom Jave a završiti programima.

U prvom poglavlju s naslovom uvod definisan je cilj seminarskog rada obrazložena struktura rada te opisan sadržaj određenih poglavlja.

Drugo poglavlje ima naslov nastanak programskog jezika gdje ćemo saznati kako i zašto je Java nastala i šta je ćini tako važnom i kako se mijenjala tokom godina. Dalje, ćemo definisati programski jezik Java i navesti njene prednosti i elemente. Onda ćemo obrazložiti zašto učiti programski jezik java.Dalje ćemo se upoznati sa mašinskim jezikom Dava Virtual Machine i sa pisanjem aplikacija u Javi i na kraju programiranje u Javi.

Posljednje poglavlje sa naslovom zaključak završni je dio seminarskog rada u kojem je sažet cijeli rad. Ma kraju rada iza posljednjeg poglavlja nalazi se popis literature koja je korištena pri izradi ovog seminarskog rada

NASTANAK PROGRAMSKOG JEZIKA JAVA

Java je programski jezik, koji je razvila kompanija *Sun Microsystems* pocetkom devedesetih godina. Mnogi koncepti Jave su bazirani na jeziku *Oberon* (autora Niklausa Virta, tvorca Paskala, Module i drugih jezika, i Hanspetera Musenbaha). Izbacili su koncept modula i uveli pakete kakve danas znamo, koji se oslanjaju na fajl sistem i uveli formalno koncept klasa iz objektno-orijentisane paradigme. Osim toga jezik ima sintaksu iz C i C++-a, ali je mnogo strožiji pri prevodenju, dizajniran tako da bude nezavistan od platforme, i sa pojednostavljenim upravljanjem memorijom. Pretpostavlja se da je ovo uradeno zbog popularnosti jezika C, ali i zbog jednostavnosti nekih struktura. Prva verzija je zvanicno objavljena 1995. Godine.[1]

Zvaničnog najavljivanja Jave u proleće 1995, doprinijelo je dosta osoba u 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!

Iako je postala nerazdvojivi deo mrežnog okruženja Interneta, treba istači da je Java na prvom mestu programski jezik. 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 namjenjog različitim elektronskim uređajima u domaćinstvu, kao što su mikrotalasne pečnice i daljinski upravljaći. Možda i sami pogađate da se kao kontroleri ovakvih uređaja koriste različiti tipovi procesora. Internet je pomogao da se Java probije u prve redove programirača, 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,

1 http://tutoriali.org/programski_jezik_java.html

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 prikazivaće 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 naåin otvorila vrata novom obliku programa: apletu. [2]

DEFINISANJE JAVE

Java je objektno orijentiran i univerzalan programski jezik, što znači da ju možete pisati i pokretati na bilo kojem operativnom sustavu, a kod pišete u skladu s objektno orijentiranom paradigmom.[3] -

Možemo reći da je Java:

- specifikacija programskog jezika i standardni zbir klasa
- implementacija navedenog programskog jezika i njegovih pratećih datoteka (*libraries*) u okolici prevođenja i izvođenja (*compile and run time enviroment*) za izradu i izvršavanje aplikacija
- implementacija navedenog programskog jezika kao podskup ugrađenog koda u *HTML* stranicama (*applet*)
- implementacija navedenog programskog jezika kao dodatak animaciji i interakciji kod *3D* objekata i scena (VRML 2.0)

2 Herbert Schildt, JAVA J2SE 5, MIKRO KNJIGA, ZAGREB, 2006

3 www.linuxzasve.com/uvod-u-java-programiranje-prvi-dio#sthash.N6tC9IlM.dpuf

Svaku od danih podstavaka prezentira drugačija implementacija Jave. Svaka od njih ima svojih prednosti kao i ograničenja. Važno je razumjeti iako Java programski jezik podržava neke vrste manifestacija, to ne znači da će pojedina manifestacija biti dopuštena ili čak moguća u svim implementacijama. Gledano s praktične strane, performansa i sigurnost nam ograničavaju što danas možemo napraviti unutar Java programskog okruženja. Programski jezik se koristi za izdavanje instrukcija računaru da obavi konkretne poslove.

POPULARNOST JAVE

Programski jezik Java stekao je veliku popularnost zbog svoje elegancije i savremene koncepcije. Programi napisani u Javi mogu se naći u mobilnim uređajima , Internet stranama i mnogim drugim okruženjima ljudskog djelovanja.[4]

Prvi razlog za popularnost je njena cena – potpuno je besplatna!. Mnogi drugi programski jezici prodaju ce po ceni od više stotina ili hiljada dolara, što je za većinu ljudi glavna prepreka da počnu da uče programiranje.

Drugi razlog za popularnost Jave je to što je Java programi mogu da se izvršavaju na skoro svakom tipu računara. Kažemo da su Java programi nezavisni od platforme na kojoj se izvršavaju. Java može da se koristi za razvoj raznovrsnih aplikacija. Postoje jednostavni tekstualno-zasnovani programi koji se nazivaju konzolnim aplikacijama. Takvi programi podržavaju samo tekstualni unos i ispis na monitoru vašeg računara. Takođe, možete da pravite aplikacije sa grafičkim korisničkim interfejsom (engl. Graphical User Interface – GUI). Ove aplikacije raspolažu sa menijima, paletama alatki, dugmadima, trakama za pomeranje sadržaja drugim kontrolama koje reaguju na miša. Primeri GUI aplikacija koje ste u svom radu na računaru već koristili su programi za obradu teksta, programi za rad sa tabelama i računarske igrice. Takođe, možete praviti i aplikacije koje se nazivaju apleti (engl. Applets). To su male GUI aplikacije koje mogu da se izvršavaju unutar web stranice. Apleti daju dinamičnost web stranicama. Mislim da ste već uvideli univerzalnost programskog jezika Java. Na ovom kursu krenućemo od prostih konzolnih aplikacija. To će nam omogućiti da se koncentrišemo na učenje

4 Dejan Živković Java programiranje.Univerzitet Singidunum Beograd 2011.god

osnova Jave bez upuštanja u svet grafičkog korisničkog interfejsa. Drugo popularno svojstvo Jave je to što je ona objektno orijentisana. To je fini način da se kaže da su Java programi sačinjeni od više osnovnih delova (komponenti) koji mogu da se više puta koriste. To za vas kao Java programera znači da možete da pravite i menjate velike programe bez većih komplikacija. Tokom ovog kursa više puta ćete se sresti sa terminom objekt (engl. Object), a tokom izlaganja kursa taj koncept će vam postati jasan. Poslednja prednost Jave je to što je ona prost jezik,u poređenju sa drugim programskim jezicima i zbog toga se relativno lako uči. Ta jednostavnost je neophodna da bi se podržala nezavisnost Java aplikacija od tipa platforme (sposobnost da se izvršava na svakom tipu računara). Međutim, ta jednostavnost ne znači da Java nije moćan programski jezik. Možete pomoću Jave da uradite sve ono što možete da uradite sa bilo kojim mnogo složenijim programskim jezikom.

Postoji puno drugačijih programskih jezika koji se koriste ili se tek razmatraju moguće upotrebe glede Internet pretražioca. Ono što daje velik korak prednosti Javi pred ostalim programskim jezicima je njena podržanost (postoji na svim operativnim sistemima gdje postoji Netscape program, što znači mnogo operativnih sistema), besplatnost za nekomercijalnu upotrebu (barem do ovog časa) te prenosivost izvršnog tj. bajtnog koda uz garantirano izvršavanje bez obzira na arhitekturu ili operativni sistem računala (naravno, ako se radi o istoj verziji Java prevodioca i interpretera te uz uvjet da ne koristimo native metode proširivanja standardnog zbira Javinih instrukcija).
Po primjerima iz prakse, dosad je programiranje HTML-baziranih servisa činila papazjanija sačinjena od FORM oznake i forme unutar HTML stranice, perl i sh skripti ili C-programa za obradu podataka pribavljenih sa forme, te driver programa za pristup bazi podataka pisanog najčešće u C-u. Ako ste se željeli baviti programiranjem mrežnih HTML-baziranih servisa, obično je preporuka glasila da pored osnovnog znanja barem HTML 2.0 specifikacija naučite programirati u C-u i perl-u a koristiti i pisati sh skripte ćete ionako "usput" naučiti.
Sada je nužno i dovoljno znati da vam od HTML specifikacija treba

```
<html><body><applet code="MyApplet.class" width=XXX
height=YYYY></applet></body></html>
```

.. te znanje programiranja u Javi.

Sama Java je daleko od savršenog programskog jezika. Zabilježene su duge rasprave o tome kako bi bilo izvedivo napisati drugačiji compiler ili točnije cross-compiler program koji bi prevodio neki drugi, jednostavniji i bolji izvorni kod u postojeći Javin bajtni kod. Na taj način se ne bi izgubilo apsolutno ništa od trenutnih dobrih karakteristika kao što je prenosivost bajtnog koda i njegovo izvršavanje na svim podržanim platformama računala.

Osim toga, veliki dijelovi Jave kao programskog jezika su već odbačeni kao neadekvatni, gledajući unatrag prema nižim verzijama. Izvrstan primjer toga je AWT (*Abstract Window Toolkit* -- zbir grafičkih alata) za koji sami tvorci Jave tvrde da je hack (na brzinu napravljen samo da proradi) i za to optužuju Netscape koji ih je manirama robo vlasnika tjerao na pridržavanje rokova te da što ranije završe sa projektom.

ELEMENTI PROGRAMSKOG JEZIKA JAVE

Skup znakova koji koristimo u jeziku Java čine mala i velika slova engleskog alfabeta Pravi se razlika između malih i velikih slova kako u službenim delovima programa tako i unutar običnih tekstova. U nastavku teksta pod pojmom slova podrazumevaju se mala i velika slova bilo kog alfabeta, a pod pojmom cifre decimalne cifre. Mada jezik Java predviđa upotrebu slova svih alfabeta, u službenom delu jezika koriste se samo slova engleskog alfabeta. Takođe, postoje radna okruženja koja ne podržavaju upotrebu slova bilo kog alfabeta. Ona obično podržavaju upotrebu slova engleskog alfabeta i, eventualno, još jedne grupe alfabeta. Na primer: latinična slovazapadno evropskih jezika, latinična slova istočnoevropskih jezika ili ćirilična slova. Leksički simboli (tokens) su nedeljivi nizovi znakova. U jeziku Java dele se na identifikatore, konstante, ključne reči, operatore i separatore. Leksički simboli mogu da se pišu, uz nekoliko izuzetaka, spojeno ili međusobno razdvojeno proizvoljnim brojem „belih” znakova. Beo znak između leksičkih simbola je neophodan, ako bi se njegovim izostavljanjem dobio drugi postojeći leksički simbol. U bele znakove (white spaces) spadaju znak za razmak, tabulacija, vertikalna tabulacija, prelazak u novi red i prelazak na novi list. U širem smislu, u bele znakove se ubrajaju i komentari. Komentari (comments) su proizvoljni tekstovi koji su namenjeni čitaocu teksta programa radi lakšeg razumevanja namene i funkcionisanja programa. Te tekstove prevodilac zanemaruje.

U jeziku Java postoje tri vrste komentara: • Komentari započeti sa // obavezno traju do kraja reda i tu se završavaju. Zgodni su za kratke komentare iza službenih elemenata s početka reda. • Komentari stavljeni između /* i */ mogu da se stave između bilo koja dva leksička simbola (kao i beli znakovi) i mogu da se protežu i kroz više redova. Mogu da posluže kao kratki umeci unutar redova ili da sadrže duža objašnjenja. • Komentari stavljeni između /** i */ su takozvani dokumentacioni komentari. Sadržaj takvih komentara program javadoc iz kompleta JDK koristi za sastavljanje dokumentacije programa. Mada i dokumentacioni komentari mogu da se umeću unutar redova, oni se obično protežu na više redova. Naredbe (statements) su nizovi leksičkih simbola. Dele se na deklarativne i izvršne naredbe. Deklarativnim naredbama (declarative statements) definišu se neki elementi programa (podaci, funkcije, klase itd.), izvršnim naredbama (executive statements) izvode se elementarne obrade. Programi su nizovi naredbi pomoću kojih se ostvaruju složene obrade podataka. Jezik Java ne postavlja nikakve uslove za raspoređivanje naredbi po redovima teksta programa. Jedna naredba može da se proteže u više redova i u jednom redu može da bude više naredbi. Osnovno načelo treba da bude što veća preglednost teksta celokupnog programa.[5]

ZAŠTO UČITI PROGRAMSKI JEZIK JAVU?

Postoji više razloga za to. Prvo, ukoliko znate da programirate, razumećete bolje kako računari rade. Drugo, pisanje programa je dobra vežba za razvijanje veštine razmišljanja – morate dobro logički da razmišljate da biste napisali računarske programe.. Takođe, morate pomalo biti i

5 Laslo Kraus PROGRAMSKI JEZIK Java sa rešenim zadacimaAKADEMSKA MISAO Beograd, 2013

perfekcionista, računari nisu toliko pametni i zato zahtevaju strogo precizne instrukcije da bi obavili svoje poslove. Treće, računarski programeri su veoma traženi i zarađuju mnogo novca. Na kraju, pisanje računarskih programa je zabavno. Posebno je zadovoljstvo kada vidite svoju ideju kako „živi" na ekranu računara. Zbog jednostavnosti Jave, brzo ćete naučiti kako se pišu Java programi. Međutim, to što ćete brzo napisati svoj prvi Java program ne znači da ste naučili sve što treba da znate o Javi. Ovaj kurs je samo kratak uvod u Javu, stoga smatrajte ga kao prvi korak na putu koji vodi do zvanja Java programera. Potrebni resursi Da biste pisali i izvršavali Java programe, potreban vam je Java Software Development Kit (Java SDK). To je besplatan proizvod koji možete preuzeti preko Interneta. To jednostavno znači da ćete iskopirati tzv. instalacionu datoteku na svoj računar, pokrenuti instalaciju i zatim početi sa radom. Ovaj proizvod možete naći na adresi http://java.sun.com. U sledećem nastavku biće opisano kako da preuzmete i instalirate ovaj softver. Java za mlade programere (2) Preuzimanje i instaliranje programa Da biste pisali i izvršavali Java programe, potreban vam je Java Software Development Kit (SDK). Radi se o besplatnom proizvodu koji možete preuzeti preko Interneta. To jednostavno znači da možete da kopirate tu datoteku na svoj računar i zatim instalirate taj paket. Potrebno je zato da uradite sledeće: Pokrenite svoj pretraživač weba (Internet Explorer, Netscape ili neki drugi) i odete na web sajt firme Sun Microsystems: http://java.sun.com Taj sajt sadrži obilje korisnih informacija o Javi. Što više budete programirali u Javi, to ćete sve češće posećivati ovaj sajt u potrazi za rešenjima svojih problema.[6]

6 http://download.tutoriali.org/Tutorials/JAVA/Java_za_mlade_programere.pdf

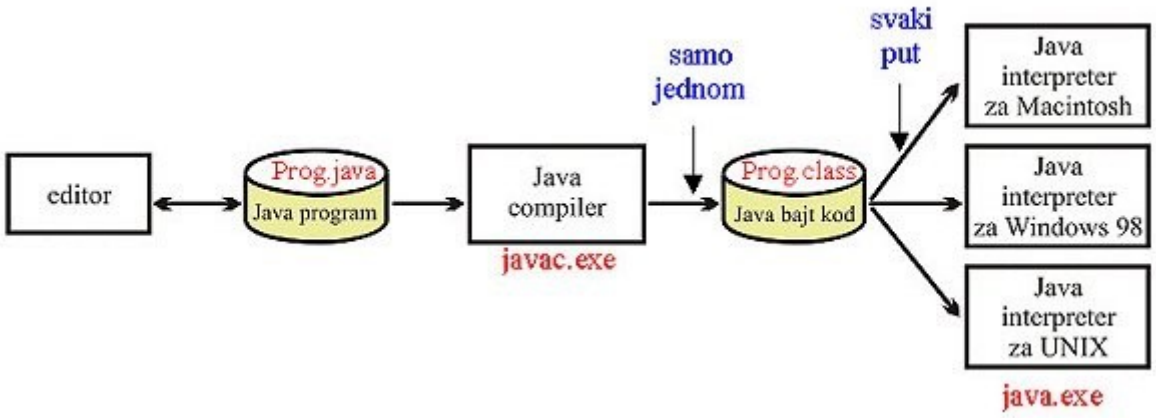
THE JAVA VIRTUAL MACHINE - JVM

Projektanti Jave su se odlučili na korištenje kombinacije kompilacije i interpretiranja. Programi pisani u Javi se prevode u mašinski jezik, ali u mašinski jezik računara koje zapravo ne postoji. Ovaj takozvani prividni (virtual) računar se zove "Java Virtual Machine". Mašinski jezik za Java Virtual Machine se zove Java bytecode. Ipak, jedna od glavnih prednosti Jave je da zapravo može biti korištena na bilo kojem računaru. Sve što je na tom računaru potrebno je interpreter za Java bajt kod. Takav interpreter oponaša Java virtual machine na isti način kao što prividno računar oponaša PC.

Mašinski jezik sastoji se od naredbi koje CPU može direktno izvršavati.

Kompajler (compile)prevodi program visokog nivoa u izvršni mašinski program.Jednom preveden program može se izvršavati neograničen broj puta ali samo na jednoj vrsti računara. Interpreter je program koji prevodi instrukciju po instrukciju izvornog koda i daje mogućnost korišćenja programa u mašinskom jeziku namenjenog za jedan tip računara, na sasvim drugom računa

Ipak, jedna od glavnih prednosti Jave je da zapravo može biti korištena na bilo kojem računaru. Sve što je na tom računaru potrebno je interpreter za Java bajt kod. Takav interpreter oponaša Java virtual machine na isti način kao što prividno računar oponaša PC. Java interpreter je potreban za svaku vrstu računara, ali nakon što računar dobije Java bajt kod interpreter, može izvršavati bilo koji Java bajt kod program. A isti taj Java bajt kod program može biti izvršen na bilo kojem računaru koje ima takav interpreter. Ovo je jedna od glavnih osob ršavati na više različitih vrsta računara.



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) oživjeli i 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. Izdaçem J2SE 5 stvorena je nova Java, prilagođena potrebama programskog okruženja koje se stalno mjenja. Java ponovo predvodi razvoj programskih jezika.[7]

7 Herbert Schildt, JAVA J2SE 5, MIKRO KNJIGA, ZAGREB, 2006

JAVA JEZIK ZA PISANJE APLIKACIJA

Javu ne koristimo samo kao dodatak vlastitim *Web HTML* stranicama tako da rade interesantne efekte sa slikama ili računaju koliko ste dana ili minuta stari. Javu možemo koristiti isto kao *C++* za pisanje *stand-alone* aplikacija.

Postupak je sljedeći:

- ***PREVODIMO JAVA IZVORNI OD U TZV BAJTNI KOD***
- ***IZVRŠAVAMO BAJTNI KOD INTERPRETIRANJEM UNUTAR JAVINE IRTUALNE MAŠINE (JVM)***

Vidimo da je po strukturi implementacije programskog jezika *Java* negdje na prijelazu između dosad kristalno jasnih definicija pojmova *kompajler* i *interpreter*. Ona spada u obje klasifikacije dok uistinu nije potpuno niti jedna od njih.

Sâm *bajtni kod* je dosta manji od ekvivalentnog izvršnog koda, recimo, *C*-a, ali brzina njegovog interpretiranja daleko zaostaje naspram brzine izvršavanja ekvivalentnih programa pisanih u *C*-u. Alternativno rješenje je u korištenju tzv. *just-in-time* interpretiranja, gdje *Javina Virtuelna Mašina* prebacuje *bajtni kod* u *native kod* prije samog izvršavanja. To za manje aplikacije pruža razumnu performansu uz zadržanu prenosivost izvršnog koda.

Međutim, čak i toliko naglašavana prenosivost možda ne počiva na čvrstim temeljima. Iole veće aplikacije će vjerojatno uslijed ograničenja nametnutih standardnim *Java* okruženjem biti prisiljene koristiti vlastita korisnička sučelja u vidu *native library* datoteka (*JNI*) čijim se funkcijama proširuju mogućnosti *Jave*. U tom slučaju mada je osnovna aplikacija prenosiva, morat će se prilagođavati od operativnog sistema do različitih arhitektura računala *library* datoteke pisane u *C/C++*-u bez kojih osnovna aplikacija neće moći raditi. [8]

PODACI I NAREDBE KAO OSNOVNI DIJELOVI PROGRAMA

8 http://web2tools-technologies.wikispaces.com/Java

Podaci i naredbe su dva osnovna dela programiranja . Za rad s podacima potrebno je razumevanje varijabli i tipova, , a za rad s naredbama potrebno je razumeti upravljačke strukture i potprograme.

Varijabla je samo lokacija u memoriji (ili nekoliko lokacija promatranih kao jedinica) kojoj je dodeljeno ime . Programer treba voditi računa samo o imenu, vođenje računa o memorijskom mestu dužnost je kompajlera.

Program je niz naredbi U običnom odvijanju računar izvodi ove naredbe redom kako se pojavljuju jednu za drugom. Ovo je vrlo ograničen način jer bi računar vrlo brzo ostao bez naredbi koje treba izvršiti. Upravljačke strukture su posebne naredbe kojekoje mogu izmijeniti tok odvijanja programa.

Postoje dvije osnovne vrste upravljačkih struktura: petlje, koje omogućavaju ponavljanje niza naredbi i grananja koja omogućuju računaru da odluči između više različitih postupaka ispitivanjem uslova koji se javljaju za vreme izvršavanjaprograma.

Na primjer, ako se želi ispisati natpise s imenom za svako ime u adresaru moglo bi se napisati "Uzmi prvo ime i adresu i ispiši natpis; uzmi drugo ime i adresu i" što vrlo brzo postaje jako smiješno i moglo bi uopće ni ne raditi ako unaprijed nije poznato koliko imena zapravo ima. Ono što bi zapravo htjeli reći je nešto kao: "Dok god ima imena za obradu, uzmi slijedeće ime i adresu i ispiši natpis." Ovakvo ponavljanje se u programu izražava petljom.

Potprogram se sastoji od naredbi za izvršavanje nekog zadatka okupljenih u celinu s imenom. Ime se kasnije koristi kao zamena za čitav niz naredbi. Na primjer, ako je jedan od zadataka koje program mora izvršiti crtanje kuće na ekranu, potrebno je naredbe okupiti u podprogram i dati mu prikladno ime, npr. "nacrtajKucu()". Nakon toga, na bilo kojem mjestu u programu gdje je potrebno nacrtati kuću dovoljno je napisati jednu naredbu: nacrtajKucu();

Ovo će imati učinak jednak kao ponavljanje svih naredbi za crtanje kuće na svakom mjestu. Prednost nije samo u tome da je manje kucanja. Organiziranje programa u podprograme također pomaže organiziranju razmišljanja i napora u razvoju programa. Za vrijeme pisanja

podprograma za crtanje kuće moguće se koncentrirati isključivo na problem crtanja kuće, bez razmišljanja o ostatku programa. Jednom kad je podprogram gotov, može se zaboraviti na detalje crtanja kuća - taj problem je riješen. Podprogram postaje kao ugrađeni dio jezika koji je moguće koristiti bez razmišljanja o tome što se događa unutar podprograma.

Organizovanje programa u podprograme također pomaže organiziovanju razmišljanja i napora u razvoju programa.
Varijable, tipovi, petlje, grananja i podprogrami su osnova onog što bi se moglo nazvati tradicionalnim programiranjem. Osim toga, kako programi rastu javlja se potreba za dodatnim strukturama za rešavanje njihove složenosti. Jedno od takvih sredstava je objektivno orijentisano programiranje.

Program je niz naredbi. U običnom odvijanju računalo izvodi ove naredbe redom kako se pojavljuju jednu za drugom. Ovo je očito vrlo ograničen način jer bi računalo vrlo brzo ostalo bez naredbi koje treba izvršiti.

Upravljačke strukture su posebne naredbe koje mogu izmijeniti tok odvijanja programa. Postoje dvije osnovne vrste upravljačkih struktura: petlje, koje omogućavaju ponavljanje niza naredbi i grananja koja omogućuju računalu da odluči između više različitih postupaka ispitivanjem uvjeta koji se javljaju za vrijeme izvršavanja programa.

Na primjer, moguće je da ako je vrijednost varijable "glavnica" veća od 10000, tada se "kamata" računa množenjem sa 0.05; a ako nije tada se kamata računa množenjem glavnice sa 0.04. Program treba neki način zapisa ove odluke. U Javi to je moguće ostvariti korištenjem sljedeće "if" naredbe:

```
if (glavnica > 10000)
                kamata = glavnica * 0.05;
        else
                kamata = glavnica * 0.04;
```

Detalji za sada nisu bitni, važno je zapamtiti da računalo može ispitati uvjet i odlučiti na osnovu toga što dalje)

Petlje se koriste kad istu radnju treba izvršiti više od jednom.
Na primjer, ako se želi ispisati natpise s imenom za svako ime u adresaru moglo bi se napisati "Uzmi prvo ime i adresu i ispiši natpis; uzmi drugo ime i adresu i" što vrlo brzo postaje jako smiješno i moglo bi uopće ni ne raditi ako unaprijed nije poznato koliko imena zapravo ima.

Ono što bi zapravo htjeli reći je nešto kao: "Dok god ima imena za obradu, uzmi slijedeće ime i adresu i ispiši natpis." Ovakvo ponavljanje se u programu izražava petljom.

JAVA PROGRAM

Program je niz naredbi koje računalo izvršava da bi obavilo neku zadaću. Da bi računalo moglo izvršavati naredbe, one moraju biti pisane na računalu razumljiv način, tj. u programskom jeziku. Programski jezici se razlikuju od ljudskog po svojoj jasnoći i strogosti što je u programu dozvoljeno, a što nije. Pravila koja određuju što je dozvoljeno zovu se sintaksa jezika. Sintaksna pravila određuju osnovni rječnik jezika i način na koji se programi mogu stvarati koristeći petlje, grananja i podprograme. Sintaksno ispravan program je onaj koji je moguće kompilirati ili izvršiti. Programi koji sadržavaju sintaksne greške će biti odbačeni uz poruku o grešci. Dakle, za postati uspješan programer potrebno je detaljno poznavati sintaksu korištenog programskog jezika.

Sintaksa je, ipak, samo dio priče. Naime, nije dovoljno samo napisati program koji radi, potreban je program koji daje ispravan rezultat, dakle program mora imati smisao. Smisao programa se zove semantika. Semantički ispravan program je onaj koji radi točno ono što programer traži od njega.

Pri uvođenju novih jezičnih elemenata bit će objašnjena i sintaksa i semantika elementa. Zapamtite sintaksu i pokušajte steći osjećaj za semantiku iz razumijevanja danih primjera. Pokušajte napisati i svoje kratke programe da bi provjerili svoje razumijevanje.

Naravno, čak i kad se upoznate sa svim pojedinačnim elementima jezika niste postali programer. Još uvijek morate naučiti kako stvarati složene programe za rješavanje određenih problema. Za to su potrebni i iskustvo i vještina.

Objašnjavanje Jave ćemo početi na uobičajen način: pisanjem programa koji ispisuje poruku "Hello World!". Naizgled jednostavan problem, ali ipak vrlo važan korak u svladavanju novog programskog jezika (pogotovo ako vam je to prvi jezik). Potrebno je da razumijete osnovne postupke:

1. unos programskog teksta u računalo

2. kompiliranje programa i
3. izvršavanje kompiliranog programa.

Ovi koraci su različiti za razna računala i Java programerska okružja. U osnovi se taj postupak svodi na pisanje programa u nekom tekstualnom editoru i snimanje programa u neku datoteku. Nakon toga određenom naredbom se program kompilira, nakon čega dobivate poruku o sintaksnim greškama ili kompilirani program. U slučaju Jave program se kompilira u bajt kod, umjesto u strojni jezik. Na kraju pokrećete program odgovarajućom naredbom. Vaše programsko okruženje može obaviti neke od ovih koraka za vas, ali se u pozadini zasigurno odvijaju ista tri koraka.

POSTUPAK RAZVOJA PROGRAMA

1. **Točno odrediti problem koji se želi riješiti.**

Programi se obično pišu da bi izvršili određeni zadatak, ali zadatak ne mora biti uvijek jasan sam po sebi. Potrebno je prikupiti dodatne podatke da bi se zadatak mogao točno odrediti. Jasno određivanje problema otklanja mogućnosti nesporazuma i olakšava postupak razvoja programa.

2. **Odrediti ulaze koje će program tražiti i izlaze koje će program stvarati**.
Ulazi i izlazi programa moraju biti određeni da bi se program dobro uklopio s drugim djelovima razvojnog postupka u jedinstvenu cjelinu.

3. **Rastaviti program na klase i pripadajuće metode**.
Odrediti jednu ili više klasa i njihovo djelovanje, međusobno i sa vanjskim svijetom. Za svako međudjelovanje odrediti zasebnu metodu.

4. **Razviti algoritme koji će biti primjenjeni u pojedinim metodama**.
Algoritam je opis postupka korak po korak do konačnog rješenja problema. Potrebno je pronaći logičan način za podjelu većih problema na manje sve dok se čitav zadatak ne podijeli na niz malih, jednostavnih i lako razumljivih zadataka. Nakon toga, ti mali zadaci se ponovo rastavljaju dok se ne dođe dijelova koji se mogu iskazati Java naredbama. Ovaj postupak se najčešće izvodi korištenjem pseudokoda.

5. **Prevođenje algoritma u Java naredbe**.
Ako je rastavljanje problema dobro obavljeno, ovaj korak se svodi na jednostavno zamjenjivanje pseudokoda odgovarajućim Java naredbama.

6. **Testiranje konačnog Java programa**.
Najduži i najvažniji dio razvojnog postupka. Dijelove programa je potrebno, ako je moguće, prvo testirati pojedinačno, a zatim i program u cjelini. Potrebno je provjeriti da program ispravno radi sa svim dozvoljenim vrstama ulaznih podataka. Često se događa da se program pisan i testiran samo na uobičajenom ulaznom skupu ruši ili daje netočne rezultate samo zbog korištenja različitog ulaznog skupa. Ako program sadrži grananja potrebno je provjeriti sva moguća grananja da bi se provjerilo ispravnost rada programa u svim mogućim uvjetima.

OSNOVNI JAVA APLET

Aplet je posebna vrsta Java programa namjenjena distribuciji preko Interneta i automatskomizvršavanju u čitačima Weba (engl. Web browsers) koji podržavaju Javu
Točnije, aplet je objekt koji pripada klasi java.applet.Applet ili jednoj od podklasa te klase.

Aplet je dio grafičkog sučelja, prikazuje se u prozoru (bilo mrežnog pretraživača ili nekog drugog programa), čini ga pravokutno područje u kojem su sadržani drugi elementi poput tipki ili tekstualnih polja. Može prikazivati grafičke elemente kao što su slike, pravokutnici ili linije, te može odgovarati na određene događaje (na primjer kad korisnik klikne negdje u apletu).

Klasa Applet, definirana u paketu java.applet, služi samo kao osnova za izradu podklasa. Objekt tipa Applet ima neka osnovna svojstva, ali zapravo ne radi ništa korisno, to je prazno područje na ekranu koje ne odgovara ni na kakve događaje. Da bi dobio upotrebljiv aplet, programer mora definirati podklasu koja nasljeđuje klasu Applet. U klasi Applet je definirano nekoliko metoda tako da ne rade baš ništa, pa programer mora nadjačati bar neke od ovih metoda i dati im neko značenje.

Apletu nije potrebna main() metoda jer on i nije samostalni program, iako više metoda u apletu podsjećaju na main() time što je zamišljeno da ih poziva sistem, a programer treba definirati odgovor na te pozive.

PROGRAMIRANJE U JAVA

Svakome tko je imao prilike usporediti C++ i Java izvorne kodove bit će odmah jasno da je Java bazirana na programskom jeziku C++. Tvorci Jave su željeli napraviti programski jezik koji bi bio jednostavniji za naučiti i koristiti nego C++. Iz tog razloga bili su prisiljeni odustati od dosta pristupa i idejnih rješenja koje programeri generalno smatraju zbunjujućima (znate li nekog tko koristi templateove u C++-u) te dodati nove mogućnosti kao što je primjerice garbage collection. Krajnji rezultat je novi programski jezik koji je uistinu lakši za savladati

od C++-a. Ugrađenim mogućnostima poput garbage collection i eliminiranjem pointerske aritmetike uspjeli su odstraniti najčešći izvor grešaka koje bi se javljale prilikom programiranja u C ili C++ programskim jezicima. No, kako ćemo vidjeti u daljnjem tekstu, ova poboljšanja nisu prošla besplatno...

C++ je trebao biti poboljšani C. Njegove mogućnosti korištenja objektno orijentiranih tehnika otvarale su mogućnost razvoja puno većih i bolje organiziranih programa. Pri njegovoj izradi tvorci su se pridržavali sljedećeg:

• C++ izvorni kod mora podržavati istu sintaksno/semantičku strukturu C-a te koliko je god moguće nadograditi se na sam C i podržavati tehnike programiranja korištene od strane C-programera
• Izvršni kodovi pisani u C++-u moraju biti barem isto toliko efikasni i brzi kao C izvršni kodovi kako bi se omogućila njegova primjena i u vremenski kritičnim rješenjima

Nadalje, programirajući u Javi ograničeni ste na korištenje već definiranih sučelja (interface). Java je u tom dijelu dosta slaba. Postojeće klase daju mogućnost Java programu izvršavanje samo sljedećih operacija:

• čitanje i pisanje datoteka
• crtanje točaka i drugih primitivnih 2D objekata u boji
• čitanje i manipulaciju slika
• kreiranje korisničkih sučelja (npr. korištenje više prozora na desktopu)
• komuniciranje preko mrežnih servisa (ne samo HTTP)
• audio-prikaz zvučnih datoteka

ZADACI U PROGRAMSKOM JEZIKU JAVA

Slijedi program "Hello World!". Ne očekujte da ćete sve razumijeti - neke dijelove ćete razumijeti tek nakon nekoliko lekcija.

```
/**
* HelloWorldApp klasa implementira aplikaciju koja
* ispisuje "Hello World!" na standardni izlaz.
*/

public class HelloWorldApp {
public static void main(String[] args) {
// Ispiši "Hello World!"
System.out.println("Hello World!");
}
}
```

Kada pokrenete ovaj program poruka "Hello world!" (bez navodnika) će biti ispisana na ekranu.

Naredba koja zapravo ispisuje poruku je:

```
System.out.println("Hello World!");
```

Ova naredba je primjer poziva podprograma. Koristi "ugrađeni" podprogram imena System.out.println da bi obavio zadatak. Prisjetite se da se podprogrami sastoje od naredbi za izvršavanje nekog zadatka, okupljenih i sa određenim imenom. Ime se koristi da bi mogli taj podprogram pozvati kad god treba izvršiti taj zadatak. Ugrađeni podprogram je već definiran kao dio jezika i time dostupan u bilo kojem programu.

Znatiželja vas mora voditi kroz ostatak programa. Dio programa su komentari. Računalo u potpunosti zanemaruje komentare u programu, oni su samo za ljude. Unatoč tome, komentari su vrlo važni, bez njih razumijevanje programa može biti jako teško. Java ima dvije vrste komentara. Prva vrsta počinje s: // i proteže se do kraja reda. Računalo zanemaruje // i sve nakon toga u istom redu. Druga vrsta komentara započinje s /* i završava s */ i može se protezati preko više redova.

Sve ostalo u programu je neophodno prema sintaksnim pravilima Jave. Svo programiranje u Javi se obavlja u klasama. Prva linija u programu kaže da se klasa zove HelloWorld. Ime klase

HelloWorld ujedno služi i kao ime programa. Klasa nije program sama po sebi, da bi mogla postati program klasa mora sadržavati podprogram imena main oblika:

```
public static void main(String[] args)
    izrazi
```

Kad naredite Java interpreteru da izvrši program, on poziva main() podprogram i izvršava naredbe unutar njega. Ove naredbe tvore skriptu koja govori računalu što točno treba raditi dok se program izvršava. Main() podprogram može pozivati podprograme definirane unutar iste, pa čak i drugih klasa, ali main() podprogram određuje kako i kojim redom će se koristiti drugi podprogrami.

Ime u prvoj liniji je ime programa, i ujedno ime klase. Ako se klasa zove HelloWorld, treba biti snimljena u datoteci HelloWorld.java, koja dakle sadrži source kod programa. Kada se ova datoteka kompilira, nastat će datoteka HelloWorld.class koja sadrži Java bajt kod. Java intrepreter izvršava Java bajt kod i za pokretanje programa ne trebate source kod.[9]

Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva Marko Topolnik, Mario Kušek

.

Prvi primjer

Pogledajmo prvi program napisan u Javi

Ispis: Prvi program

```
1 public class FirstExample
2 {
3 public static void main(String[] args)
4 {
5 System.out.println("Ovo radi!!");
```

9 Uvod u programski jezik JavaSkripta uz kolegij Informacija, logika i jezici

6 }

7 }

Program počinje definiranjem klase FirstExample (linija 1). Prvo se definira dostupnst klase koja je u ovom slučaju javna (public). Nakon toga slijedi ključna riječ class koja označava da se radio o klasi. Iza toga se stavlja naziv klase (FirstExample). Nakon toga dolazi tijelo klase koje se stavlja u vitičaste zagrade (linije 2–7). Tijelo klase se može sastojati od atributa i metoda.

U ovom slučaju imamo samo metode i to jednu (main). U lijini 3 je deklarirana metoda main. Ona je posebna metoda koja služi za pokretanje programa. Ima jedan parametar args. To je polje objekata String koje predstavlja parametre pokretanja. Svaka metoda započinje i završava vitičastim zagradama (linije 4 i 6). U tijelu metode je linija 5 koja ispisuje tekst "Ovo radi!!". Za sada je dovoljno zapamtiti da konstrukcija System.out.println ispisuje tekst koji je proslijeđen kao parametar.

Ako želimo napraviti program koji nas "pozdravlja" tako da mu pošaljemo ime u naredbenom retku onda to izgleda kao u ispisu 2.2.

Ispis 2.2: Parametar komandne linije

```
public class ArgumentExample
{
public static void main(String[] args)
{
System.out.println("Lijepi pozdrav " + args[0] + "!");
```

Razlika je u nazivu klase koja se sada zove ArgumentExample i u liniji u kojoj ispisujemo tekst imamo sljede´ce: "Lijepi pozdrav " + args[0] + "!". Ovaj dio koda spaja tekstualni niz u jedan. Prvi dio niza je: Lijepi pozdrav. Na taj dio se spaja argument koji je poslan kao parametar prilikom pokretanja i na njega je spojen znak !. Dakle, kod args[0] predstavlja prvi

element polja (indeksi počinju od 0).

Sada ga trebamo prevesti i onda pokrenuti sljedećom linijom:
> java ArgumentExample Amel
Ispis je:

Lijepi pozdrav Amel!

ZAKLJUČAK

Programski jezik Java je značajan u svijetu informatike. Java ima svoje mane – nasuprot nekih jezika je donekle sporija ali je vremenom napredovala. Java svoje nedostatke nadoknađuje tehničkim mogućnostima. Programski jezici se razlikuju od ljudskog po svojoj jasnoći i strogosti što je u programu dozvoljeno, a što nije. Pravila koja određuju što je dozvoljeno zovu se sintaksa jezika. Sintaksna pravila određuju osnovni rječnik jezika i način na koji se programi mogu stvarati koristeći petlje, grananja i podprograme. Programiranje u Javi putem objekata čini programeru život lakšim, sintaksa je pojednostavljena a način pisanja koda je dotaknuo apstraktnu razinu te dodatno približio programiranje ka ljudskom načinu

razmišljanja.Apleti, nude tržištu nešto novo i intresntnije. Java je vrlo dobar novi programski jezik i velika vijest među programerima.. Činjenice da se danas već udomaćio pojam NC (Network Computer), te da SUN tvrtka već naveliko prodaje svoje Java radne stanice bazirane na Java procesoru samo govore u prilog prijašnjoj tvrdnji. Mišljenje da će Java naslijediti **C++** kao jezik izbora glede programiranja generalno te da je Java u stvari ono što je trebao biti C++.

LITERATURA

- Joshua Bloch, Efikasno programiranje na Javi MIKRO KNJIGA, ZAGREB,2004
- Stephen J. Chapman: Java for Engineers and Scientist, Prentice Hall, NJ, 2000.
- Bruce Eckel: Thinking in *Java*, http://www.bruceeckel.com
- Dejan Živković Java programiranje.Univerzitet Singidunum Beograd 2011.god
- Herbert Schildt, JAVA J2SE 5, MIKRO KNJIGA, ZAGREB, 2006
- Laslo Kraus PROGRAMSKI JEZIK Java sa rešenim zadacima AKADEMSKA MISAO Beograd, 2013
- Metodološka zbirka zadataka za učenje C++ Mahir Zajmović

- Milosavljević, Vidaković: Java i Objektno-orijentisano programiranje
- Skripta uz kolegij Informacija, logika i jezici Zagreb Uvod u programski jezik Java Zagreb 2008

Internet;

- http://java.sun.com/docs/books/tutorial/
- http://tutoriali.org/Java.html
- http://tutoriali.org/programski_jezik_java.html
- http://download.tutoriali.org/Tutorials/JAVA/Java_za_mlade_programere.pdf
- http://web2tools-technologies.wikispaces.com/Java
- www.linuxzasve.com/uvod-u-java-programiranje-prvi-dio#sthash.N6tC9IlM.dpuf