LINKgroup

Distance Learning System

Java-Core

Uvod u Javu

Šta je Java

- Java je višenamenski programski jezik
- Osnovne karakteristike su mu:
  - Portabilnost
  - Jednostavnost
  - Objektna orjentacija
  - Pouzdanost
- Nije vezan za operativni sistem
- Java je i platforma koja služi za pokretanje programa koji su napisani u Javi

Istorijat Jave

- 1991. godine James Gosling počinje rad na programskom jeziku Oak
- Sledeće godine, Sun Microsystems predstavlja Star7, PDA uređaj sa touchscreenom i animiranim grafičkim korisničkim okruženjem, na kojem OAK debituje
- Oak, 1995. godine menja naziv u Java
- Sa „eksplozijom” WWW-a, Java izlazi u prvi plan, zato što je Webu bila potrebna portabilnost koju je Java omogućavala.

Sintaksa Jave

- Java nasleđuje sintaksu iz programskog jezika C, a objektni model adaptira iz jezika C++
- Java nije „internet verzija C++", jer nije dizajnirana kako bi zamenila C++, već kako bi rešila određene probleme koji nisu mogli biti rešeni, ili su mogli, ali na znatno teži način. Jasno je da danas oba programska jezika „žive" paralelno, i tako će biti i u budućnosti
- Najveći konkurent jeziku Java je jezik C# koji egzistira na Microsoft .NET tehnologiji. Smatra se da, poznajući Javu, programer poseduje predispozicije za brzo upoznavanje sa C#-om, i obrnuto

Kako Java funkcioniše

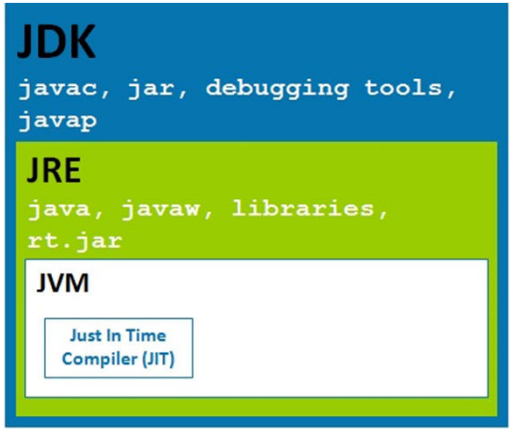
- Java program započinje kao izvorni kod
- Fajl koji sadrži java izvorni kod ima ekstenziju **.java**.
- Kompajliranje izvornog koda koji obavlja Java kompajler (**javac**).
- Rezultat ovog kompajliranja je fajl sa ekstenzijom .class. Sadržaj ovog fajla naziva se bytecode.
- JVM pretvara bytecode u mašinski jezik.
- Jednom napisan kod možemo u istom obliku izvršavati na različitim platformama uz pomoć JVM (Java Virtual Machine)

Kompajliranje / Interpretacija

- Izvršavanje bilo kog programa događa se uvek u jednom od dva oblika: **interpretacija** ili **izvršavanje** kompajliranog koda.
- **Interpretacija** - prevođenje i izvršavanje koda u realnom vremenu
- **Kompajliranje** - kod će biti prvo preveden u izvorni jezik okruženja, a zatim startovan svaki put u prevedenom obliku

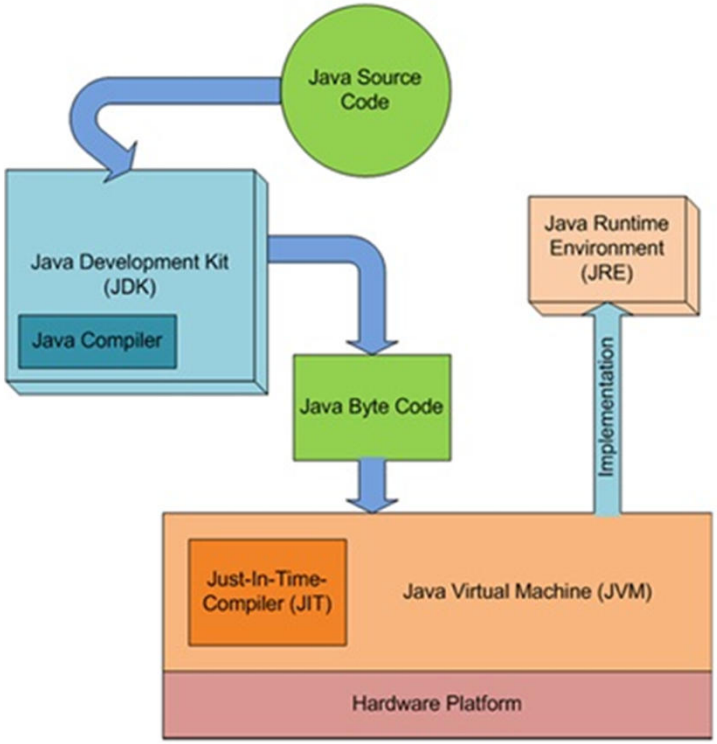
Java Virtual Machine

- Jedini preduslov da Java program funkcioniše u nekom okruženju jeste postojanje JVM (Java Virtual Machine)
- Java Virtualna Mašina predstavlja jednu vrstu softverskog procesora sa svojim setom instrukcija.
- Sastoji se od ogromnog broja biblioteka
- „glue" kod, koji služi da se ostvari interakcija sa operativnim sistemom
- Java Virtual Machine se može naći u sklopu **JRE** i **JDK**-a.
- **JRE** izvršava programe napisane u Java jeziku
- **JDK** - set alata za razvoj aplikacija



Java proces izvršavanja

- JIT (Just In Time Compiler) služi da ubrza interpretaciju koda, tako što prepoznaje naredbe koje se često ponavljaju.



Vežba 1 – (jcex012014)

- Prethodna objašnjenja su jasna, ali kako to izgleda u praksi? Kako napraviti jedan Java program i startovati ga?

Vežba 1 – (jcex012014)

- Da bi Java program bio startovan, na računaru mora postojati Java Virtualna Mašina (JVM).
- JVM može biti instaliran zasebno, ili u sklopu JRE ili JDK paketa. Ukoliko imamo nameru da kreiramo Java programe, a ne samo da ih koristimo, najbolje je da na računaru obezbedimo JDK.
- JDK možemo preuzeti sa Oracle-ovog sajta:

Vežba 1 – (jcex012014)

- Iz liste distribucija, odabiramo, preuzimamo i instaliramo onu koja odgovara našem operativnom sistemu

Java SE Development Kit 8u25

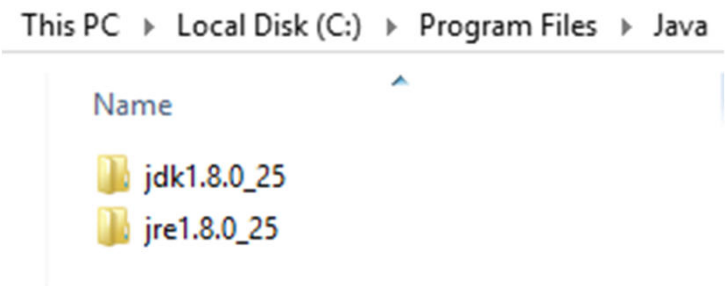
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux x86	135.24 MB	jdk-8u25-linux-i586.rpm
Linux x86	154.88 MB	jdk-8u25-linux-i586.tar.gz
Linux x64	135.6 MB	jdk-8u25-linux-x64.rpm
Linux x64	153.42 MB	jdk-8u25-linux-x64.tar.gz
Mac OS X x64	209.13 MB	jdk-8u25-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	137.01 MB	jdk-8u25-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	97.14 MB	jdk-8u25-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	137.11 MB	jdk-8u25-solaris-x64.tar.Z
Solaris x64	94.24 MB	jdk-8u25-solaris-x64.tar.gz
Windows x86	157.26 MB	jdk-8u25-windows-i586.exe
Windows x64	169.62 MB	jdk-8u25-windows-x64.exe

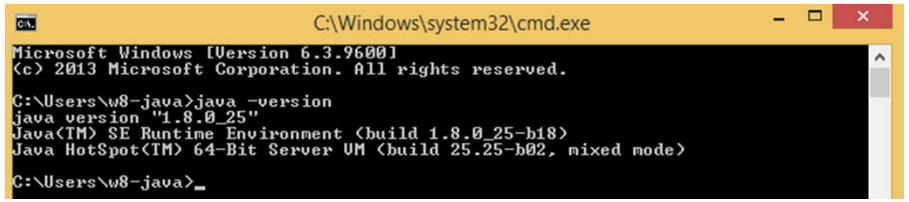
Vežba 1 – (jcex012014)

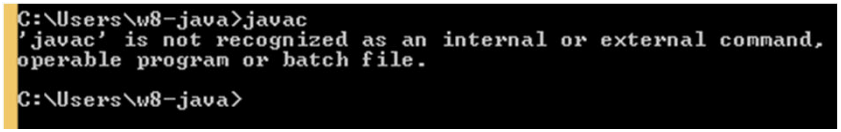
- Instalacija JDK-a ne zahteva naročitu interakciju sa korisnikom, pa će se kompletna procedura instalacije odigrati gotovo automatski
- Nakon završene instalacije, na disku će se naći direktorijum Java sa dva poddirektorijuma
- Direktorijum **jdk1.8.0_25** sa slike je direktorijum jave, poznat i kao Java Home directory



Vežba 1 – (jcex012014)

- Nakon prethodnih koraka, java će biti instalirana na sistemu. Ako otvorimo konzolu, i iz bilo kog dela fajl sistema aktiviramo naredbu **java –version** java će biti prepoznata

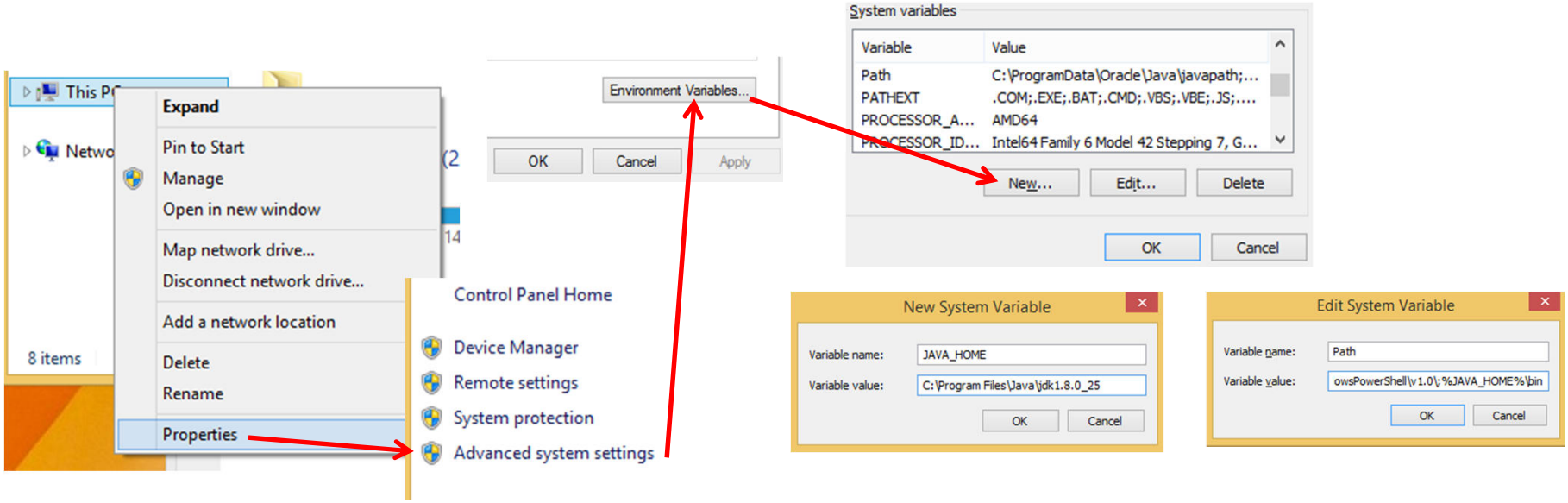




- Ipak, ako na isti način pokušamo da startujemo naredbu **javac**, doći će do greške
- Operativni sistem i mnogi programi će hteti da znaju tačnu putanju do ovog direktorijuma pa je moramo označiti u sistemskim podešavanjima operativnog sistema

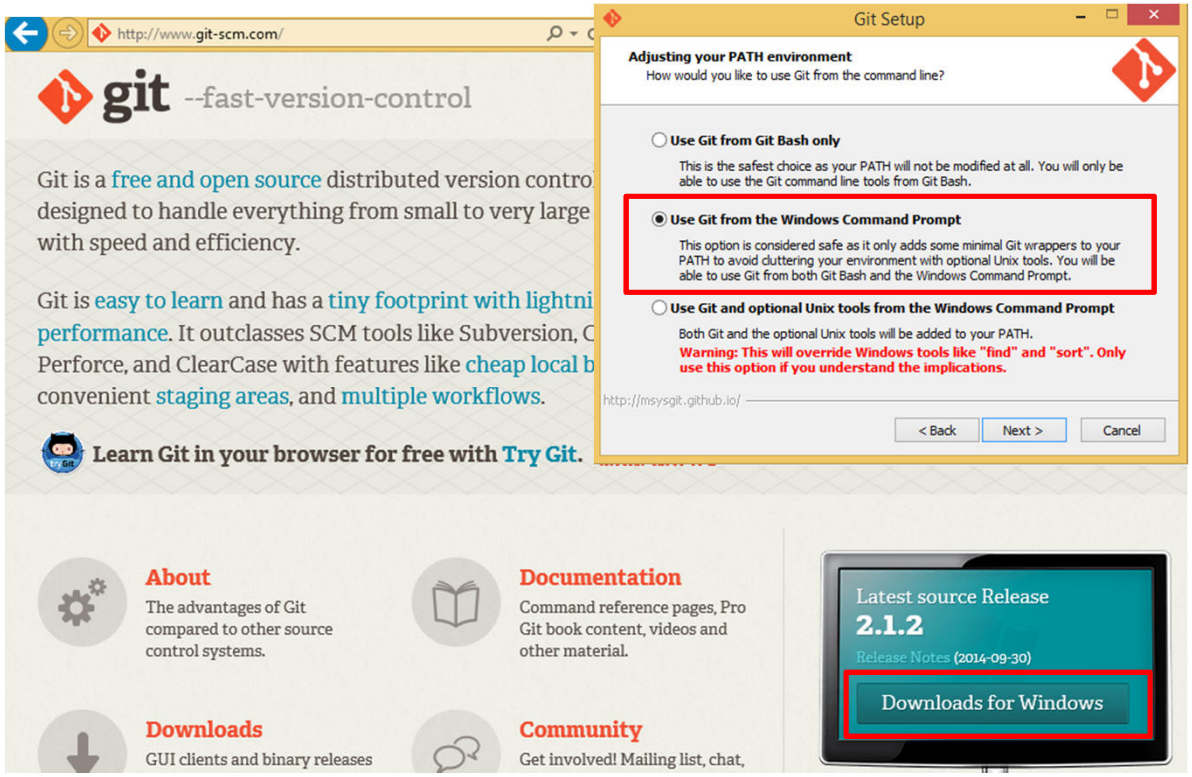
Vežba 1 – (jcex012014)

- Da bi sistem video putanju do java izvršnih fajlova, treba dodati sistemsku promenljivu JAVA_HOME i izmeniti sistemsku promenljivu Path



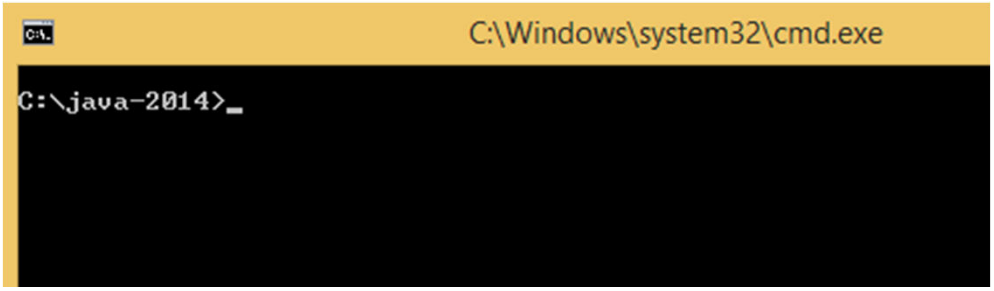
Vežba 1 – (jcex012014)

- Iako nije neophodan, veoma preporučljiv alat za praćenje slajdova/predavanja je Git te i njega treba instalirati



Vežba 1 – (jcex012014)

- Za ovaj i buduće primere, najbolje je kreirati poseban direktorijum na fajl sistemu



Vežba 1 – (jcex012014)

- Primer(e) je takođe moguće preuzeti sa git-a.
- Pre preuzimanja primera sa git-a, treba inicijalizovati git

```
C:\Windows\system32\cmd.exe
C:\>git clone https://vmaric@bitbucket.org/vmaric/java-2014.git_
```

https://vmaric@bitbucket.org/vmaric/java-2014.git

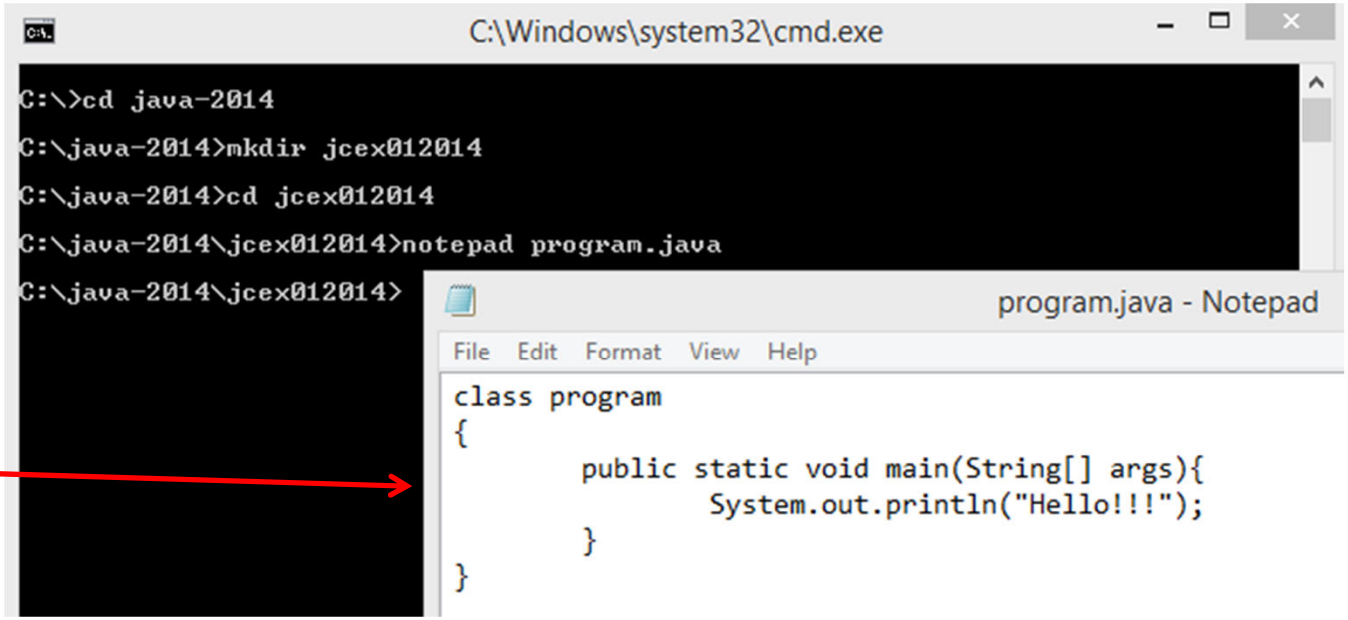
Eventualno kucati: git config --global http.sslVerify false

- Izlaz uspešno izvršene naredbe **git clone**

```
C:\Windows\system32\cmd.exe
C:\>git clone https://vmaric@bitbucket.org/vmaric/java-2014.git
Cloning into 'java-2014'...
remote: Counting objects: 7, done.
Unpacking objects: 100% (7/7), done./4)
remote: Compressing objects: 100% (4/4), done.
remote: Total 7 (delta 0), reused 0 (delta 0)
Checking connectivity... done.
```

Vežba 1 – (jcex012014)

- Za najjednostavniji Java program, dovoljan je jedan Java fajl
- Najbolje za svaki primer kreirati zaseban direktorijum, kako bi rad bio jednostavniji i pregledniji.
- Unutar direktorijuma kreirajte fajl sledećeg naziva i sadržaja



Vežba 1 – (jcex012014)

- Direktorijum i fajl takođe možete kreirati i putem git-a, tako što ćete se pozicionirati u koreni direktorijum za vežbanje (u primeru: java-2014), a zatim otkucati naredbu **git checkout jcex012014**

C:\Windows\system32\cmd.exe

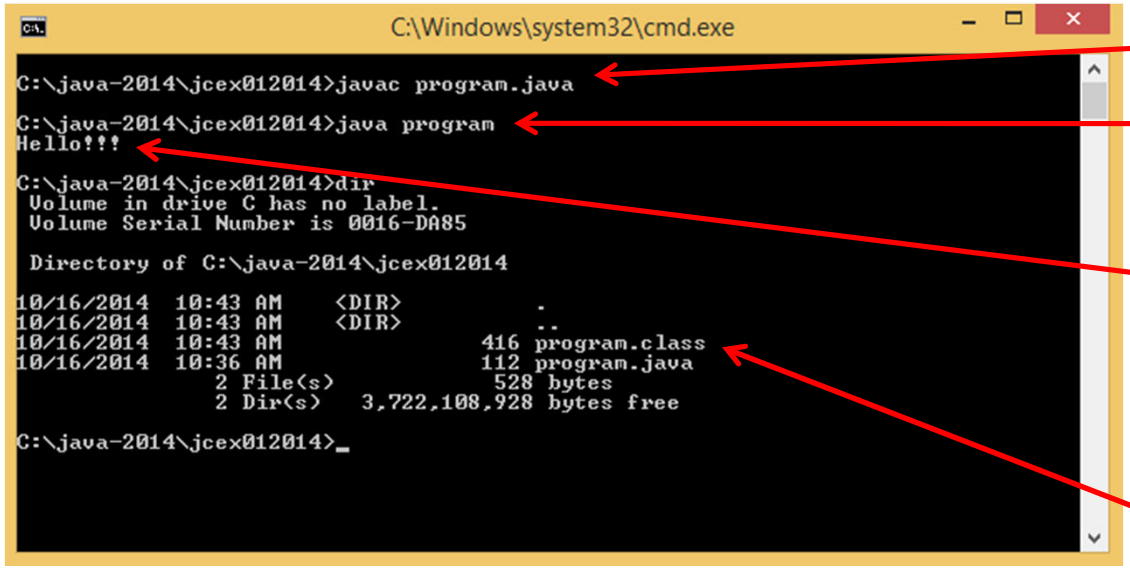
```
C:\java-2014>git checkout jcex012014
Branch jcex012014 set up to track remote branch jcex012014 from origin.
Switched to a new branch 'jcex012014'

C:\java-2014>cd jcex012014

C:\java-2014\jcex012014>dir
 Volume in drive C has no label.
 Volume Serial Number is 0016-DA85

 Directory of C:\java-2014\jcex012014

10/16/2014  10:36 AM    <DIR>          .
10/16/2014  10:36 AM    <DIR>          ..
10/16/2014  10:36 AM               112 program.java
               1 File(s)            112 bytes
               2 Dir(s)   3,722,137,600 bytes free

C:\java-2014\jcex012014>
```

Ovaj koncept se može upotrebiti i za sve ostale numerisane primere, pri čemu se kao parametar checkout opcije navodi kod primera (na primer, ovaj primer ima kod **jcex012014**).

(Procedura neće biti ponovo objašnjavana u narednim primerima)

Vežba 1 – (jcex012014)

- U oba slučaja, krajnji rezultat je pozicioniranje u direktorijumu jcex012014 i postojanje fajla program.java, da bi program bio preveden u bytecode, a zatim startovan potrebno je izvršiti sledeće naredbe respektivno:



javac program.java

java program

Rezultat:

Hello!!!

- Nakon izvršenih linija u direktorijumu će biti kreiran fajl **program.class**

Java edicije

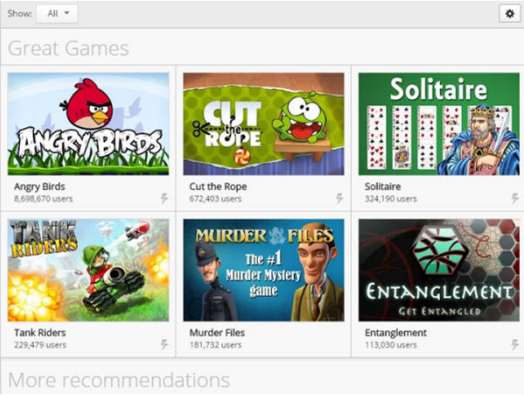
- Iako Java nije platformski zavisna, ipak postoje različite edicije i oblici njenog izvršavanja.
  - **Standard Edition**
  - **Enterprise Edition**
  - **Micro Edition**

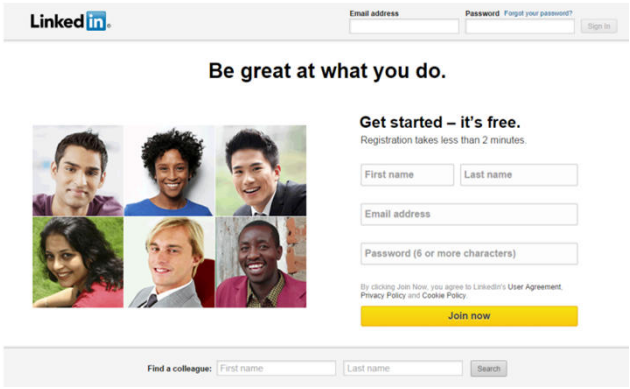
- Ove edicije imaju istu osnovu, ali se razlikuju po tome što svaka ima dodatni specijalizovani set biblioteka za određenu oblast.

- **Standard Edition** se koristi za kreiranje različitih vrsta desktop aplikacija. **Enterprise Edition** je namenjen kreiranju robusnih serverskih aplikacija koje uključuju i XML Web servise, distribuirane Web aplikacije i slično. **Micro Edition** je namenjena proizvodnji aplikacija za mobilne telefone i ostale ručne i pokretne uređaje, za koje se pretpostavlja da nemaju snagu izvršavanja analognu desktop i serverskim platformama

Java tehnologije

- Šta sve možemo napraviti pomoću Jave?
- Multiplatformske desktop aplikacije, web aplikacije, aplikacije za mobilne uređaje, RIA web aplikacije...



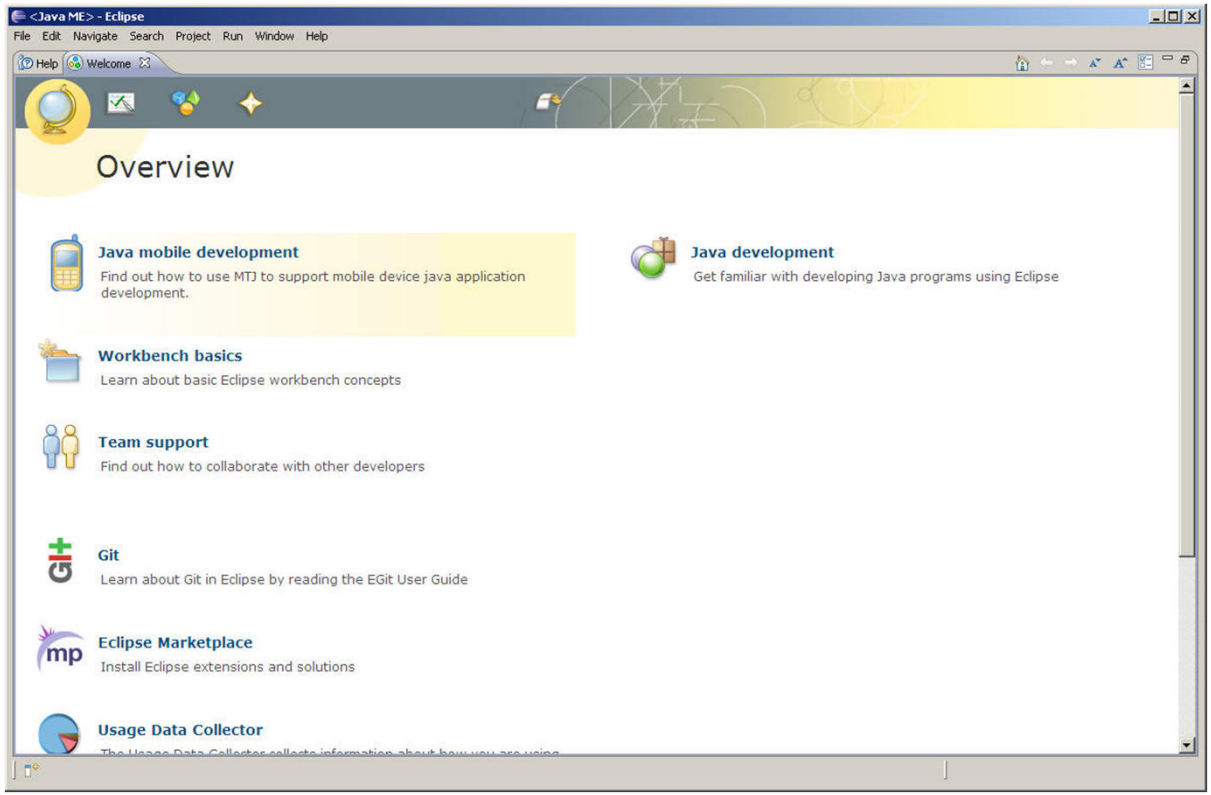


Okruženja za rad u Javi

- U primeru koji je urađen na prethodnim slajdovima, za kreiranje Java programa, koristi se notepad. Ali, kao i za ostale programske jezike, i za Javu postoji više okruženja za programiranje na tržištu
- Njihova razlika je u ceni (neki su besplatni, a neki komercijalni) i u opcijama.

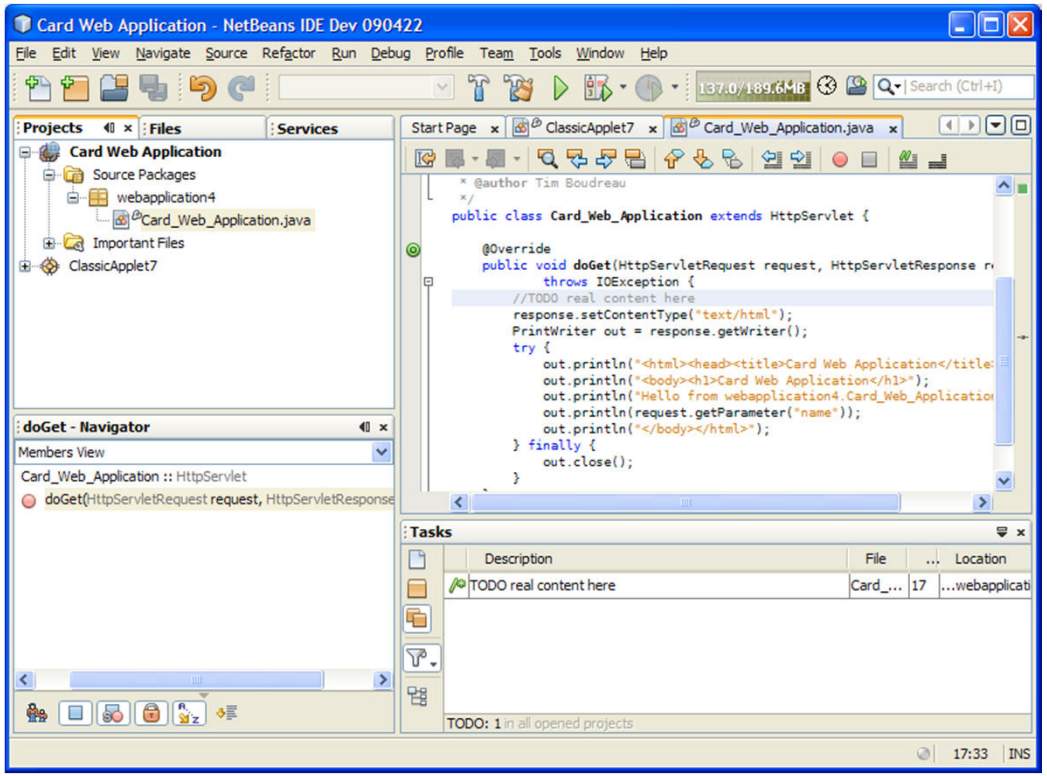
Eclipse (http://www.eclipse.org/)

- Eclipse je najpopularnije okruženje za rad u Javi i drugim jezicima
- Ima izuzetno veliku podršku zajednice, i najlakše je pronaći rešenje za problem nastao u ovom programu
- Eclipse je besplatan



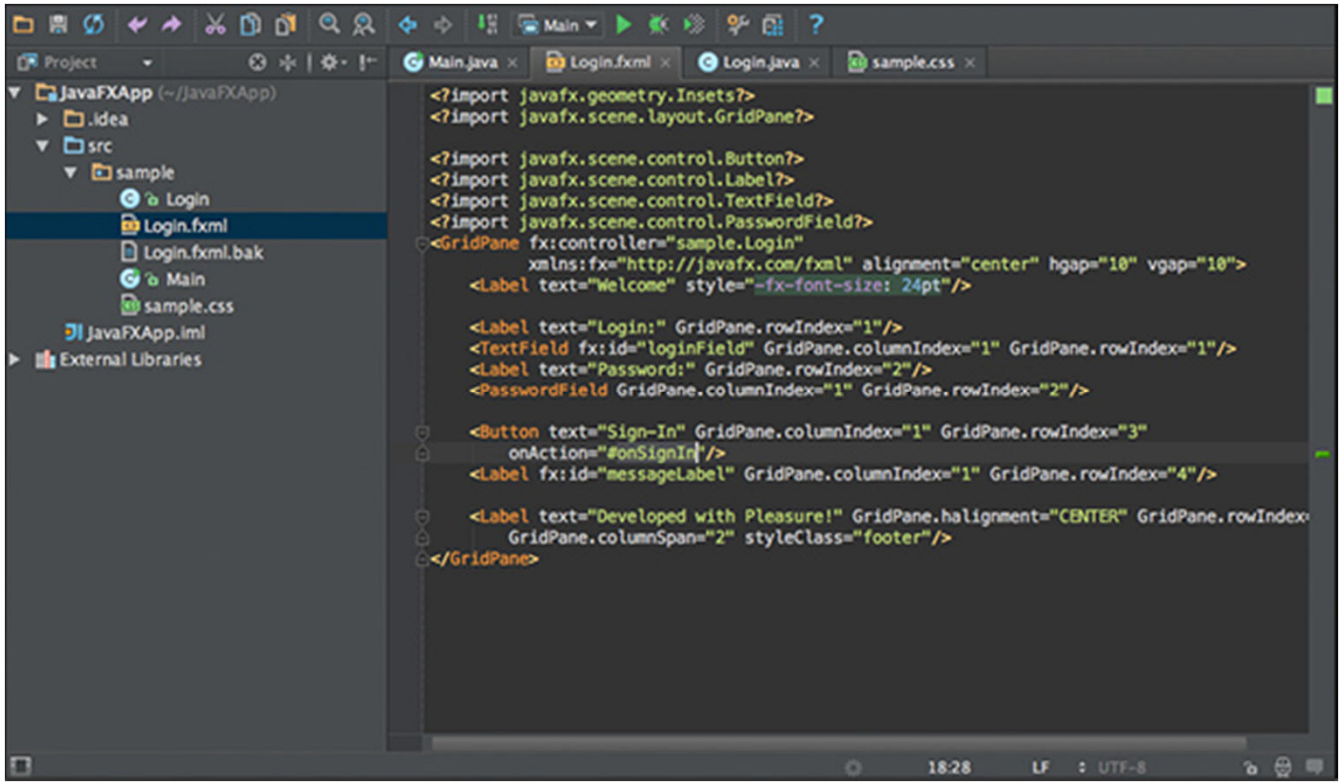
NetBeans (https://netbeans.org/)

- NetBeans je takođe veoma popularno Java okruženje sa velikom podrškom zajednice
  - Za razliku od Eclipse-a, NetBeans je Oracle-ov direktan partner, te u tom smislu i oracle preporučuje njegovo korišćenje
- Besplatan je kao i Eclipse, i podržava i druge jezike osim Jave



IntelliJ IDEA (https://www.jetbrains.com/idea/)

- Intellij IDEA je popularan alat za Java programiranje
  - Postoji u komercijalnoj i besplatnoj varijanti
  - Veoma je popularan, ali ima nešto slabiju podršku zajednice od ostalih okruženja



Prateće tehnologije

- Java je programski jezik, ali, kao što smo naveli, podrazumeva i razne prateće tehnologije
- U izgradnji jedne Java aplikacije, najčešće ćemo se koristiti različitim tehnologijama, koje nekada jesu, ali nekada i nisu blisko vezane za Javu
- Koje ćemo tehnologije koristiti u izgradnji aplikacije, zavisi od tipa aplikacije, i njene arhitekture
- Ipak, i pored arhitektonske raznolikosti aplikacija, najverovatnije da će naša aplikacija završiti sa nekim od prepoznatljivih tehnoloških šablona

Prateće tehnologije i okviri

Konzolne aplikacije

- Java Spring Framework

Desktop aplikacije

- Swing
- LibGdx
- PlayN

Web aplikacije

- Java Spring framework
- Java Server Faces
- GWT

Web / app serveri

GlassFish

JBoss

Apache Tomcat

jetty://

Baze podataka i or mapiranje

OR mapiranje

MyBatis

HIBERNATE

Baze podataka

ORACLE

PostgreSQL

MySQL

redis

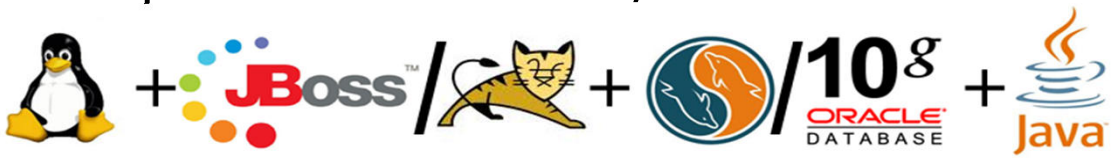
Nismo sami

- Java nije jedina tehnologija kojom se mogu kreirati aplikacije. Na sledećoj listi su kombinacije različitih Java i ne-Java (web) tehnologija.

Linux-Apache-PHP-MySql

Windows Server-IIS-ASP-MS SQL

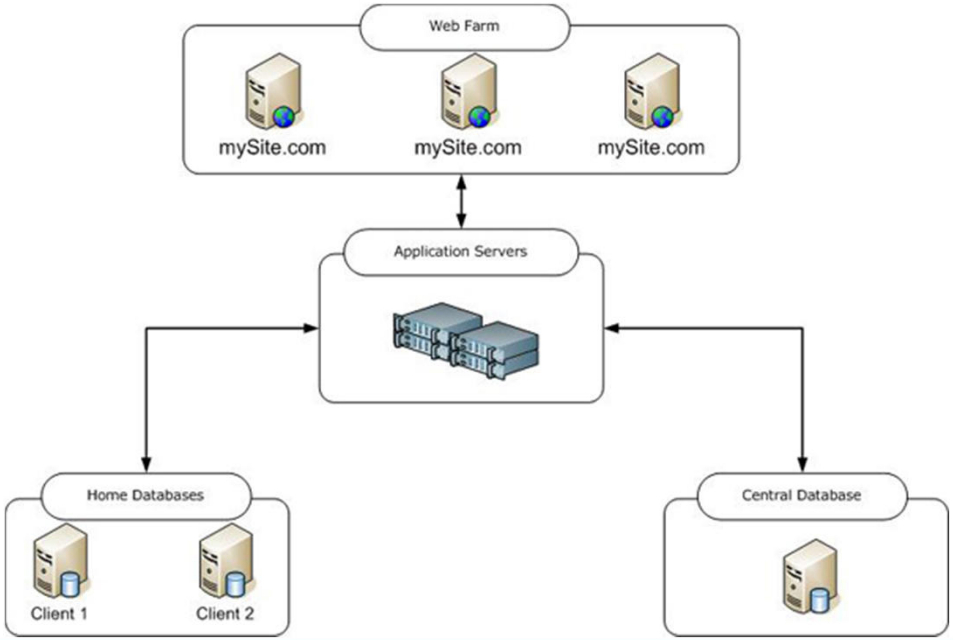
Lunux-Apache-TomCat-JSP/servlet

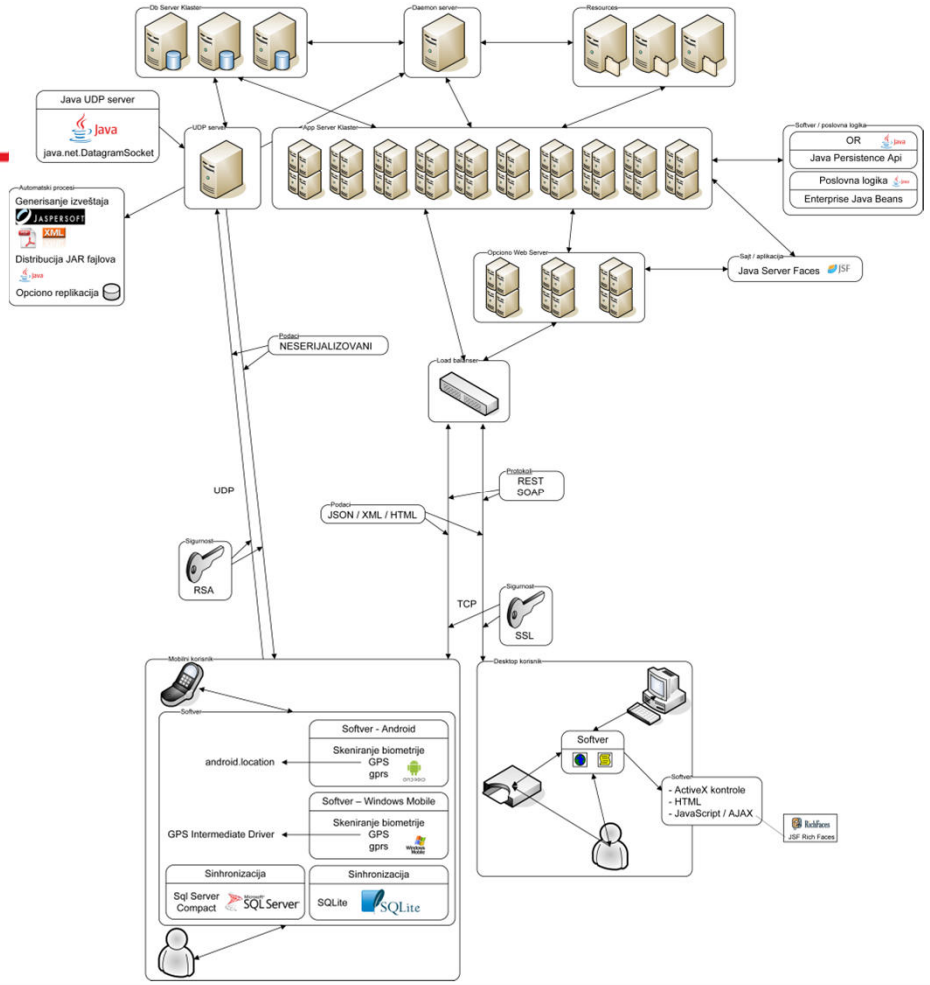


Vertikalna arhitektura aplikacije

Vertikalna arhitektura aplikacije

Horizontalna arhitektura aplikacije





LINKgroup

Vežba 2 – (jcex022014)

- Pokušajmo da napravimo jednostavan java program koji sabira dva broja i prikazuje rezultat na izlazu
- Očekivani izlaz programa:

```
C:\Windows\s
C:\java-2014\jcex022014>java Add
5

C:\java-2014\jcex022014>_
```

- *Podsetimo se još jednom da do programa možemo doći kucanjem konzolnih naredbi:*

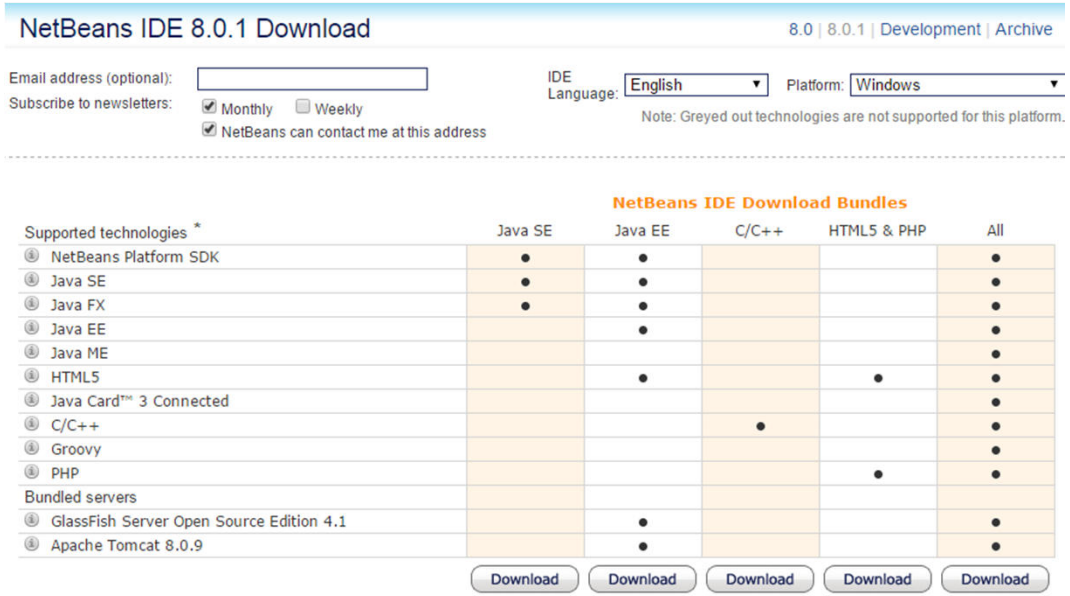
  git checkout jcex022014

  cd jcex022014

Podešavanje i upotreba okruženja

- Na prethodnim slajdovima, prikazana su neka od okruženja za kreiranje Java aplikacija
  - U našoj nastavi, biće korišćen program NetBeans
  - NetBeans je popularno okruženje, ne samo za Javu, već i za druge jezike
- Na slajdu se vidi da postoji više verzija ovog programa

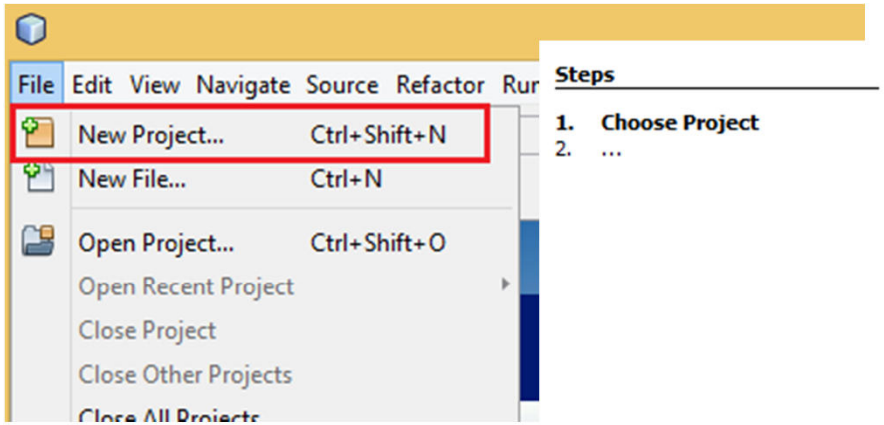
NetBeans se može preuzeti sa adrese:
https://netbeans.org/dowloads/

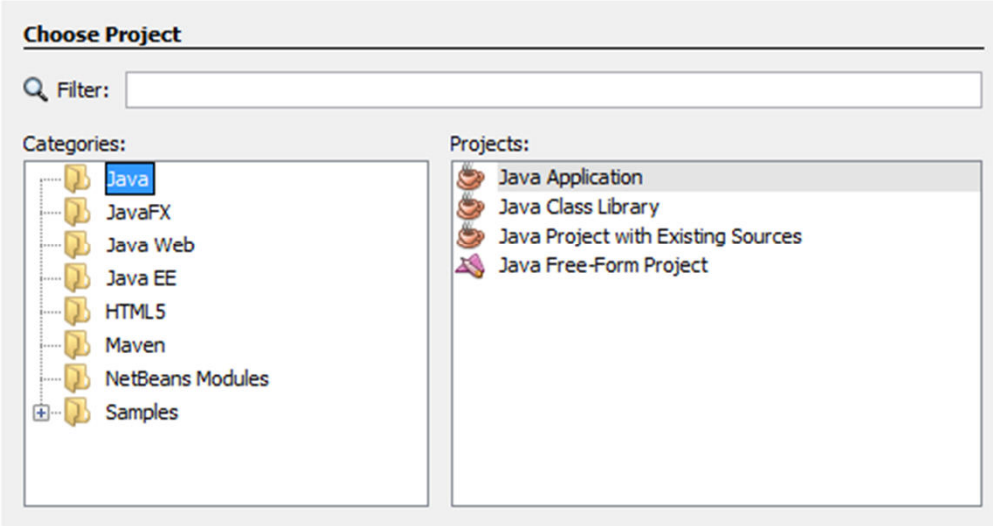


- Za početak možemo skinuti samo Java SE, ali će nam kasnije svakako trebati i Java EE (pa nije zgoreg odmah skinuti tu verziju / paket)

Kreiranje NetBeans projekta

- NetBeans svaki Java program tretira kao jedan **projekat**, pa da bi kreirali program sličan onome iz prethodnog primera, treba da kreiramo novi projekat

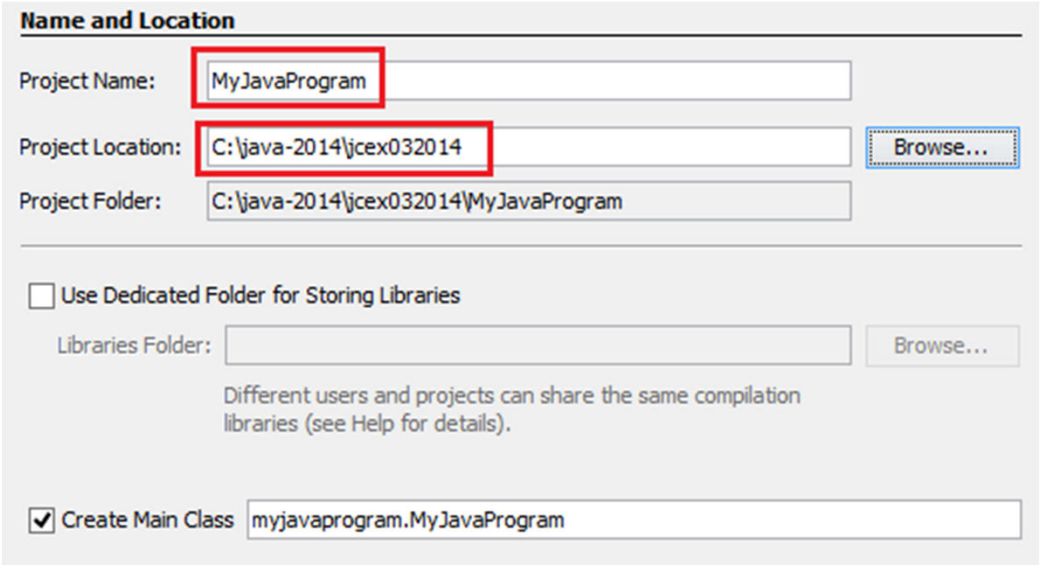




Kreiranje NetBeans projekta

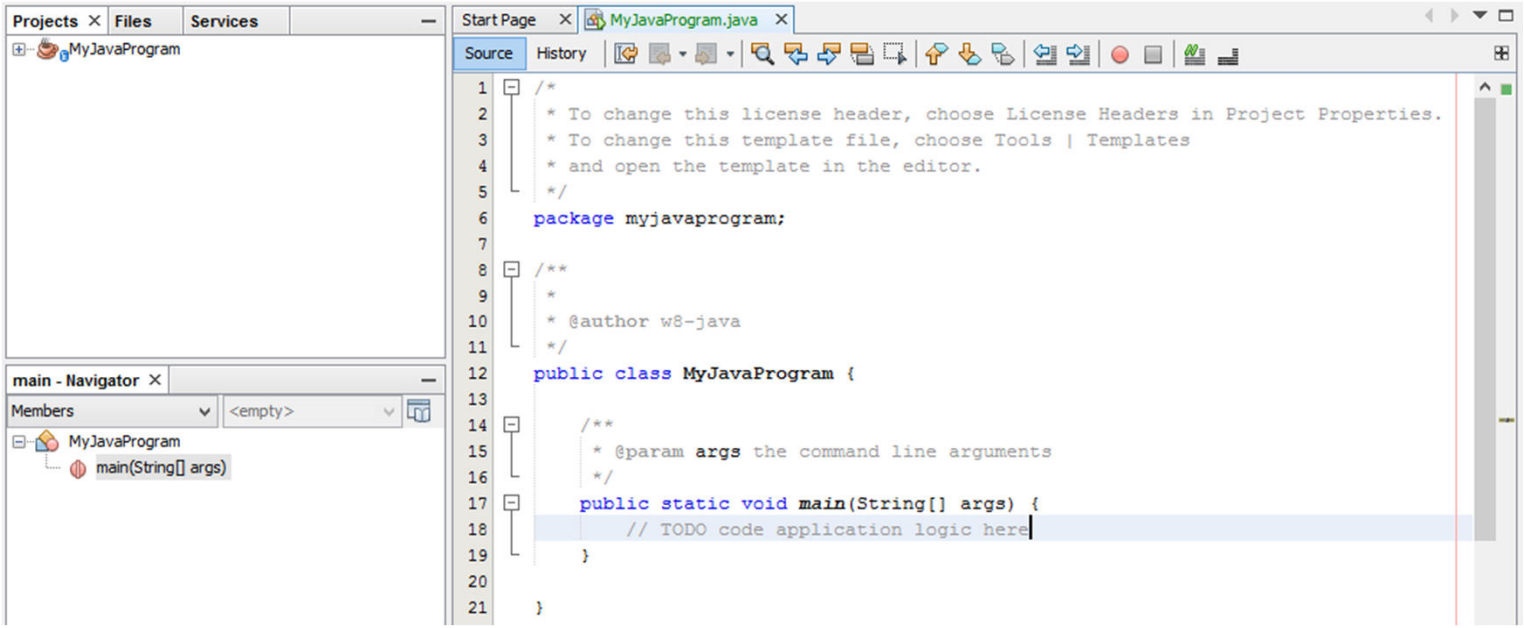
- Prilikom kreiranja projekta, NetBeans ima nekoliko pitanja za nas, na koje još uvek nemamo odgovore

- Kako će se zvati projekat?
- Gde će se nalaziti projekat?
- Da li da bude kreirana glavna klasa aplikacije i kako da se zove?



Pisanje i pokretanje programa

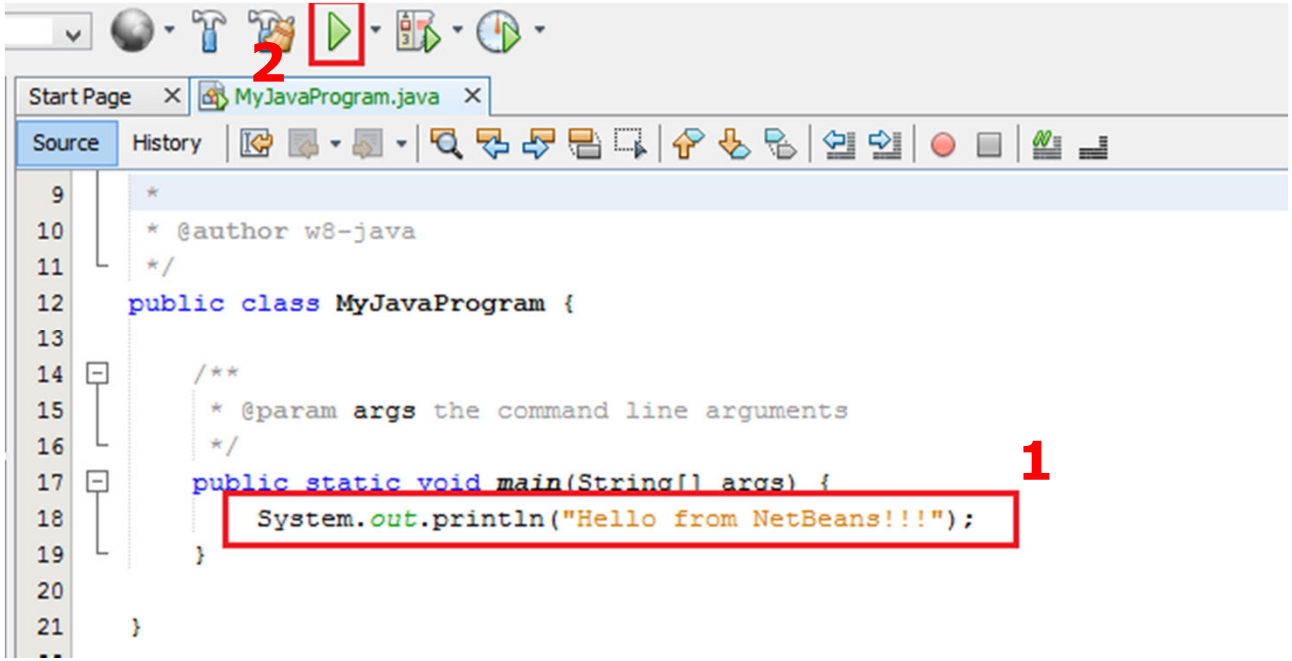
- Okruženje zatim kreira fajl sa sadržajem (i nazivom), sličnim onome koji smo sami napravili u prethodnom primeru:



Pisanje i pokretanje programa

- [1] U prostor između vitičastih zagrada, treba dodati navedeni kod kako bi bio kreiran program sličan onome iz prethodnog primera.

- [2] Zatim treba pritisnuti zeleni trougao (strelicu) levim tasterom miša



Izlaz programa (jcex032014)

- Ispravno urađen postupak, treba da ima sledeći izlaz

Start Page | MyJavaProgram.java

Source | History

```
9    *
10   * @author w8-java
11   */
12  public class MyJavaProgram {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          System.out.println("Hello from NetBeans!!!");
```

Output

java-2014 - C:\java-2014 | MyJavaProgram (run)

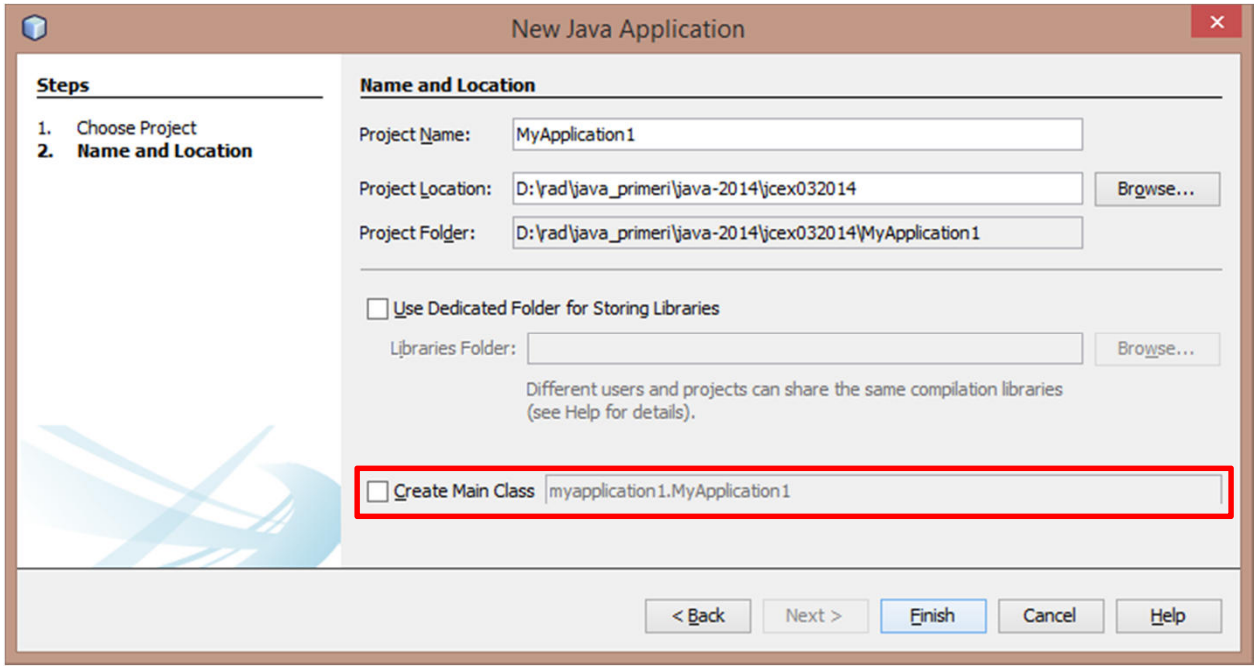
```
run:
Hello from NetBeans!!!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Vežba 3 (jcex032014)

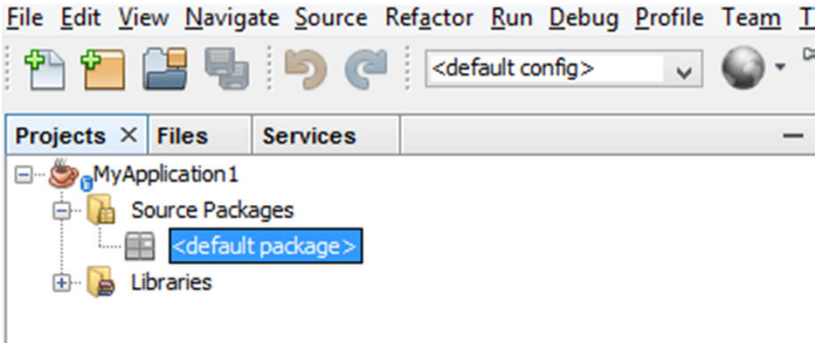
- U prethodnom primeru, NetBeans je dosta toga uradio za nas. Pokušajmo u nastavku da analiziramo urađeno, i uradimo neke stvari samostalno

Primer bez automatskog kreiranja klase

- Pokušajmo da napravimo još jedan projekat, ali ovaj put bez automatskog kreiranja glavne klase aplikacije

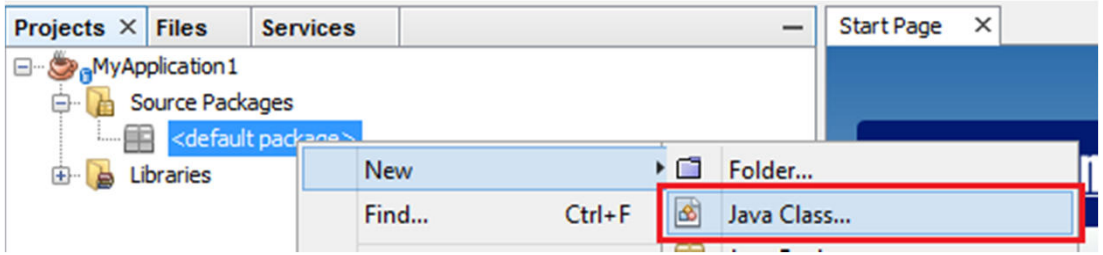


Primer bez automatskog kreiranja klase



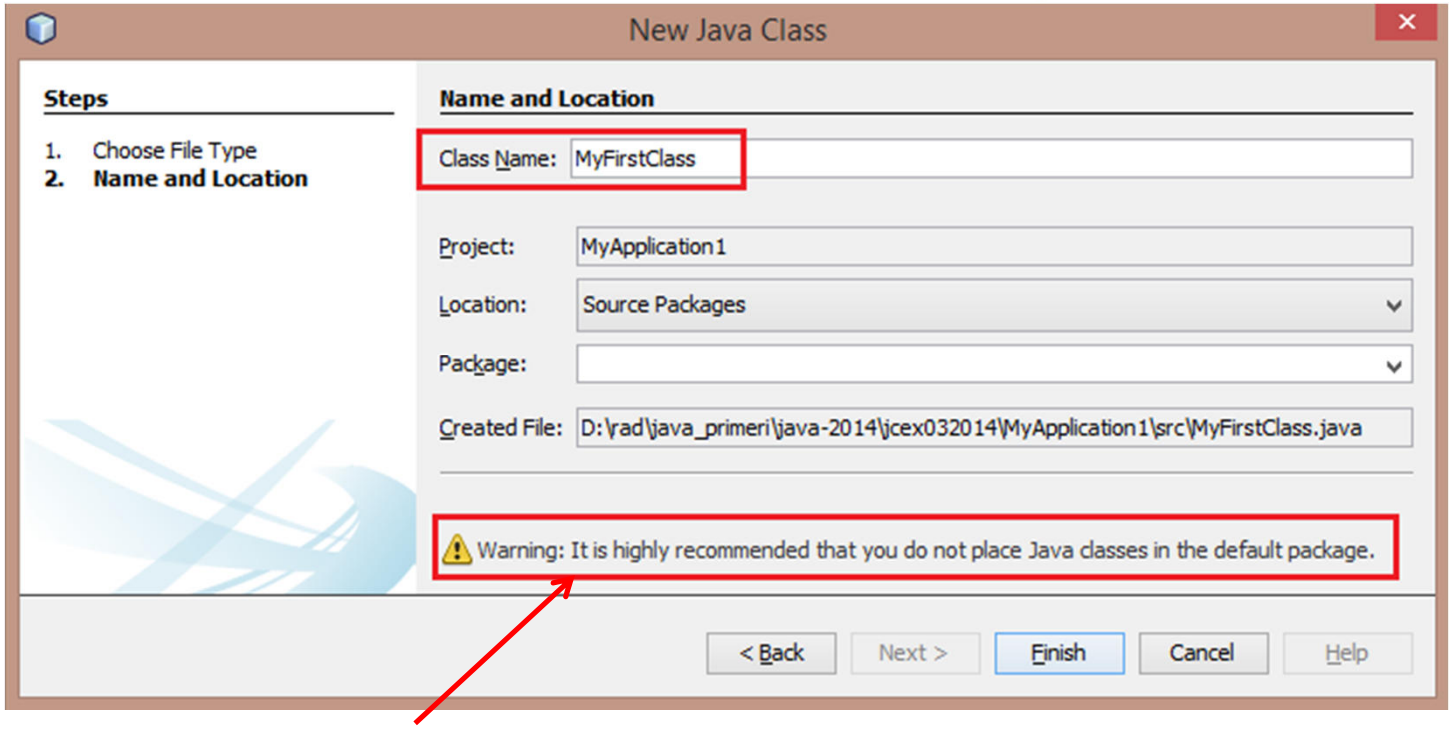
- Projekat sada nema predefinisane klase – prazan je
- Ako bi pokušali da ga startujemo, NetBeans bi nam tražio lokaciju glavne klase projekta

- Iako ovaj program možemo prevesti, da bi on mogao da radi bilo šta, moramo prvo kreirati bar jednu klasu



Primer bez automatskog kreiranja klase

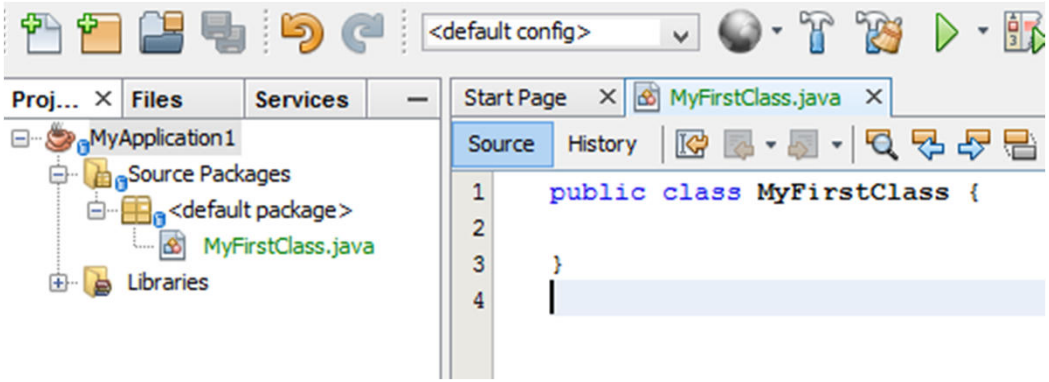
- U otvorenom panelu, treba uneti proizvoljan naziv klase (na primer MyFirstClass)

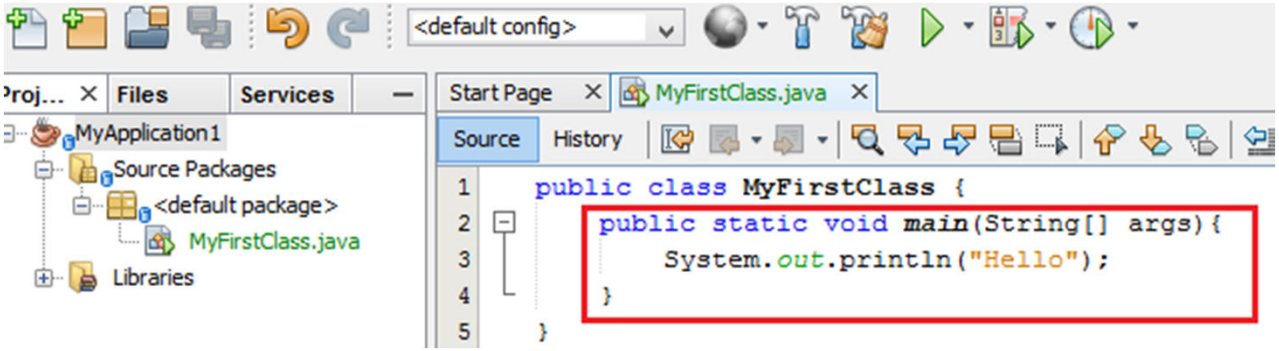


NetBeans neće biti „zadovoljan“ ako klasu ne smestimo u neki paket

Primer bez automatskog kreiranja klase

- Sadržaj kreiranog fajla je prepoznatljiv iz prethodnih primera, osim što fail **main** metod

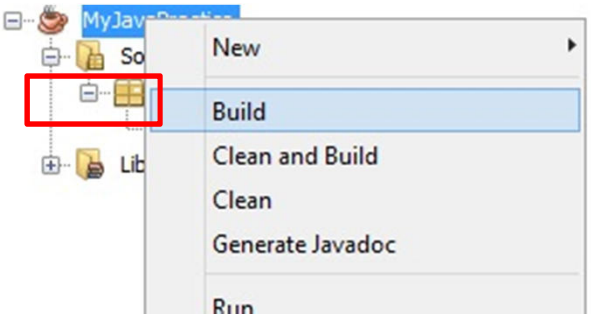




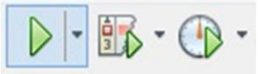
Dodajmo isti sadržaj kakav smo dodavali i u prethodnim lekcijama

Prevođenje, izvršavanje i distribuiranje

Ako želimo samo da prevedemo, odnosno kompajliramo projekat potrebno je desnim klikom ići na naziv projekta na levoj strani ekrana i iz padajućeg menija odabrati Build

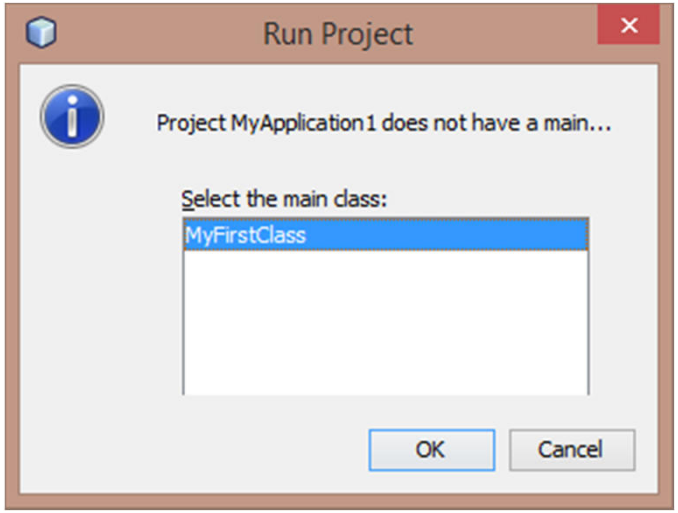


Ako želimo i prevođenje i pokretanje, i to možemo učiniti na sledeći način: iz trake sa alatima u gornjem delu prozora potrebno je odabrati dugme Run Project, ili iskoristiti prečicu sa tastature F6.



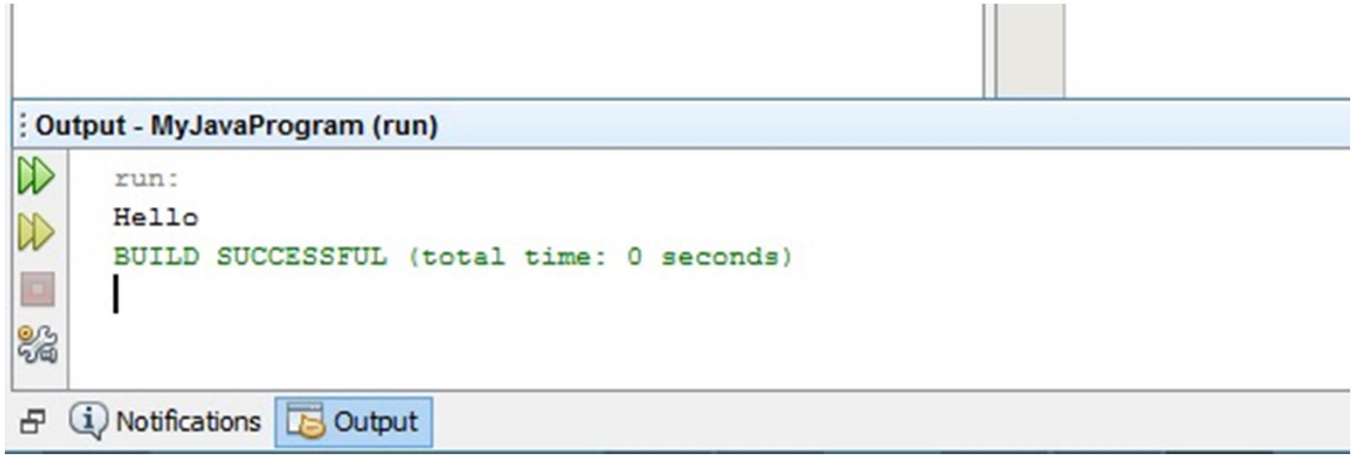
Prevođenje, izvršavanje i distribuiranje

- S obzirom na to da nemamo definisanu glavnu klasu, okruženje prikazuje prozor u kome je potrebno odabrati neku od dostupnih klasa koja će se koristiti kao glavna klasa.
- Pošto je, u našem slučaju, u projektu prisutna samo jedna klasa, MyFirstClass, biramo nju.



Prevođenje, izvršavanje i distribuiranje

- Konačan rezultat prikazuje se u Output prozoru, na dnu NetBeansa.



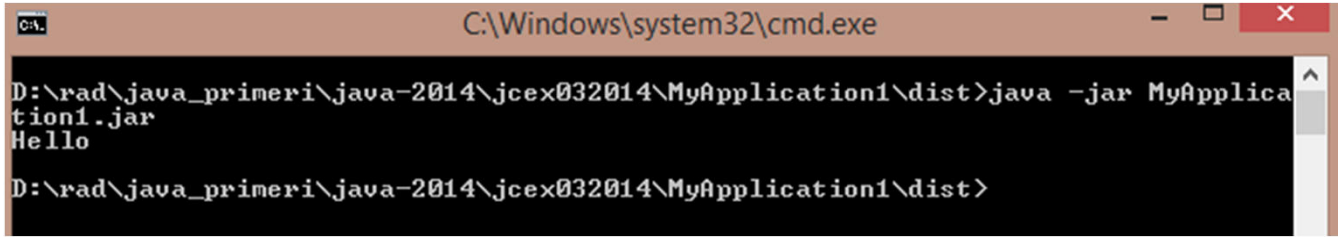
Prevođenje, izvršavanje i distribuiranje

- aplikacija koju smo kreirali, može se pokrenuti i izvan NetBeans integrisanog razvojnog okruženja. Potrebno je otvoriti Command Prompt i pristupiti \dist folderu o okviru foldera u koji je smešten naš projekat. U našem slučaju, to je:

  putanja do projekta\MyApplication1\dist

- U komandu liniju je zatim potrebno upisati:

  java -jar MyApplication1.jar

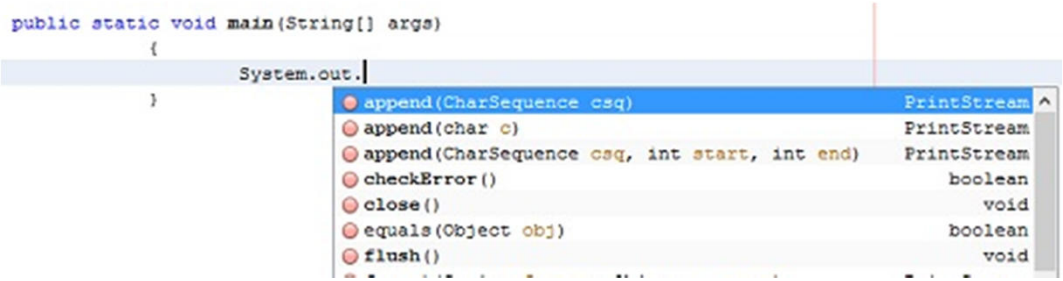


Korišćenje NetBeans razvojnog okruženja

- Svako razvojno okruženje, pored toga što postoji da bi olakšalo prevođenje, izvršavanje i distribuciju same aplikacije, postoji i da bi maksimalno olakšalo sam proces kodiranja
- NetBeans, nudi veliki broj mogućnosti za pomoć programeru prilikom rada
  - **Smart Code Completion**
  - **Code formating**
  - **Code coloring and marking**
  - **Debug helpers**
  - **Javadoc / hints**

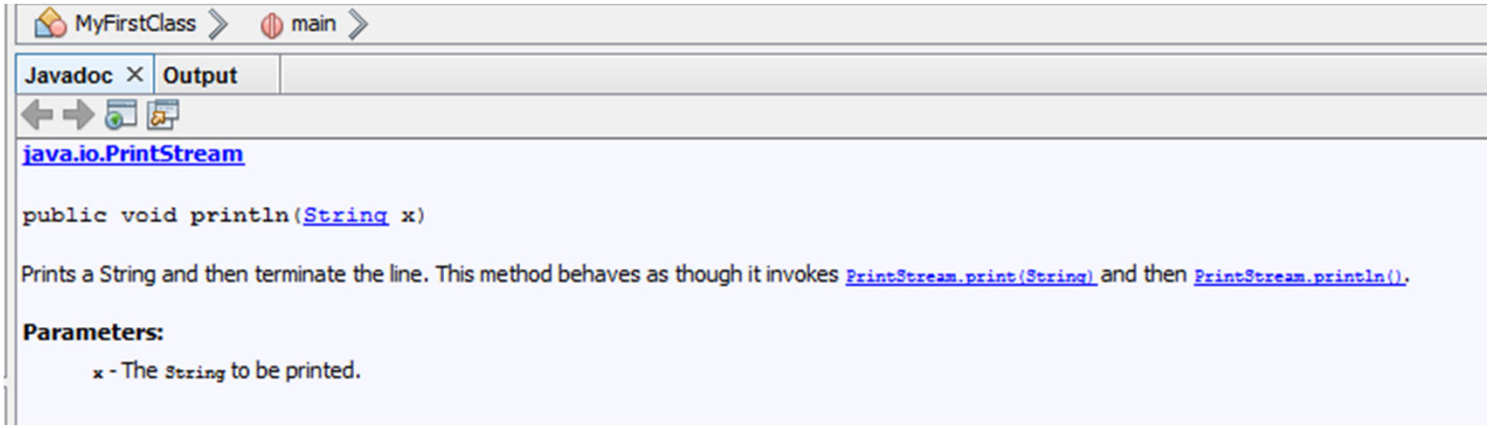
Smart Code Completion

- Ogleda se u tome da je razvojno okruženje u stanju da određene sekvence koda generiše i tako nam uštedi vreme prilikom kodiranja.
- Primetili ste verovatno prilikom pisanja primera iz prethodne lekcije koji prikazuje poruku Hello na ekranu, da u trenutku kada ukucate tačku posle reči System ili out na ekranu dobijate određeni padajući menu. Ovo je u stvari Smart Code Completion.



Javadoc

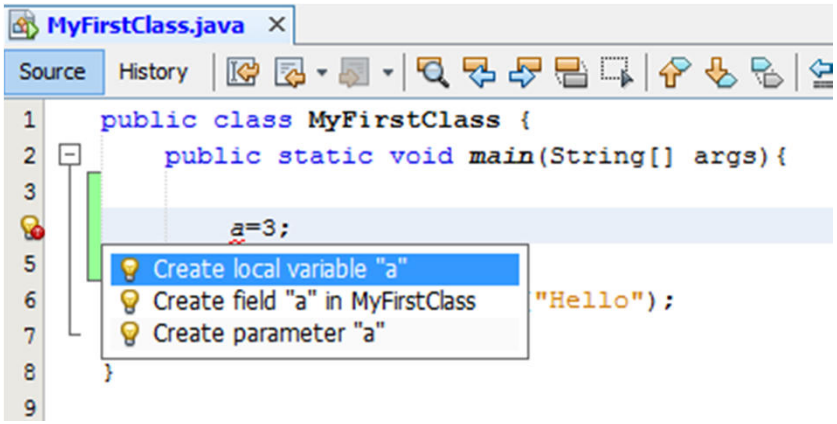
- Javadoc predstavlja način da se omogući automatsko generisanje web stranica sa dokumentacijom projekta. Takođe, sve klase koje postoje u okviru JDK-a objašnjene su pomoću javadoca.



Za aktivaciju javadoc prikaza, potrebno je iz padajućeg menija **Window** odabrati **IDE Tools**, a zatim **Javadoc Documentation**.

Korišćenja saveta (Hints)

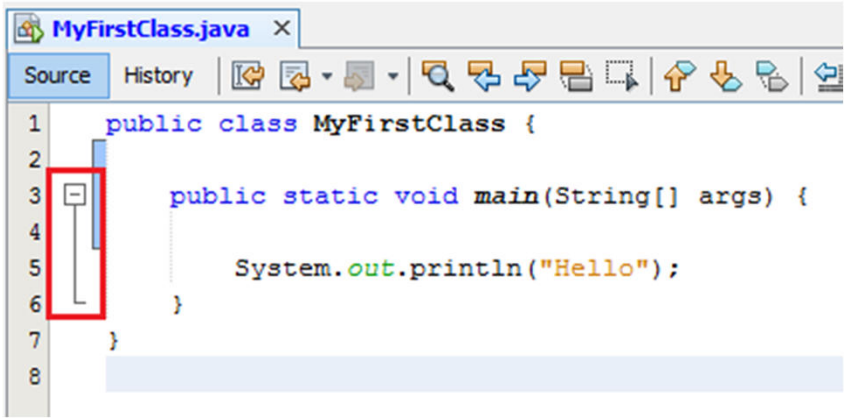
- U toku kucanja koda u NetBeansu, Java editor automatski, konstantno proverava unos i sugeriše kako da ispravimo greške ili učinimo kod boljim. Saveti se prikazuju u okviru leve margine editora. Ukoliko, na primer, napišete sledeću liniju koda:



Formatiranje koda

- Da bismo na najlakši način izvršili formatiranje koda, možemo iskoristiti prečicu sa tastature **Alt-Shift-F** ili iz menija odabrati **Source > Format**.

  - Grupisanje koda znatno olakšava pregled velike količine koda, a samim tim olakšava održavanje. Blokovi koda koji se mogu skupiti, odnosno raširiti obeleženi su sivim linijama sa znakovima plus/minus pored leve margine editora.



Markiranje i bojenje koda

- NetBeans predstavlja različite elemente koda, različitim bojama
- Kao dodatak bojenju, okruženje vrši i markiranje sličnih elemenata
- Opcija markiranja se može posmatrati kao određena opcija pretrage, jer markira svako pojavljivanje određenog elementa u kodu, pa samim tim olakšava njegovo pronalaženje

```
7   /**
8    *
9    *
10   */
11  public class Student extends Teacher {
12  // fields
13      private int id;
14      private String name;
15      private int age;
16      private int unusedField; //unused field
17
18      public Student(int id, int age, String name) {
19          this.id = id;
20          this.name = name;
21          this.age = age;
22      }
23  @Deprecated
24      public int getAge() {
25          return age;
26      }
```

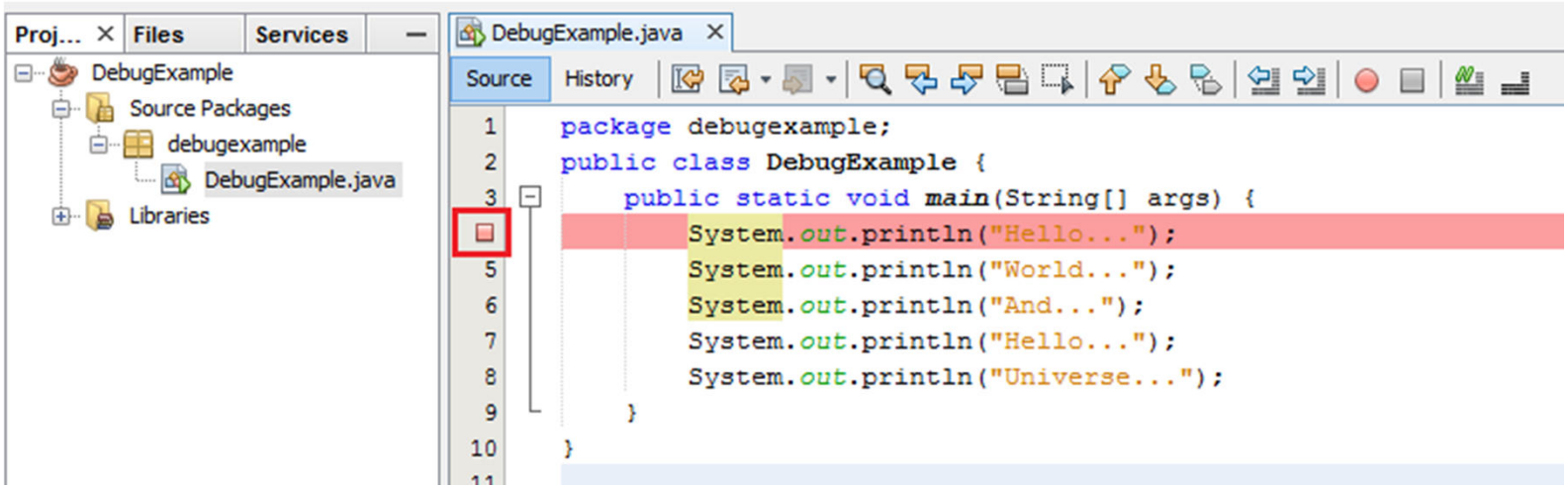
Testiranje i debagovanje aplikacije

- Jedna od bitnijih osobina okruženja je i rukovanje greškama, odnosno, njihovo prepoznavanje. Sistemi prepoznavanja grešaka umnogome olakšavaju i ubrzavaju rad i sastavni su deo svakog ozbiljnijeg programerskog okruženja.
- Korišćenjem dibagera program se može izvršiti liniju po liniju i na taj način pratiti promena vrednosti određenih parametara, u cilju pronalaska logičkih grešaka u programu.
- Tačka prekida predstavlja oznaku u izvornom kodu koja se postavlja u okviru leve margine editora i označava da se izvršavanje koda pauzira u liniji za koju je ova tačka definisana.
- Kada se program zaustavi u definisanoj tački, može se izvršiti inspekcija trenutnih vrednosti promenljivih u programu ili nastaviti izvršavanje programa, liniju po liniju.

```
3   public class TestClass {
4
5       public static void main(String[] args) {
6
        System.out.println("hello 1");
8       System.out.println("hello 2");
9       }
10  }
11
```

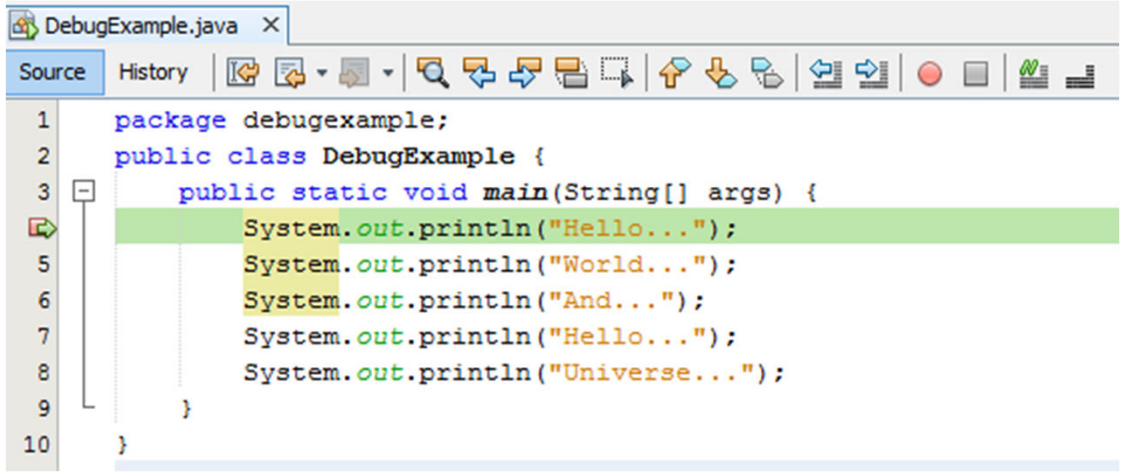
Vežba 4 (jcex042014)

- Za vežbu, prvo treba kreirati Java Application projekat, isti kao i u prethodnim primerima, a zatim dodati nekoliko linija u main metod.
- Na levoj margini koda, treba pritisnuti levi taster miša na poziciji na kojoj želimo da zaustavimo program



Vežba 4 (jcex042014)

- Da bi smo sada pokrenuli aplikaciju u Debug Modu, treba kliknuti na sledeći taster: i aplikacija će se zaustaviti na naznačenoj poziciji:



Vežba 4 (jcex042014)

- Jednom, kada se izvršavanje programa zaustavi, dalja kontrola se može obavljati sledećim kontrolama:

Step Over – izvršava jednu liniju koda; ako je linija poziv metode, izvršava metodu bez ulaska u kod metode

Step Into – izvršava jednu liniju koda; ako je linija koda poziv metode, izvršavanje ulazi u kod metode i staje na prvoj liniji u okviru metode

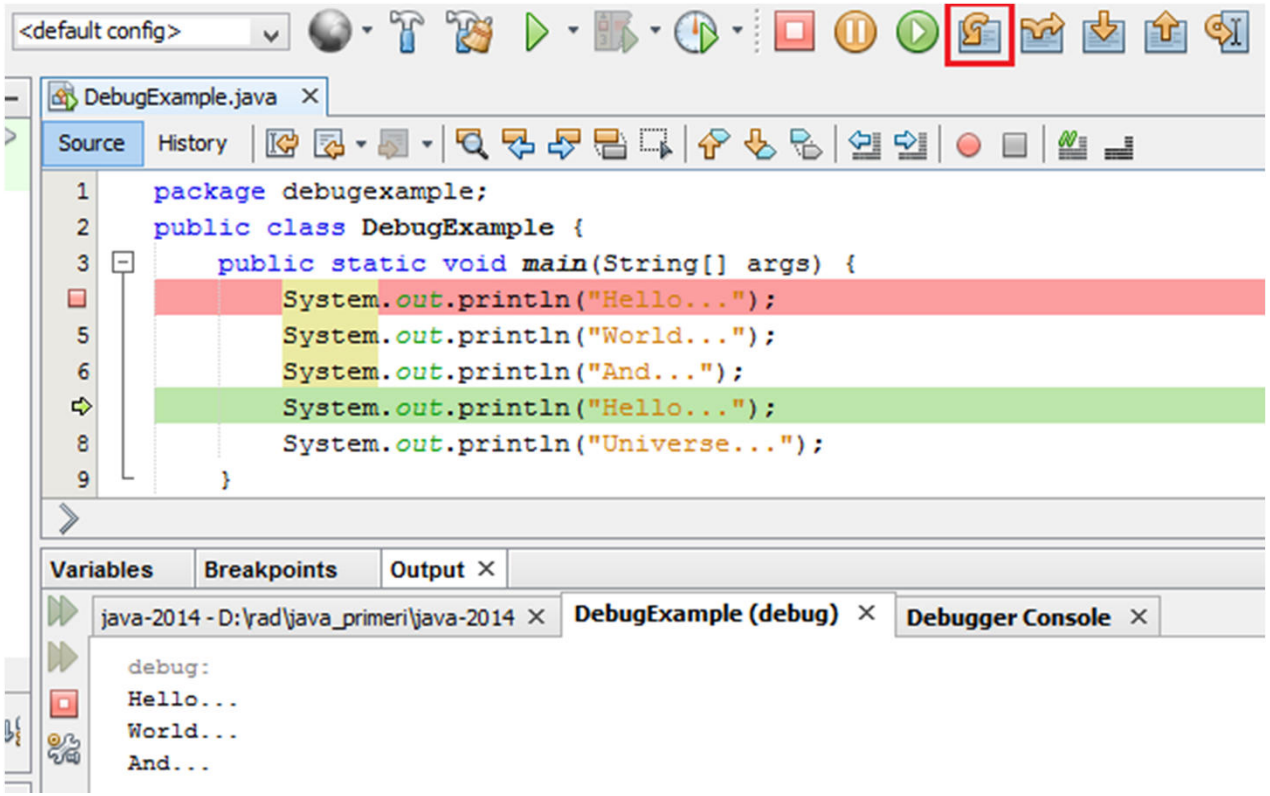
Step Out – izvršava jednu liniju koda; ipak, ako je linija koda u okviru metode, izvršava se kompletan kod metode i izvršavanje se zaustavlja na mestu sa kojeg je metoda pozvana

Run to Cursor – izvršava program do linije gde je lociran kursor

Continue – izvršava program do sledeće tačke prekida, ako postoji ili do kraja programa

Vežba 4 (jcex042014)

- Krećite se kroz program korak po korak i pokušajte da ispratite rezultat na izlazu aplikacije (u Output prozoru)



Sintaksa jezika Java

- Sintaksa bilo kog jezika predstavlja skup pravila koje korisnik jezika mora da ispoštuje kako bi ostali korisnici tog jezika mogli da ga razumeju

- Programski jezik Java, kao i većina drugih modernih jezika, najveći deo sintakse preuzima iz jezika C/C++. Ljudi koji su kreirali Javu izabrali su C++ kao uzor, zato što je ovaj jezik predstavljao jedan od najrasprostranjenijih u tom trenutku, i na taj način su mnogim programerima olakšali učenje ovog jezika

- C++ i Java su sintaksno veoma slični. Ipak, Java omogućava nešto veću slobodu u pisanju. Tačnije, ne obavezuje nas da se bavimo stvarima koje nisu u kontekstu projekta, a moraju se ispoštovati da bi program funkcionisao, kao što je rukovanje memorijom, i slično

Sastavni delovi Java programa

Svaki program u Javi sastoji se od sledećih elemenata:

- **linije**
- **blokovi**
- **identifikatori**
- **komentari**
- **prazni prostori**

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

```
{ }
```

```
public class JavaApplication5
```

```
// TODO code application logic here
```

Linije (izjave)

- Linija predstavlja najmanji samostalni deo jednog programa. Svaki Java program sastoji se iz linija koda. Veličina nekog koda se meri u linijama (100 linija koda, 1 245 linija koda itd.). Često pravilo programske sintakse jeste da svaka linija koda mora biti završena oznakom **;**

int x=1;

Nije validna linija:

int x=1

Linije (izjave)

- U posebnim slučajevima, oznaka **;** se ne mora pojaviti na kraju linije.
- Ti slučajevi su specifične naredbe za kontrolu toka ili petlje
- Na primer:

```
if(x==10)
    System.out.println("x is 10");
```

ili:

```
for(int i=0;i<10;i++)
    System.out.println("x is 10");
```

Blokovi koda

- Blokovi su delovi koda oivičeni vitičastim zagradama
- Ovi delovi koda mogu i ne moraju imati sadržaj (ali je druga varijanta besmislena)

Nema baš puno smisla

```
{
    //block 1 - no content
}
{
    System.out.println("Hello from block 2");
}
```

Komentarisanje koda

- Komentari koda omogućavaju da u kodu nešto napišemo, ali da to ne bude izvršeno niti tretirano kao deo programa
- Zašto bi tako nešto uradili?
  - Da bi vodili beleške o delovima programa
  - Da bi olakšali rad našim kolegama
  - Da bi privremeno isključili/uključili neki deo koda

```
// This is a comment
int a = 1;
```

Komentarisanje koda

- Komentari u kodu mogu se označiti na nekoliko načina
- Komentarisanje jedne linije (označava se sa //)

```
//My comment
```

ili

```
int x; //My comment
```

- Komentarisanje više linija (označava se sa /* */)

```
/*
I would say
someting smart here
*/
```

Ili za dokumentaciju:

```
/**
 * For documentation
 */
```

Case sensitivity (osetljivost na mala i velika slova)

- Java je osetljiva na velika i mala slova
- To znači da identifikator koji je napisan malim slovima, ne predstavlja isto što i identifikator napisan velikim slovima
  - ***class Hello* ...** nije isto što i ***class hello ...***
- Pored osetljivosti na velika i mala slova, moramo dakođe paziti da u identifikatore ne **smeštamo prazne karakere** (space), kao ni **specijalne karaktere**, te da ih **ne započinjemo brojevima**

Vežba 5 (jcex052014 Ex1)

- Dat je sledeći kod koji sabira dva broja:

```
int operand1=2;
int operand2=3;
int sum=operand1+operand2;
System.out.println(sum);
```

- Potrebno je napraviti opis koda za dokumentaciju.

Vežba 6 (jcex052014 Ex2)

- Dat je sledeći kod:
- Kod daje sledeći izlaz:

```
int a=2,b=3;
System.out.print("First operand: ");
System.out.println(a);
System.out.print("Second operand: ");
System.out.println(b);
System.out.print("Sum: ");
System.out.println(a+b);
```

First operand: 2
Second operand: 3
Sum: 5

- Potrebno je obezbediti da program na izlazu prikazuje samo rezultat (broj 5), ali da sve linije koda ostanu dostupne za buduću upotrebu.

Zadatak 1

- Ispraviti sledeći program, tako da ne prijavljuje sintaksne greške:

```
public static void main(String[] args) {
    Initializing variables here:
    int x = 25, y = 35, z = 45;
    int[] currentPosition = { x,y,z ;
    System.out.println("Your current position is: ")
    System.out.printf("x:%d,y:%d,z:%d\n",
            currentPosition[0],currentPosition[1],currentPosition[2])
}
```