Dr Marijan Mijatović
Nezavisni Univerzitet Banja Luka
Fakultet za informatiku, BiH

UDC 045(004.42)
Stručni članak

BUDUĆNOST PROGRAMIRANJA U JAVI

SAŽETAK: U današnjem vremenu svijeta informacijskih tehnologija i programskog jezika java ima svoje posebno mjesto u programiranju. A samim tim izbroj klasa kroz program. Java programski jezici su ti koji služe za izradu i izvršavanje aplikacija. Važno je napomenuti da je programski jezik Java baziran na programskom jeziku C++, programiranje u Javi se ograničava na korištenje već definiranog sučelja, ključne riječi programiranja je građa samog programskog jezika Java i C++ jezika. Java programski jezik je napredne generacije, i u sve većem broju se koristi za programiranje.

KLJUČNE RIJEČI: Java programiranje, programski jezici, C++, potprogrami, pseudokod.

UVOD

Činjenica koja nam otežava jednostavno definiranje ovog pojma jeste ta da je *Java* u suštini puno različitih stvari. Osim toga, pravi potencijal *Jave* u mnogome ovisi o kojoj *Javi* u stvari govorimo.

Za *Javu* danas možemo reći da je:

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

Svaku od zadanihstavki prezentiradrugačija implementacija *Jave*. Svaka od njih ima svoje 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 ograničavaju, što danas može napraviti unutar *Java*programskog okruženja.

OSNOVNI DIJELOVI PROGRAMA

Dva su osnovna dijela programiranja: podaci i naredbe. Za rad sa podacima potrebno je razumjevanje varijabli i tipova, a za rad sa naredbama potrebno je razumjeti upravljačke strukture (controlstructures) i podprograme (subroutines). Velikidio ovih predavanja posvećen je usvajanju ovih modela. Varijabla je samo lokacija u memoriji (ili nekoliko lokacija promatranih kao jedinica) kojemu je dodijeljeno ime da ga se u programu može lako pozivati i koristiti. Programer treba voditi računa samo o imenu, vođenje računa o memorijskom mjestu dužnost je kompajlera – prevoditelja. Programer ne smije gubiti iz vida da ime upućuje na neku vrstu mjesta u memoriji koje može spremiti podatke, čak i ako ne mora znati gdje je to točno u memoriji.

U Javi i većini ostalih jezika varijabla ima tip (type) koji ukazuje na vrstu podataka koje može spremiti. Jedna vrsta varijabli može sadržavati integer – cjelobrojne vrijednosti, dok druga sadrži floatingpoint - brojeve s decimalnim mjestima. Računalo pravi razliku između integer 17 i floatingpoint 17.0, zapravo u računalu izgledaju sasvim različito. Također postoje i tipovi za pojedinačne znakove, nizove znakova, kao i za manje uobičajene tipove kao što su datumi, boje, zvukovi ili bilo koji drugi tip podataka koje bi program mogao spremiti.

Programski jezici uvijek imaju naredbe za stavljanje i vađenje podataka u i iz varijabli i za obradu tih podataka. Na primjer, sljedeća „izjava dodjeljivanja“, koja se može pojaviti u Java programu, kaže računalu da uzme podatak iz varijable „glavnica“, pomnoži taj broj sa 0.07 i spremi rezultat u varijablu „kamata“:

```
kamata = glavnica * 0.07;
```

Također postoje i ulazne naredbe za uzimanje podataka od korisnika ili iz datoteka na diskovima računala i izlazne naredbe za slanje podataka u suprotnom smjeru. Ove osnovne naredbe – za pomicanje podataka s mjesta na mjesto i za obradu podataka su dijelovi za izradu svih programa. Ovi dijelovi su složeni u složene programe korištenjem upravljačkih struktura i potprograma.

Program je niz naredbi. U običnom odvijanju računala izvodi ove naredbe redom kako se pojavljuju jednu za drugom. Ovo je očito vrlo ograničen način jer bi računalo vrlo brzo ostao 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;
```

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 jedan puta. Na primjer, ako je potrebno 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.

Veliki programi su tako složeni da bi ih bilo gotovo nemoguće napisati bez da ih se „razbije“ u lakše ostvarive dijelove. Potprogrami su jedan od načina ostvarivanja tog „razbijanja“. Potprogram se sastoji od naredbi za izvršavanje nekog zadatka okupljenih u cjelinu s imenom. Ime se kasnije koristi kao zamjena za čitav niz naredbi. Na primjer, ako je jedan od zadataka koje program mora izvršiti crtanje kuće na ekranu, potrebno je naredbe okupiti u potprogram 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 potprograme također pomaže organiziranju razmišljanja i napora u razvoju programa. Za vrijeme pisanja potprograma za crtanje kuće moguće se koncentrirati isključivo na problem crtanja kuće, bez razmišljanja o ostatku programa. Jednom kad je potprogram gotov, može se

zaboraviti na detalje crtanja kuća – taj problem je riješen. Potprogram postaje kao ugrađeni dio jezika koji je moguće koristiti bez razmišljanja o tome što se događa unutar potprograma.

Varijable, tipovi, petlje, grananja i potprogrami su osnova onog što bi se moglo nazvati tradicionalnim programiranjem. Osim toga, kako programi rastu javlja se potreba za dodatnim strukturama za rješavanje njihove složenosti. Jedno od najučinkovitijih sredstava je objektno orijentirano programiranje OOP.

RAZVOJ PROGRAMSKOG JEZIKA

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 dijelovima 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.

PROGRAMIRANJE U JAVI

Svako tko je imao prilike usporediti *C++*i *Java* izvorne kodove bilo mu je 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 te dodati nove mogućnosti kao što je primjerice *garbagecollection*. Krajnji rezultat je novi programski jezik koji je uistinu lakši za savladati od *C++*-a. Ugrađenim mogućnostima poput *garbagecollection* i eliminiranjem *pointerske aritmetike* uspjeli su odstraniti najčešći izvor grešaka koje bi se javljale prilikom programiranja u *C* ili*C++*programskim jezicima.

*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

Dalje, 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,

ZAKLJUČAK

Java je vrlo dobar novi programski jezik i velika vijest među programerima. Informatički časopisi još uvijek ne prestaju pisati o i oko *Jave* te kako će upravo *Java* promijeniti dosadašnje poimanje kratice *WWW* (*World Wide Web*), korisnik/poslužitelj modela razvoja aplikacija te ekonomskog modela prezentacije i korištenja programa počevši od tabličnih kalkulatora pa sve do video-igrica. 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. Postoji već utvrđeno 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

[1] Joshua Bloch: (2004). Efikasno programiranje na JAVI, Zagreb.
[2] Oracle Corporation. (2016). https://java.com/en/download/faq/whatis_java.xml.
[3] Baltes-Götz B., Götz J., (2012). Einführungindas Programmieren mit Java.
[4] Fain, Y. (2016). Programiranje Java, IT Expert.
[5] Topolnik, M., Kušek, M., (2016). *Uvod u programski jezik Java*.
[6] Čupić, M. (2016). *Programiranje u Javi*.
[7] Imtiaz, A. (2016). Absoluteintroduction to ObjectOrientedProgrammingin Java.
[8] Meloan, S. (2016). JavaOne 2012 Review: Make the Future Java.
[9] Potter, P. (2016). How many Java developers are intheworld.
[10] Stephen, J. Chapman: Java for EngineersandScientist, Prentice Hall, NJ, 2000.
[11] Krsnik, R. (1996). *Fizika 1, Udžbenik za nastavu fizike u 1. razredu gimnazije*, II. Izdanje. Zagreb: Školska knjiga.
[12] Morin D., (2004). IntroductoryClassicalMechanics, withProblemsandSolutions, http://www.physics.harvard.edu/people/facpages/morin.html
[3] Persson, A. (2005). HistoryofMeteorology 2 – TheCoriolisEffect: Four centuries of conflict between common sense and mathematics.
[14] Goldstein, H. C. Poole, J. Safko. (2000). *Classicalmechanics*, 3. izdanje, AddisonWesley.
[15] Flanagan, D. (2005). *Java in a Nutshell*, 5. izdanje, O'Reilly.
[16] Schildt, H. (2004). *Java 2: The Complete Reference*, 5. izdanje, McGraw-Hill.

Marijan Mijatović, Ph.D.

JAVA PROGRAMMING LANGUAGE

Summary

In today's world of information technology and programming language, java has it sown special place in programming. Also the team and thesum of the classes through the program Java programming languages are the ones used to create and execute applications. It is important to note that the Java programming language is based on the C ++ programming language, Java programming islimited to the use of the already definedinter face, programmingkeywordsistheveryessenceoftheprogramminglanguage Java and C ++ languages. Java programming language is an advanced generation, and is use dinin creasing numbers for programming.

Key words: Java programming, programming languages, C ++, subprograms, *pseudocode*.