PREDMET

Osnove Java Programiranja

Čas1-2

Upoznavanje sa radnim okruženjem

Oktobar 2011.

SADRŽAJ

Kompajliranje 3
Izvršavanje Java aplikacije 3
Proces izvršavanja Jave 4
Razvojna okruženja za Javu 5
IDE alati 6
NetBeans 6

Čas 1-2

Upoznavanje sa radnim okruženjem

Kompajliranje

Kompajliranje Java programa je proces pretvaranja tekstualnog fajla sa ekstenzijom java u binarni kod koji može da se izvršava u okviru Java Virtualne Mašine (JVM). Ekstenzija kompajliranog fajla je class. Binarni fajl class se prilikom startovanja izvršava (interpretira) u okviru JVM-a. JVM binarne naredbe JVM-a pretvara u letu u binarne naredbe konkretnog operativnog sistema (OS).

Da bi pisali programe u programskom jeziku Java nije nam neophodno razvojno okruženje. Dovoljno je da na računaru bude instalirana Java. Java postoji u dva oblika kao JRE (Java Runtime Enviroment) i kao JDK (Java Developer Kit). JRE je dovoljno da bi startovali Java aplikacije ali u okviru JRE ne postoje alati za kompajliranje i razvoj aplikacija. Ako želimo da programiramo u javi neophodno je da imamo instaliran JDK na računaru na kome želimo da programiramo. Tekstualni java fajlovi su u stvari najobičniji tekstualni falovi koje možemo da napravimo u najobičnijem text editoru i da ih snimimo kao obične tekstualne fajlove sa ekstenzijom java.

Ako je na računaru instaliran JDK onda bi iz komandne linije trebalo da nam budu dostupne naredbe javac i java. Pomoću naredbe javac možemo da kompajliramo naš program. Ako u kodu nema grešaka biće kreiran binarni class fajl. Ako u kodu postoje greške onda će na ekranu, nakon pokretanja naredbe za kompajliranje, biti prikazan spisak grešaka sa mestom u kodu gde su načinjene.

Gotovu, kompajliranu java aplikaciju možemo da startujemo pomoću naredbe java. Prilikom startovanja java programa ne stavlja se ekstenzija. Dakle ako hoćemo da pokrenemo program koji se zove Krediti i nalazi se u fajlu Krediti.class onda treba da pokrenemo naredbu:

java Krediti

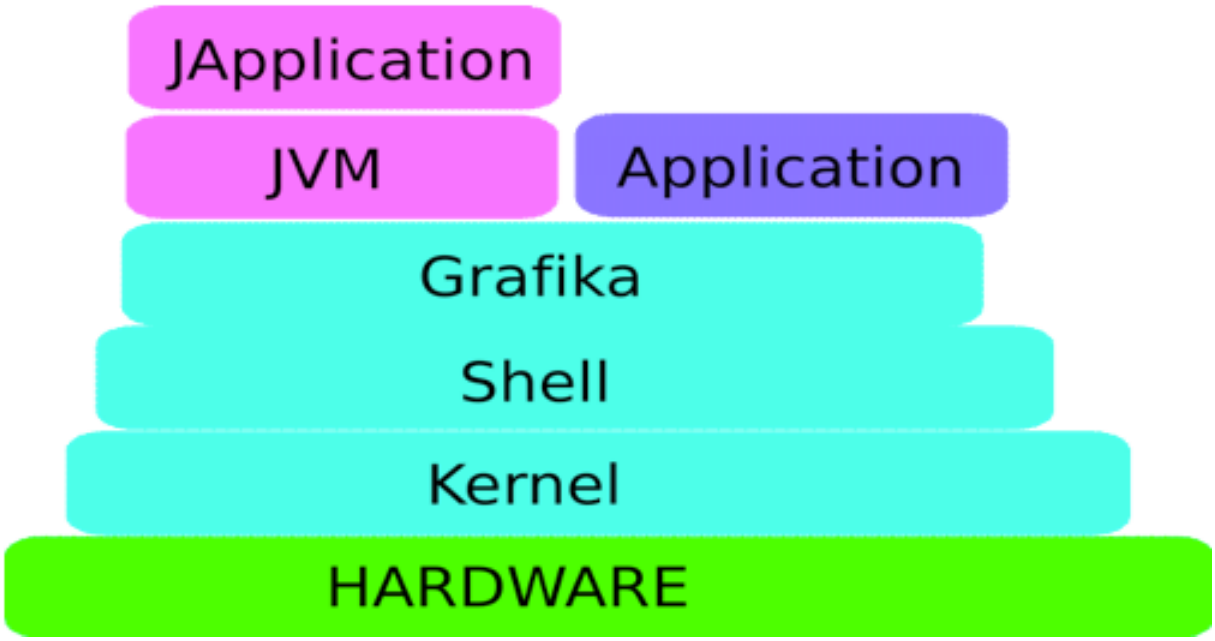
Izvršavanje Java aplikacije

Kada pokrenemo java aplikaciju ona se izvršava u okviru JVM (Java Virtual Machine). JVM je vrsta interpretera koja java binarni fajl prevodi na naredbe za konkretan operativan sistem.

Osnova svakog računara je Hardware koji je zadužen za uzimanje podataka od računara obradu tih podataka po zadatim naredbama OS-a i prikaz podataka korisniku. Komunikacija sa Hardverom je ostvarena preko drajvera ili je ugrađena u jezgro operativnog sistema. Svaki operativni sistem ima jezgro koji je osnova operativnog sistema i koji komunicira sa Hardware-om preko odgovarajućih drajvera. Naredbe Kernela (jezgra OS-a) su suviše računarski orijentisane pa iz tog razloga obično postoji sloj u OS-u koji je zadužen za komunikaciju sa kenelom. Ovaj sloj OS-a se zove shell. Mnogi programi mogu da se pokreću u shell-u i to je korisniku obično prikazano kao komandna linija. S obzirom da svaki savremeni operativni sistem funkcioniše u grafičkom modu na shell se obično oslanja nekakav grafički server koji je zadužen za prikaz grafike i rad sa svim grafičkim programima. Kernel, Shell i Grafika predstavljaju delove operativnog sistema. Neki operativni sistemi imaju jasno odvojene ove delove (svi Linux sistemi i Mac OS X), dok je kod nekih OS-a ta razlika u slojevima sakrivena od korisnika (Windows).

Programi se obično oslanjaju na grafičke biblioteke OS-a tako da grafičke aplikacije koje su pisane za dati operativni sistem se izvršavaju na vrhu OS piramide (prikazano na slici). U Windows operativnom sistemu to je ujedno i jedini način izvršavanja bilo koje aplikacije.

Java Virtuelna Mašina se u windows sistemu izvršava na vrhu OS priamide a naša aplikacija se izvršava na samoj JVM. Ovo utiče jednim delom na brzinu izvršavanja Java aplikacija.



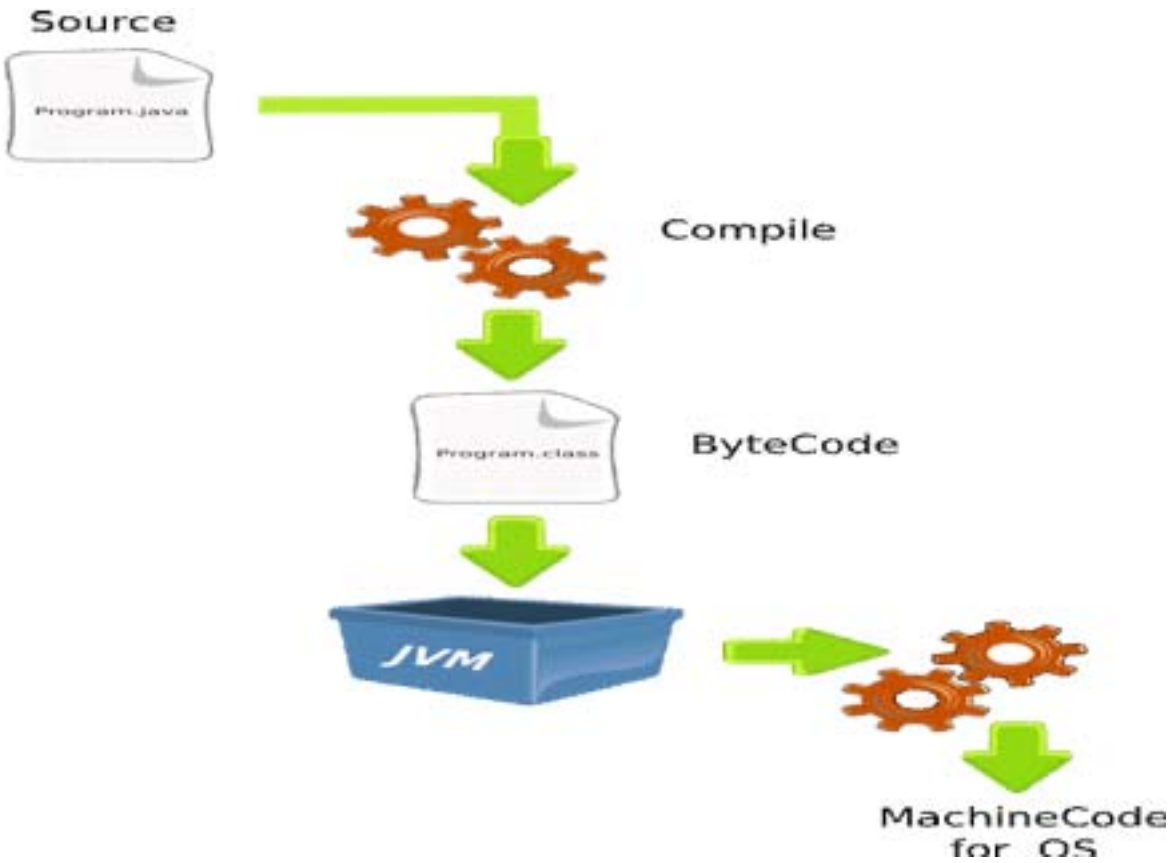
Kod operativnih sistema gde slojevitos OS-a nije sakrivena u mogućnosti smo da startujemo program na bilo kom nivou OS-a (naravno u skladu sa sigurnosnim mehanizmima operativnog sistema). Na Linux sistemima i Mac OS X-u možemo da napišemo aplikaciju koja se izvršava u shell-u i u celosti preskače grafički podsistem. Isto tako smo u mogućnosti da napišemo i program koji se obraća direknto kerenelu.

Kada su Java aplikacije u pitanju na Mac OS X sistemu JVM je podešena tako da pristupa svim delovima OS-a tako da JVM funkcioniše kao da je deo samog OS-a. To znači da svaka java aplikacija startovana na Mac OS X opertativnom sistemu je ravnopravna u brzini izvršavanja u odnosu na bilo koju drugu aplikaciju na sistemu.

Na Linux sistemema od verzije Linux kernela 2.6 moguće je podesiti JVM da radi u istom režimu kao i na Mac OS X sistemu. Ako se u kernelu ne podesi da JVM radi kao deo OS-a onda na Linux sistemu se java aplikacije izvršavaju na isti način kao i na Windows sistemu.

Proces izvršavanja Jave

Na slici je prikazan proces kompajliranja i izvršavanja programa pisanog u Javi. Source kod je običan java tekstualni fajl. Nakon kompajliranja od našeg Java tekstualnog fajla nastaje ByteCod (fajl sa ekstenzijom class). Ovakav fajl se pokreće u JVM-u i izvršava tako što JVM u letu interpretira naredbe iz ByteCod-a i pretvara ih u naredbe za konkretan operativni sistem.



Razvojna okruženja za Javu

Iako programe u Javi možemo pisati u bilo kom editoru pisanje aplikacija na takav način je dosta naporno. Lako se prave greške u kodiranju a kasnije se teško otklanjaju. Pravljenje većih aplikacija na ovaj način je previše komplikovano i teško za održavanje tako da programeri skoro nikad ne programiraju na ovaj način.

Kao pomoć programerima razvijani su razni alati za razvoj aplikacija. Najjednostavniji oblik ovakvih alata su programerski tekst editori. Oni su slični običnim tekst editorima ali raspoznaju Java naredbe i delove java koda pa su u stanju da farbaju različitim bojama java naredbe, metode, promenjive, konstante... Pored toga u okviru ovakvih editora obično postoje naredbe za automatsko pokretanje kompajlera i aplikacije. Mnogi operativni sistemi (Linux i Mac OS X) su ove mogućnosti ugradili u osnovne tekst editore.

Text editori mogu da budu dovolji za manje projekte ali na većim projektima nam treba i dodatni alati osim samog dobrog editora. Za takve projekte obično koristimo neki od IDE alata. Savremeni IDE alati imaju mogućnost dodavanja alata pomoću plugin-ova. Na taj način se njihova funkcionalnost može dodatno povećati. Svi IDE alati podrazumevaju da u okviru njih postoje odlični editori koda ali pored toga i razni drugi alati (debug-er, refactoring alati, test alati, ...)

Razlika između RAD alata i IDE alata je ponekad suviše tanka. Osnovna definicija RAD alata je da kreiramo aplikaciju bez kucanja koda. Međutim većina IDE alata ima dodatke koji im pomažu da se deo koda automatski generiše. Sa druge strane alati koji se reklamiraju kao RAD alati u sebi imaju i odlične editore koda kao i druge dodatke koje imaju klasični IDE alati.

Razvojna okruženja možemo podeliti u tri grupe:

- Text editori
- IDE (Integrated Developer Enviroment)
- RAD (Rapid Application Development)

IDE alati

- NetBeans (SUN microsystem)
- Eclipse (IBM)
- IntellJ idea
- JDeveloper (Oracle)
- JBuilder (Borland)
- XCode (Apple – samo za Mac OS X)

Postoji veliki broj razvojnih alata za Java programski jezik ovde smo nabrojali samo najpoznatije i najveće. U okviru same jave postoje alati koji vam omogućavaju, na primer, da pokrenete kompajler iz samog java programa. Tako da je relativno lako napraviti sopstveno razvojno okruženje. Naravno savremena razvojna okruženja podrazumevaju veoma puno dodatnih alata i plugin-ova, što zahteva velike timove za razvoj. Većina razvojnih okruženja je pisano u programskom jeziku Java tako da isto razvono okruženje možete da koristite bez obzira na kom operativnom sistemu radite. Na datom spisku razvojnih okruženja izuzetak je jedino XCode koji nije pisan u Javi već u Apple-ovom Object C-u pa funkcioniše samo na Mac OS X operativnom sistemu. XCode se dobija sa OS X operativnim sistemom i besplatan je za sve kupce Apple-ovih računara.

Iako su SUN microsystem i IBM imali svoja razvojna okruženja koja su se plaćala vremenom su besplatna open source rešenja preuzela primat pa su ove dve velike korporacije odlučile da odustanu od svoji razvojnih alata i da se uključe u razvoj nekog od ovih besplatnih alata otvorenog koda. SUN Microsystem je pomogao razvoj NetBeans razvojnog okruženja a IBM je podržao Eclipse. Ove dve kompanije najviše utiču i na razvoj samog Java programskog jezika.

Oracle korporacija je najpoznatija po svojim izuzetnim bazama podataka, međutim oni se intezivno bave i razvojem aplikacija koje se oslanjaju na njihovu bazu podataka i te aplikacije razvijaju u Javi. Iz tog razloga je Oracle odlučio da razvije sopstveni alat za razvoj koji je izuzetno obiman i u sebi sadrži više nego što se za druge alate može naći. Još jedna velika prednost Oracle-ovog JDevelop razvojnog alata je i izuzetna podrška za Oracle baze podataka, po informacijama sa Oracle-ovog sajta jedino iz Jave može da se pristupi svim mogućnostima njihovih najnovijih 10g baza podataka. JDeveloper je potpuno besplatan i može se skinuti sa Oracle-ovog sajta.

Najveće komercionalno razvono okruženje JBilder će verovatno prestati da postoji i pridružti svoj kod NetBeans projektu. Ovo razvojno okruženje je najstarije na tržištu i dugi niz godina je bio primarni alata profesionalaca. Intezivan razvoj razvojnih okruženja otvorenog koda je uticao da Borland nema šta da ponudi programerima za novac koji traži tako da su programeri prešli na besplatna i često kvalitetnija rešenja.

Jedini komercionalni alat koji još uvek uspeva da održi korak sa rešenjima otvorenog koda je IntellJ idea. Ovaj razvojni alat ima svoje verne korisnike jer ima najkvalitetniji editor sa izuzetnom lakoćom korišćenja. Ovaj alat nije popularan zbog raznih dodataka već zato što osnovne stvari koje su potrebne programeru radi perfektno. Uz to ima i umerenu cenu tako da se mnogi programeri odlučuju za kupovinu ovog alata i pored toga što postoje odlična besplatna rešenja.

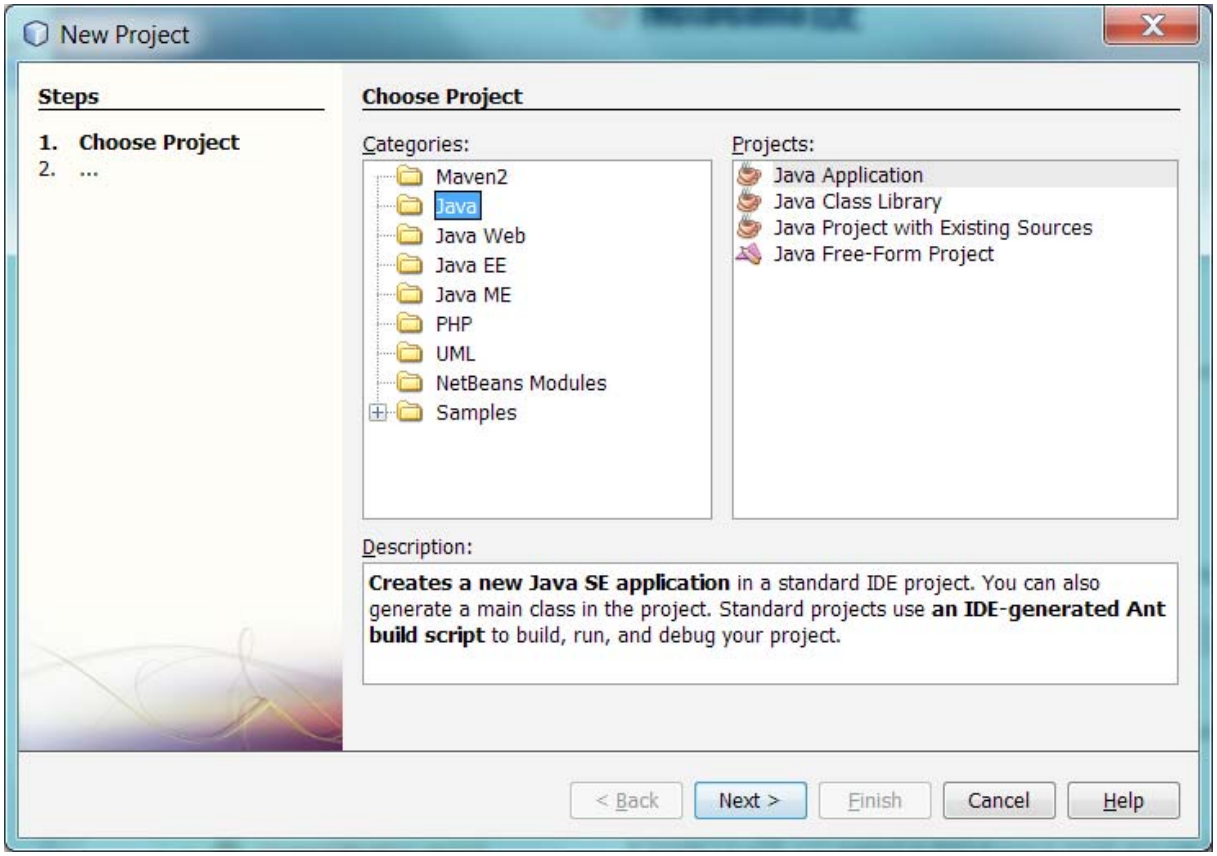
NetBeans

Sa gornje strane se nalazi meni i toolbar na kome su najčešće korišćene komande. Sa leve strane su Projects i Files. U njima vidite spisak trenutno učitanih projekata i fajlova. NetBeans dozvoljava da imamo istovremeni pristup većem broju projekata dok je jedan od njih glavni. Glavni projekat je ispisan podebljanim slovima. Sa donje strane se nalaze opcije koje se koriste tokom rada, npr. Output prikazuje šta trenutno radi kompajler ili pokrenuti program. Ovde se ispisuju informacije o greškama. Watches i Usages su od koristi prilikom traženja grešaka.

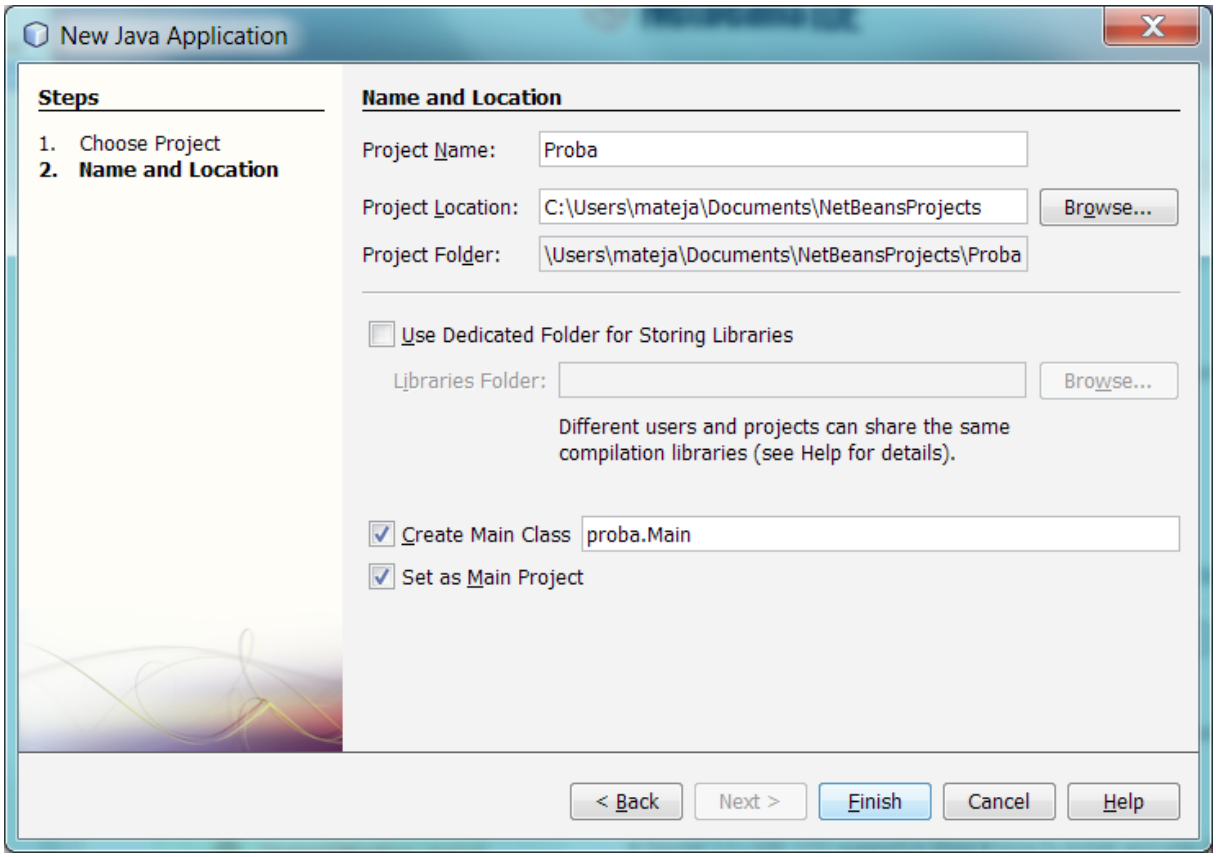
Sa desne strane se nalaze pomoćni alati i plug-inovi (oni mogu da budu i levo i dole)

Svaki program koji pravimo u NetBeansu mora da se nalazi u nekom projektu. Tako da kad god počinjemo nov program mi prvo počinjemo od novog projekta. NetBeans ume da podesi i izgeneriše odgovarajuće fajlove u zavisnosti od toga da li razvijamo običnu java aplikaciju, web aplikaciju, plugin

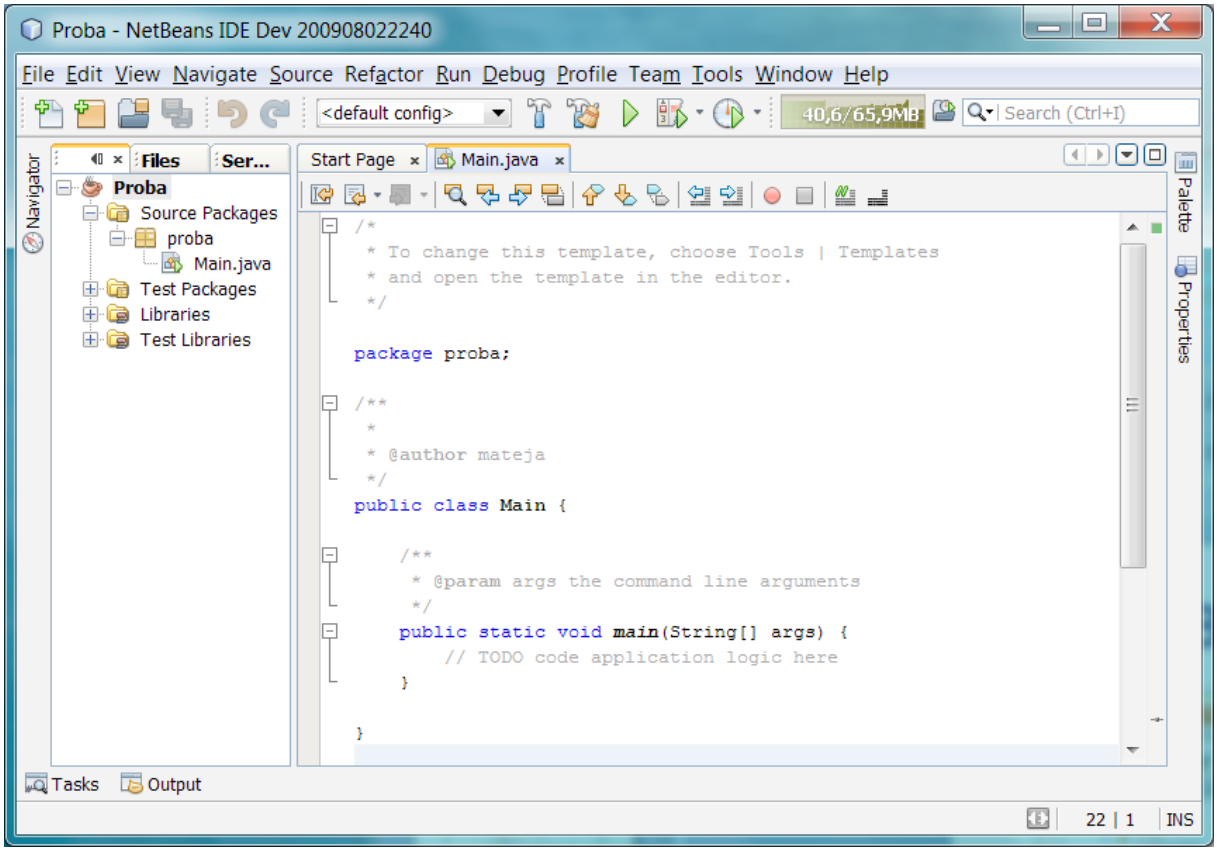
za netBeans.... Naravno NetBeans nam nudi i pomoć ako želimo da učitamo projekat rađen u drugom razvojnom okuženju.



Sledeći korak u kreiranju novog programa (projekta) je određivanje gde na računaru će taj projekat da se nalazi i kako će se zvati. NetBeans će nam pomoći i, ako to želimo, automatski kreirati početnu klasu sa main metodom. Po želji će ovaj novi projekat postaviti za glavni projekat u spisku projekata.

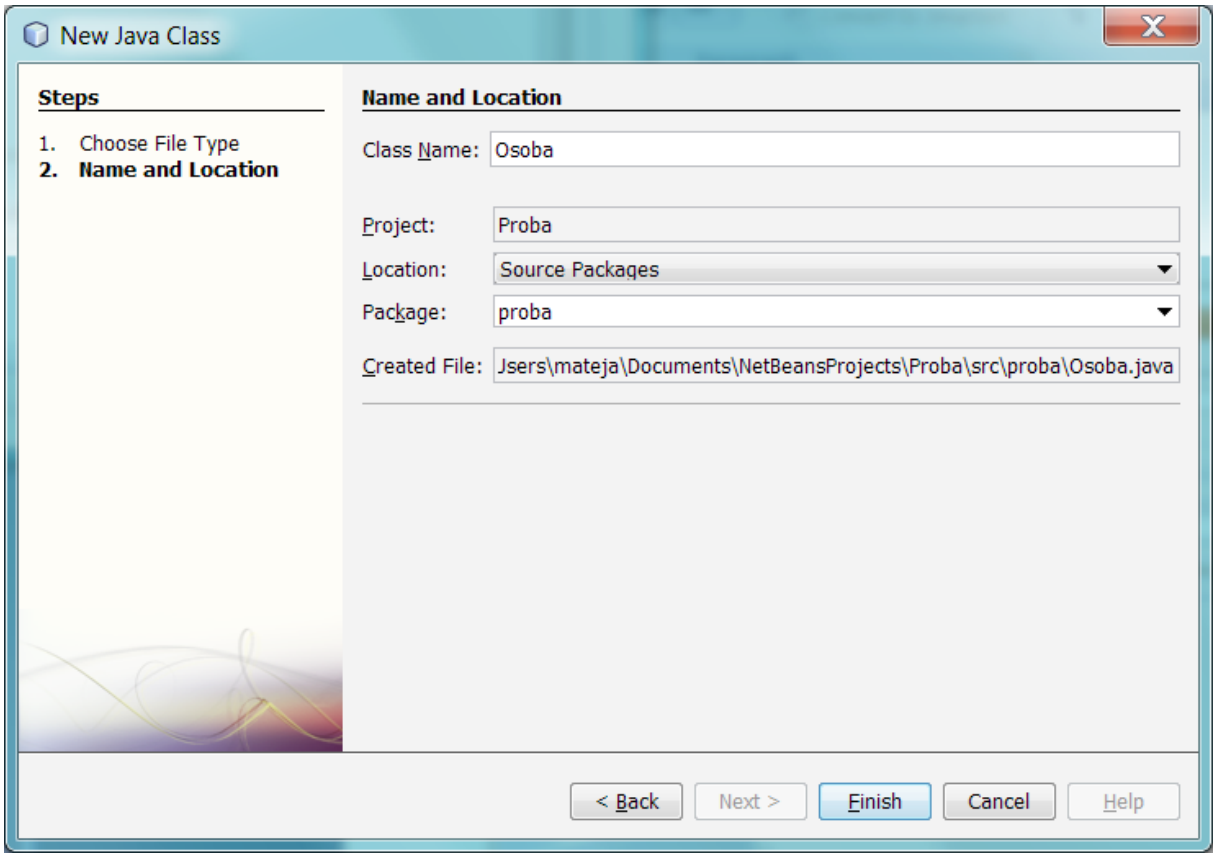


Nakon što smo kreirali novi projekat u NetBeans-u će se pojaviti taj novi projekat sa već kreiranom klasom pod nazivom Main koja u sebi ima i main metodu. Naš novi projekat je automatski povezan i sa JDK-om kao i sa nekim plugin-ovima. Na slici se vidi JUnit (test Framework).

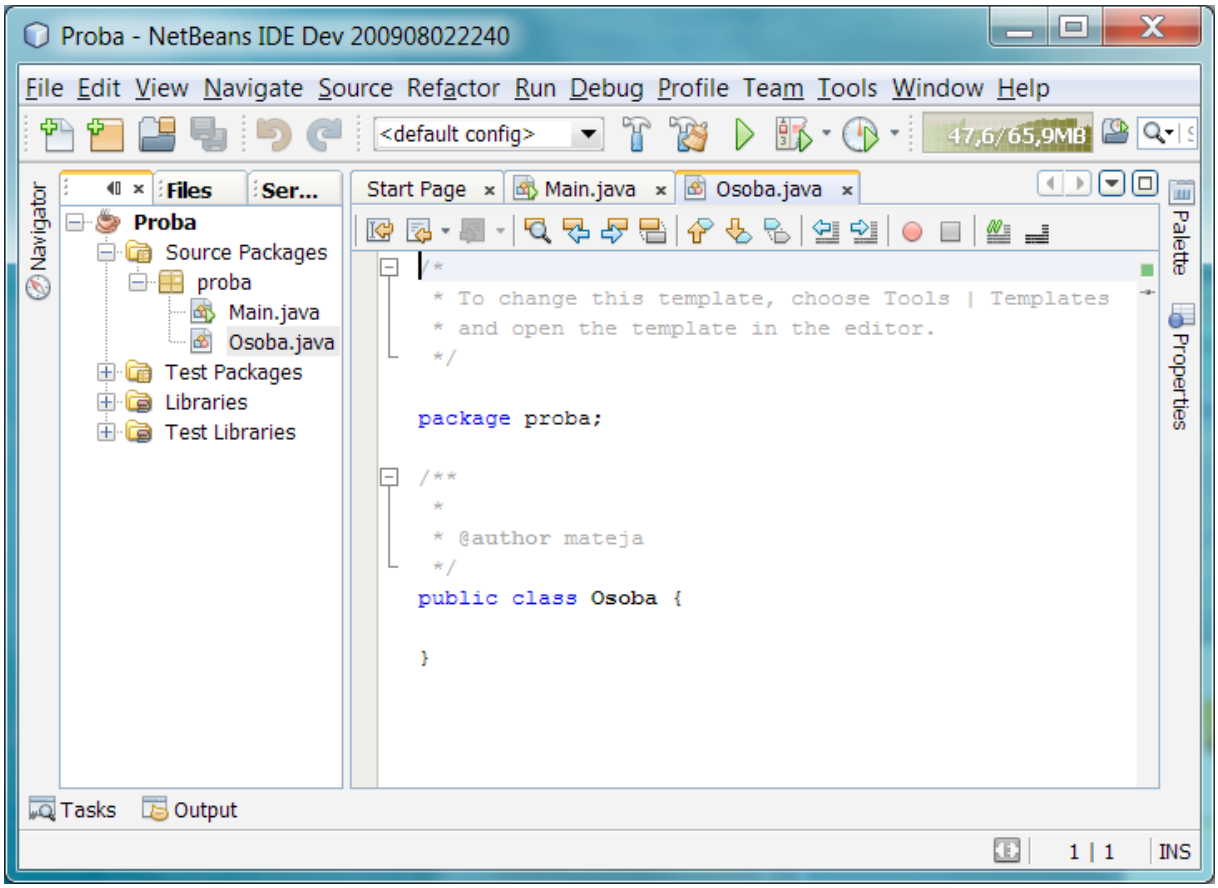


Ako želimo da kreiramo novu klasu treba da desnim tasterom miša kliknemo na naziv paketa u okviru projekta i da iz menija odaberemo New->Java Class. Na ovaj način možemo da kreiramo i java Interface, Java paket, običan fajl,...

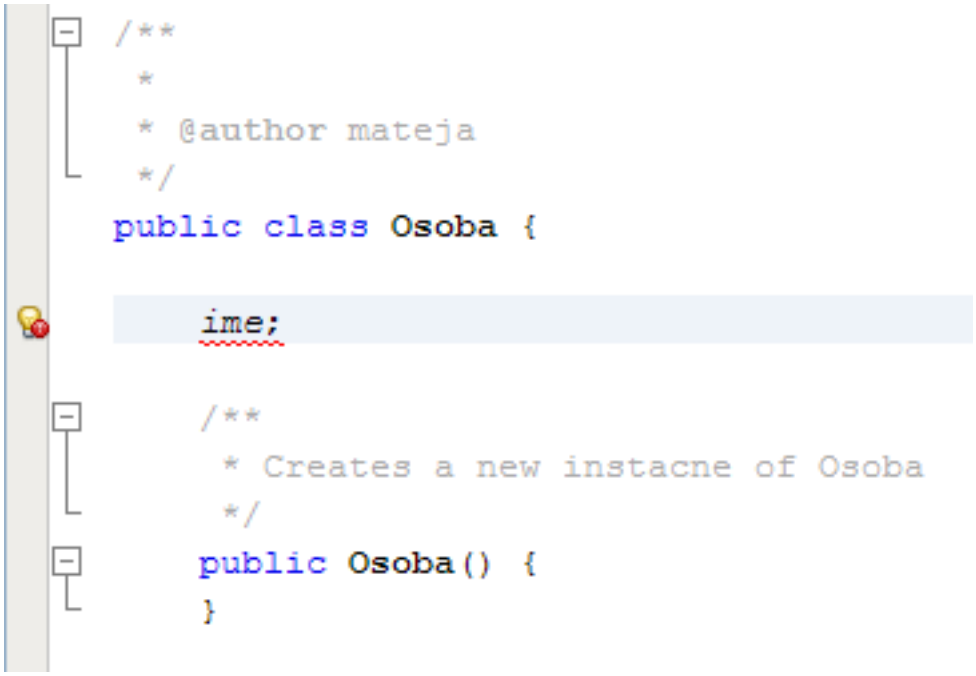
Pojavljuje nam se dijalog za unos informacija vezanih za novu java klasu. Tu treba da unesemo naziv klase i kom paketu pripada. Odabirom opcije Finish kreiramo klasu.



Kada NetBeans kreira novu klasu ona već ima deklaraciju klase i prazan konstruktor, kao i osnovne komentare.



Ako prilikom pisanja koda pogrešimo, napravimo sintaksnu grešku, razvojno okruženje će podvući crvenim deo koda gde smo napravili grešku i staviti oznaku sastrane da se na tom mestu nalazi greška. Ovo se dešava u letu dok mi kucamo kod, tako da možemo odmah da vidimo gde smo pogrešili. Ovakav pristup se zove intime kompajliranje i značajno povećava efikasnost prilikom kodiranja. Ponekad će razvojno okruženje dati predlog kako da otklonite grešku. Predlog će se pojaviti kao lampica na koju kad kliknete dobijate spisak predloga za rešavanje greške.



Kada želimo da startujemo aplikaciju nije potrebno da posebno pokrećemo kompajler a posebno aplikaciju. Klikom na jedno dugme (play/run) na toolbar-u NetBeans će da izvrši kompajliranje

programa a nakon toga i njegovo pokretanje što je najčešći način pokretanja programa prilikom razvoja.