Java za mlade programere (1)

Šta je Java?

Java je računarski **programski jezik** koji je razvila firma Sun Microsystems. Programski jezik se koristi za izdavanje instrukcija računaru da obavi konkretne poslove. Java, mada relativno novi jezik, nastao 1995. godine, izuzetno je popularna.

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 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.

Zašto učiti Javu?

Možemo postaviti i pitanje Zašto učiti programski jezik? 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 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:

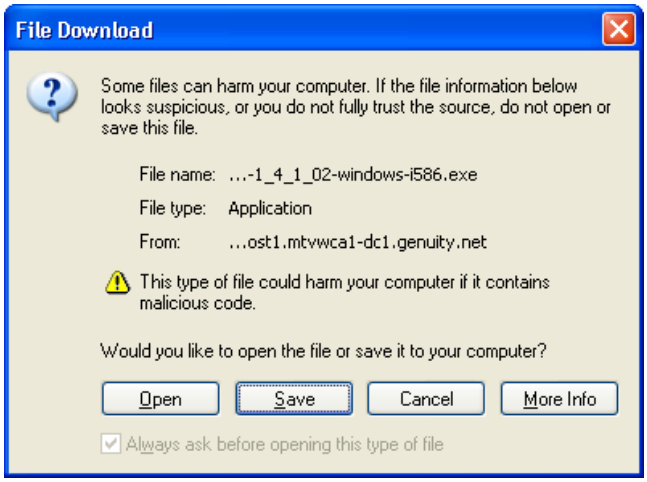
Pokrenite svoj pretraživač weba (Internet Explorer, Netscape ili neki drugi) i odete na web sajt firme Sun Microsystems: http://java.sun.com

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.

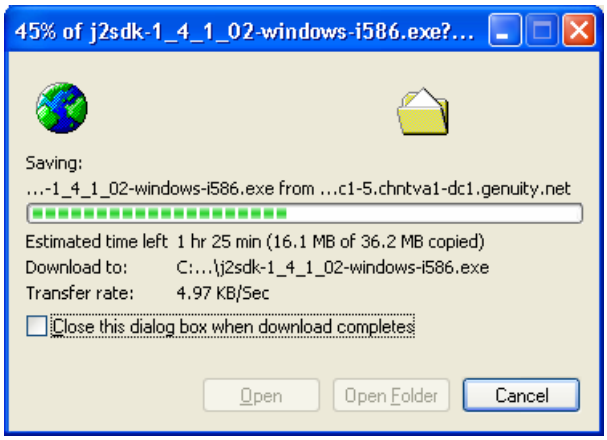
Na Sunovom sajtu treba da potražite link koji omogućava preuzimanje paketa koji se naziva J2SE SDK (Java 2 Standard Edition Software Development Kit). Za naše potrebe treba da preuzmete verziju J2SE 1.4.1. Kada pronađete stranicu sa linkovima odaberite onu koja odgovara za instaliranje na vašu računarsku platformu (Windows).

Pošto aktivirate link, pojaviće se sporazum o licenci (Licence Agreement). Potrebno je da kliknete na Accept ukoliko želite da preuzmete paket.

Kliknite mišem na link za preuzimanje datoteke J2SDK i na ekranu će se pojaviti sledeći prozor:

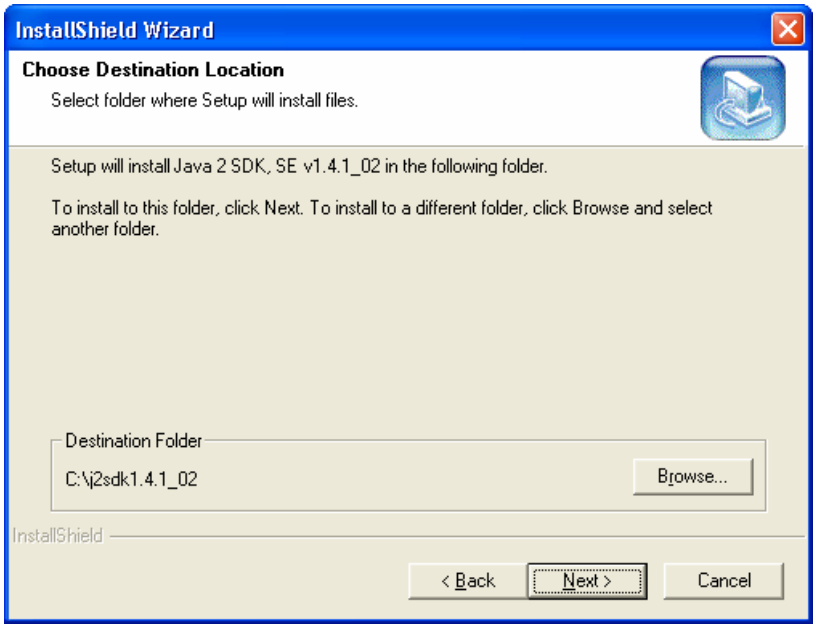


Kliknite mišem na dugme Save i izaberite lokaciju na kojoj želite da snimite preuzetu datoteku. Potom ćete na ekranu videti kako preuzimanje napreduje.

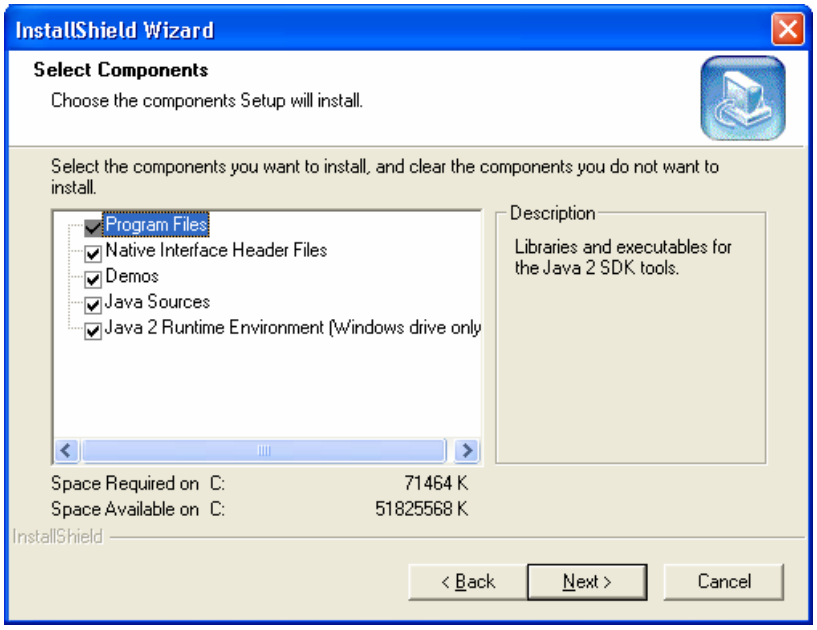


Java SDK je veoma velika datoteka (j2sdk-1_4_1_02-windows-i586.exe), veličine oko 36 MB, stoga će preuzimanje preko modema i komutirane veze potrajati.

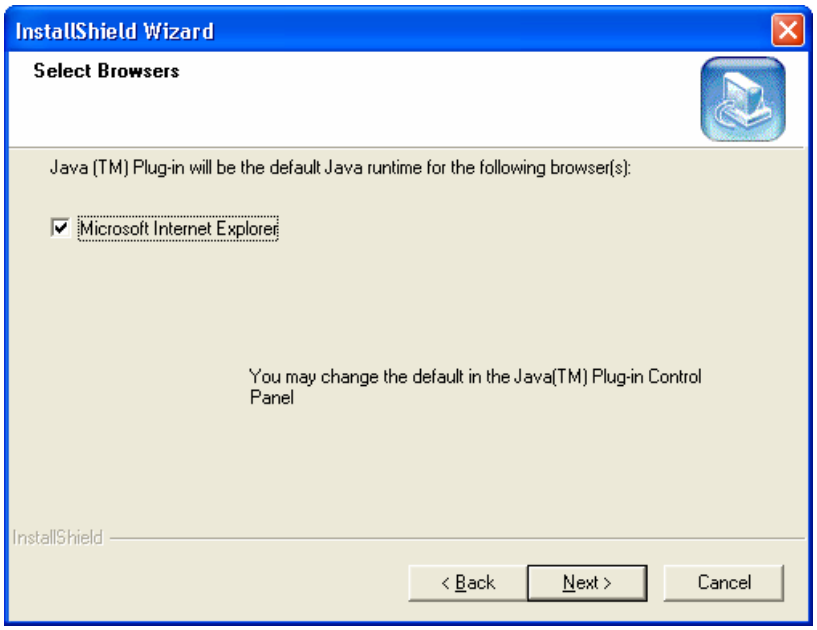
Kada ste preuzeli SDK, spremni ste za instaliranje paketa na svoj računar. Izaberite dugme Start na paleti poslova a zatim Run. Pronađite direktorijum (folder) gde ste snimili datoteku i selektujte datoteku. Kliknite na Open. Java SDK instalacioni program će raspakovati neke datoteke iuvodni ekran sa sporazumom o licenci će se pojaviti na ekranu. Po prihvatanju licence, instalacioni program će vas pitati gde želite da se obavi instalacija.



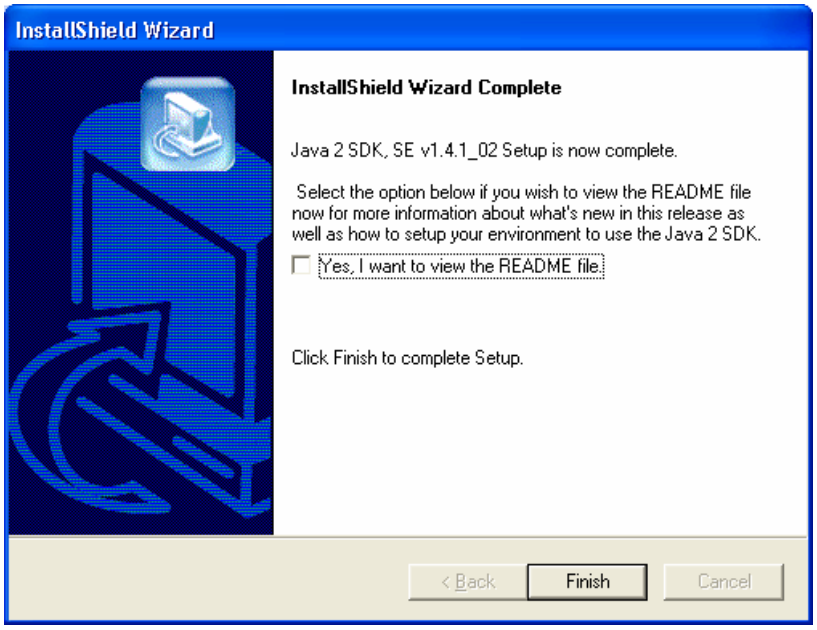
Predlažem da instalaciju Java SDK obavite u podrazumevanom direktorijumu (c:\j2sdk1.4.1_02 u ovom slučaju), stoga kliknite mišem na dugme Next.



Nemojte ovde ništa menjati. Želimo da instaliramo sve komponente. Kliknite mišem na Next da biste videli ekran koji vam omogućava izbor podrazumevanog Internet pretraživača.



Kliknite mišem na Next i instalacija počinje. Kada se završi videćete sledeći prozor:



(Možda će vam biti postavljeno pitanje da ponovo pokrenete računar, a možda se to pitanje i neće pojaviti.) Kliknite mišem na dugme Finish i time ste okončali instalaciju.

JCreator – integrisano razvojno okruženje

Proces pravljenja i izvršavanja Java programa ima tri jasno razdvojene etape:

- Pisanje programa
- Kompajliranje (prevođenje) programa (generiše se datoteka koju računar može da razume)
- Izvršavanje programa.

Za sada, ne brinite se mnogo oko toga šta se tačno dešava u svakoj od ovih etapa. Jedan od načina da kompletirate ove tri etape je da prvo pišete program koristeći neki osnovni editor teksta i snimite dobijeni tekst u datoteku. Potom, etape kompajliranja i izvršavanja mogu se obaviti kucajući u komandnoj liniji posebnih komandi predviđenih za te etape. Mali broj Java programera piše programe na taj način. Skoro svi programeri razvijaju svoje programe koristeći tzv. Integrisano razvojno okruženje (Integrated Development Environment – IDE). Postoji više IDE proizvoda za razvoj Java programa., neki su veoma složeni, a neki veoma prosti. Mi ćemo koristiti verziju koja je besplatna. Radi se o okruženju koje se naziva JCreator. Radi se o

interfejsu koji se lako koristi a može besplatno (verzija Lite) da se preuzme preko Interneta. Preporučujem da koristite ovaj softver zato što će vam olakšati programerski deo posla i omogućiti da se koncentrišete na učenje elemenata Java jezika.

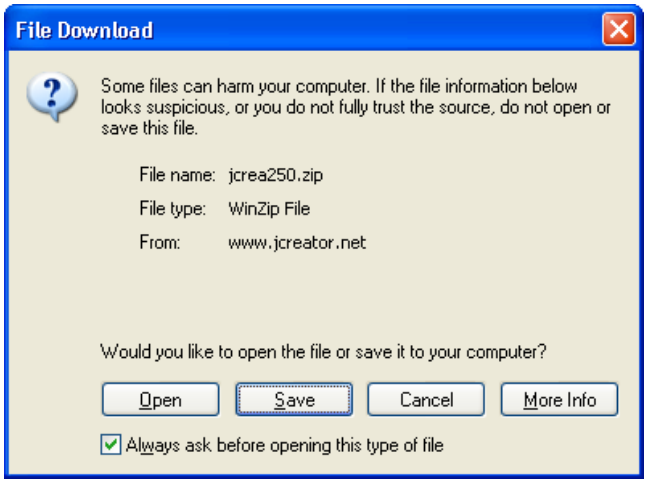
Pre nego što preuzmete i instalirate JCreator, moram da vas upoznam sa jednom činjenicom. JCreator radi samo u Windows okruženju, to znači da korisnici Linuxa i Mac OS moraju da pronađu drugo rešenje.

Ukoliko koristite Windows i izaberete da koristite JCreator, pristupite preuzimanju tog okruženja. JCreator je razvila firma Xinox Software. Postoje dve verzije ovog proizvoda, JCreator LE (Light Edition) I Pro (Professional Edition). Mi ćemo koristiti LE zato što je besplatan.

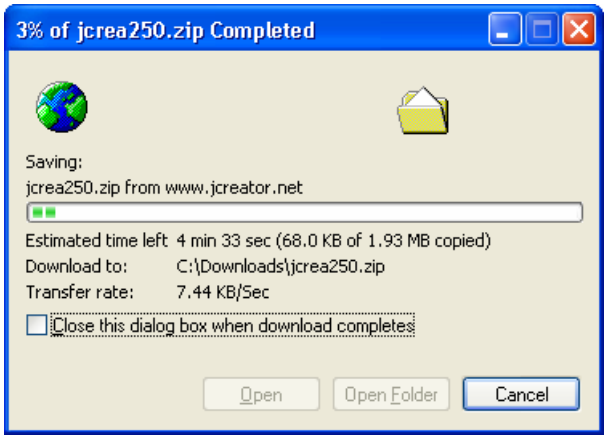
Da biste preuzeli ovaj softver uradite sledeće:

Idite na JCreator web sajt: http://www.jcreator.com
Potražite link za preuzimanje Jcreatora LE (Freeware).

Na web stranici za preuzimanje kliknite mišem na link i na ekranu će se pojaviti sledeći prozor:

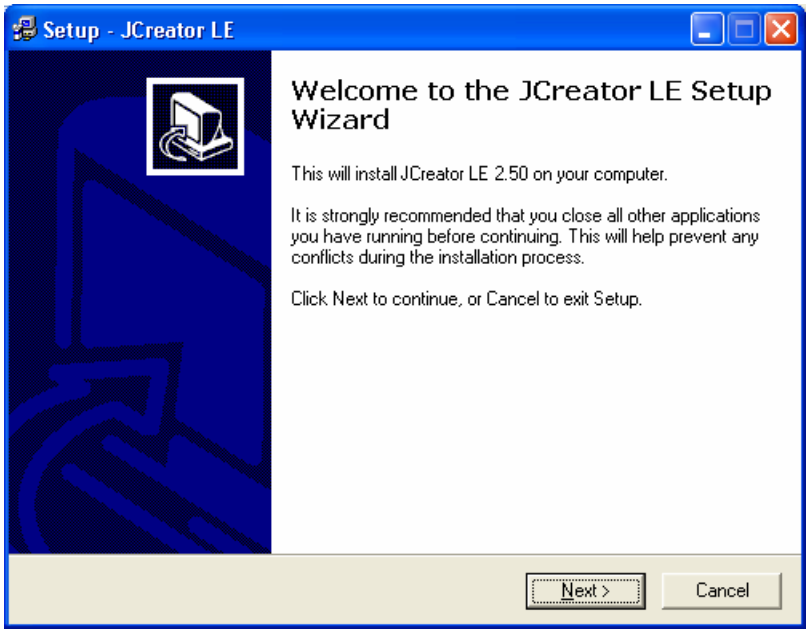


Kliknite mišem na dugme Save i izaberite lokaciju na kojoj želite da sačuvate preuzetu datoteku. Pojaviće se na ekranu prozor koji prikazuje kako preuzimanje napreduje.



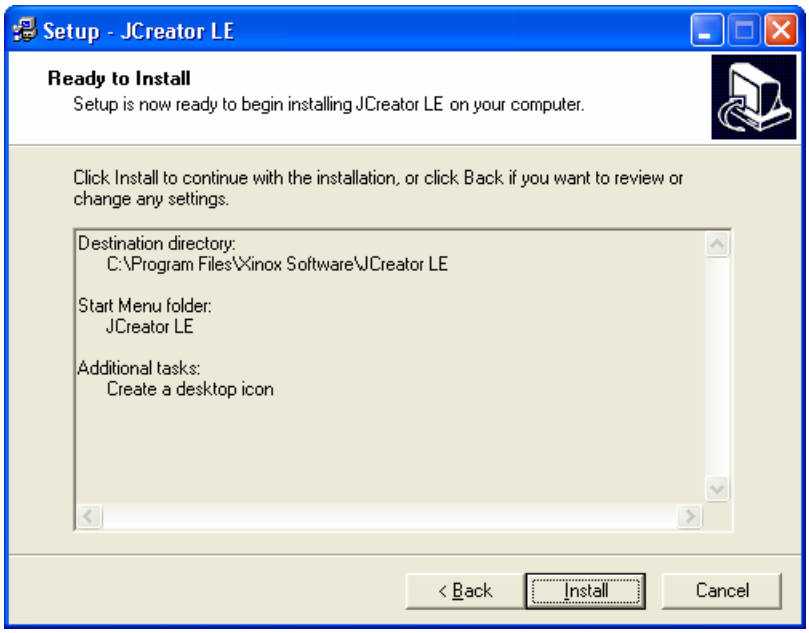
Ovog puta radi se o datoteci relativno male veličine (manje od 2 MB), tako da preuzimanje neće dugo trajati.

Po preuzimanju komprimovane datoteke, raspakujte je pomoću WinZipa. Dvo-kliknite na datoteku Setup.exe da bi započeli instalaciju. Na ekranu će se pojaviti prozor sledećeg izgleda:

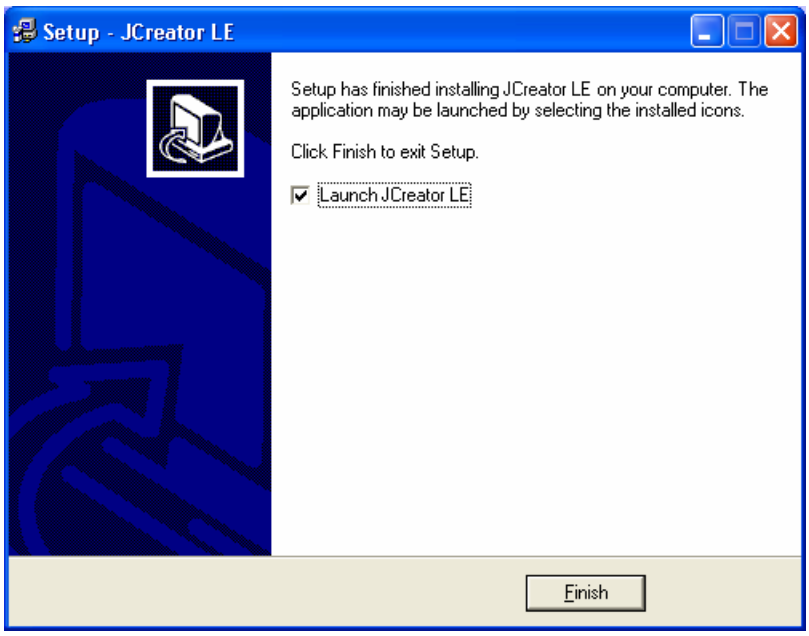


Kliknite mišem na dugme Next i videćete sporazum o licenci (naravno, treba da se saglasite da bi se instalacija nastavila).

Na ekranu će se pojaviti za redom nekoliko prozora posredstvom kojih će vas instalacioni program pitati za odredište za neke datoteke i da li želite da se ikona programa pojavi na radnoj površini (Desktopu).

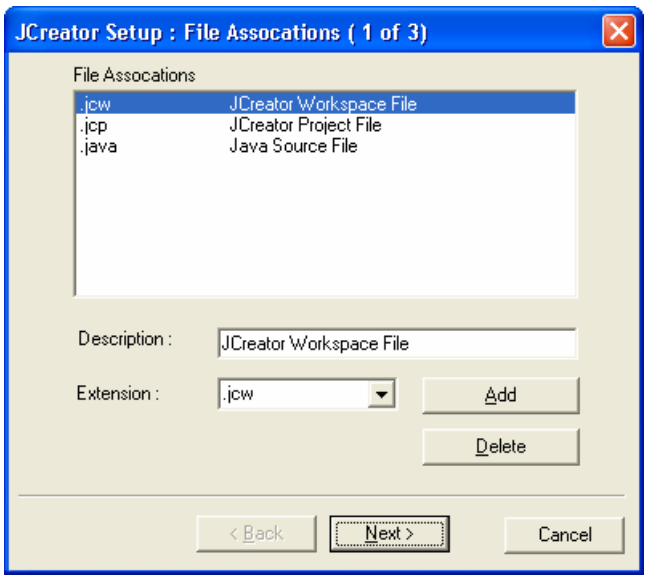


Kliknite mišem na Install i počinje instalacija. Kada se završi videćete sledeći prozor.

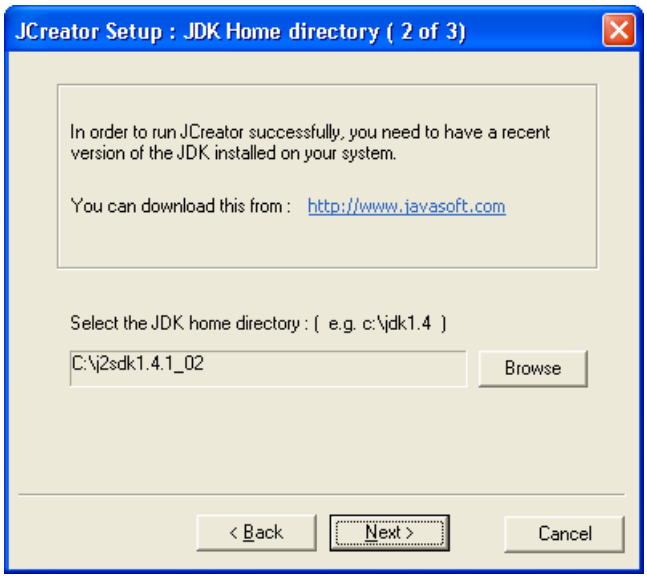


Kliknite mišem na dugme Finish (potvrdite opciju Launch JCreator LE).

JCreator će se aktivirati i postaviti vam još nekoliko pitanja. Prvo, pitaće vas za pridruživanje datoteka:

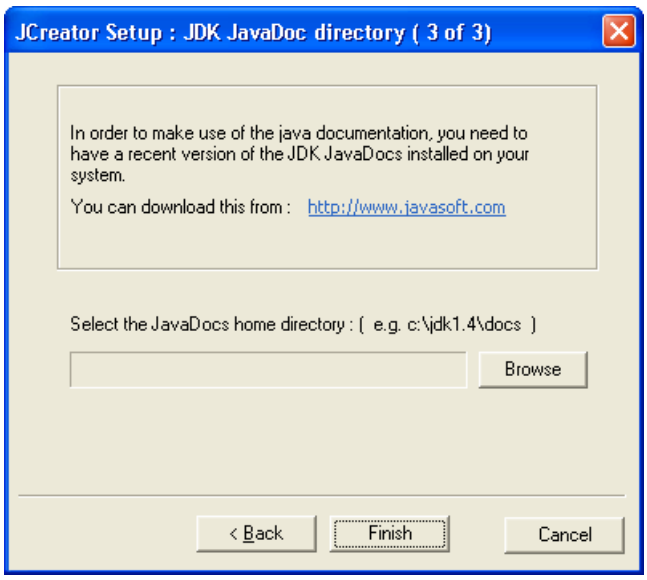


Prihvatite ponuđeno klikom miša na Next. Sada će se pojaviti jedan veoma važan prozor:



JCreator od vas traži informaciju o instaliranoj verziji Java SDK (odnosno JDK – Java Development Kit). Kliknite mišem na dugme Browse (ako je potrebno) da biste definisali putanju do Java direktorijuma (verovatno, c:\j2sdk1.4.1_02), a zatim kliknite na Next.

Na ekranu se pojavljuje prozor



JCreator želi da zna gde se nalazi Java SDK dokumentacija. Za potrebe ovog kursa dokumentacija nije neophodna, zato nisam ni naveo da je preuzmete sa Sunovog sajta. Na kraju, kliknite mišem na dugme Finish i potom će se JCreator pokrenuti.

U sledećem nastavku upoznaćemo se sa interfejsom programa JCreator i nastaviti dalje učenje Jave.

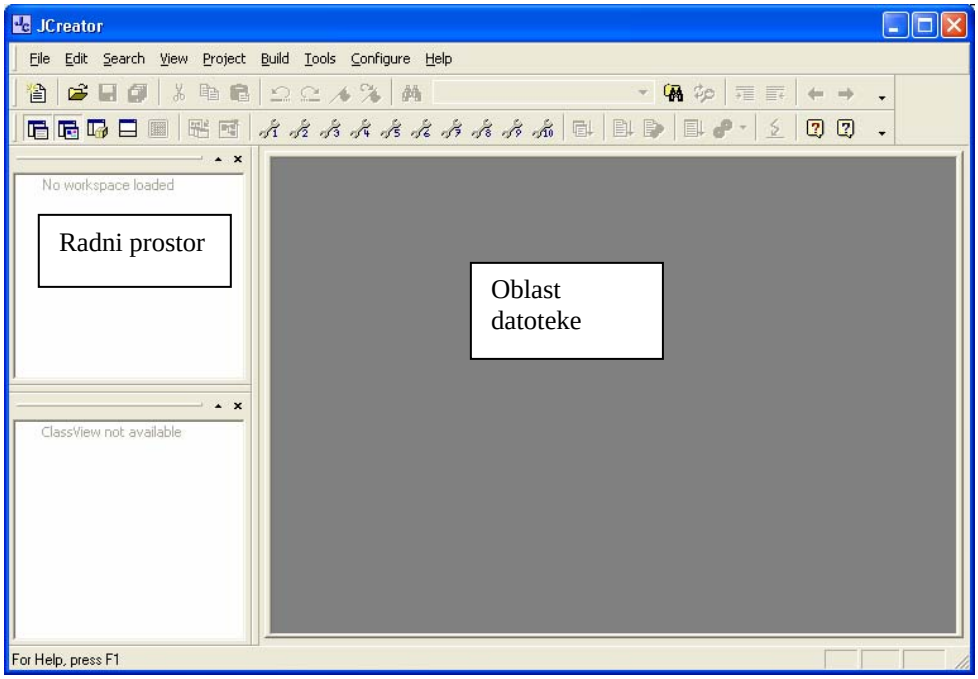
Java za mlade programere (3)

Pokretanje JCreatora

Kada je instaliran, JCreator se pokreće sledećom procedurom:
Kliknite mišem na dugme Start na Windows paleti poslova.
Izaberite u meniju Programs stavku JCreator LE.
Kliknite mišem na JCreator LE.

Ukoliko ste izvukli prečicu na radnu površinu, tada ćete JCreator pokrenuti tako što ćete dvo-kliknuti mišem na ikonu ovog programa.

JCreator je pokrenut i na ekranu se pojavljuje nekoliko prozora.



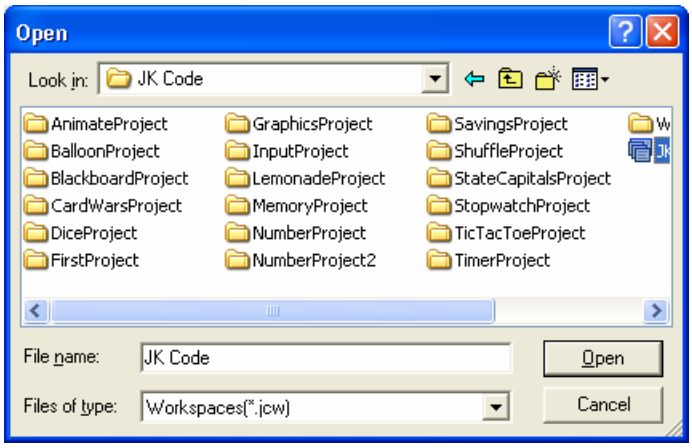
Ovaj ekran prikazuje JCreatorovo integrisano razvojno okruženje (engl. Integrated Development Environment - IDE). O integrisanom razvojnom okruženju ćemo naučiti nešto više u nekom od sledećih nastavaka. Za sada,

koristićemo ga da iztestiramo Java instalaciju i da vidimo kako se učitava i pokreće neki program. Obratite pažnju na lokacije radnog prostora, oblast datoteke i glavni meni. Radni prostor nam prikazuje koji Java programi su nam na raspolaganju. Oblast datoteke se koristi za prikaz aktuelnog koda, a glavni meni za kontrolu pristupa datoteci i funkcije uređivanja datoteke. Meni se takođe koristi za kompajliranje i izvršavanje programa.

Otvaranja Java projekta

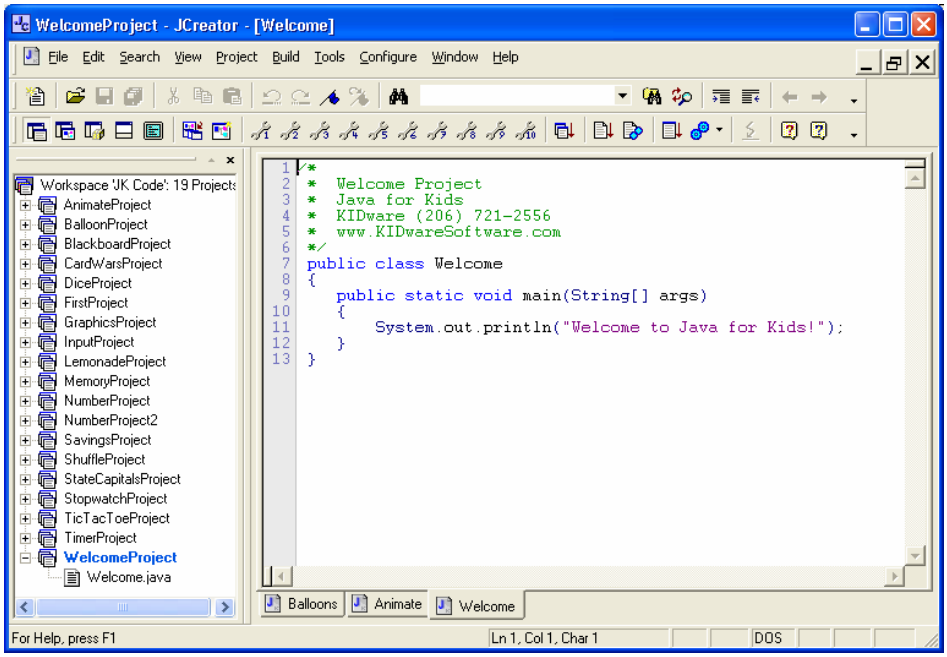
Ono što želimo sada da uradimo je da otvorimo projekat. Računarski programi (aplikacije) napisani u Javi često se nazivaju projektima. Projekti uključuju sve informacije u datotekama koje su potrebne našem programu. Java projekti su grupisani u radnim prostorima.

Neka je JCreator pokrenut. Prvi korak kod otvaranja projekta je otvaranje radnog prostora koji sadrži projekat koji nas interesuje. Sledite ove korake:
Izaberite u meniju File opciju Open Workspace. Na ekranu se pojavljuje prozor Open:



Svi projekti u slučaju koji ćemo razmatrati su sačuvani u radnom prostoru pod nazivom \JavaKids\JK Code. Pređimo u taj direktorijum i izaberimo JK Code radni prostor. Kliknimo Open.
Biće prikazano više projekata u radnom prostoru. Pronađimo projekat WelcomeProject. Uradimo desni klik mišem na nazivu tog projekta i iz pomoćnog

menijanizaberimo stavku Sets as Active Project. Klikom miša na znak + pored naziva projekta razvimo čvorište projekta. Sada ćemo videti da u njemu postoji jedna datoteka pod nazivom Welcome.java. Ako se sadržaj fajla ne pojavi u oblasti prikaza datoteke, dvo-kliknite na tu datoteku da bi se otvorila.

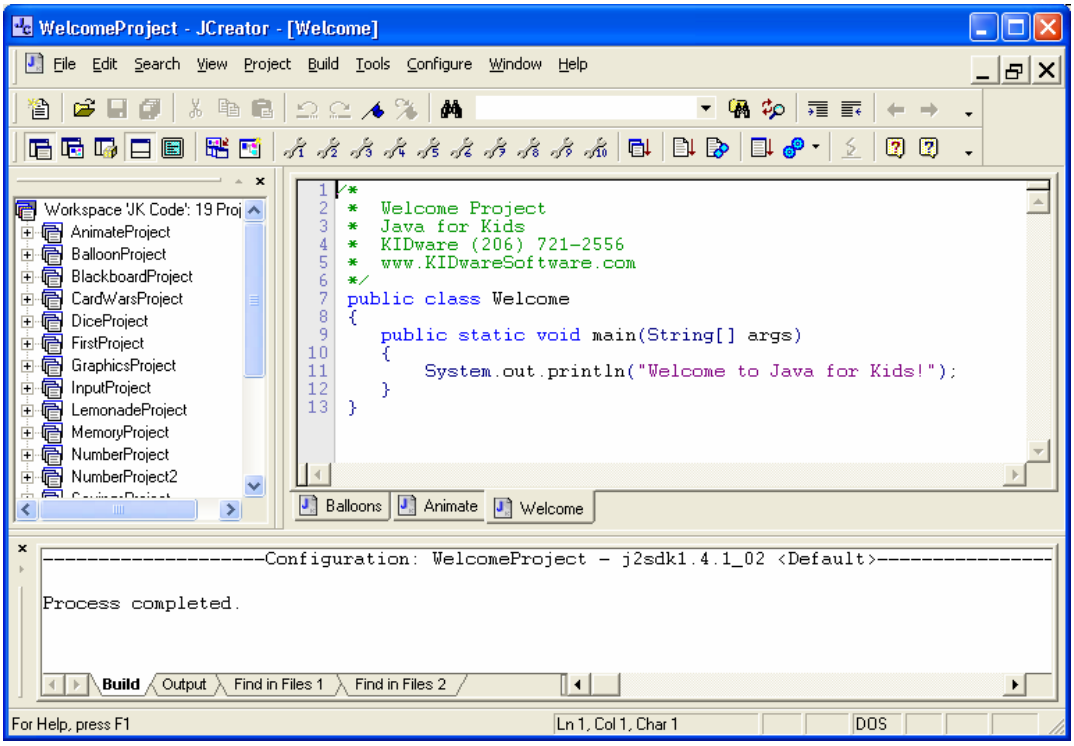


Konačno imate priliku da vidite svoj prvi Java program; kasnije ćemo objasniti šta tih nekoliko redova koda radi. Za sada, želimo samo da vidimo da li možemo da pokrenemo taj program.

Kompajliranje i izvršavanje Java projekta

Pošto razvijete Java projekat, prirodno je da želite da pokrenet ili izvršite taj program. To se ostvaruje u dva koraka: kompajliranjem i izvršavanjem. Ponovo, o tome ćete više saznati u nekom od sledećih nastavaka. Za sada, proveravamo da li je sve instalirano kako treba.

Da biste kompajlirali projekat Welcome, izaberite u glavnom meniju stavku Build, a zatim Compile Project. Druga varijanta je da pritisnete samo taster F7 na tastaturi. Na ekranu treba da se pojavi novi prozor koji je prikazan na sledećoj slici.



Ako je sve instalirano kako treba, u donjem prozoru treba da vidite reči Process completed. Time nas JCreator obaveštava da je projekat uspešno kompajliran.

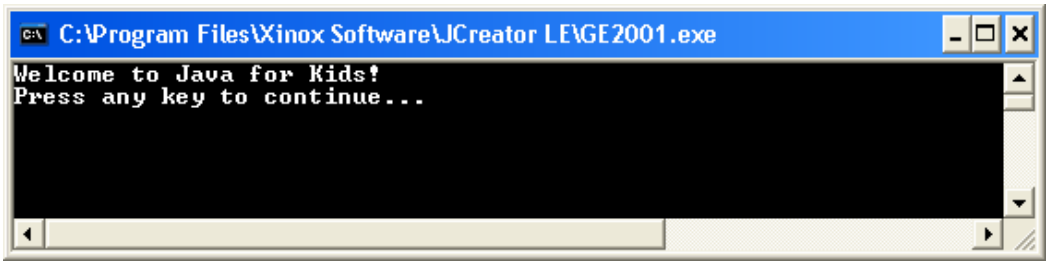
Ako program nije kompajliran, najverovatnije da JCreator nije mogao da pronađe Java SDK. Da biste proverili da li JCreator „zna" gde je Java SDK, pokušajte sledeće:

U glavnom meniju izaberite stavku Configure.
Izaberite stavku Options.
U okviru za dijalog Open kliknite mišem na JDK Profiles.

U prozoru treba da bude direktorijum gde je Java instalirana. Ako ga nema, izaberite stavku New i sledite instrukcije kako biste pronašli direktorijum. Ako je direktorijum naveden ali to nije odgovarajući direktorijum, označite navedeni direktorijum i izaberite Edit. Uradite potrebnu korekciju naziva i lokacije direktorijuma.

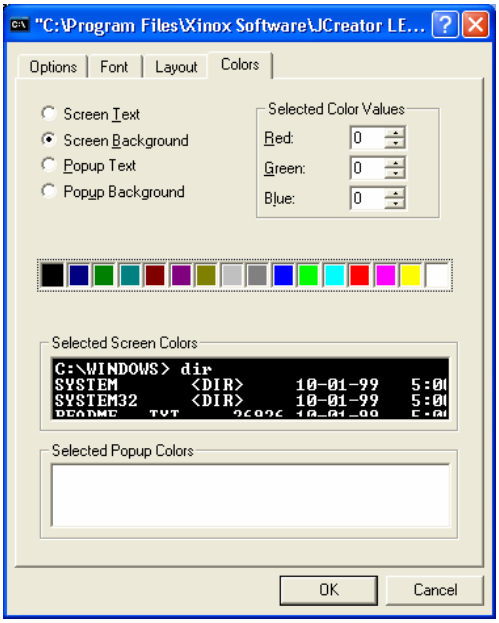
Da li ste sada spremni da pokrenete svoj prvi projekat? Da biste to uradili, izaberite u glavnom meniju Build, a zatim stavku Execute Project (a možete isto dejstvo

proizvesti i pritiskom na taster F5 na tastaturi). Na ekranu treba da se pojavi prozor sa Welcome porukom, kao na sledećoj slici.



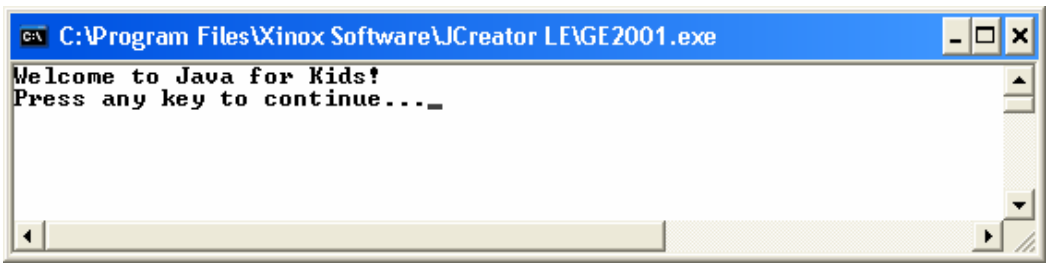
Ako se to desilo, sve je instalirano kako treba. Ako nije tako, nešto nije instalirano kako treba i moraćete kompletno da proverite instalaciju Jave i JCreatora kao biste bili sigurni da je sve urađeno na odgovarajući način.

Pre nego što zaustavimo izvršavanje programa, napravimo jednu malu izmenu na izlaznom prozoru. Inicijalna postavka okruženja je da izlazni prozor prikazuje („štampa”) beli tekst na crnoj pozadini. To je u redu, ali nije baš racionalno. Naime, ako nastavimo tako da radimo, prilikom štampanja ovih prozora potrošili bi mnogo tonera. Možemo promeniti boje i to vrlo lako. Kliknimo mišem na malu ikonu u gornjem levom uglu izlaznog prozora. Izaberimo u pomoćnom meniju stavku Properties. Kada se na ekranu pojavi okvir za dijalog, izaberimo karticu Colors.



Na ovoj kartici možemo definisati mnogo stvari, između ostalog i novi Screen Background (pozadina ekrana) i novu boju za Screen Text (ekranski teksta). Izbor se obavlja tako što prvo označite odgovarajuće radio dugme pored tih naziva a zatim izaberete boju sa ponuđene palete. Izaberite boje koje želite, a ja preporučujem svetlu pozadinu sa tamnim tekstom. Kada ste obavili izbor kliknite na dugme OK. Na ekranu se pojavljuje sledeći okvir za dijalog:

Navedite gde želite da snimite ova svojstva koja će JCreator koristiti za sve vaše projekte i potom kliknite na OK. Izlazni prozor će sada biti obojen novim bojama.



Da biste zaustavili izvršavanje projekta, pritisnite bilo koji taster ili kliknite mišem na dugme X u gornjem desnom uglu prozora.

Napuštanje JCreatora

Kada ste završili rad sa Java projektom, preostalo je još da napustite JCreator razvojno okruženje. To je ista procedura kao i kod svih drugih Windows aplikacija:
U glavnom meniju izaberete stavku File.
Izaberete sada stavku Exit (na dnu menija File).

JCreator će zatvoriti sve otvorene prozore i bićete vraćeni na radnu površinu Windowsa. Možete koristiti i alternativni način, tj. kliknuti mišem na dugme X u gornjem desnom uglu glavnog prozora JCreatora.

Java za mlade programere (4)

Osnove Java programa

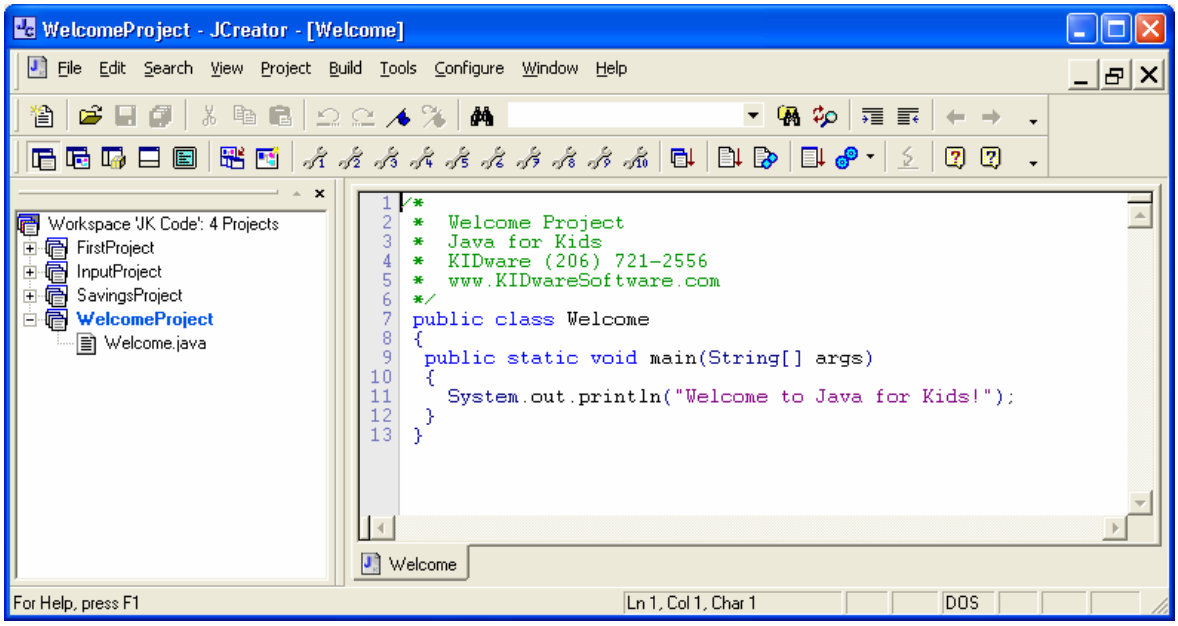
U prethodnim nastavcima opisali smo kako da se pripremi računar za kreiranje i izvršavanje Java programa. U ovom nastavku naučićete kako izgleda osnovna struktura Java programa, zatim ćete naučiti neka osnovna pravila za pisanje Java programa. Kreiraćemo i snimiti projekat pomoću Jcreatora, razvojnog okruženja koje smo upoznali u prethodnom nastavku. Na kraju, naučićete šta se dešava kada kompajlirate (prevodite) i izvršavate Java program. Ovime ćete steći neophodna znanja da napravite svoj prvi Java program.

Struktura Java programa

Java, kao i svaki drugi jezik (računarski ili govorni), ima svoju terminologiju. Pogledajmo zato strukturu Java programa i naučimo tu novu terminologiju. Java program (ili projekt) čini nekoliko datoteka. Ove datoteke se nazivaju **klasama**. Svaka od tih datoteka ima Java kôd koji obavlja neki specifičan zadatak (zadatke). Svaka datoteka klase ima oznaku tipa datoteke (tzv. Ekstenziju) **.java**. Naziv datoteke koji se koristi za snimanje klase mora da odgovara

nazivu klase. Jedna klasa u svakom projektu će imati nešto što se nekad zove glavni metod (engl. **main method**). Kagdog izvršavate Java program, vaš računar će prvo potražiti glavni metod da bi nešto počelo da se dešava. Prema tome, da biste izvršili program, direktno navodite klasu koja sadrži glavni metod.

Pogledajmo kako to izgleda u primeru **Welcome Project,** koji smo imali prilike ranije da vidimo. Pokrenimo prvo **JCreator**. Welcome Project bi trebalo da je još aktivan projekt. Ako nije tako, uradite desni klik mišem na folder **WelcomeProject** u radnoj oblasti i izaberite stavku **Sets as Active Project.** Trebalo bi da vidite:



Ovaj projekt ima jednu datoteku koja se naziva **Welcome.java**. Uočite da, kao što je zahtevano, naziv **Welcome** odgovara nazivu klase koji se vidi u kodu (public class **Welcome**). Ako se kôd ne vidi, uradite dvo-klik mišem na datoteku koja ima naziv **Welcome.java**. Ako projekt ima još klasa, one bi trebalo da budu izlistane ispod foldera **WelcomeProject**. Takođe, uočite u kodu oblast koja ima reč **main**. To je glavni metod koji nam je potreban u jednoj od klasa projekta.

To je zaista sve što je potrebno da znamo o strukturi Java programa. Samo da vas podsetim da **program** (ili projekt, koristićemo oba termina) sačinjava više datoteka koje se nazivaju **klasama**, a koje sadrže aktuelni Java kôd. Ona klasa od koje sve počinje je **glavna** klasa. Još jedna stvar, projekti se grupišu u radne oblasti (engl. **Workspaces**). Sa ovim znanjem možemo da počnemo sa razmatranjem **Welcome** programa red po red kako bi shvatili šta je Java program uopšte.

Welcome Project - ponovo

Trebalo bi da je još uvek aktivan **JCreator** u kome se izvršava **Welcome** program. Ako nije tako, pokrenite JCreator, otvorite **WelcomeProject** (desni klik na naziv

projekta i izbor stavke **Set as Active Project**) i dvo-kliknite mišem na datoteku **Welcome.java**. Trebalo bi da vidite sledeći kôd:

```
/*
 *  Welcome Project
 *  Java for Kids
 *  KIDware (206) 721-2556
 *  www.KIDwareSoftware.com
 */
 public class Welcome
 {
   public static void main(String[] args)
   {
     System.out.println("Welcome to Java for Kids!");
   }
 }
```

Prođimo kroz ovaj kôd red po red kako bi objasnili njegovu strukturu.

Prvih šest redova programa su:

```
/*
 *  Welcome Project
 *  Java for Kids
 *  KIDware (206) 721-2556
 *  www.KIDwareSoftware.com
 */
```

U tim redovima je **komentar** (engl. **Comment**). Ti redovi obezbeđuju neke informacije o programu, tj. O kom

programu se radi i informaciju za kontakt. Komentar počinje sa simbolom **/*** i završava se sa simbolom ***/**. Ovi redovi su poznati i kao **zaglavlje programa** (engl. **program header**). Dobro je da uvek stavite zaglavlje u svoje Java programe u kojima je informacija šta rade i ko ih je napisao. Java kompajler ignoriše svaki komentar – njihova uloga je samo da pruže objašnjenje.

Prvi red posle komentara sadrži sledeće:

```
public class Welcome
{
```

Taj red sadrži definiciju naše klase nazvane **Welcome**. Ključna (službena; engl. keyword) reč **public** određuje da li drugi delovi programa mogu da pristupaju ovoj klasi. **Ključne reči** su deo svakog programskog jezika – to su rezervisane reči i ne mogu se koristiti u regularnim Java izrazima. Leva vitičasta zagrada (**{**) se koristi da označi početak definicije klase. Uočićete brzo da se mnogo vitičastih zagrada koristi u Java programima!

Sledeći red je:

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

U ovom redu se kreira glavni metod koji smo ranije pominjali. Nemojte se za sada brinuti šta sve te reči znače. Bitno je da uočite da tako počinje glavni metod u kome pišemo Java kôd koji želimo da se izvrši kada se program pokrene. U ovom kursu uglavnom ćemo stavljati sav naš kôd u glavni metod. Uočite da se druga leva vitičasta zagrada koristi za početak definisanja glavnog metoda.

Jedini **Java iskaz** (naredba; engl. statement) u glavnom metodu je:

System.out.println("Welcome to Java for Kids!");

Podsetite se kako je izgledalo izvršavanje Welcome projekta u prethodnom nastavku. Kada ste ga pokrenuli, videli ste u izlaznom prozoru poruku koja je glasila - **Welcome to Java for Kids!** Prethodni red koda je ispisao tu poruku. U tom redu **System** je klasa ugrađena u Javu, **out** je objekat klase (koji se odnosi na izlazni prozor (engl. output window)). Reč **println** (izgovara se print line) prikazuje jedan red teksta. Tekst koji se prikazuje je pod navodnicima. Uočite da se iskaz završava sa (**;**) – u Javi, takođe, postoji mnogo tačka-zareza! U ovom prostom primeru, glavni metod ima samo jedan iskaz. Naravno, naredni primeri će imati mnogo više iskaza. **Metodi** su mesto gde Java programi obavljaju zadatke. Pored pisanja sopstvenih metoda, možete koristiti

bilo koji od mnogobrojnih metoda koji su ugrađeni u Java jezik. Učićete o tim metodima kako bude kurs odmicao.

Posle tog reda slede dva reda od kojih svaki ima desnu vitičastu zagradu (**}**). Prva zagrada okončava glavni metod, dok druga okončava definiciju klase. Biće potrebno da uvek pazite da imate u Java programima vitičaste zagrade uparene, tj. da svakoj levoj odgovara jedna desna.

Mada je ovo veoma kratak i prost program, on ilustruje koje su glavne komponente u Java programu. Potrebno vam je zaglavlje programa, definicija klase i glavni metod. Takođe, potrebno je da vodite računa i snimite datoteku klase pod istim nazivom pod kojim je klasa definisana. Ta datoteka ima oznaku tipa (tzv. ekstenziju) **.java**.

Neka pravila Java programiranja

Pogledajmo još jednom kôd **Welcome** projekta kako bismo ukazali na neka osnovna pravila Java programiranja. Evo tog koda:

```
/*
 *  Welcome Project
 *  Java for Kids
 *  KIDware (206) 721-2556
 */
```

```
public class Welcome
{
  public static void main(String[] args)
  {
    System.out.println("Welcome to Java for Kids!");
  }
}
```

A evo i pravila:

- Java kôd zahteva perfekciju. Sve ključne reči moraju biti korektno zapisane. Ako napišete **printline** umesto **println**, čovek može da zna na šta ste mislili, ali računar ne.
- Java razlikuje mala i velika slova (engl. case-sensitive), što znači da se velika i mala slova smatraju različitim znakovima. Kada pišete kôd, vodite računa kako koristite slova. U Javi reči **Main** i **main** su potpuno različite.
- Java ignoriše beline (engl. "**white space**"; blankove). Često ćete koristiti beline da bi vaš kôd izgledao čitljivije.
- **Vitičaste zagrade** se koriste za grupisanje. One markiraju početak i završetak programskih sekcija. Vodite računa da vaši Java programi imaju jednak broj levih i desnih vitičastih zagrada. Sekcija koda između dve odgovarajuće vitičaste zagrade naziva se **blok**.

- Dobra praksa je da uvlačite kôd bloka prilikom pisanja programa. To olakšava praćenje koda. Uočite da je u datom primeru svaki blok uvučen tri blanka (beline). Ako koristite JCreator, on će automatski umesto vas uvlačiti kôd u blokovima.
- Svaki Java iskaz se završava tačka-zarezom. Iskaz je programski izraz koji proizvodi neku akciju (na primer, ranije pomenuti iskaz **Println**). Uočite da nisu svi Java izrazi iskazi (na primer, red koji definiše glavni metod nema tačku-zarez).

Naučićete još o Java programiranju ako nastavite da pratite ovaj kurs.

Pripremio Dragan Marković

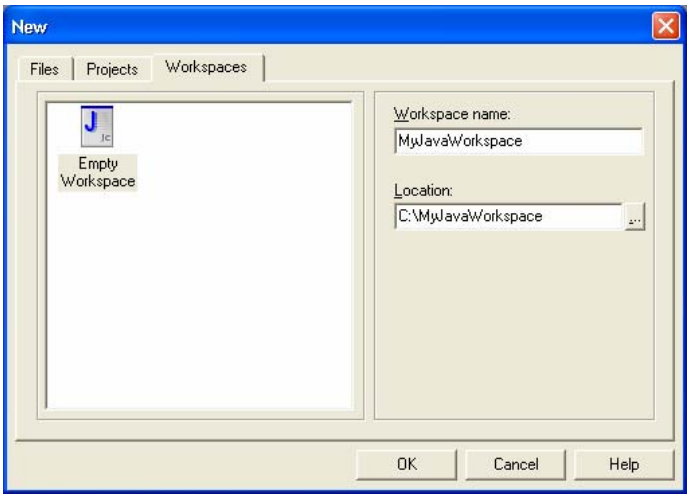
Java za mlade programere (5)

Kreiranje Java projekata pomoću JCreatora

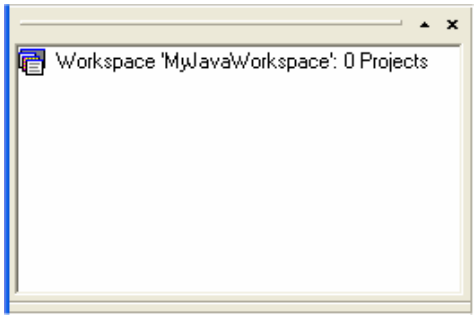
Na ovom „času” počećemo da učimo programski jezik Java i da pišemo svoje Java programe. Da bi ste mogli da to uradite potrebno je da znate kako se kreira novi projekat pomoću JCreatora. Takođe, potrebno je da znate kako da dodate datoteke svom projektu. Sada ću pokazati kako se to radi. Ponovo ćemo kreirati projekt **Welcome** u vašem radnom prostoru.

Ako Jcreator nije pokrenut, pokrenite ga. Radni prostor koji sadrži projekt Welcome trebalo bi da je još uvek tu. Uklonićemo ovaj radni prostor i kreirati novi. (Treba samo da koristite radni prostor **JK Code** kada želite da referencirate kôd vezan za ovo uputstvo. Za sve svoje projekte, koristićete sopstveni radni prostor). Da biste uklonili radni prostor, o oblasti za prikaz radnog prostora označite tekući radni prostor (**JK Code**). U glavnom meniju izaberite **File** a zatim **Close Workspace**.

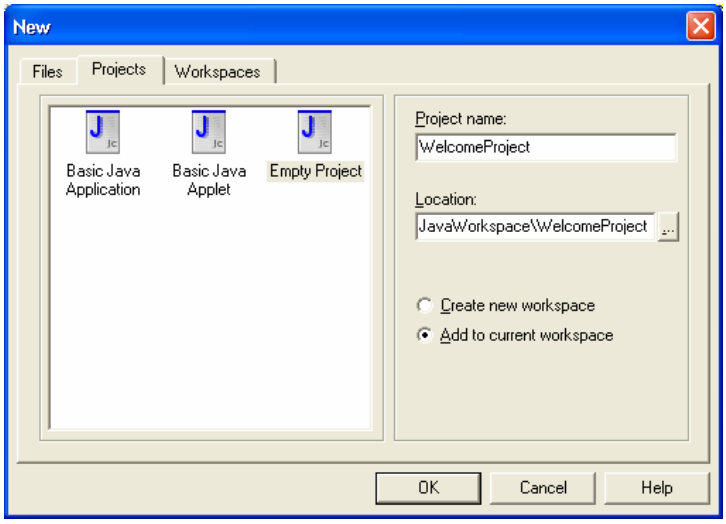
Sada, kreirajte svoj radni prostor – dodelite mu naziv **MyJavaWorkspace**. U glavnom meniju izaberite **File** a zatim **New**. Kada se pojavi prozor **New**, kliknite na jezičak **Workspaces** :



Kliknite na (**...**) pored okvira **Location** i izaberite **c:** disk (ili neku drugu lokaciju). Upišite **MyJavaWorkspace** u polje **Workspace name**. Kliknite mišem na dugme **OK** i kreiraće se prazan radni prostor u folderu pod nazivom **MyJavaWorkspace** na vašem disku **c:** . Ovaj novi radni prostor pojavljuje se u prikazu radnog prostora programa JCreator:

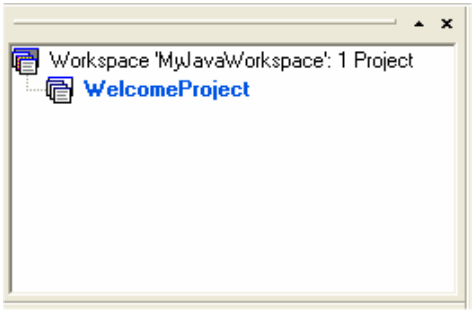


Možemo sada da dodamo projekt u radni prostor. Obratite pažnju na ove korake pošto ćete ovo ponavljati svaki put kada je potrebno da kreirate novi Java projekt. Uradite desni klik u prikazu radnog prostora i izaberite **Add new Project** :

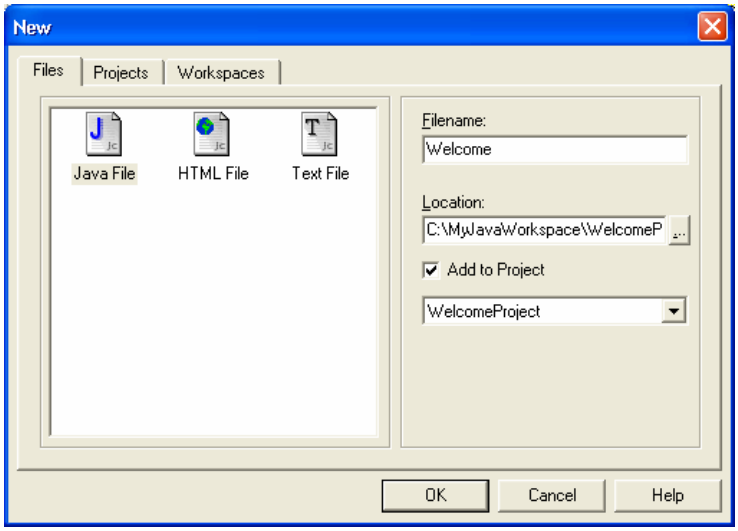


Izaberite jezičak kartice **Projects**, zatim **Empty Project** i upišite **WelcomeProject** u polje **Project name** (kao što je prikazano). Kliknite na **OK** da biste dodali projekt.

Prozor za prikaz radnog prostora trebalo bi sada da prikaže projekt (**WelcomeProject**) u radnom prostoru:

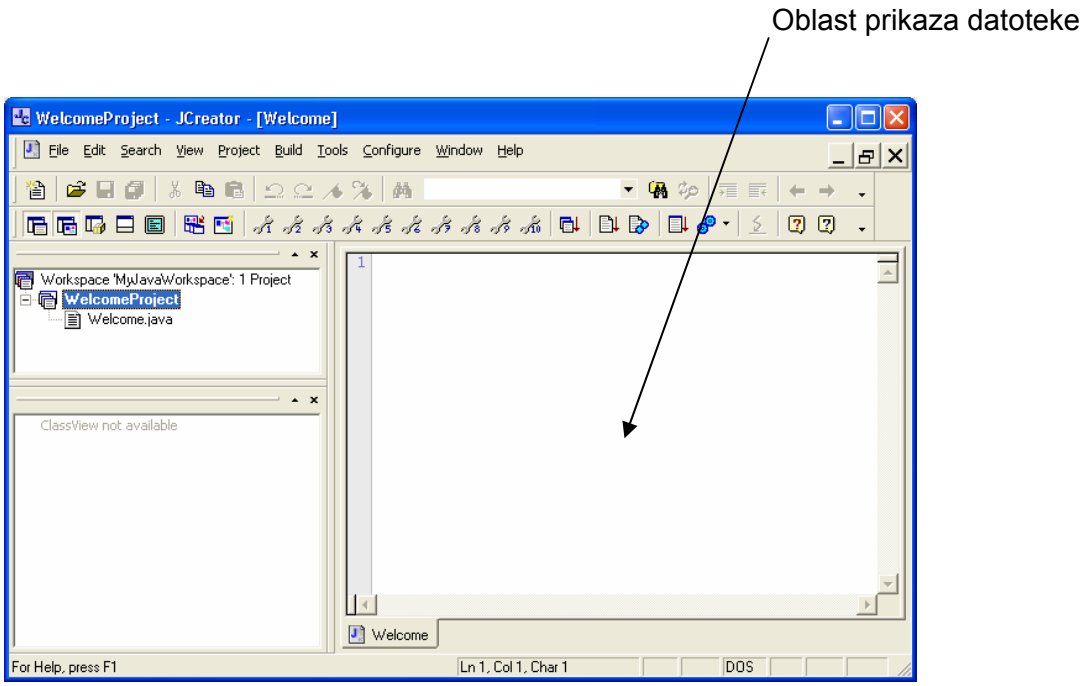


Po proceduri, sada treba da dodamo datoteku (našu datoteku sa Java programom) projektu. Označite **WelcomeProject** levim klikom na naziv u prikazu radnog prostora. Odaberite **File** iz glavnog menija JCreatora a zatim stavku **New**. Pojavljuje se sledeći prozor:



Kao što je prikazano, izaberite karticu **Files**, izaberite **Java File** i upišite **Welcome** u polje Filename (dobro je da steknete naviku da nazivi vaših datoteka imaju drugačije nazive od foldera projekta i radnog prostora). Kliknite na **OK**.

Razvite **WelcomeProject** (kliknite na znak +) i videćete datoteku **Welcome.java** koja je dodata projektu. U oblasti za prikaz datoteke desno od prikaza radnog prostora je editor gde ćete upisivati Welcome.java kôd:



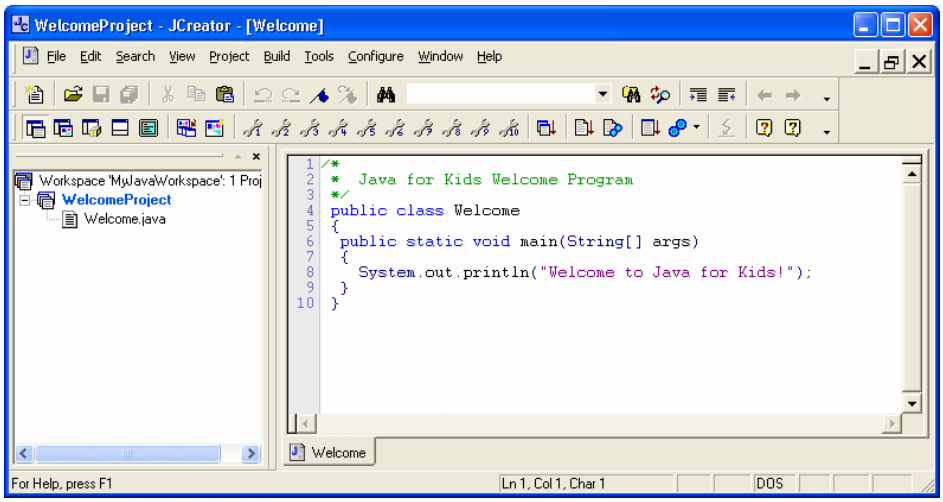
Pišite liniju po liniju koda i obratite pažnju da pišete sve kako je ovde dato (imajući u vidu pravila koja ste ranije naučili). Evo koda još jednom:

```
/*
 *  Java for Kids Welcome Program
 */
 public class Welcome
 {
   public static void main(String[] args)
   {
     System.out.println("Welcome to Java for Kids!");
   }
 }
```

Kako pišete uočićete da posle pisanja svake leve vitičaste zagrade (**{**), JCreator editor automatski uvlači sledeći red, a čim upišete odgovarajuću desnu vitičastu zagradu (**}**), uvlačenje se povlači nazad za jedan nivo. To znači da editor poštuje pravilo uvlačenja svakog bloka koda. Inicijalno, JCreator uvlači za 4 znaka (mesta) na svakom nivou. Da biste promenili ovu vrednost, izaberite u glavnom meniju stavku **Configure** a zatim **Options**. Odaberite **Java Editor** i upišite novu vrednost za **Tabs Size**. Mi ćemo na ovom kursu koristiti vrednost 2 a ne 4 (stvar ličnog ukusa).

Druga stvar koju treba da uočite je da editor koristi različite boje za različite stvari u kodu. Zeleni tekst predstavlja komentare. Kôd je u crnoj boji a ključne (rezervisane) reči u plavoj. Ova šema boja ponekda može da vam pomogne da otkrijete greške koje ste možda napravili prilikom kucanja.

Kada završite kucanje, trebalo bi da vidite:



Pokušajte da kompajlirate (prevedete) svoj program (izaberite **Build**, zatim **Compile Project**, ili samo pritisnite taster <**F7**>). Trebalo bi ubrzo da vidite reči **Process completed**. Ako nije tako, proverite da li ste sve upisali kako je navedeno u prethodnom listingu. Da vas podsetim, greške se ne tolerišu.

Kada je vaš program kompajliran, možete ga aktivirati biranjem opcije **Build**, zatim **Execute Project** (ili pritisnite taster <**F5**>). Trebalo bi ponovo da vidite poruku **Welcome to Java for Kids!** . Takođe, vidite i sami kako je zaista lako dobiti Java program pomoću JCreatora.

Snimanje Java projekata u JCreatoru

Pre nego što napustimo JCreator, potrebno je da znamo kako se snimaju projekti koje smo napravili. Postoje dve stvari koje treba razmotriti: snimanje projekata i snimanje radnih

prostora (foldera koji sadrže projekte). Kadgod kompajlirate i/ili izvršavate Java projekt, JCreator automatiski snima i izvorne datoteke i datoteke kompajliranog koda. Prema tome, veći deo vremena ne morate brinuti o snimanju svojih projekata – Jcreator je to preuzeo na sebe. Ako želite da snimite kôd koji ste upravo napisali (pre kompajliranja), jednostavno izaberite u glavnom meniju **File** a zatim kliknite na **Save All**. Ili, samo kliknite na dugme **Save All** na paleti alatki:

Treba da snimite radni prostor svaki put kada napravite promenu, na primer, ako dodate/obrišete datoteke projekta ili dodate/obrišete projekte. Da snimite radni prostor odaberite u glavnom meniju **File** a zatim **Save Workspace**. Ako pokušate da napustite JCreator a niste snimili projekte ili radne prostore, JCreator će vas preko okvira za dijalog upozoriti na to i pružiti vam mogućnost da snimite datoteke pre napuštanja.

Kompajliranje i izvršavanje Java programa

U primeru koji smo upravo završili, u tri koraka smo obavili kreiranje i izvršavanje Java programa:

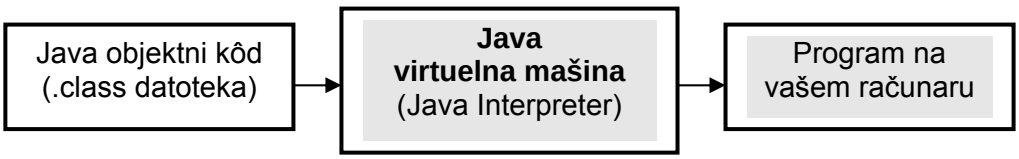
- **Pisanje** koda
- **Kompajliranje** koda
- **Izvršavanje** koda

Vidimo da pisanje koda obuhvata poštovanje uspostavljenih pravila (mada ćemo stvarno početi sa pisanjem sopstvenog koda kasnije). Takođe, znamo kako da kompajliramo i izvršavamo program, ali šta se stvarno dešava kada mi radimo te dve stvari. Sada ćemo to razmotriti.

Da vas podsetim, jedna od velikih prednosti Jave je što će Java programi raditi na raznorodnim mašinama bez promena u kodu. Kažemo da je Java **nezavisna od platforme**. Ova nezavisnost je moguća zato što se Java program ne izvršava direktno na vašem računaru, već na virtuelnom računaru koji je instaliran na vašem računaru zajedno sa Java Software Development Kitom (SDK). Taj virtuelni računar se naziva **Java virtuelna mašina**. Pre nego što Java virtuelna mašina može da razume vaš Java program, on mora da bude pretvoren ili preveden u jezik koji ona razume. Ova konverzija se obavlja kada kompajlirate svoj Java program. Kompajliranje Java programa kreiradatoteku sa **objektnim kodom** koji Java virtuelne mašina može da razume. Ta datoteka će imati oznaku tipa datoteke (tzv. ekstenziju) **.class**. Na sledećoj slici je grafički prikazan proces kompajliranja:

Kada izvršavate svoj Java program, ove **.class** datoteke obrađuje nešto što se naziva **Java interpreter unutar** virtuelne mašine. Ovaj interpreter razgovara sa vašim

računarom i prevodi vaše Java naredbe u željene rezulatate. To je kao magija! Na sledećoj slici je grafički prikazan proces izvršavanja programa:



Java za mlade programere (6)

JCreator i Java datoteke

Sada ćemo se upoznati kako su te sve informacije vezane za strukturu programa, datoteke, kompajliranje i izvršavanje uklopljenje u JCreator, koji je naše razvojno okruženje. Ranije smo videli da su Java projekti grupisani u radne prostore. A projekti se sastoje od **.java** datoteka.

Koristeći My Computer ili Windows Explorer, pređimo u folder koji sadrži **Welcome Project** koji ste upravo napravili, kompajlirali i izvršili. Trebalo bi da vidite sledeće datoteke:

Welcome.java
Welcome.class
WelcomeProject.jcp
src_welcomeproject.txt

Welcome.java je izvorni kôd koji se pojjavljuje u oblasti za prikaz datoteke JCreatora. **Welcome.class** je kompajlirana verzija tog izvornog koda (Welcome.java) – to je datoteka koja je potrebna Java virtualnoj mašini.
WelcomeProject.jcp je projektna datoteka Jcreatora koja se koristi za evidenciju koje sve datoteke čine projekat a **src_welcomeproject.txt** je druga datoteka Jcreatora koja

je potrebna za organizaciju. Pored toga, u foldeu vašeg radnog prostora biće i datoteka:

MyJavaWorkspace.jcw

(Naravno, ovde se podrazumeva da ste radnom prostoru dodelili naziv MyJavaWorkspace). To je još jedna datoteka Jcreatora koja se koristi za evidenciju projekata u radnom prostoru **JK Code**.

Imajte na umu da su ovde prave Java datoteke samo one koje imaju oznaku tipa **.java** i **.class**. Ostale datoteke stvara i menja naše konkretno razvojno okruženje, u ovom slučaju JCreator. Ako želite da delite svoj Java program sa prijateljem ili da ga prebacite u neko drugo razvojno okruženje, jedine datoteke koje treba da prebacite su **.java** datoteke. Te datoteke može da koristi svaki Java programer ili programsko okruženje da bi napravio program koji može da se izvršava.

Kreiranje Java programa

Ranije sam rekao da **Java naredba** nešto radi. U primeru **Welcome** videli smo da naredba štampa neku informaciju (“Welcome to Java for Kids!”). Svaki program

koji budemo pravili biće sačinjen od više Java naredbi koje računar treba da obradi. Pravljenje računarskog programa pomoću Jave (ili bilo kog drugog jezika) je direktan proces. Imate konkretan zadatak koji želite da računar obavi za vas. Saopštavate računaru logičan, proceduralan skup koraka koje treba da preduzme da bi obavio zadatak.

Relativno je lako ispisati korake rešenja na našem govornom jeziku. Međutim, za nas je malo teže to saopštiti računaru na njegovom jeziku. Bilo bi zaista lepo kada bi mogli samo da napišemo "Zdravo računaru, evo dva broja – saberi ih i kaži mi koliki je zbir". Čovek može da razume ove instrukcije, ali računar ne može. Zašto? Prvo, računaru je potrebno da se kaže na vrlo specifičan način kako da uradi zadatak, tj. razlaganjem na logičke korake. Za ovaj prost primer sabiranja ti koraci bi bili:

1. Zadaj vrednost prvog broja.
2. Zadaj vrednost drugog broja.
3. Saberi prvi broj sa drugim, rezultat je zbir, tj. treći broj.
4. Saopšti zbir.

Sada je potrebno da to saopštimo računaru na njegovom jeziku. Prevodimo svaki korak rešenja u naredbu (ili

naredbe) računarskog jezika. Na ovom kursu računarski jezik je **Java**. Da bi mogli da kažete računaru kako da uradi bilo koji zadatak, potrebno je dobro da znate Java programski jezik. Vaše znanje Jave će vam omogućiti da prevedete programske korake u jezik koji računar može da razume.

Druga stvar o kojoj treba da vodite računa kada pišete Java programe je logika i preciznost. Računar će slediti vaše instrukcije – čak i kada su pogrešne! Prema tome, kako budete napredovali u učenju Jave tako ću vam objasniti potrebu da budete precizni. Kada jednom napišete precizan i logičan Java kôd, računar će veoma brzo i dobro uraditi svoj posao. A on može da uradi veoma zanimljive stvari. Pogledajmo par primera pisanja programerskih zadataka kao niz koraka za ilustraciju nekih stvari koje računar može da uradi.

Šta treba da uradite ako vaš šef zatraži od vas prosečnu ocenu sa testa koji je radilo 352 učenika vaše škole? Koraci koje treba da uradite su sledeći:

1. Odredite ocenu svakog učenika.
2. Saberete svih 352 ocene da bi dobili sumu.
3. Podelite sumu sa 352 da biste dobili prosečnu ocenu.

4. Saopštite šefu prosečnu ocenu.

Nije tako teško, zar ne? Uočite da se drugi korak može dalje razdeliti u manje korake. Da biste sabrali 352 ocene treba da:

1. Počnete sa prvom ocenom.
2. Dodate drugu ocenu, zatim treću, pa četvrtu itd.
3. Stanete kada su sve ocene dodate.

U ovim koracima računar bi trebalo da radi isti posao (dodavanje broja) 352 puta. Računari su veoma dobri kod poslova koji se ponavljaju – videćete kasnije da se ovi poslovi ponavljanja nazivaju petlje (engl. **looping**). Kasnije ćete pisati programski kôd za ovaj primer.

Računari su veoma dobri kod igranja igara sa vama (zato su video igre toliko popularne). Da li ste ikada igrali neku akcionu igricu tipa "War?" Vi i drugi igrač uzimate karte iz standardnog špila. Onak koji ima „jaču" kartu nosi kartu drugog igrača. Zatim svaki uzima drugu kartu i nastavlja se proces upoređivanja sve dok u špilu ima karata. Ko sakupi više karata kada se igra završi je pobednik. Igranje ove igre zahteva sledeće korake:

1. Promešati špil karata.
2. Dati kartu prvom igraču.
3. Dati kartu drugom igraču.
4. Odrediti čija karta je jača i proglasiti pobednika.
5. Ponavljati proces davanja (deljenja) karata igračima sve dok ima karti u špilu.

Stvari su sada nešto komplikovanije ali računar je sposoban da obavi taj zadatak. Prvi korak zahteva da računar promeša špil karata. Kako da saopštite računaru da to uradi? Dobro, pre nego što se kurs završi saznaćete i to. Za sada, bitno je da znate da je u pitanju nekoliko programskih koraka. Smestićemo Java program za takav specijalan zadatak u sopstvenu oblast koja se naziva **metod** (slično glavnom metodu sa kojim smo se susreli u primeru Welcome). Ovo omogućava da nešto lakše pratimo program a takođe i da koristimo ovaj kôd u drugim programima, što je prednost **objektno orijentisanog programiranja**. Uočite da korak 4 zahteva od računara da donese **odluku** – utvrđivanje koja je karta jača. Računari su veoma dobri kod donošenja odluka. Na kraju, korak 5 traži ponavljanje deljenja karata sve dok ih ima u špilu – drugi primer **petlje**. Ovaj program ćete, takođe, praviti kasnije.

Ako svi ovi koncepti nisu u ovom trenutku jasni, to je u redu. Oni će postati jasniji kako kurs bude odmicao. Želeo sam samo da vam dam neke ideje koje možete da realizujete kao Java programe.

Da se ukratko podsetimo, za svaki Java program koji pravite, najbolje je prvo da napišete niz logičkih koraka koje želite da računar sledi kod obavljanja poslova koji su potrebni vašem programu. Zatim da pretvorite te korake u Java jezik što će dati vaš Java program – to je zaista prosto. Sledeći čas će početi sa osnovnim elementima Jave, a zatim ćete učiti sve više i više o Javi. Počećemo polako ali ćete na kraju kursa sasvim solidno „govoriti” Javu.

Java za mlade programere (7)

Posle dužeg vremena stigli smo i do srži Java projekta - Java jezika. U ovoj lekciji biće reči o promenljivama (nazivu, tipu, deklarisanju), dodeljivanjima, aritmetičkim operacijama i tehnikama za rad sa konkretnim tipom promenljive koja se naziva string.

Promenljive

Svi računari rade sa nekom vrstom informacije. Brojevi, tekst, datumi i slike su uobičajeni tipovi informacija sa kojima oni rade. Računarskim programima je potreban prostor da smeste informacije dok sa njima rade. Šta ako nam je potrebno da znamo koliko košta deset banana ako je cena jedne 25 dinara? Nama je potreban prostor da smestimo broj banana, cenu banane i rezultat množenja ta dva broja. Da bismo smestili takve informacije, koristimo nešto što se naziva **promenljiva**. Ona se naziva promenljiva zato što informacija tu smeštena može da se menja za vreme izvršavanja programa. Promenljive su primarni metod za premeštanje informacija u Java projektu. Prilikom korišćenja promenljivih određena pravila se moraju poštovati.

Nazivi promenljivih

Svakoj promenljivoj koju koristite u svom projektu morate dodeliti naziv (ime). Pravila za imenovanje promenljivih su:

- Mogu se koristiti samo slova, brojevi i znak podvlake (_) (mada se on vrlo retko koristi).
- Prvi znak mora biti slovo.
- Ne mogu se koristiti službene (rezervisane) reči Java programskog jezika (na primer, ne možete dodeliti promenljivoj naziv **println** ili **System**).

Po konvenciji, nazivi promenljivih počinju sa malim slovom. Ako se naziv promenljive sastoji od više reči, tada su reči spojene a svaka reč posle prve počinje sa velikim slovom.

Najvažnije pravilo je da se koriste nazivi promenljivih koji nešto znače. Trebalo bi da možete da identifikujete informaciju smeštenu u promenljivoj po nazivu promenljive. Za naš primer sa bananama, dobri nazivi bi bili:

Količina	**Naziv promenljive**
Jedinična cena banane	bananaCost
Broj poručenih banana	numberBananas
Cena svih banana	totalBananaCost

Uočite konvenciju, tj. naziv svake promenljive počinje sa malim slovom a velikim slovom počinje svaka potonja reč u nazivu promenljive.

Tipovi promenljivih

Potrebno je da znamo **tip** informacije koju čuva (drži) svaka promenljiva. Da li sadrži broj? Da li broj ima decimalnu tačku (zarez)? Da li sadrži tekstualnu informaciju? Pogledajmo neke tipove promenljivih.

Prva promenljiva je tipa **int** (integer). Ovaj tip promenljive se koristi za predstavljanje celih, ne-decimalnih, brojeva. Primeri tih brojeva su:

1 -20 4000

Uočite da pišemo 4,000 (anglosaksonski oblik zapisivanja brojeva) kao 4000 u Javi – ne možemo koristiti zareze kod velikih brojeva. U našem primeru sa bananama, **numberBananas** treba da bude promenljiva tipa **int**.

Šta ako promenljiva koju želimo da koristimo treba da čuva broj koji ima decimalnu tačku (tj. decimalni zarez). Na ovom kursu, takve promenljive će biti tipa **double**. U tehničkom žargonu kažemo da su takve promenljive dvostruke preciznosti (double-precision), brojevi sa pokretnim zarezom (floating point numbers – decimalna tačka je stvar koja se "pomera"). Sve što vi treba da znate o promenljivama tipa

double je da one drže brojeve sa decimalnom tačkom. Primeri tih brojeva su:

-1.25 3.14159 22.7

U našem primeru sa bananama, promenljive **bananaCost** i **totalBananaCost** treba da budu promenljive tipa **double**.

U Java programiranju se koristi i **bulov** (**boolean)** tip promenljive. Taj tip je dobio naziv po poznatom matematičaru (Boole). Može da ima jednu od dve vrednosti: **true** (tačno) ili **false** (netačno). Videćemo da su promenljive tog tipa osnova računarevih sposobnosti za donošenje odluka. Ako želite da znate da li je banana trula, možete imenovati bulovu promenljivu **isBananaRotten**. Ako je to tačno, banana je zaista trula.

Sledeći "tip" promenljive koji koristimo nije uopšte tip. Preciznije, to je Java klasa – **String** klasa (činjenica je da počinje sa velikim slovom). String promenljiva je upravo to – ona čuva string (nisku, listu) različitih znakova. String može biti ima, string brojeva, rečenica, pasus, bilo koji znaci uopštel. Često, string neće sadržati bilo kakav znak (prazan string). Mi ćemo mnogo koristiti stringove u Javi, zato treba dobro da ih znate. Stringovi su uvek omeđeni (ograničeni) navodnicima. Primeri za string su:

“I am a Java programmer” “012345” “Title Author”

Deklarisanje promenljivih

Kada imenujete promenljivu i odredite kog tipa želite da bude, morate te informacije proslediti Java projektu. Potrebno je da **deklarišemo** svoje promenljive. Java naredba koja se koristi za deklarisanje promenljive imenovane **variableName** a tipa **type** je:

type variableName;

Nemojte zaboraviti tačku-zarez (;) – svaka Java naredba se završava sa time. Potrebna nam je naredba deklaracije kao ova za svaku promenljivu u našem projektu. To izgleda mnogo zametno, ali vredi potruditi se. Odgovarajuće deklarisanje promenljivih olakšava programiranje, smanjuje mogućnost pojavljivanja grešaka i olakšava menjanje programa.

Gde treba da se stave deklaracije promenljivih? U prvih nekolikoa projekata, pisaćemo samo kôd unutar glavnog metoda (**main method)** Java projekta. Stoga će deklaracije promenljivih biti smeštene posle reda u kome je definisan glavni metod. Usled toga promenljive imaju **lokalni karakter** (**local scope)**, što znači da su na raspolaganju samo metodu u kome su definisane. Ovaj nivo oblasti definisanosti je sasvim

zadovoljavajući za naše početne projekte. Primeri deklaracije promenljivih:

```
int numberBananas;
double bananaCost;
double totalBananaCost;
boolean isBananaRotten;
String myBananaDescription;
```

Uočite da su deklaracije **int**, **double** i **boolean** napisane malim slovima, a **String** velikim slovom.

Java dozvoljava da deklarišete više promenljivih istog tipa u jednom redu razdvajajući nazive sa zarezom. Na primer, možemo kombinovati dve od prethodnih deklaracija (za double promenljive) u:

```
double bananaCost, totalBananaCost;
```

U glavnom metodu Java programa, ove deklaracije promenljivih treba da se pojave na vrhu:

```
public static void main(String[] args)
{
  int numberBananas;
  double bananaCost, totalBananaCost;
  boolean isBananaRotten;
  String myBananaDescription;

      [Ostatak glavnog metoda]
```

}

Sada ćemo razmotriti kako se dodeljuju promenljivama vrednosti.

Naredba dodeljivanja

Najprostija i najviše korišćena naredba u Javi je naredba dodeljivanja (**assignment** statement). Ta naredba ima oblik:

variableName = variableValue;

Uočite da samo jedna promenljiva može biti na levoj strani operatora dodeljivanja (**=**). Evo nekoliko prostih primera dodeljivanja korišćenjem naše "banana" promenljive:

numberOfBananas = 22;
bananaCost = 0.27;
isBananaRotten = false;
myBananaDescription = "Yes, we have no bananas!";

Stvarne vrednosti ovde dodeljene promenljivama nazivaju se **literali**, pošto one literarno pokazuju svoje vrednosti.

Operator dodeljivanja možete prepoznati kao znak jednako koji koristite u aritmetici, ali se on ne naziva tako u računarskom programiranju. Zašto? U stvari, desna strana (**variableValue** u našem primeru) operatora dodeljivanja nije ograničena na literale. Svaki ispravan Java izraz, sa bilo kojim

brojem promenljivih ili drugih vrednosti, može biti sa desne strana operatora. U tom slučaju, Java prvo izračunava **variableValue**, a zatim dodeljuje rezultat **variableName**. Ovo je veoma važan koncept programiranja i treba da ga zapamtite – "izračunajte desnu stranu, a zatim dodelite levoj strani". Takođe je važno da imate na umu da ako **tip** od **variableValue** ne odgovara **tipu** od **variableName**, Java će konvertovati (ako je to moguće) variableValue u odgovarajući tip. Na primer, ako je variableName tipa **int** (celobrojna) a variableValue je izračunato **25.6**, variableName će imati vrednost **25** (uzima se celobrojna vrednost). Pogledajmo sada neke operatore koji mogu da pomognu kod izračunavanja Java izraza.

Aritmetički operatori

Jedna od stvari u kojoj su računarski programi jako dobri je aritmetika. Oni mogu veoma brzo da sabiraju, oduzimaju, množe i dele brojeve. Mi treba da naučimo kako da napravimo da naš Java projekt radi aritmetiku. Postoji pet **aritmetičkih operatora** koje ćemo koristiti iz Java jezika.

Sabiranje se obavlja pomoću znaka plus (**+**) a **oduzimanje** se obavlja pomoću znaka minus (**-**). Evo nekoliko prostih primera:

Operacija	**Primer**	**Rezultat**
Sabiranje	7 + 2	9

Sabiranje	3 + 8	11
Oduzimanje	6 - 4	2
Oduzimanje	11 - 7	4

Množenje se obavlja pomoću znaka zvezdice (*****) a **deljenje** se obavlja pomoću znaka u vidu kose crte (**/**). Evo primera:

Operacija	**Primer**	**Rezultat**
Množenje	8 * 4	32
Množenje	2 * 12	24
Deljenje	12 / 2	6
Deljenje	42 / 6	7

Siguran sam da ste radili sabiranje, oduzimanje, množenje i deljenje i pre nego što ste razumeli kako svaka od tih operacija radi.

Ovde ćemo koristiti još jedan operator koji se naziva operator ostatka (**%**). Taj operator daje ostatak koji se dobija prilikom deljenja dva cela broja. Možda vam to sada neće biti jasno, ali taj operator se mnogo koristi u računarskom programiranju. Evo primera:

Primer	**Rezultat deljenja**	**Ostatak - rezultat**
7 % 4	1 ostatak 3	3
14 % 3	4 ostatak 2	2

25 % 5 5 ostatak 0 0

Proučite ove primere kako biste razumeli kako operator ostatka radi u Javi.

Šta se dešava ako naredba dodeljivanja ima više od jednog aritmetičkog operatora? Da li to nešto menja? Pogledajmo primer:

7 + 3 * 4

Koji je rezultat? Zavisi kako računate. Ako krenete sa leva na desno i prvo saberete 7 i 3, zatim pomnožite sa 4, rezultat je 40. Ako prvo pomnožite 3 i 4, zatim saberete 7, rezultat je 19. Konfuzija? Međutim, Java sprečava mogućnost pojave konfuzije svojim pravilima prvenstva (prioriteta, engl. **precedence**). To znači da postoji određeni redosled po kome se aritmetički operatori primenjuju. Taj redosled je sledeći:

1. Množenje (*) i deljenje (/)
2. Ostatak (%)
3. Sabiranje (+) i oduzimanje (-)

Prema tome, kod naredbe dodeljivanja prvo se urade sva množenja i deljenja, zatim operacija ostatka i na kraju, sabiranja i oduzimanja. U našem primeru (7 + 3 * 4),

množenje će se uraditi pre sabiranja tako da će rezultat koji daje Java biti 19.

Ako dva operatora imaju isti nivo prioriteta, na primer, množenje i deljenje, operacije se u naredbi dodeljivanja obavljaju sa leva na desno. Primer:

```
24 / 2 * 3
```

Prvo se obavlja deljenje (24 / 2) koje daje 12, zatim množenje (12 * 3), tako da je rezultat 36. A šta ako želimo da uradimo množenje pre deljenja – da li je to moguće? Da - pomoću Java **operatora za grupisanje** - zagrade **()**. Pomoću zagrada u naredbi dodeljivanja dajemo prioritet operacijama u zagradama i one se prve obavljaju. Prema tome, ako napišemo naš primer kao:

```
24 / (2 * 3)
```

množenje (2 * 3) će biti prvo obavljeno, što daje 6, zatim deljenje (24 / 6), koje daje željeni rezultat, 4. Možete koristiti zagrada koliko hoćete, ali one se uvek moraju koristiti u paru – svaka leva zagrada mora imati svoju desnu. Ako ugnježdavate zagrade, tj. imate jedan skup unutar drugog, izračunavanje počinje iz najdubljeg skupa i ide ka spolja. Evo primera:

```
((2 + 4) * 6) + 7
```

Prvo se radi sabiranje 2 i 4, što daje 6, koje se množi sa 6, i rezultat je 36. Taj rezultat se zatim sabira sa 7, tako da je konačni rezultat 43. Možete koristiti zagrade čak i ako one ne menjaju prioritet. Često, one se koriste da bi izraz bio razumljiviji.

Evo nekoliko primera Java naredbi dodeljivanja sa aritmetičkim operatorima:

```
totalBananaCost = numberBananas * bananaCost;
numberOfWeeks = numberOfDays / 7;
averageScore = (score1 + score2 + score3) / 3.0;
```

Ovde treba da uočite nekoliko stvari. Prvo, uočite da zagrade kod **averageScore** izračunavanja primoravaju Javu da sabere tri rezultata pre deljenja sa 3. Takođe, uočite upotrebu "belina", razmaka koji odvajaju operatore od promenljivih. To je uobičajena praksa u Javi koja pomaže da kôd bude čitljiviji. Na ovom kursu imaćete prilike da se susretnete sa mnoštvom primera naredbi dodeljivanja.

Java za mlade programere (8)

Nadovezivanje stringova

Možemo da primenimo aritmetičke operatore na numeričke promenljive (tip **int** i tip **double**). Takođe, može se operisati i sa string promenljivama. Često u Java projektima uzimate string promenljivu sa jednog mesta i dodajete je na kraj drugog stringa. Ovo se naziva **nadovezivanje** (engl. concatenation) **stringova**. Operator nadovezivanja je znak plus (**+**) i lako se koristi. Na primer:

newString = “Java for Kids ” + “is Fun!”;

Posle ove naredbe, string promenljiva **newString** će imati vrednost “Java for Kids is Fun!”.

Uočite da je operator nadovezivanja identičan sa operatorom sabiranja. Potrebno je da uvek obezbedimo da ne dođe do konfuzije kada koristimo oba operatora. String promenljive čine veliki deo Jave. Kako se budete razvijali kao programeri biće potrebno da postanete „bliski” sa stringovima i operacijama sa njima.

Komentari

Poželjno je da uvek sledite odgovarajuća pravila programiranja kada pišete svoj Java kôd. Jedno od tih pravila je da na odgovarajući način komentarišete svoj kôd. Možete da stavite ne-izvršne naredbe (koje računar ignoriše) u svoj kôd koje objašnjavaju šta vaš program radi. Ovi **komentari** mogu biti od pomoći u razumevanju vašeg koda. Takođe, oni olakšavaju buduće promene vašeg koda.

Da biste stavili komentar u svoj kôd, koristite simbol za komentar, dve kose crte (**//**). Sve što je napisano posle simbola za komentar računar će ignorisati. Možete imati kao komentar poseban ceo red Java koda, kao u ovom slučaju:

```
// Postavi broj banana
numberBananas = 14;
```

Ili, možete staviti komentar u isti red sa naredbom dodeljivanja:

```
numberBananas = 14; // Postavi broj banana
```

Takođe, možete imati komentar koji zahvata više redova. Počnite komentar sa simbolom (**/***) i završite ga sa simbolom (***/**):

```
/*
  Ovo je veoma dug komentar
  Zauzima dva cela reda!!
*/
```

Vi, kao programer, treba sami da odlučite koliko želite da komentarišete svoj kôd. Pokušaću u projektima koji se budu ovde razmatrali da obezbedim adekvatne komentare.

Programski izlaz

Skoro da ste spremni da napravite svoj prvi Java program. Ali, potrebna je još jedna stvar. Upoznati smo sa kao se imenuju i deklarišu promenljive i kako se radi matematika sa njima, ali kada dobijemo rezultate, kako ti rezultati mogu da budu prikazani? Na ovom „času" koristićemo metod koji ste već videli u našem malom programu Welcome, Java metod **println** (da vas podsetim, izgovara se „print lajn"). Ono što taj metod radi je štampanje stringa rezultata u jednom redu:

```
System.out.println(stringValue);
```

U ovom izrazu, **stringValue** mogla bi biti promenljiva tipa String koja se negde izračunava (možda pomoću operatora nadovezivanja) ili literal (stvarna vrednost). U primeru Welcome, mi smo koristili literal:

System.out.println(“Welcome to Java for Kids!”);

I videli da se kao izlaz na ekranu pojavilo **Welcome to Java for Kids!**

Šta ako želite da izlaz bude numerička informacija? To je sasvim lako. Metod println će automatski konvertovati numeričku vrednost u string za potrebe izlaza. Na primer, pogledajte ovaj mali segment koda:

```
numberBananas = 45;
System.out.println(numberBananas);
```

Ako pokrenete izvršavanje ovog segmenta, na ekranu će se kao izlaz pojaviti **45**.

Takođe, možete kombinovati tekstualnu informaciju sa numeričkom informacijom pomoću operatora nadovezivanja. Na primer:

```
numberBananas = 45;
System.out.println(“Number of Bananas is ” +
numberBananas);
```

Štampaće na ekranu kao izlaz **Number of Bananas is 45**. Numerički podatak (**numberOfBananas**) konvertovan je u string pre nego što je nadovezan na tekstualni podatak.

Prema tome, prilično je lako kao izlaz dobiti tekstualnu i numeričku informaciju. Budite spremni na to da se ponekad mogu pojaviti i izvesni problemi. Da vas podsetim na činjenicu da je operator nadovezivanja identičan sa aritmetičkim operatororm sabiranja. Pogledajte sledeći segment koda:

```
numberBananas = 32;
numberApples = 22;
System.out.println("Pieces of fruit " +
numberBananas + numberApples);
```

Možda mislite da sa ovom naredbom štampate ukupan broj voća (numberBananas + numberApples = 54). Međutim, ako izvršite ovaj kôd, dobićete **Pieces of fruit 3222**. Ono što se desilo je da je Java konvertovala oba numerička podatka u string pre nego što je obavljeno sabiranje. Tada znak plus koji ih razdvaja ima ulogu operatora nadovezivanja i stoga se dobija 3222. Da biste štampali sumu, potrebno je da pomoću zagrada izvedete prisilno numeričko sabiranje:

```
numberBananas = 32;
```

```
numberApples = 22;
System.out.println(“Pieces of fruit ” +
(numberBananas + numberApples));
```

U ovom slučaju, dve numeričke vrednosti se sumiraju pre konvertovanja u string i dobija se željeni izlaz **Pieces of fruit 54**. Prema tome, vidimo da metod println nudi pogodan način za dobijanje izlaza i sa numeričkom i sa tekstualnom informacijom, ali mora korektno da se koristi.

Uočite još jednu stvar u ovom primeru. Zadnji red koda izgleda kao da ima dužinu za dva reda! Ovo je se pojavilo zbog osobine programa za obradu teksta da prelama red. U pravom Java programu taj red će se pojaviti kako je i ukucan, kao jedan red. Imajte to na umu dok čitate tekstove ovog serrijala.

Java za mlade programere (9)
Projekt – Sendvič žurka

Vaš razred je odlučio da napravi žurku. Napravljena su dva veoma velika „podmornica" sendviča i vaš zadatak je da definišete koliko svaki gost (student) može da pojede. Svakako, to možete da uradite pomoću kalkulatora, ali ovoga puta ćemo koristiti Javu. Projekat ćemo snimiti pod nazivom **FirstProject** u folderu ovog kursa (**\JavaKids\JK Code**).

Dizajniranje projekta

Predpostavimo da znate dužinu svakog „podmornica" sendviča. Da bismo olakšali sečenje, reći ćemo da će svaki gost dobiti celi broj inča (ili centimetara) od sendviča (bez decimala). Uz ovaj podatak, možete izračunati koliko gostiju se može nahraniti od jednog sendviča. Ako je ukupni broj veći od broja gostiju, tj. đaka u razredu, svi će jesti i sve je u redu. A ako nije tako, moraćete da uradite podešavanje. Koraci u programu bi trebalo da budu sledeći:

1. Postaviti vrednost za broj inča sendviča koje gost može pojesti.
2. Odrediti dužinu oba sendviča.
3. Odrediti koliko gostiju može da se nahrani od svakog sendviča.

4. Povećati ili smanjiti broj inča sve dok se ne postigne da ceo razred, tj. svi gosti mogu da jedu.

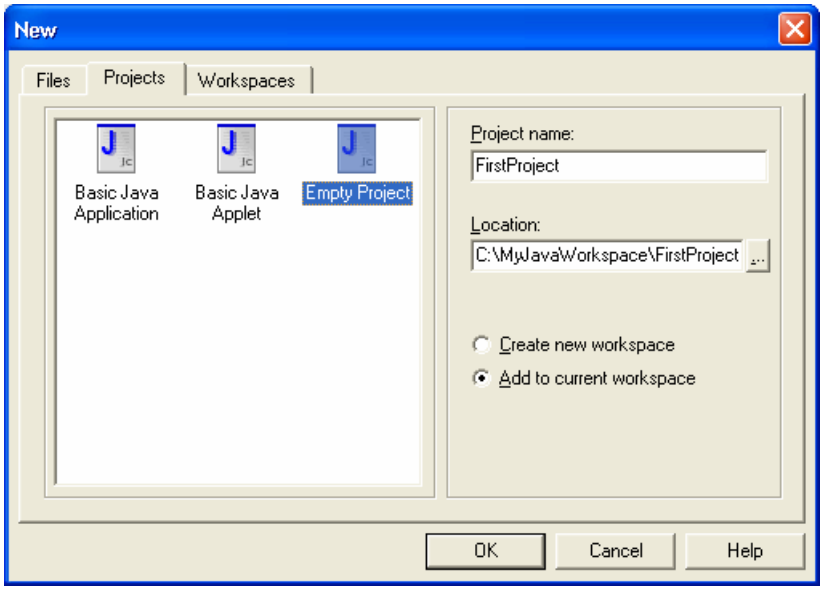
Prevedimo svaki od ovih koraka u Java kôd i postepeno izgradimo naš projekt. Pošto je ovo vaš prvi projekt, razmotrićemo svaki korak (kreiranje novog projekta i dodavanje datoteke (fajla)) i pisaćemo i diskutovati kôd.

Razvoj projekta

Pokrenite **JCreator** i proverite da li je vaš radni prostor (**MyJavaWorkspace**) otvoren. **WelcomeProject** bi trebalo da je tu. Ako nema vašeg radnog prostora, kliknite mišem na **File**, zatim **Open Workspace** da bist ga otvorili. Kreirajte novi projekt:

- Kliknite mišem na **File**
- Izaberite **New**
- Kliknite mišem na jezičak **Projects**
- Selektujte **Empty Project**
- Dodelite naziv: **FirstProject**

Prozor vašeg Jcreatora trebalo bi da ima sledeći izgled:

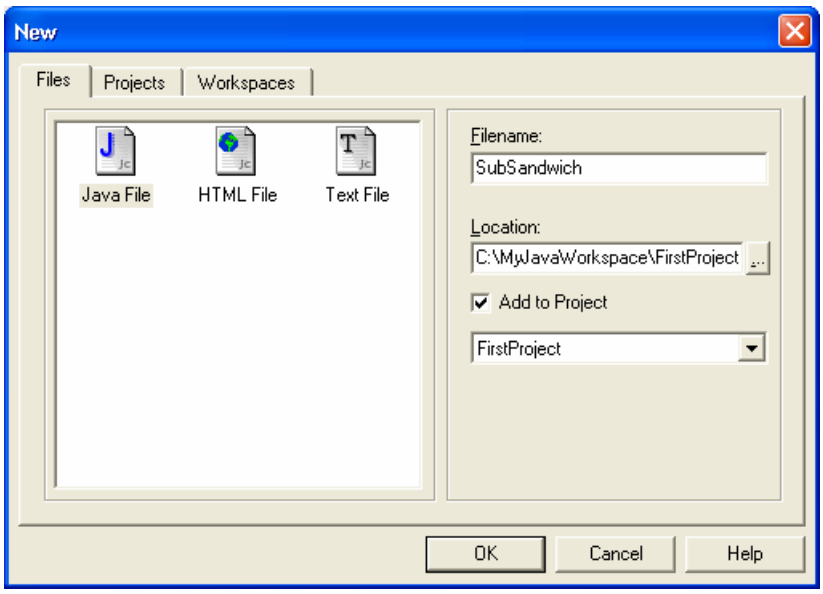


Kliknite na dugme **OK**. Uradite desni klik nad projektom **FirstProject** u prikazu radnog prostora i selektujte **Sets as Active Project**.

Dodajte fajl projektu:

- Kliknite mišem na **File**
- Izaberite **New**
- Kliknite na jezičak **Files**
- Selektujte **Java File**
- Dodelite naziv: **SubSandwich**

Prozor vašeg Jcreatora trebalo bi da ima sledeći izgled:



Kliknite na **OK** i prazan fajl bi trebalo da se pojavi u oblasti prikaza fajla. To je mesto gde ćete upisivati svoj kôd.

Prvo, upišite sledeće zaglavlje kao višeredni komentar:

```
/*
 *  Sub Sandwich Project
 *  Java for Kids
 */
```

Sada, upišite u red definiciju klase i levu vitičastu zagradu za otvaranje klase (**{**). Da vas podsetim, zagrade se koriste za definisanje blokova sa kodom:

```
public class SubSandwich
{
```

Sada sledi definicija glavnog metoda i njegova pripadna zagrada:

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

Uočite, kako upisujete nove blokove koda, JCreator uvlači odgovarajuće blokove. Proverite da pišete svaki red tačno kako je prikazano. Za računarsko programiranje vezano je mnogo kucanja – korisno je da pohađate kurs daktilografije kako biste usavršili kucačku veštinu. Dobra tehnika kucanja znači brže i bez grešaka kucanje, a time i vađi Java programi neće imati kucačke greške.

Spremni ste da počnete da pišete Java kôd. Koristićemo u ovom programu pet promenljivih: jednu za to koliko svaki student može da pojede, dve za dužine sendviča, i dve za to

koliko gostiju može da se nahrani od jednog sendviča. Sve promenljive će biti celobrojnog (integer) tipa. Upišite sada njihove deklaracije (svaku deklaraciju završite sa znakom tačka-zarez):

int inchesPerStudent;
int lengthSandwich1, lengthSandwich2;
int students1, students2;

Postavite vrednosti za neke promenljive (takođe, stavite komentar za ono što radite):

```
// set values
inchesPerStudent = 5;
lengthSandwich1 = 114;
lengthSandwich2 = 93;
```

Mi smo postavili navedene vrednosti, vi možete postaviti neke druge ako želite. Uočite da smo pretpostavili da svaki gost može da pojede 5 inča od sendviča.

Zatim, pomoću običnog deljenja računamo koliko gostiju može da se hrani iz svakog sendviča:

```
// odredite koliko gostiju može da se hrani iz svakog ---
//sendviča
students1 = lengthSandwich1 / inchesPerStudent;
students2 = lengthSandwich2 / inchesPerStudent;
```

Uočite da će **students1** i **students2** biti (kako smo želeli) celi (integer) brojevi. Prikažite rezultate pomoću metoda **println**:

```
// ispiši rezulatet
System.out.println("Letting each student eat " +
inchesPerStudent + " inches");
System.out.println((students1 + students2) + " students
can eat these two sandwiches!");
```

Uočite kako nadovezivanje stringova funkcioniše. Takođe, uočite da sumiramo broj gostiju pre ispisivanja. Na kraju,

završite program sa dve desne zagrade (}), jednom za zatvaranje metoda i drugom za zatvaranje klase.

Konačni kôd u JCreatoru treba da ima sledeći oblik:

```
/*
 *  Sub Sandwich Project
 *  Java for Kids
 */
 public class SubSandwich
 {
   public static void main(String[] args)
   {
     int inchesPerStudent;
     int lengthSandwich1, lengthSandwich2;
     int students1, students2;

     // set values
     inchesPerStudent = 5;
     lengthSandwich1 = 114;
     lengthSandwich2 = 93;

     // determine how many students can eat each
sandwich
     students1 = lengthSandwich1 / inchesPerStudent;
     students2 = lengthSandwich2 / inchesPerStudent;

     // print results
     System.out.println("Letting each student eat " +
inChesPerStudent + " inches");
     System.out.println((students1 + students2) + "
students can eat these two sandwiches!");

   }
 }
```

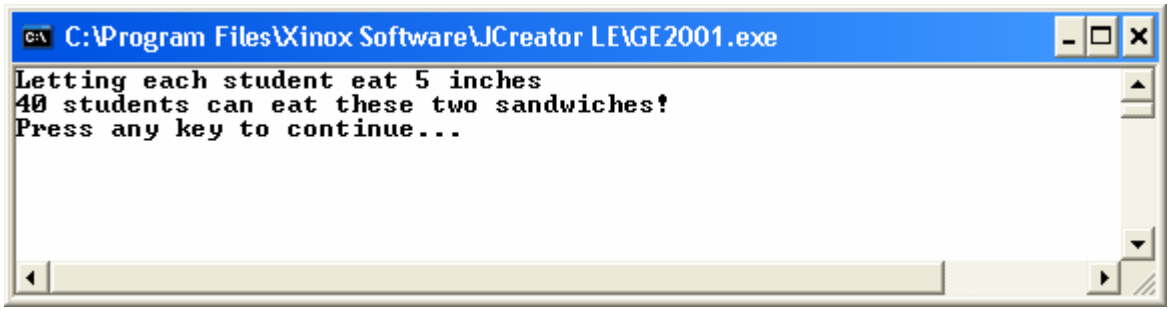
Dva puta proverite da ste svaki red upisali kako treba.

Java za mlade programere (10)

Pokrenite projekt

Snimite svoj projekt (kliknite mišem na dugme **Save All** na paleti alatki). Kompajlirajte svoj kôd tako što ćete pritisnuti taster <**F7**>, ili izaberite opciju **Build** iz glavnog menija i kliknite na stavku **Compile Project**. Treba da dobijete poruku **Process completed**. Ako se to ne desi, bilo koja poruka o grešci koja se pojavi verovatno će ukazivati na grešku u kucanju koda. Još jednom proverite svoj kôd – tačka-zarez postoje na odgovarajućim mestima kao i znaci navoda, nema neslaganja kod imenovanja promenljivih i funkcija.

Kada je kôd kompajliran, pokrenite njegovo izvršavanje pritiskom na taster <**F5**>, ili izaberite u glavnom meniju **Build**, zatim **Execute Project**. Na ekranu bi trebalo da se pojavi sadržaj prikazan na sledećoj slici:



Čestitamo – napisali ste svoj prvi Java program!!

Probajte još neke kombinacije

Kod prethodnog izvršavanja videli smo da 40 učenika (studenata) može da jede. Šta ako treba da nahranite više ili manje osoba? Podesite promenljivu **inchesPerStudent** i odredite broj osoba koje mogu da se nahrane za tu novu vrednost. Posle svakog podešavanja (izmene), moraćete da ponovo kompajlirate i izvršavate program.
Pretpostavimo da sendviči prilično koštaju po inču veličine. Izmenimo program tako da on izračunava i cenu sendviča. Odredite sa kolikom sumom novca svaki učenik treba da učestvuje da bi pokrio troškove svog jela. Možete koristiti **double** tip promenljive da bi mogli da koristite decimale. Pokušajte sami da to uradite!

Pošto smatramo da svaki učenik treba da pojede ceo broj sendviča, može da se desi da kod računanja imamo ostatke za svaki sendvič. Kako bi mogli da izračunamo tu količinu? To bi bila mala primena operatora ostatka koga smo ranije upoznali. Treba samo malo kôd prepraviti.
Prvo, definišimo promenljivu koja izračunava veličinu ostatka:

int inchesLeftOver;

Sada je na redu kôd koji izračunava vrednost:

```
// izracunava leftovers
inchesLeftOver = lengthSandwich1 %
inchesPerStudent + lengthSandwich2 %
inchesPerStudent;
System.out.println("Postoji " + inchesLeftOver + "
inca ostatka.");
```

Dodajte ovaj kôd svom projektu, ponovo ga kompajlirajte i izvršite. Da li vidite ukupni ostatak u vrednosti od 7 inča? Da li znate zašto kod računanja **inchesLeftOver** nismo sabrali obe dužine sendviča pre primene operatora ostatka?

Zaključak

Još jednom čestitamo, završili ste svoj prvi Java projekt. Naučili ste dosta o Java naredbama i dodeljivanjima, pa i nešto malo o aritmetici. Trebalo bi da ste sada spremni za novi poduhvat, tj. projekat. Sledeći put ćemo učiti još novih stvari i pozabaviti se detaljnije Java projektima.

Java za mlade programere (11)

Dizajniranje projekta

Spremni smo da započnemo prilično detaljan projekat u Javi. Ovde će biti predloženi neki projekti a vi ćete verovatni imati i svoje ideje koje ćete možda realizovati. Međutim, pre ne go što počnemo projekat, nije loše da malo razmislimo o onome što ćemo da radimo. Dobro razrađena ideja, tj. dobro dizajniran projekat, uštedeće kasnije mnogo vremana a i rezultati će biti bolji.

Dobro dizajniranje projekta nije nešto tako teško, to je više stvar dobre prakse. Osnovna ideja je da se kreira projekat koji se lako koristi, lako razume i koji nema grešaka. Morate priznati, to ima smisla, zar ne? Zato potrošite neko vreme na razmišljanje o onome što želite da vaš projekat radi. Koje su informacije potrebne vašem projektu? Da li računar neke informacije definiše? Utvrdite koje programske korake treba da sledite da biste ostvarili željene zadatke.

Trudite se da Java kôd u vašim metodima bude čitljiv i lako razumljiv. Ovo će biti kasnije od velike koristi kada budete morali nešto da menjate. Poštujte opšte prihvaćena pravila programiranja – ta pravila ćete saznati kako budete dalje učili da programirate u Javi. Pažljivo analaizirajte svoj koôd kako bi

bio bez grešaka. To možda zvuči kao suvišna opaska, ali verujte da većina programa ima greške.

Jednostavnu ideju da program bude upotrebljiv, jasno napisan i bez grešaka, kao i lako izmenljiv nije uvek lako postići. Pažljivo planiranje i planiranje unapred pomaže da se taj cilj ostvari. Ovde ćemo pokušati da vam kod svakog projekta koji ovde razmatramo damo potpun uvid u ono što se dešava. Uvek ćemo pokušati da vam objasnimo zašto nešto radimo.

Druga bitna stvar kod dizajniranja projekta je da uvek treba raditi u etapama, postepeno. Nemojte pokušavati da izgradite ceo projekat odjednom i tek tada da ga testirate. Takav pristup povećava mogućnost pojavljivanja velikog broja grešaka. Zato radite po fazama, pišite pomalo koda, kompajlirajte ga i testirajte kako biste se uverili da radi kako treba. Polako dodajte sve više i više koda. Posle svakog dodavanja koda obavezno sledi testiranje. Nastavite sa tim pristupom sve dok ne kompletirate svoj projekat. Brzo ćete uvideti da ovaj pristup "laganog hoda" kod kreiranja Java projekta olakšava programiranje.

Java – još jedna lekcija

Do sada smo naučili prilično toga o Javi. To je bilo neophodno kako biste naučili osnovne koncepte i mogli da počnete sa pisanjem svog prvog programa. Sada ćemo krenuti dalje, naučićemo kako se inicijalizuju promenljive i nešto o matematičkim funkcijama.

Inicijalizacija promenljive

Ranije smo razmatrali potrebu da se deklariše svaka promenljiva koja se koristi u Java programu. Opšta naredba koja se koristi za deklarisanje promenljive je oblika:

type variableName;

Ovde mi kažemo da se promenljiva **variableName** deklariše da je tipa **type**. Tipovi promenljivih sa kojima ćemo se susretati su **int** (celi brojevi), **double** (decimalni brojevi), **boolean** (vrednosti tačno i netačno) i **String** promenljive. Evo nekoliko primera deklaracije promenljivih:

```
int numberLightBulbsPerPack;
int numberPacks;
double costOfPack;
boolean anyBurnedOut;
String myQuestion;
```

Kada deklarišete promenljivu, njoj se dodeljuje neka lokacija u memoriji računara u kojoj se nalazi neka nepredvidiva vrednost. Često, to je sasvim dovoljno, ukoliko ne zaboravite da promenljivoj dodelite neku korisnu vrednost negde u svom programskom kodu. Postoje slučajevi kada treba da dodelite inicijalnu vrednost promenljivoj kada je deklarišete. To je prilično dobra programerska praksa da radite takvu inicijalizaciju, ako znate vrednost koju promenljiva treba da ima (često to nećete znati). Postoje i situacije kada će Java insistirati da inicijalizujete promenljive. U tim situacijama radi se o jednostavnom proširenju deklaracije:

```
type variableName = variableValue;
```

Naredaba deklaracije u ovom slučaju je i naredba inicijalizacije, kreira se promenljiva pod nazivom **variableName** tipa **type** i

dodeljuje joj se početna (inicijalna) vrednost od **variableValue**. Vodite računa da dodelite vrednost odgovarajućeg tipa. Ne možete dodeliti decimalnu vrednost promenljivoj celobrojnog tipa! Ove naredbe se smeštaju zajedno sa uobičajenim deklaracijama na vrhu metoda da bi se obezbedila lokalna definisanost.

Evo nekoliko primera deklarisanja i inicijalizacije promenljivih u Java programu:

```
int numberLightBulbsPerPack = 8;
int numberPacks = 7;
double costOfPack = 2.45;
boolean anyBurnedOut = false;
String myQuestion = "How many Java programmers does it take to change a light bulb?";
```

I sami vidite kako je lako koristiti takve naredbe. Kao programer, biće potrebno da odlučite kada želite da inicijalizujete promenljive a kada ne.

Matematičke funkcije

U jednom od prethodnih nastavaka, susreli smo se sa aritmetičkim operatorima koji nam omogućuju da obavimo osnovno sabiranje, oduzimanje, množenje i deljenje. Kao i drugi računarski jezici, Java ima mogućnosti da obavlja veoma moćna matematička izračunavanja. Ugrađene Java

matematičke **funkcije** (nazivaju se i **metodi**) vrlo često se koriste u takvim izračunavanjima.

Ne očekujem da ste matematički genije i zato ćemo se pozabaviti samo sa tri matematičke funkcije. Prvo, šta je to uopšte **funkcija**? Funkcija je rutina koja izračunava neku vrednost za vas na osnovu date informacije. Funkcija se koristi u sledećem formatu:

functionValue = functionName(argumentList);

functionName je ime funkcije a **argumentList** je spisak (lista) vrednosti (**argumenata**, razdvojenih zarezima) koje su obezbeđene funkciji kako bi mogla da radi. U ovoj naredbi dodeljivanja, functionName koristi vrednosti iz argumentList da izračuna rezultat I dodeli ga promenljivoj pod nazivom **functionValue**. Moramo se postarati da promenljiva functionValue bude istog tipa kao vrednost koju izračunava funkcija functionName.

Kako da znate koje Java matematičke funkcije postoje, koje tipove informacija obezbeđuju i koje argumente koriste? Potražite informacije u Java referentnim priručnicima i na Sunovom Java websajtu. Kao što sam već pomenuo, razmatraćemo samo tri matematičke funkcije. Metodi koji podržavaju matematičke funkcije implementirani su u Java klasi **Math**. Prema tome, da biste koristili konkretnu funkciju, pišete **Math**, zatim tačku i posle toga ime funkcije.

Prva funkcija koju ćemo razmatrati je funkcija apsolutne vrednosti. U matematici, apsolutna vrednost je pozitivni deo broja. Java funkcija je:

Math.abs(argument)

gde je **argument** broj čiju apsolutnu vrednost izračunavamo. Argument može biti ili **int** ili **double** tipa a vraćena vrednost će biti istog tipa kao argument. Evo nekoliko primera:

Primer	**Rezultat**
Math.abs(7)	7
Math.abs(-11)	11
Math.abs(-3.14)	3.14
Math.abs(72.1)	72.1

Da li vam je ikada trebao **kvadratni koren** nekog broja? Kvadratni koren je broj koji pomnožen samim sobom daje originalni broj. Na primer, kvadratni koren od 4 je 2, pošto je 2 puta 2 četiri. Na vašem kalkulatoru postoji dugme (√) koje će to da uradi za vas. U Javi,kvadratni koren je:

Math.sqrt(argument)

gde je **argument** broj čiji kvadratni koren izračunavamo. Argument mora da bude ne-negativan **double** broj i vraćena vrednost je **double**. Evo nekoliko primera:

Primer	Rezultat
Math.sqrt(4.0)	2.0
Math.abs(36.0)	6.0
Math.abs(72.1)	8.491

Treća funkcija koju ćemo ovde koristiti je stepenovanje. Kod stepenovanja, broj se množi samim sobom određen broj puta. Ako množimo broj samim sobom četiri puta, kažemo da je broj dignut na 4-ti stepen. Java funkcija za stepenovanje je:

Math.pow(argument1, argument2)

Uočite da **pow** (skraćenica za power) funkcija ima dva argumenta. **argument1** je broj koji množimo samim sobom **argument2** puta. Drugim rečima, ova funkcija diže argument1 na argument2 stepen. I argument i vraćena vrednost su brojevi tipa **double**. Evo nekoliko primera:

Primer	**Rezultat**
Math.pow(4.0, 2.0)	16.0
Math.pow(-3.0, 3.0)	-27.0
Math.pow(10.0, 4.0)	10000.0

U ovde datim primerima, argumenti nisu imali decimalne delove. To je namerno urađeno da bi primeri bili jasniji. Vi niste ograničeni na takve vrednosti. Možete, na primer da koristite ovu funkciju da izračunate 7.654 na 3.16!! (Uzgred, odgovor je 620.99)

Za one koje matematika malo više interesuje, rećiću da postoji još Java funkcija. Možete pokušati sami da ih koristite. Postoje trigonometrijske funkcije I inverzne trigonometrijske funkcije, funkcije koje pretvaraju radijane u stepene I obratno, funkcije koje nalaze ekstremne vrednosti, funkcije za zaokruživanje, logaritam itd..

Java za mlade programere (12)

Ulazni metodi programa

U primeru Sub Sandwich Project pravili smo klasu, zatim postavljali vrednosti promenljivih u kodu i izvršili program da bi videli rezultat. Rezultat je ispisao Javin izlazni metod **println**. Ako želimo da koristimo druge vrednosti, potrebno je da promenimo kôd, ponovo kompajliramo program i ponovo izvršimo. To je prilično zametno! Bilo bi lepo ako bi omogućili korisniku da upiše vrednosti dok se program izvršava i prepustimo računaru da uradi računanje na osnovu ulaznih vrednosti. Na taj način nema promene koda i ponovnog kompajliranja da bi se dobio novi rezultat. Nama je potrebno da tako nešto imamo u svojim programima.

nažalost, Java nema opšti metod koji podržava upisivanje ulaznih vrednosti preko tastature. Da li treba da budemo nesrećni zbog toga? Ne. Napisaćemo sopstvenu ulaznu rutinu pomoću Jave. Mada taj kôd nije teško napisati, to je prilično van okvira ovog kursa. Zato će ovde biti dat taj kôd. Vi zatim možete da koristite taj kôd u bilo kom programu koji zahteva upis vrednosti preko tastature. Korišćenje tog koda je slično korišćenju matematičkih funkcija iz prethodne lekcije.

Kôd ulaza je sadržan u klasi pod nazivom **Typeit** (fajl **Typeit.java**). Klasa Typeit sadrži tri metoda:

inInt(prompt) prihvata ceo (**int**) broj
inDouble(prompt) prihvata decimalni (**double**) broj
inString(prompt) prihvata string (**String**) vrednost

Svaki metod ima odzivnik (**prompt**) u obliku string poruke koja se pojavljuje kada se metod izvršava. Kada se odzivnik pojavi, korisnik upisuje zahtevani ulaz i pritiska taster **<Enter>** kako bi računar prihvatio vrednost. Pogledajte primer.

Recimo da imate program gde želite da saznate koliko korisnik ima godina. Dva reda Java koda obavljaju taj posao:

```
int userAge;
userAge = Typeit.inInt("Koliko imate godina?");
```

Uočite da kod korišćenja **inInt** metoda, prvo pišete ime klase **Typeit**, zatim tačku in a kraju ime metoda. Kada se ovaj kôd izvrši, korisnik će videti na ekranu odziv u sledećem obliku:

Koliko imate godina?

Korisnik upisuje celobrojnu vrednost i pritiska taster <**Enter**>. U ovom trenutku ulazna vrednost je dodeljena promenljivoj **userAge**. Metod inInt obezbeđuje da korisnik može da upiše samo celobrojnu vrednost. Korišženje druga dva metoda **inDouble** i **inString** je slično.

Postavlja se pitanje kako da naš program zna da su ovi ulazni metodi za tastaturu na raspolaganju? Vaš program treba da pristupi klasi **Typeit**. Najlakši način da se ovo postigne je da fajl **Typeit.java** bude deo vašeg projekta. Da vas podsetim projekat je sastavljen od fajlova (klasa). Do sada, naši projekti su imali samo po jedan fajl (klasu). Sa te tačke gledišta, većina naših projekata će imati **dva fajla** – naš programski fajl (fajl sa metodom **main** gde pišemo Java kôd) i fajl **Typeit.java** koji sadrži naše ulazne metode.

Kako da ubacimo ovaj drugi fajl? Pogledajmo jedan primer kako bi pokazali kako se dodaje Typeit.java u project i kako se koristi svaki od ulaznih metoda.

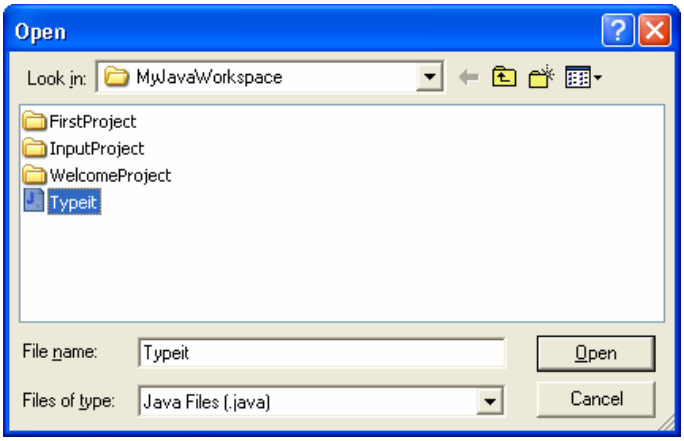
Primer ulaznih metoda

Pokrenite JCreator, otvorite **MyJavaWorkspace** i sledite uobičajene korake za kreiranje novog projekta pod nazivom **InputProject**. Aktivirajte InputProject. Ako je potrebno podsetite se kako se kreira projekt. Dodajte fajl **InputTest** u projekt. To je fajl u kome ćemo pisati naš kôd za testiranje ulaznih rutina. Sada možemo da dodamo drugi fajl (klasu), **Typeit.java**.

Kao prvi korak, napustite na trenutak JCreator i kopirajte **Typeit.java** u svoj radni proctor, tj. folder

(**MyJavaWorkspace**). Time dobijate sopstvenu kopiju klase. Vratite se u JCreator. Sledite navedene korake:

- U pregledu radnog prostora uradite desni klik na **InputProject**
- Izaberite **Add Files**
- Prebacite se u folder **MyJavaWorkspace** i izaberite **Typeit.java**. Pojavljuje se prozor **Open**:



- Kliknite na **Open** i fajl se dodaje vašem projektu.

Sada bi vaš projekt (**InputProject)** trebalo da sadrži dva fajla (**InputTest.java** and **Typeit.java**). Podsetimo se bitnog koraka. Dodavaćete **Typeit.java** skoro svim projektima koje budemo pravili ubuduće na ovom kursu. A sada otvorite fajl **Typeit.java** (dvoklik na fajl u pregledu radnog prostota) i pogledajte šta se u njemu nalazi. Trebalo bi da vidite mnošto Java koda:

```
/*
 * Program: Typeit
 * Class that provides keyboard input
 * Java for Kids
 * KIDware (206) 721-2556
*/

import java.io.*;

public class Typeit
{

  public static void printPrompt(String prompt)
  {
    // print the prompt
     System.out.print(prompt + " ");
     System.out.flush();
  }

  public static void inputFlush()
  {
    int dummy;
    int bAvail;

    // flush the data stream
    try
    {
      while((System.in.available()) != 0)
      {
```

```
    dummy = System.in.read();
   }
  }
  catch(java.io.IOException e)
  {
   System.out.println("Input error");
  }
}


public static String inString(String prompt)
{
  // returns input string
  inputFlush();
  printPrompt(prompt);
  return inString();
}

public static String inString()
{
  int aChar;
  String s = "";
  boolean finished = false;

  while(!finished)
  {
   try
   {
    aChar = System.in.read();
    if (aChar < 0 || (char)aChar == '\n')
    {
     finished = true;
    }
    else if ((char)aChar != '\r')
    {
     s = s + (char) aChar;
    }
   }
   catch(java.io.IOException e)
   {
    System.out.println("Input error");
```

```
      finished = true;
     }
    }
    return s;
  }


  public static int inInt(String prompt)
  {
    // returns input int
    while(true)
    {
     inputFlush();
     printPrompt(prompt);
     try
     {
      return
Integer.valueOf(inString().trim()).intValue();
     }
     catch(NumberFormatException e)
     {
      System.out.println("Invalid input. Not an integer");
     }
    }
  }


  public static double inDouble(String prompt)
  {
    // return input double
    while(true)
    {
     inputFlush();
     printPrompt(prompt);
     try
     {
      return
Double.valueOf(inString().trim()).doubleValue();
     }
     catch(NumberFormatException e)
     {
```

```
      System.out.println("Invalid input. Not a floating
point number");
    }
   }
  }
}
```

Ne očekuje se od vas da u ovom trenutku razumete taj kôd. Ali ipak ćemo ga proanalizirati. Bez obzira na vaš nivo znanja, Ipak ćete neke delove prepoznati.

Bitno je jednu stvar uočiti a to je da glavni metod ne postoji nigde u klasi Typeit. To znači da taj kôd ne može da radi samostalno. On zahteva neku klasu za podršku da bi bio korišćen sa glavnim metodom. Taj kôd će biti napisan u našoj glavnoj klasi, **InputTest.java**. Uradimo to sada. Otvorimo fajl InputTest.java. On treba da je prazan. Koristićemo taj program da testiramo korišćenje ulaznih metoda. Probaćemo sistemom metod po metod, sledeći preporuke koje smo ranije naveli da je najbolje postepeno stvarati kôd.

Upišite uobičajene informacije u zaglavlje, definiciju klase i definiciju glavnog metoda:

```
/*
 *  Input Project
 *  Java for Kids
 */
public class InputTest
{
  public static void main(String[] args)
  {
```

Upišite neki kôd da biste dobili korisnikovu starosnu do bi dodajte dva para zagrada:

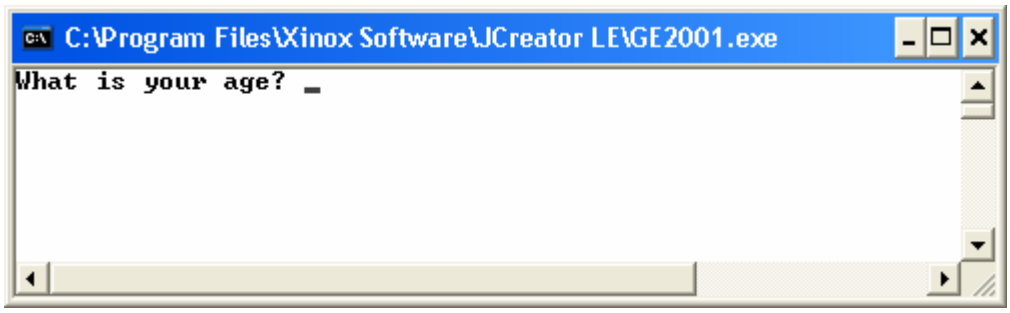
```
int ageUser;
ageUser = Typeit.inInt("What is your age?");
System.out.println("You typed " + ageUser);
```

Završeni kôd u JCreatoru bi trebalo da ima sledeći oblik:

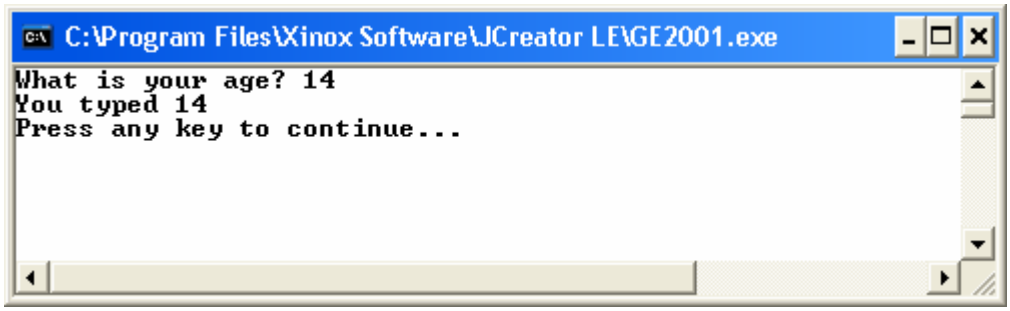
```
/*
 *  Input Project
 *  Java for Kids
 */
public class InputTest
{
  public static void main(String[] args)
  {
    int ageUser;
    ageUser = Typeit.inInt("What is your age?");
  }
}
```

Kompajlirajte projekt (pritisnite <**F7**>). Ako se program ne kompajlira, proverite da ste uključili klasu Typeit id a ste kôd upisali tačno kako je pokazano.

Izvršite projekt (pritisnite <**F5**>). Treba da vidite sledeće:



Uočite kako se prompt (komandni odzivnik) pojavljuje. Upišite vrednost i pitisnite <**Enter**>. Treba da vidite:



Rutina **inInt** radi!!

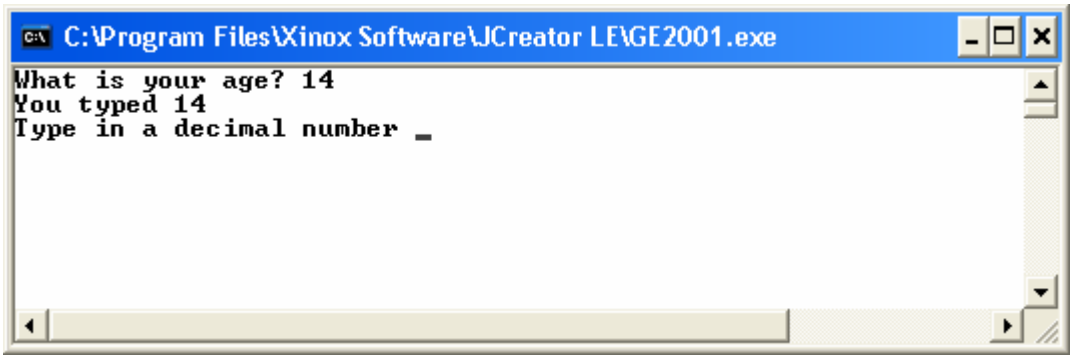
Testirajmo sada metod **inDouble**. Dodajmo deklaraciju nove promenljive:

double myDouble;

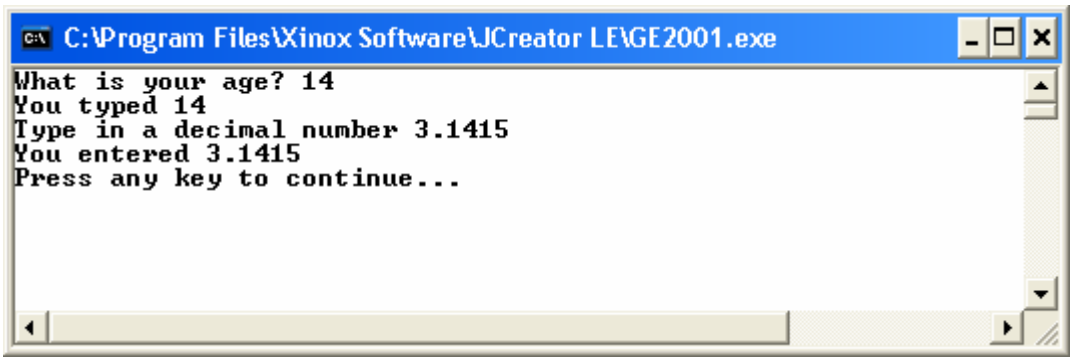
Pošto kôd pita za korisnikovu starosnu dob, dodajte sledeća dva reda:

```
myDouble = Typeit.inDouble("Type in a decimal number");
System.out.println("You entered " + myDouble);
```

Ponovo kompajlirajte i izvršite kôd. Upišite neke godine i videćete:



Upišite vrednost i pritisnite <**Enter**>. Videćete vaš unos „preslikan" kao u ogledalu:



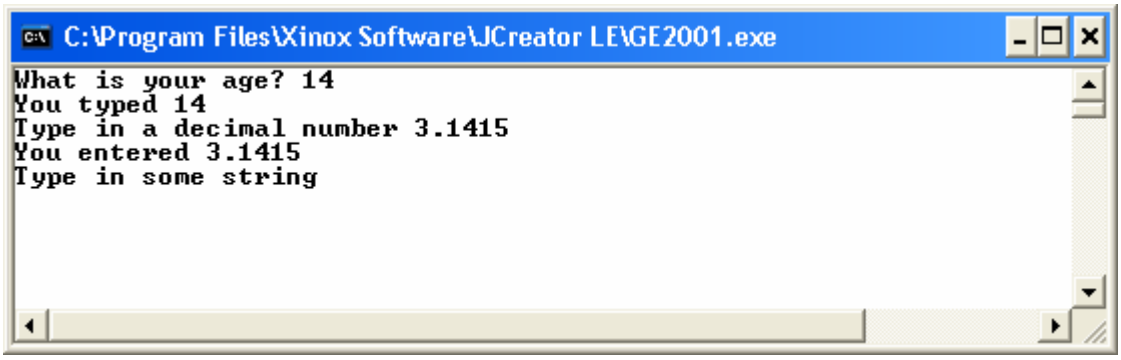
Na kraju, testirajmo **inString**. Dodajmo sledeću deklaraciju promenljive:

String myString;

A zatim dodajte seldeća dva reda koda:

```
myString = Typeit.inString("Type in some string");
System.out.println("You entered " + myString);
```

Ponovo kompajlirajte i izvršite program. Upišite svoje godine, upišite decimalni broj i videćete prompt koji vas pita za string:



Upišite string i pritisnite <**Enter**>. Evo primera ekrana:

Izgleda da svi ulazni metodi fino rade. Da li ste uočili kako je gradnja projekta u etapama (postepeno dodavanje po malo koda) dobra? Uvek sledite takvu proceduru. Kao što sam naveo, koristićemo klasu Typeit i njene ulazne metode u skoro

svim narednim aplikacijama. Ako vam nešto kasnije ne bude jasno uvek se možete vratiti na ovaj primer. Projekt je sačuvan pod imenom **InputProject**.

Pre nego što napustite ovaj primer i pređete na gradnju nekog drugog primera, pogledajte druge korisne Java koncepte. U prethodno datom izlaznom prozoru bilo bi lepo ako bi postojao prazan red između svih ulaznih zahteva za upisivanje. To će uticati da izlaz bude čitljiviji. Jedna od mogućnosti da ubacite prazan red u izlaz je da koristite metod **println** bez argumenata:

```
System.out.println();
```

Ovo može biti zametno ako vam je potrebno više praznih redova.

Lakši način je da se koristi Java „escape" sekvenca za novi red (**\n**). Jednostavno umetnete sekvencu od ova dva znaka u bilo koji string koji prosleđujete na izlaz sa **println**. Kada se to pojavi novi red se započinje. Na primer,

```
System.out.println("This is a line\n");
```

će štampati na ekranu prazan red posle iskaza **This is a line**. Dok će,

```
System.out.println("\nThis is a line");
```

štampati prazan red pre ispisivanja iskaza **This is a line** na ekranu. Uverićete se i sami da je „escape” sekkvenca za novi red veoma korisna.

Java za mlade programere (13)

Projekt – Savings Calculator

U ovom projektu, pravićemo „kalkulator za štediše”. Scenario rada kalkulatora je sledeći: svake sedmice stavljamo novac na račun, banka daje određenu kamatu i mi možemo pomoću ovog kalkulatora da izračunamo koliko ćemo novca imati na računu posle određenog broja sedmica. Ovaj projekt je snimljen pod nazivom **SavingsProject** u folderu (**\JavaKids\JK Code**).

Dizajn projekta

Po koracima, kalkulator treba da uradi sledeće:

1. Dobije iznos sedmičnog depozita.
2. Dobije broj sedmica.
3. Pomnoži ta dva broja.
4. Kao izlaz prikaže dobijeni proizvod.

Koristićemo ulazni metod **Typeit** za prihvat korisnikovog unosa. Metod **println** će se koristiti za ispis iznosa izlaza. Dodaćemo još jedan korak, tj. pitaćemo korisnika za njegovo ime (primer korišćenja metoda inString).

Razvoj projekta

Pokrenite JCreator, otvorite radni prostor i kreirajte novi projekt pod nazivom **SavingsProject**. Dodajte prazan fajl pod nazivom **Savings** (.java ekstenzija će biti po automatizmu dodata) a zatim fajl **Typeit.java** (trebalo bi da se nalazi u vašem radnom folderu).. Podsetite se primera iz prošlog broja i dodavanja klase Typeit.java svom projektu.

Otvorite prazan fajl **Savings.java**. Prvo, upišite sledeće informacije u zaglavlje, definiciju klase i definiciju glavnog metoda (sa svim potrebnim levim zagradama:

```
/*
 *  Savings Project
 *  Java for Kids
 */
 public class Savings
 {
   public static void main(String[] args)
   {
```

Koristićemo četiri promenljive u ovom programu: jednu za korisnikovo ime, jednu za iznos depozita, jednu za broj sedmica i jednu za iznos ukupne sume (totala). Upišite sada njihove deklaracije:

```
// deklarisanje i inicijalizacija promenljivih
String yourName;
double deposit = 0.0;
int weeks = 0;
double total = 0.0;
```

Sada pristupamo kodiranju prema koracima navedenim u dizajnu projekta. Svaki put kada upišemo neki kôd, možemo da se zaustavimo, kompajliramo i izvršimo program da bi videli da li je sve u redu. To je dobra praksa i trebalo bi da vam pređe u naviku. Prvo, pitamo korisnika za ime koristeći sledeći kôd:

```
// pitamo korisnika za ime
yourName = Typeit.inString("Hello, what is your name?");
```

Uočite da je **yourName** tipa string. Zatim, utvrdite koliki će biti sedmični depozit:

```
// dobijanje iznosa depozita
deposit = Typeit.inDouble("\nHow much will you deposit each week?");
```

Iznos za depozit je tipa **double**. Uočite korišćenje tzv. escape sekvence za prelazak u novi red pre pojavljivanja ispisa komandnog prompta (**\n**). Na kraju, dobijamo broj sedmica, vrednost tipa **int**:

```
// dobijanje broja sedmica
weeks = Typeit.inInt("For how many weeks?");
```

Korišćenjem ovh informacija može se izračunati ukupni depozit i prikazati korišćenjem metoda println (nemojte zaboraviti desne zagrade):

```
// izračunavanje i prikazivanje ukupne sume
```

```
total = deposit * weeks;
System.out.println("\n" + yourName + ", after " +
weeks + " weeks, you will have $" + total + " in your
savings.\n");
```

Snimite projekt klikom miša na dugme **Save All**.

Konačan kôd u prozoru Jcreatora trebalo bi da ima sledeći oblik:

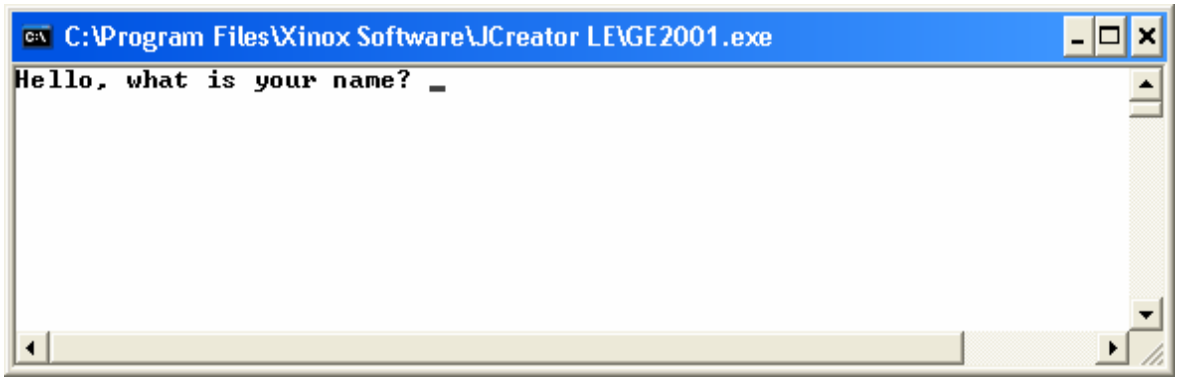
```
/*
 *  Savings Project
 *  Java for Kids
 */
 public class Savings
 {
   public static void main(String[] args)
   {
     // declare and initialize variables
     String yourName;
     double deposit = 0.0;
     int weeks = 0;
     double total = 0.0;

     // ask user name
     yourName = Typeit.inString("Hello, what is your
name?");

     // get deposit amount
     deposit = Typeit.inDouble("\nHow much will you
deposit each week?");

     // get number of weeks
     weeks = Typeit.inInt("For how many weeks?");

     // compute and display total
     total = deposit * weeks;
```

```
    System.out.println("\n" + yourName + ", after " +
weeks + " weeks, you will have $" + total + " in your
savings.\n");
  }
 }
```

Izvršavanje projekta

Kompajlirajte i izvršite projekt. Ako se projekt ne izkompajlira uspešno, pokušajte da pronađete gde su greške pomoću poruka o greškama koje će se pojaviti ukoliko ima grešaka. Jedna moguća greška da program ne može "resolve the symbol Typeit". Ta greška će se pojaviti ukoliko ste zaboravili da dodate fajl Typeit.java svom projektu. Zato još jednom proverite da li ste dodali taj fajl svom projektu.

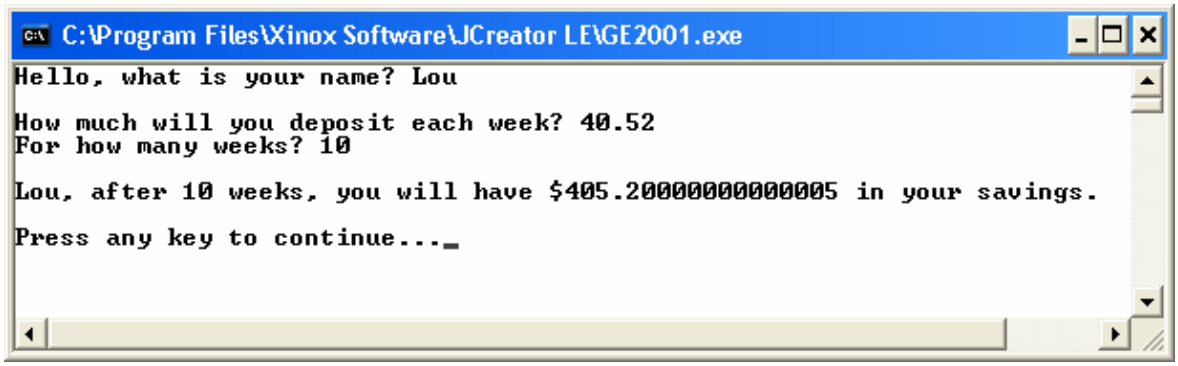
Kada program radi kako treba vi treba na svom ekranu da dobijete prikaz dat na sledećoj slici:



Upišite svoje ime, iznos depozita i broj sedmica. Vaša ukupna suma biće prikazana u obliku lepo formatiranog string. Uočite kako su ime, depozit, sedmice i ukupna suma stavljeni zajedno

(nadovezani) u jednu rečenicu, zajedno sa znakom za dolar ($). Proverite da li je odgovor tačan. Da vas podsetim, veliki korak u programiranju je proveravanje da li sve radi kako treba! Ako ste naveli da štedite 10 dolara nedeljno u narednih 10 nedelja i vaš program kaže da ćete tada imati milion dolara, znate da negde nešto nije u redu!

Kada sam probao program dobio sam:



Uočite da je depozit 40.52 (znak za dolar se ne unosi) u toku 10 sedmica, a program saopštava da ću imati $405.20000000000005 na računu!! Ovde se pojavljuje mnogo nula zato što računar nije sposoban da precizno izvrši aritmetiku – dobili smo ono što se naziva greška zaokruživanja (round-off error). Rezultat može biti korektnije prikazan korišćenjem samo dve decimale ($405.20). To je lako uraditi u Javi, ali je u ovom trenutku van našeg interesovanja.

Čini se da ovaj projekt nije komplikovan. I nije. Mi smo samo množili dva broja. Međutim, projekt demonstrira korake koji se koriste u svakom Java projektu. Vredno iskustvo se stiče

saznavanjem kako se čitaju ulazne vrednosti, proverava da su vrednosti odgovarajućeg tipa, rade matematiku tako da se dobijaju tačni rezultati i prikazuju ih korisniku.

Pokušajmo još neke stvari

Uobičajeno je da se na štednju zaračunava i neka kamata, tj. Banka vam plaća zato što ste joj omogućili da radi sa vašim novcem. Naš projekt je ignorisao kamatu. Međutim, izuzetno je lako prepraviti ovaj program i omogućiti izračunavanje kamate. Ovde će vam biti rečeno kako da prepravite projekt, ali neće biti to i pokazano. Pokušajte samo da to uradite:

- Definišite promenljivu **interest** da biste memorisali godišnju stopu kamate. Kamatne stope su decimalni brojevi, tako da ova promenljiva treba da bude tipa **double**.
- Dodajte još jednu naredbu za ulaz da biste omogućili korisniku da upiše kamatnu stopu.
- Prepravite kôd tako da koristi kamatnu stopu prilikom izračunavanja ukupne sume (**total)**. Kôd za to izračunavanje je:

  total = 5200 * (deposit * (Math.pow((1 + interest / 5200), weeks) - 1) / interest);

  Upišite sve u jedan red – kao što možda već znate, programi za obradu teksta izmene oblik koda zbog

preloma reda, pa je to i ovde slučaj. Ovo je ujedno vežba za korišćenje zagrada i matematičke funkcije (**pow**). Broj '5200' se koristi za konvertovanje kamatne stope sa godišnje na sedmičnu stopu.

Sada kompajlirajte i izvršite prepravljenu verziju programa. Upišite vrednosti za depozit, sedmice i kamatnu stopu. Proverite ručno da li vam računar daje tačan rezultat. (Ako ste za depozit upisali vrednost 10, za 20 sedmica, a kamatna stopa je 6.5, ukupna suma trebalo bi da bude $202.39 (sa zaokruživanjem, inače se dobija $202.3929075…). Snimite svoj projekt.

Pripremio Dragan Marković