Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva
Zavod za elektroničke sustave i obradbu informacija

Podatkovni višemedijski prijenos i računalne mreže

JAVA

Davor Živko
mbr: 0036385503
INE

Zagreb, siječanj 2005

Sadržaj:

1. Uvod

Cilj ovoga seminara je dati pregled nekih osnovnih pojmova koji se vežu uz pojam cjelokupne Java tehnologije, jezika, njegovih karakteristika, povijesti nastanka, razvoja i smjera kojim će se tehnologija razvijati.

Programski jezik Java, kao i cijela tehnologija koja stoji iza jezika, nastao je kao rezultat rada na projektu pokrenutom u tvrtci Sun 1991. godine. Projekt je bio poznat pod nazivom: "the Green Project", a stvaranje novoga programskog jezika čak nije bio ni cilj tog projekta. Cilj projekta je bio predvidjeti smjer kojim će se kretati stanje u računarstvu u to doba i tek nakon objavljivanja nekoliko demo uređaja i dugotrajnog traženja područja primjene, poput televizije i potrošačke elektronike, rezultati rada na projektu su našli svoje opravdanje na internetu. U to vrijeme internet je postajao popularan medij za prijenos kompleksnih sadržaja, poput video i audio sadržaja, putem mreže heterogenih računala koristeći HTML (hyper textual markup language).

Java tehnologija je bila dizajnirana za prijenos medijskog sadržaja putem računalne, heterogene mreže, ali je, uz to, imala mogućnost prenošenja i "ponašanja" putem mreže pomoću apleta (*applet*), zajedno sa sadržajem. Java tehnologija je službeno "objavljena" u svibnju 1995.godine i od tada je višestruko prerasla originalni koncept i postala vodeća tehnologija i okruženje za razvoj aplikacija u mrežnom računalstvu. Otkako je predstavljena, u svibnju 1995.godine, Java platforma postala je najbrže prihvaćena nova tehnologija u povijesti računalstva.

Svoj nevjerovatni uspjeh Java treba zahvaliti potpuno drugačijem i novom konceptu na kojem je razvijena. Budući da je inicijalno tržište bilo područje potrošačke elektronike, nekoliko karakteristika razvijane tehnologije bilo je nužno osigurati. Kako postoji veliki broj proizvođača potrošačke elektronike bilo je potrebno osigurati potpunu neovisnost Java platforme o hardveru i to će se kasnije pokazati kao ključna prednost Jave. Drugi parametri koje je tehnologija trebala zadovoljiti su bili pouzdanost i sigurnost. Sigurnost je bila potrebna prvenstveno jer se pretpostavljalo kako će uređaji raditi na mreži. Konačno, tehnologija je trebala biti jednostavna i kompaktna, kako bi bila prihvaćena od strane proizvođača uređaja potrošačke elektronike. Kada je tehnologija našla svoje konačno tržište, sva navedena svojstva su bila upravo ono šta je potrebno za rad na internetu.

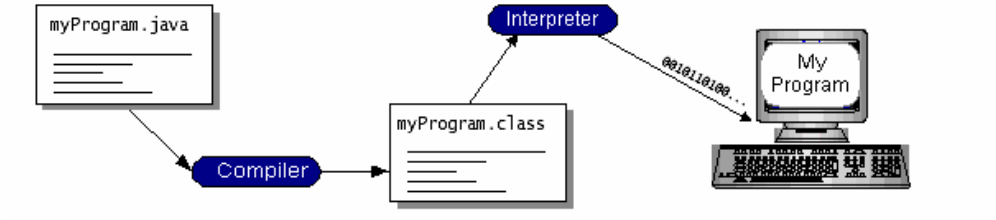
2. Java tehnologija

2.1 Java programski jezik

Java tehnologija je istovremeno i programski jezik i platforma. Programski jezik Java je viši programski jezik čije se karakteristike najčešće opisuju s riječima: jednostavan, objektno orjentiran, mrežno učinkovit, neovisan o hardveru, siguran i učinkovit.

Programski jezik Java je nastao kao dio jednoga većega projekta kojem je bio cilj razviti programski jezik za programiranje uređaja potrošačke elektronike. Zbog zahtjeva jednostavnosti i uklanjanja dugotrajnog učenja novoga jezika, Java je napravljena po uzoru na postojeći jezik C++. Budući da je većina programera radila, i radi, u tom jeziku, Java je napravljena što je bliže moguće C++-u, iako sam C++ nije odgovarao potrebama projekta. Java je odbacila neke rijetko korištene opcije C++-a i pojednostavila ga. Java je objektno orjentirana kako bi se naglasak stavio na podatke i sučelje između i prema podatcima. Objektno orjentirani koncept programskog jezika daje čistu definiciju tih sučelja i omogućuje ponovnu uporabu softverskih blokova. Java je mrežno učinkovita jer koristi opsežne biblioteke rutina za rad sa mrežnim TCP/IP (*Transfer control protocol/internet protocol*) protokolima. Java zbog svoje mrežne primjene nužno ima mnogo sigurnosnih funkcija. Najvažnija karakteristika jezika je, pak, neovisnost o hardveru na kojem se program izvodi. Taj koncept se naziva *write once, run anywhere*. Zbog činjenice da je mreža sastavljena od velikog broja računala koja se rezlikuju po svojim procesorima, količini memorije, veličini hrad-diska, brzini mrežne veze i o operativnom sustavu na računalu, nužno je, zbog pojednostavljenja cijeloga koncepta, da jezik bude neovisan o platformi na kojoj se izvodi.

Kod većine programskih jezika, napisani program je potrebno prevesti (*compile*) i interpretirati, kako bi se mogao izvoditi na osobnom računalu. Programski jezik Java je poseban po tome što se napisani program i prevodi i interpretira. S prevodiocem, program se prevodi u 'među jezik', zvan *Java bytecodes*. Taj među-jezik je neovisan o platformi na kojoj se izvodi i taj, prevedeni kod, se postavlja Java interpreteru (*Java interpreter*) u Java platformi. Interpreter prolazi kroz svaku liniju dobivenog koda i izvršava naredbe na osobnom računalu. Prevođenje se odvija samo jedanput dok se interpretiranje programa odvija svaki put kada se program izvršava na računalu. Na slici 1. prikazan je shematski prikaz ovakovog načina rada.



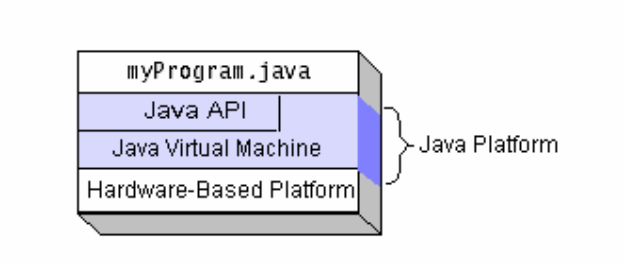
Slika 1. Shematski prikaz rada Java platforme

Dakle, Java *bytecodes* se mogu smatrati vrstom instrukcija zapisanih u strojnom kodu namijenjenih za *Java virtual machine* (Java VM). Svaki Java interpreter, bilo da se radi o razvojnom alatu ili Web pregledniku koji može pokretati aplete, je implementacija Java VM. Java *bytecodes* omogućuju koncept “write once, run anywhere” mogućim. Napisani Java program je moguće prevesti u *bytecodes* na bilo kojoj platformi koja posjeduje Java prevodioc. *Bytecodes* se zatim mogu izvršavati na svakoj implementaciji Java VM. To znači

da, ukoliko računalo posjeduje Java VM, potpuno je svejedno koji operativni sustav računalo posjeduje: Windows, Solaris, MacOS. Navedeno svojstvo je jedna od najbitnijih karakteristika Java tehnologije i predstavlja glavnu prednost u odnosu na ostale programske jezike.

2.2 Platforma

Općenito govoreći, platforma je skup hardvera i softvera na kojem se program izvodi. Većina platformi se može karakterizirati kao kombinacija sklopovlja i operativnog sustava. Java platforma se razlikuje po tome što je isključivo softverska platforma koja radi na drugim hardverski orjentiranim platformama. Klasične platforme koje su kombinacija hardvera i softvera se razlikuju po količini memorije, mrežnoj vezi i, što je najvažnije, računalnoj snazi. Java platforma je isključivo softverska i tom činjenicom je izbjegnuta ovisnost napisanih programa, ispravnosti koda i brzine izvođenja o sklopovlju računala. Sama Java platforma ima dvije glavne komponente: *Java virtual machine* i Java API (*application programming interface*). Java API se sastoji od kolekcije softverskih komponenti (poput *Graphic User Interface*-a), koje omogućuju lakši rad. Java API se organizira u biblioteke (*libraries*) srodnih klasa (*classes*). U Java terminologiji, ove biblioteke se nazivaju paketima (*packages*). Shematski prikaz strukture u kojoj se program izvodi prikazan je na slici 2.



Slika 2. Shematski prikaz strukture Java platforme

Zbog činjenice da se program pisan u Javi može izvoditi na raznim platformama on će biti sporiji od programa koji se izvodi na platformi koja je specifična za pojedini programski jezik. Uz pravilno optimiranje programa brzina izvođenja Java programa se može približiti brzini izvođenja programa koji se izvodi na svojoj prirodnoj platformi.

2.3 Karakteristike jezika

Najčešće vrste programa pisanih u Javi su applikacije (*application*) i apleti (*applets*). Aplikacije su samostalni programi koji se izvršavaju direktno na Java platformi. Posebna vrsta aplikacije zvana *server* služi za podršku klijentima na Webu. Primjeri su : Web serveri, proxy serveri, mail serveri. Apleti su manji programi koji služe prenošenju "ponašanja"i izvršavanju tog "ponašanja" unutar bilo kojeg Web preglednika koji podržava Javu. Još jedna posebna vrsta programa su *servlets*. Servleti se mogu smatrati apletima koji se izvršavaju na serverskoj strani. Oni su popularan izbor za izgradnju interaktivnih Web aplikacija.

Klasa (*class*) je temeljni gradivni blok u objektno orjentiranim programskim jezicima, poput Jave. Klasa je, u neku ruku, obrazac koji opisuje podatke i ponašanje koje se povezuje sa pojedinim instancama klase. Kada se napravi instanca klase stvara se objekt (*object*) koji se ponaša poput svih instanci navedene klase. Dakle, može se reći kako je klasa prototip po kojem se stvaraju objekti. Podatci povezani sa pojedinim objektima ili klasama spremaju se u varijablama, dok je ponašanje koje se povezuje sa pojedinim objektom ili klasom naziva

metodom (*method*). Metoda je slična funkciji ili proceduri u proceduralnim jezicima, poput C-a. Još jedno važno svojstvo koje se javlja u objektno orjentiranim jezicima je pojam naslijeđa (*inheritance*). Naime, dozvoljeno je definiranje klasa pomoću drugih klasa. U tom slučaju postoji jedna nadklasa (*superclass*) iz koje jedna ili više podklasa (*subclass*) naslijeđuju neke metode i tipove podataka. Podklase nisu ograničene samo na metode i podatke nadklase već mogu stvarati i svoje podatke i modificirati varijable i metode naslijeđene od nadklase.

Korištenjem klasa, objekata, poruka (*mesages*) koje klase izmjenjuju i međusobne povezanosti klasa putem naslijeđa, dobivamo jednu preglednu strukturu jezika koji nam omogućuje jednostavno, pregledno i strukturirano programiranje

Jedna od važnih prednosti klasa je u tome što klase mogu ograničiti i kontrolirati pristup svojim varijablama i metodama. Java podržava četiri razine sigurnosti koje se primjenjuju u ograničavanju pristupa. Shematski prikaz se nalazi na slici 3.

Specifier	class	subclass	package	world
private	X			
protected	X	X*	X	
public	X	X	X	X
package	X		X	

Slika 3. Shematski prikaz ograničavanja pristupa klasama

Prvi stupac označava kojim članovima pristup ima klasa u ovisnosti o razini privatnosti koja se koristi. Iz tablice se vidi da klasa uvijek ima pristup svojim članovima, bez obzira na razinu sigurnosti. Drugi stupac označava kojim članovima pristup imaju podklase neke klase (neovisno u kojem paketu se nalaze). Treći stupac označava imaju li članovi istoga paketa pristup članovima klase. Četvrti stupac označava, uz koju razinu privatnosti, sve klase imaju pristup članovima klase. Ukoliko se ne definira razina privatnosti, onda se ona automatski postavlja na *package*. Vidi se iz slike da je najrestriktivniji pristup *private* koji dopušta samo klasi da pristupa svojim članovima. *Public* razina sigurnosti je drugi ekstrem I vidimo da je u tom slučaju pristup članovima klase omogućen svima.

2.4 Primjeri

Nakon što su objašnjeni neki osnovni pojmovi Java platforme i objektno orjentiranih jezika, pokazati ćemo primjer aplikacije i apleta da bi se dobio preglednije gledište iznesenih pojmova. Primjeri su preuzeti iz Java *tutorial*-a. Primjer osnovne Java aplikacije:

```
/**
 * The HelloWorldApp class implements an application that
 * simply displays "Hello World!" to standard output.
 */
class HelloWorldAp {
        public static void main (String[] args) {
                System.out.println("Hello World!"); //Display the string
```

```
    }
}
```

U ovom slučaju radi se o aplikaciji koja na standardni izlaz ispisuje "Hello world!". Komentari u Javi se označavaju sa simbolima /*, /** i //. Komentare prevodioc zanemaruje i njihova jedina funkcija je objašnjavanje koda.

Prva stvar koju radimo je definicija klase *HelloWorldApp*. Navedena klasa predstavlja "glavnu" klasu tj. predstavlja cijelu aplikaciju. Klasa se omeđuje pomoću *{}* zagrada. Unutar klase definiramo *main* metodu. Svaka aplikacija mora imati *main* metodu. Ta metoda je slična *main* funkciji u C-u. Prilikom izvršavanja Java programa, interpreter poziva *main* metodu. Ona zatim poziva sve ostale metode koje su potrebne za izvršavanje programa. Preko argumenata *main* metode programu se mogu prenositi nizovi *string*-ova za upravljanje programom. *System.out.println* je klasa koja postavlja argument zadan u obliku *string*-a na standardni izlaz.

Nakon aplikacije, primjer jednostavnog apleta je napisan na sljedećim retcima:

```
import java.applet.Applet;
import java.awt.Graphics;

public class HelloWorld extends Applet {
    public void paint (Graphic g) {
        g.drawString ("Hello World!", 50, 25);
    }
}
```

Kod apleta počinje sa pozivanjem (*import*) dvaju paketa klasa. Postupak ima isti učinak kao i referenciranje biblioteka (*library*) funkcija u proceduralnim jezicima. Zatim definiramo podklasu *HelloWorld* koja je podklasa nadklase *Applet* što je označeno ključnom riječi *extends*. Nakon toga se radi implementacija klase apleta u sljedeće dvije linije koda. Svaki aplet mora implementirati jednu ili više *init*, *start* i *paint* metoda. *g.drawString* objekt iscrtava *string* koji mu se prenosi sa početkom na koordinatama koje se zadaju kao drugi i treći argument kod poziva objekta (u ovom slučaju 50 i 25).

Java apleti su namijenjeni izvođenju unutar HTML stranica i uključuju se unutar izvornog koda stranice. Apleti se uključuju pomoću <APPLET> oznake (*tag*) unutar HTML stranice. Unutar te oznake se, minimalno, definiraju lokacije aplet podklase i dimenzije prikaza apleta na ekranu. Kada internet preglednik (*browser*), sa podrškom ua Javu, naiđe na navedenu oznaku, on rezervira traženo područje na ekranu, učita aplet podklasu, kreira njenu instancu. Primjer osnovnog HTML koda koji uključuje Java aplet na stranici je prikazan dolje:

```
<HTML>
<HEAD>
<TITLE> Jednostavni program za prikaz apleta </TITLE>
</HEAD>
<BODY>
Ovdje se nalazi izlaz programa:
<APPLET CODE="HelloWorld.class" WIDTH=150 HEIGHT=25>
</APPLET>
</BODY>
</HTML>
```

3. Stanje Java tehnologije i budući razvoj

Otkako je predstavljena 1995. Java je postala najbrže prihvaćena tehnologija u povijesti računarstva. Područja primjene se neprestano šire, broj *developera* se veoma brzo povečava i taj napredak zasada ne pokazuje znakove usporavanja. Dakako da je tom izuzetnom rastu pogodovalo i okruženje u kojem se odvijao usporedni brzi razvoj *World wide weba*. Zahvaljujući karakteristikama Jave i, posebice Java apleta, koji su dali novu dimenziju mediju poput Interneta, te dvije tehnologije su se međusobno podupirale i nadopunjavale u izuzetnom rastu koji se dogodio u proteklih desetak godina. Nakon što su nedostatci HTML-a postali nezanemarivi i uz uvođenje novih tehnologija, poput XML-a (*extensible markup language*) i SGML-a (*Standard generalized markup language*), fuzija Jave i *Weba* postaje jača, uz napredak prema sljedećem koraku, koji je tzv. inteligentni web (*intelligent web*).

Nedvojbeno je da je Java postala dominantna tehnologija u razvoju aplikacija u distribuiranom, mrežnom računarstvu. Uz takvo stanje, problem tržišta više nije problem. Problem sada postaje kontrola budućeg razvoja i napretka tehnologije. Uz brojne tužbe između *Sun Microsystems*, *Microsofta*, uz interesne grupe i lobiste, smjer budućeg razvoja i karakter toga razvoja trenutno djeluju prilično magloviti. Mnoge interesne grupe *developera* se zalažu da Java ostane otvorenog standarda, no hoće li se tako i zbiti još je veliko pitanje. Glavno pitanje se vodi oko toga hoće li se uspjeti sačuvati Javin originalni i revolucionarni koncept: *write once, run anywhere*.

Unatoč borbi za prevlast nad kontrolom i budućim razvojem, nije vjerovatno da bi Java mogla pasti u zaborav. Njena rasprostranjenost, brojna područja primjene, velika baza *developera* i rast i razvoj *Weba*, koji ne pokazuje znakove usporavanja, osiguravaju budućnost Javi, u ovom ili onom obliku.

4.Literatura

1. **Sun Microsystems**, http://www.sun.com
2. **Java tutorial**, http://java.sun.com/docs/books/tutorial
3. **Xmlapps.htm**, http://www.ibiblio.org/pub/sun-info/standards/xml/why/
4. **Understanding XML and Java XML Apis.htm**, http://java.sun.com/xml/jaxp/dist/1.1/docs/tutorial/overview/
5. **Index.html**, http://java.sun.com/reference/docs/
6. **Java_hist.html**, http://ei.cs.vt.edu/~wwwbtb/book/chap1/
7. **Jw-07-winjava.html**, http://www.javaworld.com/javaworld/jw-07-1999/