# 1. Kako početi programirati u JAVA jeziku

Cilj ovog poglavlja je napisati i pokrenuti jednostavnije Java programe.

SADRŽAJ



---

## *1. O predmetu*

Sadržaj ovog predmeta je pisanje objektno orijentiranih programa u Java jeziku.
Java nije samo programski jezik već ujedno i vrlo velika biblioteka programskih rutina te izvršna okolina za izvođenje programa. Java biblioteka programskih rutina sastoji se od tisuća **klasa**. Postoje klase za rad s datotekama, klase za rad s 3D grafikom, klase za pristup bazama podataka, animaciju web stranica, itd. itd. Moglo bi se reći da se u bibliotekama nalazi veći dio onoga što će vam ikada zatrebati u programiranju. Možda ste se dosad susreli s JavaScript jezikom za web stranice. JavaScript nije Java jezik !

Ovaj predmet se zbog ograničenog broja sati ipak mora ograničiti na osnove jezika te na manji dio osnovnih klasa.

Kao dodatnu literaturu možete koristiti knjige:

Čukman, Tihomir: Java , nakladnik Alfej
Dario Sušanj, "Java: Programiranje za Internet i World Wide Web", Znak, Zagreb, 1997.

Sadržaj predmeta je prilagođen kako studentima s predznanjem tako i onima bez ikakvog predznanja programiranja. Ako dosad niste programirali bit će potrebno uložiti značajan trud u savladavanje gradiva. Java jezik zajedno s ogromnim bibliotekama jezik je namijenjen profesionalnim programerima. Štoviše Java biblioteke se stalno proširuju i mijenjaju .

Naučiti programirati u bilo kojem jeziku nije moguće bez praktičnog rada na računalu.

Stoga je od vrlo velike važnosti da primjere sa satova pokušate realizirati na računalu tijekom vježbi.

Ako ste dosad programirali u nekom programskom jeziku mogao bi vam prvi uvodni dio biti vrlo jednostavan. međutim kako lekcije budu išle broj novih sadržaja i znanja će se povećavati.

Potrudite se zadatke odraditi sami. Nemojte kopirati programski kod od kolega. Osim što varate nastavnika varate i sami sebe.

---

## *2. Programi i programiranje*

U ovome poglavlje prikazat ćemo jednostavni Java program i način kako ga pokrenuti na računalu.

Za početak ćemo ponoviti glavne dijelove računala i njihova svojstva:

1. **Memorija** je dio računala u koji pohranjujemo informacije. Pohranjene informacije možemo pisati, brisati, obnavljati,... Informacije u memoriji su niz bitova dok na višoj razini predstavljaju brojeve, tekst, slike, glazbu,...
2. **Uređaji za ulaz/izlaz (I/O).** Informacije u računalo unosimo bilo tipkovnicom ili preko medija (disketa, CD-ROM ,mreža...). Informacije iz računala prikazujemo na ekranu ili pak šaljemo na neki od medija.
3. **Procesor** koji djeluje po instrukcijama **programa**. Program se sastoji od niza operacija koje procesor izvršava. To uključuje akcije poput izvršavanja proračuna, čitanja ili pisanja po memoriji, slanja podataka na izlazna sučelja procesora, ...

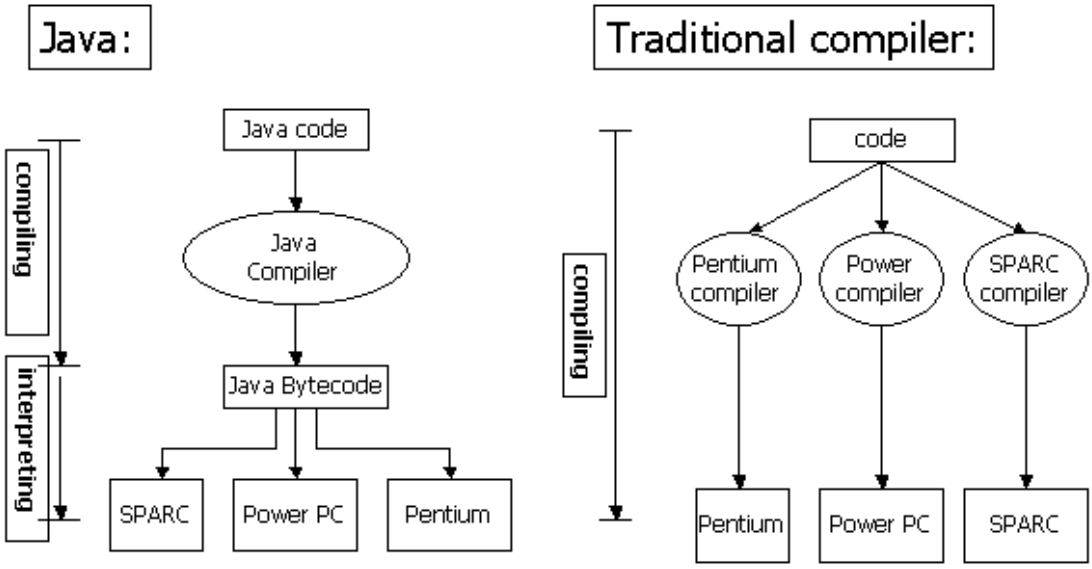
U industriji programiranja većinu velikih programa napisali su timovi programera, ali postoje i slučajevi vrlo složenih i korisnih programa koje su napisali pojedinci.

Program se piše u notaciji koja se naziva **programski jezik**. Svako računalo (procesor) ima svoj programski jezik koji nazivamo **strojni jezik** (machine code). Taj jezik je dizajniran s fokusom na elementarne operacije koje se obavljaju nad hardverom računala. Radi se o jednostavnim operacijama poput pisanja ili čitanja iz memorije, aritmetičkim operacijama nad registrima procesora, itd. Iako je teoretski svaki program moguće napisati koristeći strojni jezik to je vrlo teško čak i za jednostavne programe. U praksi se gotovo svo programiranje izvodi u jezicima koji su prilagođeni programeru. Takvi jezici se nazivaju **jezici visokog nivoa**.
Zadnjih četrdeset godina razvijeno je niz jezika visokog nivoa. Neki od ranih jezika su još u upotrebi. Jezici koji se danas koriste u komercijalnoj upotrebi su C, C++,Java, Pascal(Delphi),Basic,Fortran....
Java jezik je jedan od najmlađih jezika. Prvi put se pojavio 1995. godine. Java 2 specifikacija jezika koju koristimo u ovom predmetu pojavila se 1998.

Rekli smo da programeri pišu programe uglavnom koristeći programske jezike visokog nivoa, a da računala izvršavaju instrukcije strojnog jezika. Pitanje je što računalo radi s programom

napisanim u jeziku visokog nivoa. Najčešći način je koristiti računalni program koji nazivamo **prevodilac (compiler)**. Prevodilac prevodi program napisan u jeziku visokog nivoa u program sastavljen od strojnog jezika. Prevedeni program onda možemo pokrenuti na računalu. (U čemu se piše prevodilac ?)

U slučaju Java jezika korišten je malo drukčiji pristup u kojem se u procesu pisanja Java koda do transformacije u računalu razumljiv kod koriste dva programa. Prvo se program koji je programer napisao u Javi, pomoću prevodioca prevodi u **bytecode** program**.** Bytecode je sličan strojnom jeziku, ali je neovisan o bilo kojem računalu. Bytecode program nije više čitljiv od strane programera. Njega čita i izvršava program koji se naziva **Java virtual machine.** Prednosti pristupa u dva koraka jest da se tako proizvode programi koji se ipak izvršavaju zadovoljavajućom brzinom te se Java okolina može brzo realizirati na bilo kojem računalu.
Originalni Java program koji piše programer i kojeg prevodi prevodilac naziva se **izvorni kod**. Bytecode koji proizvodi prevodilac i interpretira Java virtual machine naziva se **objektni kod.**




Slika 1.1 Usporedba izvršavanja Java programa s tradicionalnim postupkom prevođenja

Postoje brojne razvojne okoline u kojima je moguće pisati, prevoditi i izvršavati programe. Za potrebe ovog predmeta zadržat ćemo se na najjednostavnijem sustavu koji je ujedno i podloga i za druge kompleksnije razvojne okoline.

Sustav se naziva Software Development Kit (SDK). Može se naći na stranicama

```
java.sun.com
```

Ove stranice održava kompanija Sun Microsystems koja je odgovorna za razvoj Jave. Trenutna verzija Java okoline je 1.4.0. U ovom predmetu koristit ćemo prethodnu 1.3.1. verziju.

Instalacija Java okoline se u najkraćim crtama obavlja na slijedeći način:

- Pokreni Java 2 SDK installer (datoteka j2sdk-1_3_1-win.exe), odaberi mjesto instalacije i instaliraj.
- Dodaj u PATH variablu operativnog sustava mjesto instalacije (npr. C:\jdk1.3.1\bin). Način dodavanja je ovisan korištenom operativnom sustavu.
- Provjeri (ukloni) CLASSPATH variablu (-classpath command-line prekidač je bolji način).

Detaljniji opis instalacije može se naći na stranicama java.sun.com.

---

## *3. Primjeri jednostavnih Java programa*

U C jeziku prvi program imao je funkciju ispisa jednostavne poruke na ekran.
U Javi program iste funkcionalnosti izgleda ovako:

```
public class Hello
{  /* napiši jednostavnu poruku na ekran*/
   public static void main(String[] args)
   {  System.out.println(″Hello, World!″);
   }
}
```

Ako niste vidjeli dosada neki Java program ovaj jednostavni program izgledat će vam konfuzno. Cilj ovih početnih sati predavanja je da vam se objasni struktura ovako jednostavnih programa.

Svaki Java program sadrži **naredbe** (statements). Svaka naredba opisuje neku operaciju koju računalo treba izvršiti. Operacija može biti ispis neke informacije na ekranu, može biti neka računska operacija, provjera položaja miša na ekranu, itd. Računalo jednostavno izvršava naredbu po naredbu.

U programu koji je gore napisan nalazi se samo jedna naredba:

```
System.out.println(″Hello, World!″);
```

Kad se ta naredba izvrši na ekranu će se pojaviti slijedeći ispis:

```
Hello, World!
```

Java ima različite načine pisanja poruka po ekranu bilo da pišemo po prozoru, na web stranicu, itd. U ovome slučaju koristimo jednu Java **metodu** koja se naziva

`System.out.println`. Rezultat izvršavanja bit će ispis poruke u najjednostavnijem obliku prozora kojeg nazivamo **konzola** (ili DOS prozor u Windows OS). Konzola dopušta samo jednostavan ispis teksta , redak po redak.

Slijedi program s dvije naredbe koje su tiskane podebljano (ne koristimo podebljane fontove u Java programima).

```
public class Hello
{  /* napiši jednostavnu poruku na ekran*/

   public static void main(String[] args)
   {  System.out.println("Hello, World!");
      System.out.println("See you later.");
   }
}
```

Kada pokrenemo ovaj program izvršit će se obe naredbe jedna za drugom. Prvo će se na ekranu ispisati u jednoj liniji `Hello, World!` , a nakon toga u drugoj liniji `See you later.`

Oba programa imaju formu:

```
public class Hello
{  /* komentar. */
   public static void main(String[] args)
   {
         Naredbe

   }
}
```

ne postoji ograničenje na broj naredbi u programu. Može ih biti na tisuće.
Ostatak ovog jednostavnog programa može se promatrati kao pakiranje. Ovo pakiranje ćemo objasniti poslije. Sad ćemo se zadržati na opisu **objekata** koji su neizostavni dio svakoga pa i najjednostavnijeg Java programa.

---

## *4. Objekti i Metode*

Razmotrimo naredbu:

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

koja ispisuje poruku `Hello, World!` na ekran. Gdje su tu **objekti** ? Poznavalac Java jezika vidjet će dva objekta. Prvi je objekt `System.Out` , a drugi sami niz znakova `"Hello, World!"`. Java cijelo vrijeme radi s objektima. U Java biblioteci definirano je mnogo vrsta različitih objekata koje možemo koristiti u svojim programima.

Možemo i kreirati objekte prema našim potrebama. Npr. pišemo program koji će pratiti koji su studenti na FESB-u prijavljeni na koji predmet. Tada ćemo napisati takav program koji će pokretanjem:

- za svakog studenta kreirati jedan objekt studenta
- za svaki predmet također jedan objekt predmeta.

Svaki objekt studenta sadržavat će određene podatke poput osobnih podataka studenta i liste upisanih predmeta. Objekt predmeta može sadržavati naziv predmeta i druge podatke vezane za predmet.

U isto vrijeme kad definiramo izgled objekata trebamo i definirati koje će se operacije izvršavati nad tim objektima. Što se tiče objekta student , bit će nam potrebne operacije kreiranja objekta studenta, ažuriranja liste predmeta koje je student upisao, operacije ispisa podataka o studentu na ekran, itd. Te operacije koje se izvršavaju nad objektom nazivaju se **metode**.

Dosad smo već vidjeli primjer metode. Njen puni naziv je:

```
System.out.println
```

Ovaj naziv označava metodu `println` koja pripada objektu `System.out`. `System.out` je objekt čiji je zadatak da primi poruku koju treba prikazati na ekranu. Zamislimo ga kao osobu kojoj dajemo što treba ispisati na ploči.

`Println` metoda je operacija koja se izvršava nad porukom. tako nam izraz:

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

kaže:

koristi `println` metoda za slanje poruke `"Hello, World!"` objektu `System.out`, koji će je prikazati na ekranu.

Svaki objekt pripada **klasi (class)** koja specificira od kojih podataka se objekt sastoji i koje metode posjeduje. Npr. svi nizovi znakova pripadaju klasi koja se naziva String. Klase String i System.Out definirane su u klasama koje pripadaju Java bibliotekama. Možemo kreirati i svoje klase npr. klasu Student i klasu Predmet.

Koji je odnos klasa-objekt ? Kažemo da je objekt instanca od klase. Jednostavno, klasa je opis objekta napisan u kodu. Možemo je promatrati kao kalup ili skicu prema kojoj se u tijeku izvršavanja programa kreiraju objekti. Program može kreirati više objekata ,instanci iste klase.
Java biblioteka je u potpunosti sastavljana od definicija klasa. Ako napišemo bilo koji program u Javi i on će se sastojati od klasa.

Većina klasa definira tipove objekata. Postoje samo nekoliko klasa kojima se ne definira objekt već su sastavljene samo od samostalnih metoda.

Ipak nije sve u Javi objekt. Najjednostavniji tipovi podataka poput cjelobrojnih i brojeva u pokretnom zarezu tretiraju se nešto drugačije. takvi podaci nazivaju se **primitivni tipovi podataka**.

---

## *5. Kako je konstruiran program iz primjera*

Prethodni dio pokazao nam je da se Java programi sastoje od klasa. Programi koji su navedeni kao primjer uklapaju se u tu tvrdnju, ali ipak na vrlo primitivan način. Oba programa sastoje se od jedne klase (Hello) koja se sastoji samo od jedne statičke metode tj. metode koja ne pripada nijednom određenom objektu.

Svaki metoda bilo da pripada objektu ili ne, sadrži određeni broj naredbi koje izvršavaju neku korisnu operaciju. (U drugim računalnim jezicima imamo funkcije ili procedure)

Analizirat ćemo korak po korak kompletnu metodu iz zadnjeg primjera:

```
/* napiši jednostavnu poruku na ekran*/
public static void main(String[] args)
{  System.out.println(″Hello, World! ″);
   System.out.println(″See you later. ″);
}
```

Sastoji se od slijedećih dijelova:

1. `/* napiši jednostavnu poruku na ekran*/`

   Ovo je komentar koji opisuje što ni program trebao raditi. to je jednostavno poruka za onoga tko čita izvorni kod programa. Svaki tekst između `/* i */` bit će tretiran kao komentar i Java će ga u potpunosti ignorirati.

2. `public static void main(String[] args)`

   Ovo predstavlja **zaglavlje** (heading) metode. Svaki metod ima svoj naziv. U ovom slučaju naziv metoda je riječ `main` koja se nalazi neposredno ispred zagrada. Riječi `public`, `static` i `void` pokazuju Java prevodiocu način korištenja metoda main. (Bit će objašnjeno kasnije)

   Dio u zagradama, `String[] args`, opisuje informaciju koja će biti proslijeđena metodu svaki put kad bude pozvan. Naziva se lista **parametara.** U navedenom primjeru ta je informacija ignorirana, odnosno nije korištena u programu. (koko se koristi bit će objašnjeno poslije)

3. ```
   {  System.out.println(″Hello, World! ″);
      System.out.println(″See you later. ″);
   }
   ```

   Ovo je **tijelo** (body) metoda. Uvijek se sastoji od niza naredbi zatvorenih u vitičaste zagrade, {..}. Pozivom ovog metoda izvršava se svaka od naredbi.
   ```

Sve metode sastoje se od tri navedena dijela. Strogo rečeno komentar je opcionalan. Međutim preporuča se uvijek početi s komentarom koji ukratko kaže što radi metod koji slijedi.
Taj dio nazivamo **specifikacijom**. Gornji primjer je prejednostavan da bi specifikacija bila od veće koristi, ali u većim programima to je najefikasniji način da pomognemo razumijevanju programa. Posebno je to bitno ako na programu radi više programera.

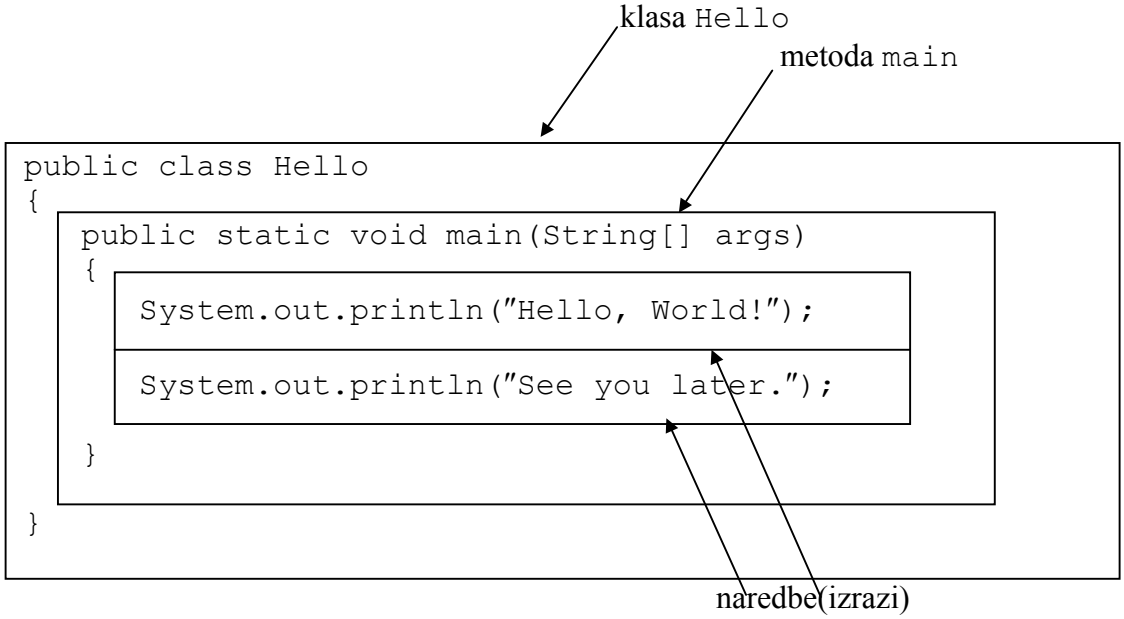
Svaki program sastoji se od određenog broja definicija klasa. U dva gornja primjera u programu je definirana samo jedna jedina klasa nazvana `Hello`. Definicija klase započinje s zaglavljem:

```
public class Hello
```

Zaglavlje je praćeno elementima koji sačinjavaju klasu, zatvorenim u vitičaste zagrade. U našim primjerima klasa se sastoji od samo jednog člana, metode nazvane `main`, koju svo već opisali. Općenito program se sastoji od jedne ili više definicija klasa od kojih svaka sadržava jednu ili više metoda.

Java programeri koriste konvenciju po kojoj naziv klase započinje velikim slovom, a naziv metode malim slovom. U ovome predmetu ćemo se nastojati strogo pridržavati navedene konvencije. Općenito bilo koji naziv u Javi (**identifikator**) sa;injen je od slova, znamenki i mora počinjati sa slovom. Za potrebe ove definicije se simboli valuta poput £ i $, i povlaka (‘_’) računaju kao slova. Duljina naziva nije ograničena.

Na slijedećoj slici prikazan je naš program koji se sastoji od dvije naredbe razdvojen okvirima koji nam pokazuju što je klasa, što je metoda a što su naredbe.




## 6. Metoda System.out.println

`System.out.println` je prva metoda iz Java biblioteke koji smo upotrijebili. Ova metoda će ispisati niz znakova na konzolu (DOS prozor). Naziv `println` je skraćenica za 'print line'.

Mjesto na ekranu na kojemu će se ispisati slijedeći znak označeno je na ekranu s malom treptajućom linijom nazvanom **kursor.** `System.out.println` metoda ima tu osobinu da ispisom teksta pomiče kursor na početak slijedeće linije. Ako želimo da kursor ostane na prethodnoj liniji koristit ćemo metodu `System.out.print.`

Drugi način određivanja kad želimo ispis u novoj liniji je korištenje para znakova `\n` u nizu znakova svaki put kad je potrebno da ispis krene u novu liniju. Npr. :

```
System.out.print(″Hello, World!\nSee you later\n″)
```

će ispisati `Hello, World!` , nakon toga pomaknuti kursor u novu liniju gdje će ispisati `See you later.` , i nakon toga pomaknuti kursor na novu liniju. Ta će naredba imati isti rezultat kao i par naredbi:

```
System.out.println(″Hello, World! ″);
System.out.println(″See you later. ″)
```

---

## *7. Kako prevesti i pokrenuti Java program*

Pretpostavimo da želite izvršiti program iz prvog primjera:
.

```
class Hello
{  /* napiši jednostavnu poruku na ekran*/
   public static void main(String[] args)
   {  System.out.println(″Hello, World!″);
   }
}
```

Prvo morate biti pristupiti računalo na koje je instalirana podrška za Javu odnosno Java SDK.

A. **Ukucajte program i pohranite ga u datoteku `Hello.java`.**
Možete koristiti bilo koji tekst editor za unos koda, npr. Notepad. Korisno je imati i neki sofisticiraniji Java editor npr. JedPlus. Takvi editori nam mogu omogućiti korisne funkcije poput sintaksnog naglašavanja ili automatskog uvlačenja teksta. Moguće je iz takvih editora pozvati i operacije prevođenja i izvršavanja.

Ako se vaš program sastoji od samo jedne klase potrebno ga je pohraniti u datoteku koja ima isti naziv kao i klasa uz dodanu ekstenziju '`.java`'. kako se naš jednostavni program sastoji od samo jedne klase nazvane Hello, datoteku **moramo** nazvati `Hello.java`.

B. **Otvori DOS prozor i postavi trenutni direktorij na direktorij gdje je datoteka s programom.**

Sve naredbe koje slijede tipkaju se u ovaj prozor. Poruke prevodioca i sve što će program ispisati odvija se također u ovome prozoru. Postoje i drugi načini koje ćemo obraditi na vježbama, ali ovo je općeniti način koji radi na svim računalima.

U ovome trenutku bit će otvorena dva prozora. Jedan s editorom (npr. Notepad), a drugi s DOS prozorom koji služi za interakciju s Java sustavom. U tom DOS prozoru prevodimo i izvršavamo program.

C. **Korištenje JDK za prevođenje programa u `Hello.java` datoteci.**

Za prevođenje programa treba utipkati:

```
javac Hello.java
```

Ako dobijete poruku da sustav ne može naći javac (Java prevodilac) znači da ili nije pravilno postavljena PATH varijabla sustava ili nije instaliran JDK.

Ako nema grešaka u programu prevodilac će proizvesti bytecode verziju vašeg programa u datoteci nazvanoj `Hello.class`.

Ako prevodilac nađe greške poput tipkanja Class umjesto class ili izostavljanja znaka ; na kraju naredbe, odbit će prevođenje i izvijestiti o pronađenim greškama.
To nazivamo **greškom prevođenja (compiler error)**. Najčešće je poruka o grešci takva da je jednostavno naći mjesto u kodu gdje smo učinili pogrešku. Ponekad iz poruke nije jasno odakle potječe greška i tada je potrebno pažljivo pregledati kod. Početnici često pogriješe tražeći grešku baš u liniji gdje je to prevodilac javio. Međutim , greška se može nalaziti i negdje prije !

Ako postoje greške kod prevođenja potrebno ih je otkloniti u editoru. nakon otklanjanja grešaka ne zaboravite snimiti datoteku.

D. **Korištenje JDK za pokretanje prevedene verzije programa koja se nalazi u `Hello.class` datoteci.**

Nakon što je prevodilac proizveo bytecode verziju programa `Hello.class`, možete ga pokrenuti u Java Virtual machine tipkanjem:

```
java Hello
```

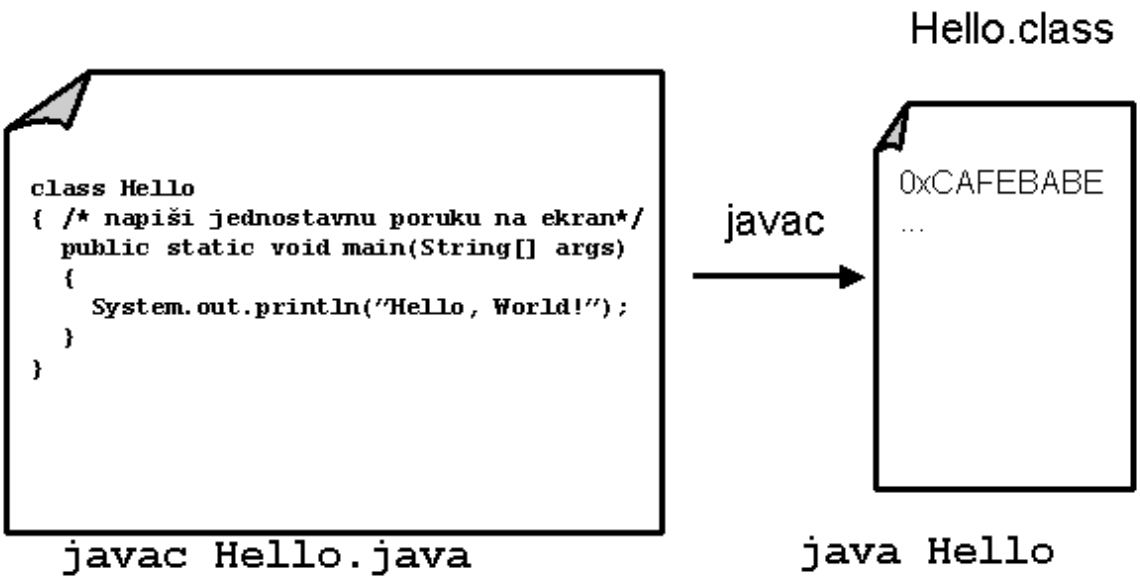
Ne tipkajte `.class` ekstenziju. JDK pretpostavlja da ste mislili na `Hello.class`.

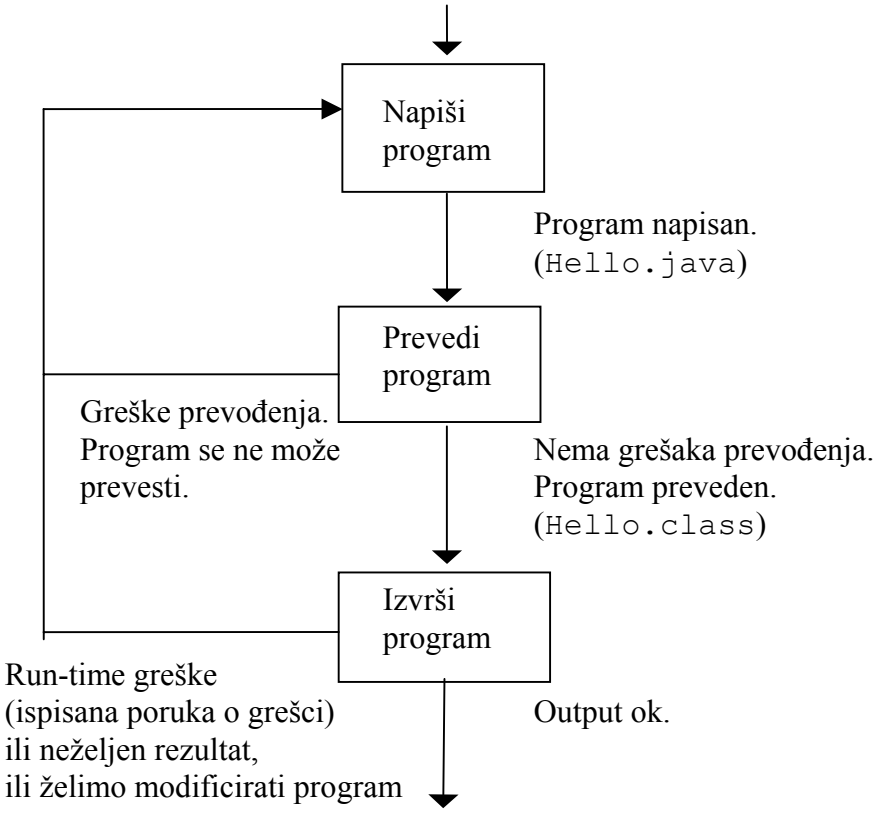
Ovaj se program nakon toga treba izvršiti i ispisati poruku u DOS prozoru. Ako želite modificirati ispis programa ili dodati još koju naredbu opet se trebate vratiti u prozor tekst editora, napraviti modifikacije i nakon toga u DOS prozoru ponoviti postupak prevođenja i izvršavanja.

U tijeku izvođenja programa može nastati greška. Takva greška naziva se **run-time greška** (run-time error) ili **greška u izvršavanju** (execution error).

U nekim slučajevima JVM neće moći izvršiti program do kraja. Tada će ispisati poruku o grešci s podatkom gdje je program došao u izvođenju prije nego što je prekinut.

Proces pisanja, prevođenja i izvođenja programa pokazan je i na slijedećim slikama:




Slika 1.2 Pisanje programa, prevođenje, izvršavanje.

## *8. Pisanje koda Java programa*

Prvo pravilo pisanja bilo kojeg programa je: pišite pažljivije što god možete. Skoro svaka pogreška u pisanju će najvjerojatnije proizvesti grešku prilikom prevođenja ili prilikom izvođenja. Prvo će je trebati detektirati, a nakon toga korigirati. Npr. svaki izraz koji koristi naredbu `System.out.println` mora završiti s ; . Ako izostavite ; dobit ćete poruku o grešci u fazi prevođenja.

U pisanju Java programa morate paziti na korištenje malih i velikih slova. Za razliku od nekih drugih programskih jezika Java razlikuje velika i mala slova (case sensitive). To znači da ne možete utipkati `Class` ili `CLASS` na početku programa. Potrebno je točno pisati `class`.

Riječ `class` ima u Java programima specijalno značenje i jedna je od **ključnih riječi** (keywords) Java jezika. java ima 47 ključnih riječi i sve se pišu malim slovima. kao što je prije napisano držat ćemo se konvencije po kojoj se nazivi klasa pišu s početnim velikim slovom, a nazivi metoda malim slovima.

Ako imamo nazive koji se sastoje od više spojenih riječi možemo početak slijedeće riječi pisati velikim slovom. Npr. klasu možemo nazvati HelloWorld, a metodu npr. ispisTeksta.

Koliko grešaka pisanja možete vidjeti u ovome kodu ? prevodilac će javiti svaku od njih.

```
public Class Hello
{  /* Write a simple message on the screen.
   public static viod main(string[] args)
   {  System.out,println('Hello, World!");
      System.out.printline("See you latef.");
   }
)
```

jedina greška koju prevodilac neće javiti je pogrešno pisanje unutar niza znakova `See you latef`. Ta greška će biti očigledna tek nakon izvršavanja programa.

Zbog svega navedenog nije se praktično oslanjati na prevodilac u otkrivanju grešaka. To je prije svega dugotrajan proces jer je potrebno svaki put nakon ispravljene greške proći kroz faze prevođenja i izvršavanja. Međutim to više vrijedi za iskusne programere nego za početnike kojima će prevodilac često biti neprocjenjiva pomoć u otklanjanju pogrešaka.

Osim što je Java prevodilac razlikovanjem velikih i malih slova zna biti dosta nezgodan u drugim pogledima je vrlo liberalan. Uopće mu nije bitan prostorni raspored vašeg koda tj. upotreba praznina i novih redova.

Slijedeći primjer bi se trebao prevesti bez greške iako je vrlo teško čitljiv:

```
public class Hello{public static void main(String[]
```

```
args){System.out.println(″Hello, World!″);System.
out.println(″See you later.″);}}
```

Iz ovoga je na prvi pogled teško ustanoviti da se program sastoji od jednog metoda koji sadrži dvije naredbe. Čak će se i slijedeći primjer pravilno prevesti:

```
public                                                class
Hello


public
static
void                              main


(String[] args)

   {  System.    out.       println

(                 ″Hello, World!″                    );
                                                System.out.
println(          ″See you later.″                   );


}                                                     }
```

Sve dok ne spojite neke riječi poput `classHello`, ili ne razdvojite `cla ss`, prevodilac se neće buniti..

Prevodilac prema svemu navedenom je vrlo fleksibilan u korištenju razmaka i novih redova. Međutim bilo tko bude čitao vaš program (i vi sami) poželjet će mnogo uredniji i jasniji kod. Zato je praznine uputno koristiti da bi se dobio jasniji kod.
Ako pogledate na originalni kod gore modificiranih programa, vidjet ćete da postoje neka pravila u pisanju. Npr. metoda `main` je pomaknuta tri mjesta unutar u odnosu na prethodni redak. To nazivamo **uvlačenje koda** (indentation). Ako ima više metoda unutar klase, stavljamo razmak između svake. Primijetite i da vitičaste zagrade imaju konvenciju o smještaju. Vidjet ćemo tijekom ovoga predmeta kako postupati s pojedinim elementima koda. Osim urednosti koja omogućava razumijevanje koda, drugi rezultat je pisanje koda s manje grešaka.

## *9. Kako rješavati zadatke*

Na računalu na kojemu dobijete korisnički račun kreirajte direktorij `JProg`.
Unutar tog direktorija za svako poglavlja otvorite direktorij `Pn` gdje je n broj poglavlja.
Dakle za prvo poglavlje otvorite direktorij P1. Kada završite poglavlje 1 unutar direktorija P1 trebali biste imati slijedeće datoteke:

```
Hello.java
Hello.class
ImeUOkviru.java
ImeUOkviru.class
Gresnik.class
Gresnik.java
Inicijali.class
Inicijali.java
```

Nakon kraja vježbi programi koji ostanu u vašem direktoriju bit će kopirani i pregledani.
Molimo vas da programe ne kopirate jer neće biti priznati.

---

## *10. Zadaci za prvo poglavlje*

Za ovo poglavlja zadana su tri jednostavna problema tj. pisanje tri jednostavna programa..

1. Ukucaj, prevedi i pokreni slijedeći Hello program koji ispisuje dvije linije na ekran:

```
/* Autor: Ime Prezime */

class Hello
{

    /* jednostavan ispis na ekran. */
   public static void main(String[] args)
   {  System.out.println("Ciao, Svite!");
      System.out.println("Vidimo se kasnije !");
   }
}
```

2. Napišite program s nazivom klase ImeUOkviru. Ovaj program treba ispisati vaše ime u okviru poput slijedećega:

```
+--------+
|  Ivan  |
+--------+
```

Uputa: nije svejedno kako ćete nazvati datoteku s kodom!

3. Ukucaj i pokreni slijedeći program te analiziraj pogreške koje će javiti prevodilac. ukucajte ga sa svim pogreškama !

```
public Class Gresnik {
   /* Write a simple message on the screen.
   public static viod main(string[] args)
   {  System.out,println('Alo, Svijete!");
      System.out.printline("Vikimo se kasnije.");
   }
}
```

Zatim ispravite pogrešaka koliko možete i pokušajte natjerati program da radi kako bi trebao.

4. Napiši program koji će ispisati vaše inicijale na ekranu na slijedeći način:

```
O       O  OOOOOO
OO     OO  O
O O   O O  O
O  O O  O  OOOOOO
O   O   O  O
O       O  O
O       O  OOOOOO
```

Klasu nazovi `Inicijali.`