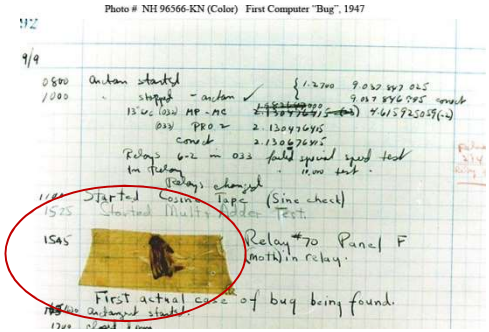
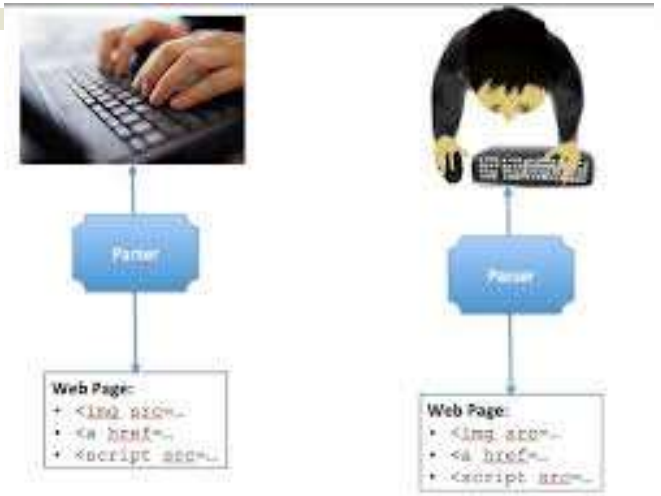
Programiranje





Nastava: prof.dr.sc. Dražena Gašpar

Datum: 06.03.2018.

Osnovni plan nastave iz Programiranja

- Osnovni pojmovi
- Povijesni razvoj programskih jezika
- Programske paradigme
- Algoritamski pristup rješavanju problema
- Tipovi podataka
- Implementiranje imperativne paradigme
- Implementiranje objektne paradigme

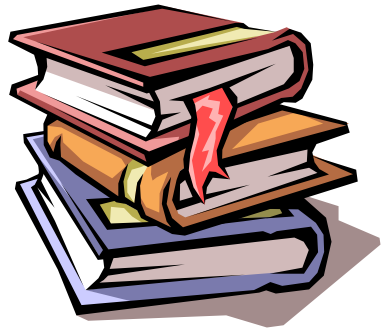
Osnovni plan nastave

- Programski jezik implementiranja programskih paradigmi

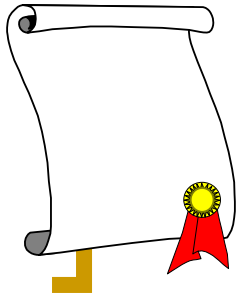
  JAVA

Bazna literatura

1. Dražena Tomić “Osnove programiranja”
2. Dario Sušanj “*Java Programiranje za Internet i World Wide Web*”, Znak Zg
3. Bruce Eckel “*Thinking in Java*”
4. **http://www.oracle.com/technetwork/java/index.html**
5. **http://www.java.com**
6. **www.tutoriali.org**



Način polaganja ispita



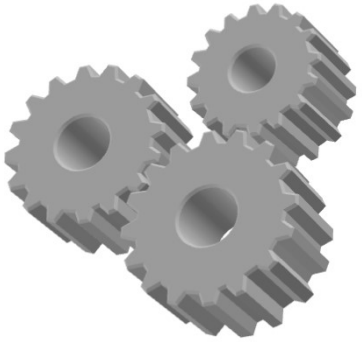
A. Kontinuirano ocjenjivanje

   Ocjenjivanje Programiranje.doc

B. Integralni ispit (pismeni+usmeni)

Definiranje programiranja

Što je to programiranje?



Što je sofware?

Je li programiranje=software ?

Definiranje programiranja

U širem smislu programiranje je procedura, postupak rješavanja nekog problema (problem solving procedure). Rezultat programiranja je program.

Program je niz naredbi koje se slijedno izvršavaju. Naredbama programa izvršava se neki zadatak. Podaci se transformiraju u informacije.

Primjer programa

```
public class Broj {
    public static void main(String[] args) {
          int broj = 5;
         if (broj >= 0)
             if (broj == 0) System.out.println("Prvi niz");
             else System.out.println("Drugi niz");
         System.out.println("Treći niz");
    }
}
```

Programiranje i software

- **Program i softver nisu sinonimi.**
- **Pod softverom se podrazumijevaju programi, postupci, pripadajuća dokumentacija i podaci koji su povezani s radom nekog računalnog sustava.**
- **Softver je širi pojam od programa.**
- **Pisanje programa je oko 10-20 % ukupnog vremena razvoja softvera. Ako se to prevede na troškove softvera u jednom životnom ciklusu, onda je to oko 10% ukupnih troškova.**

Definicija programiranja

Programiranje je uži pojam od software-a.

Programiranje je procedura tj. postupak rješavanja nekog problema uz uporabu konkretnog programskog jezika.

Rezultat programiranja je program.

Definicija programiranja

Osnovni preduvjet za programiranje:

Da se kompleksni problem koji se želi isprogramirati može raščlaniti na konačan broj nedvosmislenih koraka koje stroj (računalo) može izvršiti.

Povijest Programiranja

- Korijeni u tekstilnoj industriji (1801)

Francuz Joseph Marie Charles Jacquard, po zanimanju tkalac

program za tkalački stroj, izrađen na drvenoj bušenoj kartici

2 bitne ideje programiranja:

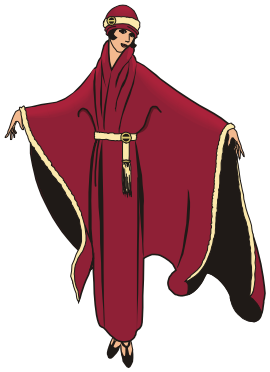
- raščlanjivanje kompleksnih zadaća na niz nedvosmislenih i konačnih koraka koje stroj može izvesti
- stroj na temelju programa može izvršavati ponavljajuće zadatke

Programski jezik

Govorni jezik: sustav glasova, gramatike, naglasaka, riječi i fraza kojima ljudi izražavaju svoje misli i osjećaje

Vezan pàs.

Vezan pâs.



- računala razumiju samo nizove brojeva

Programski jezik

Programski jezik -> formalni jezik u kojem je napisan računalni program

sastoji se od:

sintakse (načina na koji se različiti simboli jezika mogu kombinirati) i

semantike (značenja jezičnih konstrukcija)

Programski jezik

Principi dizajna jezika:

- Sintaksa

Sintaksa opisuje što čini strukturno ispravan program tj. što je gramatika jezika, koji je osnovni skup riječi i simbola dozvoljen za uporabu

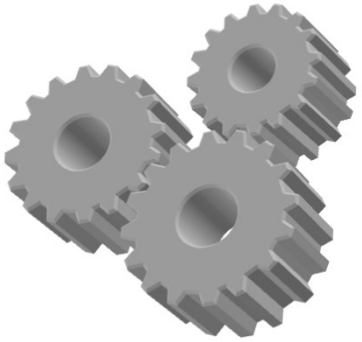
- Imenovanje i tipovi

Programski jezik sadrži potpuno izgrađen skup pravila za imenovanje varijabli, funkcija, klasa, parametara i sl., kao i njihovu “vidljivost” unutar programa. Tipovi podataka omogućavaju programerima bolje razumijevanje i uporabu operatora, kao i kvalitetniju kontrolu kompilatora.

- Semantika

Značenje programa, tj. programskih izraza definirano je semantikom programskog jezika.

Tko se smatra prvim programerom ?



Povijest programskih jezika

Prvi programer/ka :

1842. Ada Lovelace Byron

«Analitički Stroj tka algebarske uzorke na isti način kako Jacquard-ov tkalački stroj tka cvjetove i listove»

Ada je napisala skupove instrukcija koje bi se mogle izvršavati na Analitičkom Stroju.

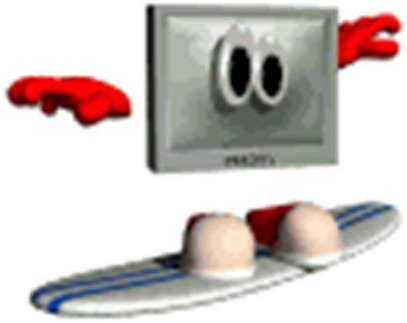
Ada - prvi programer za računala

Programski jezik Ada je u njenu čast dobio to ime.

Povijest programskih jezika

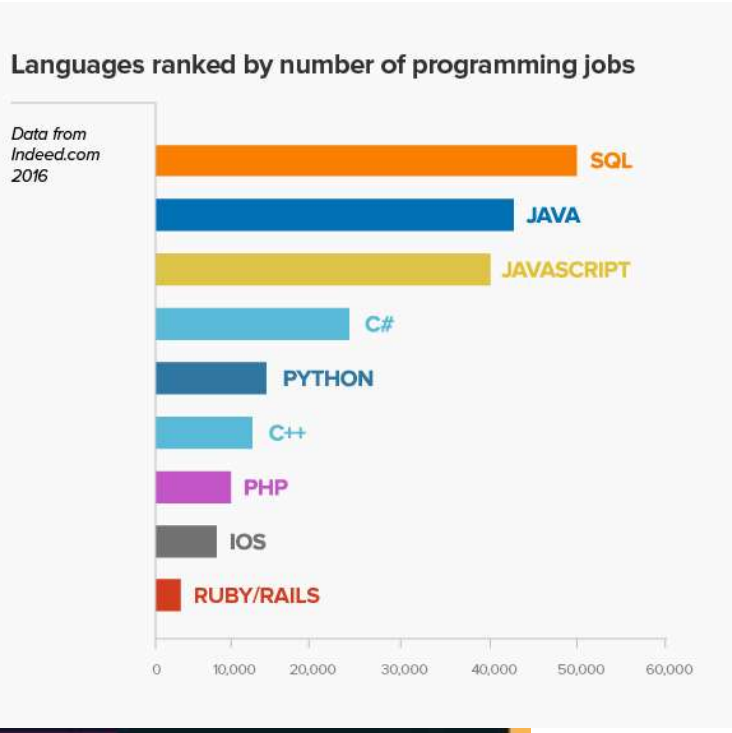
Pet generacija programskih jezika:

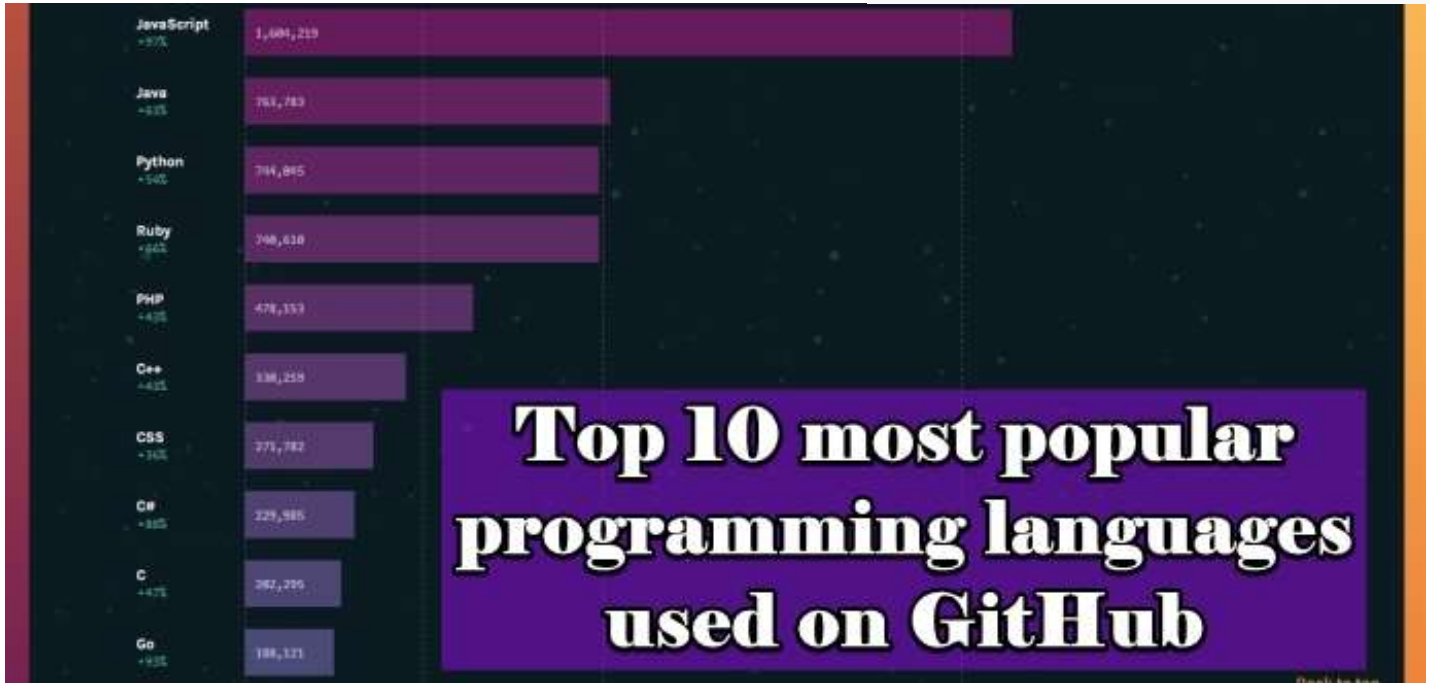
- strojni
- asembler
- proceduralni
- problemski orijentiran
- prirodni



Trenutno stanje

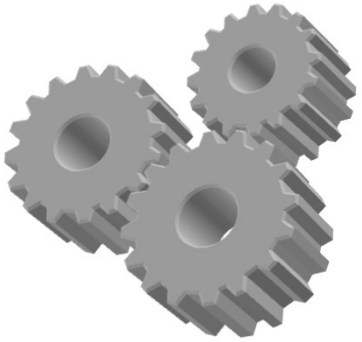
Language Rank	Types	Spectrum Ranking
1. C		100.0
2. Java		98.1
3. Python		98.0
4. C++		95.9
5. R		87.9
6. C#		86.7
7. PHP		82.8
8. JavaScript		82.2
9. Ruby		74.5
10. Go		71.9



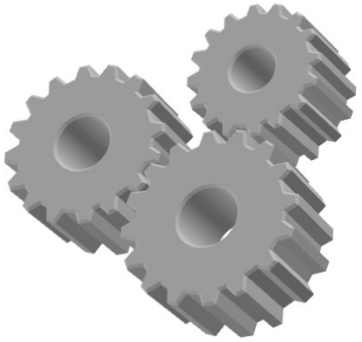


Tema: Implementiranje programskog jezika

Što se podrazumijeva pod implementiranjem programskog jezika?



Što je to
IZVORNI KOD?



Implementiranje programskog jezika

- Osnova za implementiranje je IZVORNI KOD

Izvorni kod je bilo koji niz izraza napisan u nekom, od strane čovjeka čitljivom programskom jeziku.

Izvorni kod je obično jedna ili više tekstualnih datoteka.

Transformiranje izvornog koda (čitljivog za čovjeka) u kod čitljiv od strane računala ostvaruje se ili pomoću kompilatora (prevoditelja) ili interpretera.

Implementiranje programskog jezika

BITNO:

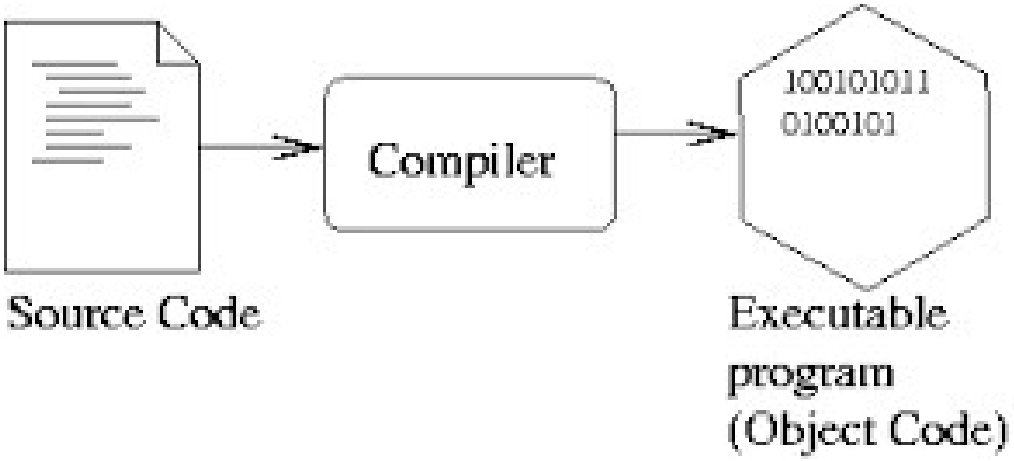
promjene programa mogu se raditi ISKLJUČIVO na izvornom programu.

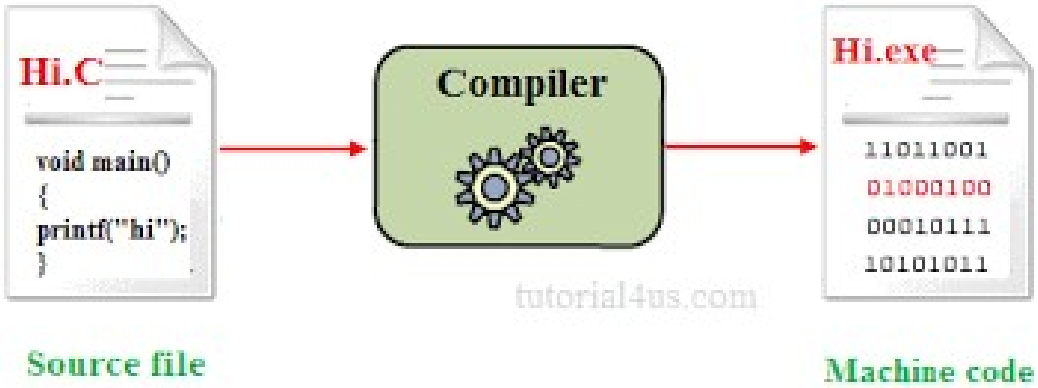
Da bi računalo moglo izvršiti zadaće napisane u izvornom programu, neophodno je taj program prevesti na jedini jezik koji računala razumiju, a to je jezik jedinica i nula (strojni programski jezik).

Implementiranje programskog jezika

- Način izvršavanja konkretnog programa na jednoj ili više konfiguracija hardware-a i software-a
- 2 osnovna pristupa:
  - Kompiliranje tj. prevođenje
  - Interpretiranje

Što je prevoditelj / compiler



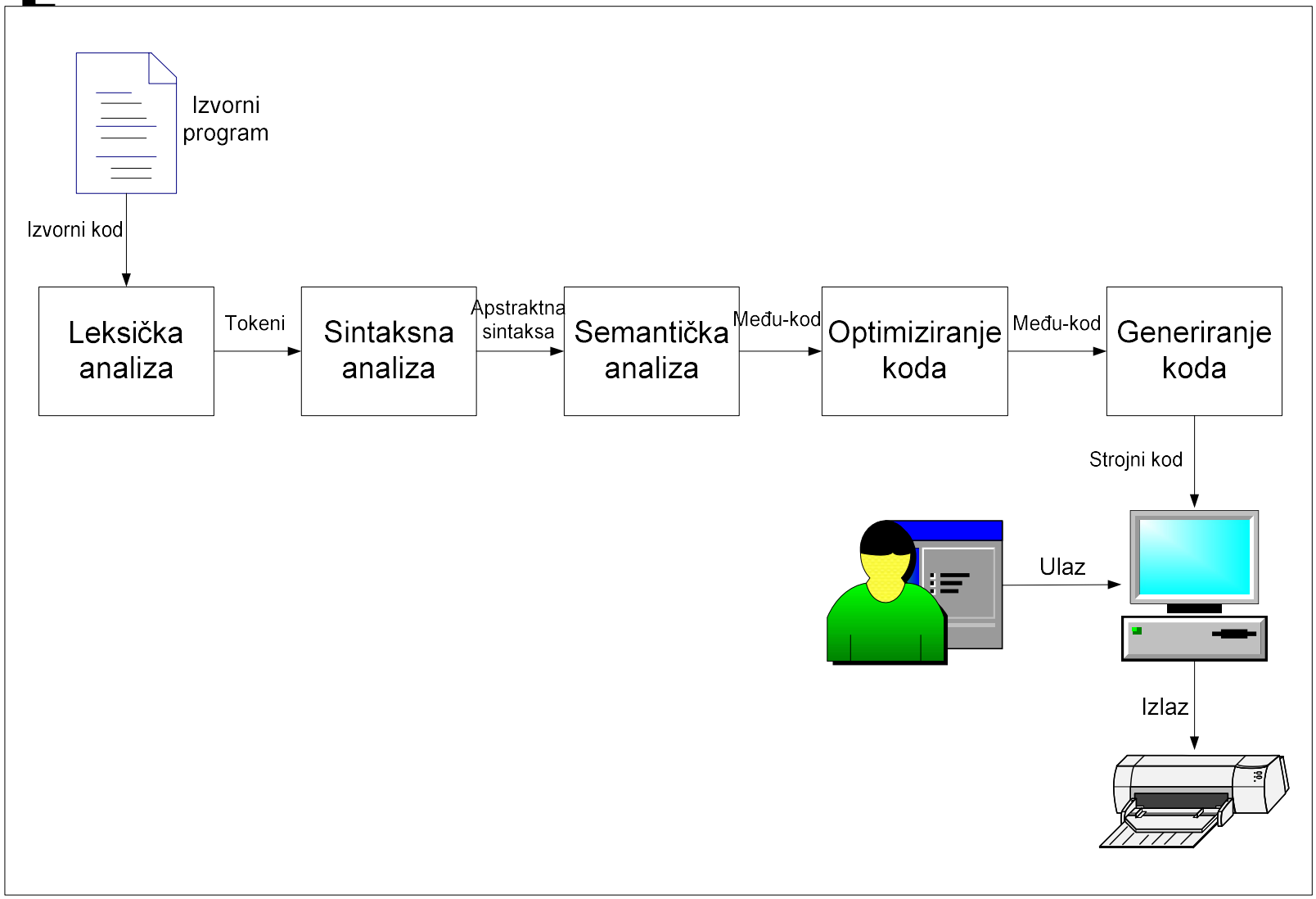


Implementiranje programskog jezika

Kompilator ili prevoditelj (engl. compiler) je računalni program (ili skup programa) koji prevodi tekst napisan u konkretnom programskom jeziku (izvorni kod) u drugi računalni jezik – objektni kod i/ili izvršni kod.

Osnovni cilj prevođenja (kompiliranja) izvornog koda je kreiranje tzv. izvršnog (engl. executable) programa tj. programa koji se može samostalno pozvati i izvršiti na odgovarajućoj računalnoj platformi.

Implementiranje programskog jezika



Interpreter

Interpreter je u biti program koji izvršava korake apstraktnog programa (međukoda) dok se izvršava na realnom stroju (računalu), odnosno to je program koji zamjenjuje dvije posljednje faze kompiliranja na način da izravno izvršava međukod.

Postoje dva opća tipa interpretera:

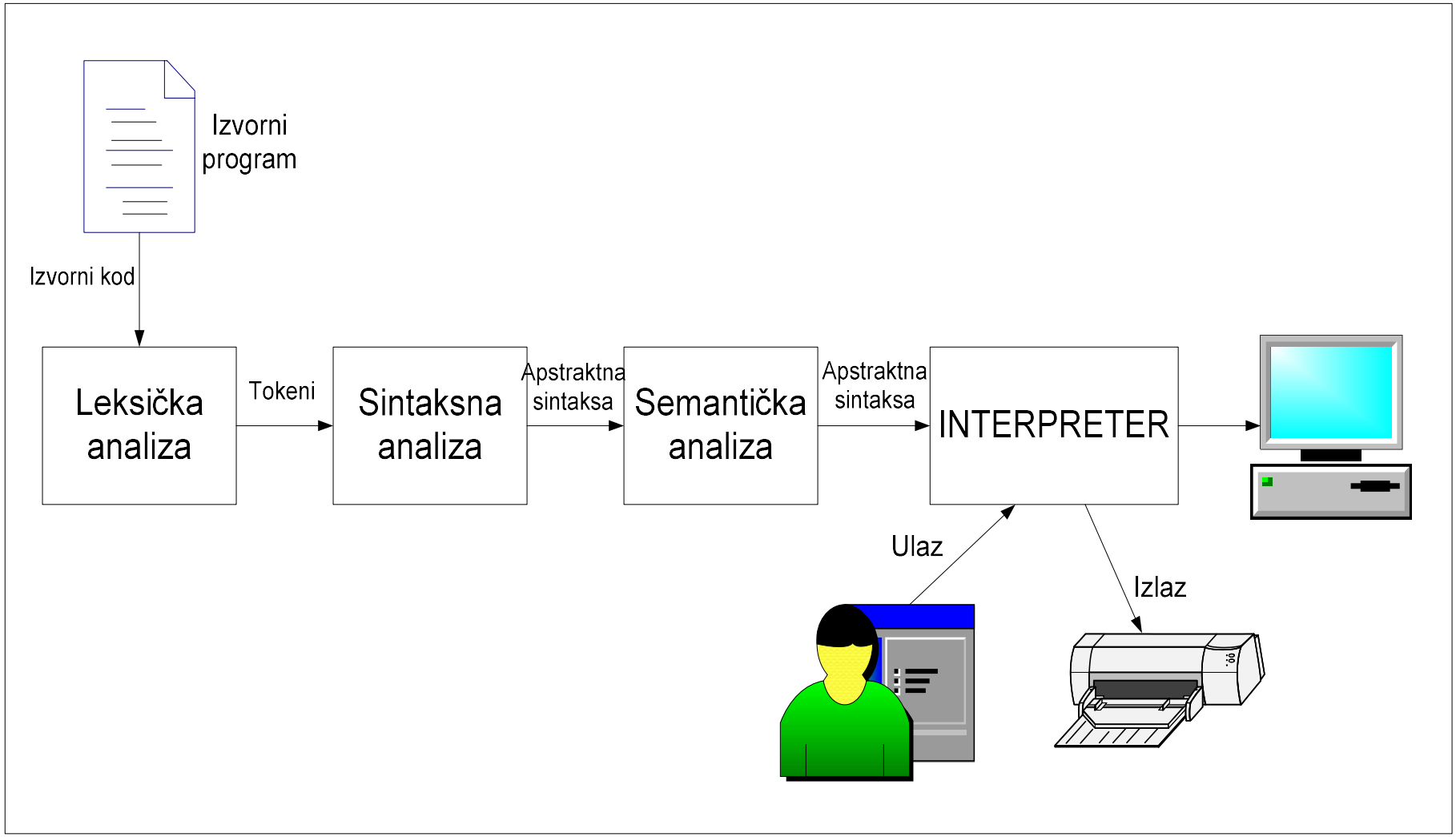
- Čisti
- Miješani.

Čisti interpreter svaki izraz tokenizira[1], raščlanjuje, semantički provjerava i interpretira pri svkom izvršavanju. Klasični primjer čistog interpretera je interpreter za Basic programski jezik.

Miješani interpreter prvo prevodi čitav program u među-kod, ali samo jedanput po izvršavanju, a onda dalje ponavlja interpretiranje međukoda bez daljnjeg prevođenja. Većina skript jezika (npr. Perl) koristi miješani interpreter.

[1] Pojam tokenizira preuzet je iz engleskog jezika a odnosi se na grupiranje programskih znakova u tokene tj. najmanje dijelove programa koji imaju neko značenje

Interpreter



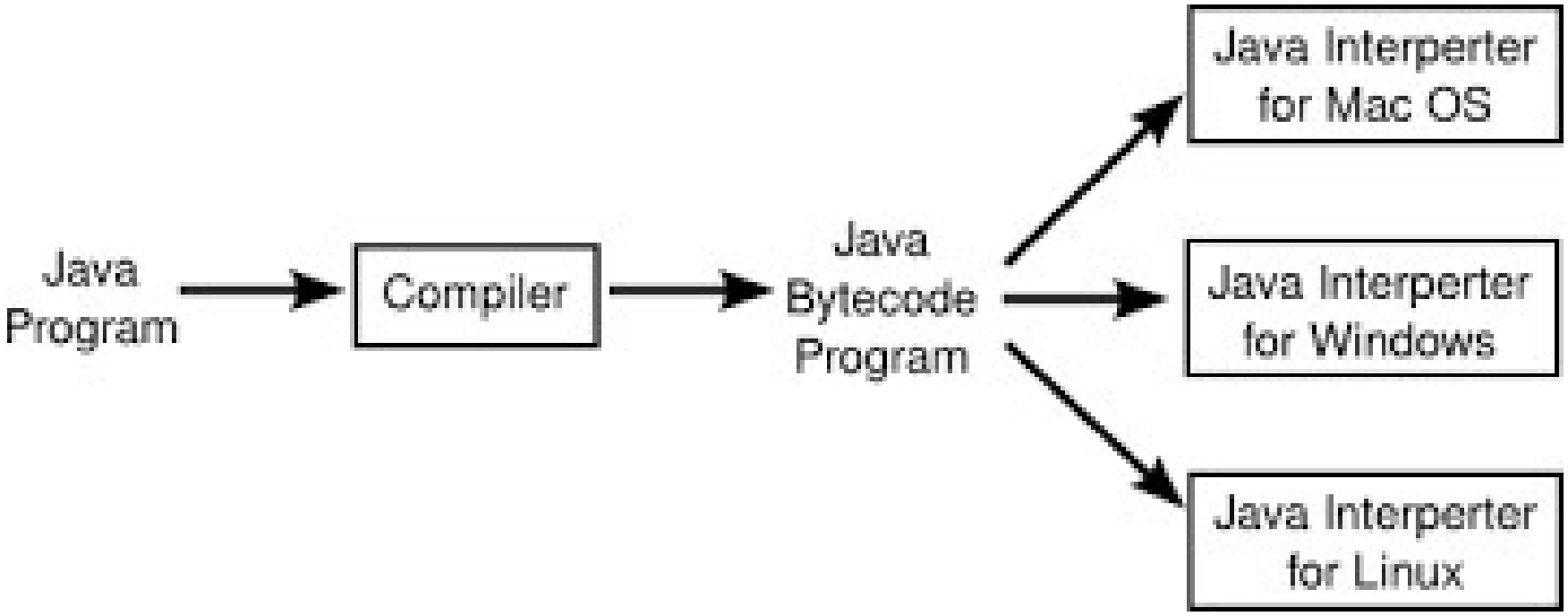
Interpreter

Poseban slučaj miješanog interpretera je virtualni stroj (engl. *virtual machine*).

Kompilacija izvršava samo jedanput, s ciljem dobivanja koda za apstraktni virtualni stroj, a zatim se taj virtualni stroj implementira pomoću interpretera za svaki različiti realni tip stroja (računala).

Najpoznatiji predstavnik ovog pristupa je programski jezik Java čiji se apstraktni stroj naziva Java virtualni stroj - JVM. Osnovna namjera Java dizajnera pri kreiranju JVM bila je osigurati što bolju fleksibilnost i portabilnost Java programa, po cijenu nešto manje efikasnsti. Naime, JVM omogućava izvršavanje svih promjene u Java programu pomoću samo jednog kompilatora, umjesto nekoliko različitih kompilatora. Java kompilator prevodi izvorni kod u vanjski oblik međukoda poznat kao bajt kod koji se interpretira pomoću JVM. Novije verzije Java programskog jezika smanjuju gubitak efikasnosti ugradnjom JIT kompilatora u JVM. Ova nadogradnja omogućuje da se JVM bajt kod prevodi „trenutno“ (engl. *on-the-fly*) u izvorni strojni kod na host stroju prije samog izvršavanja.

Java interpreter



JAVA interpreter

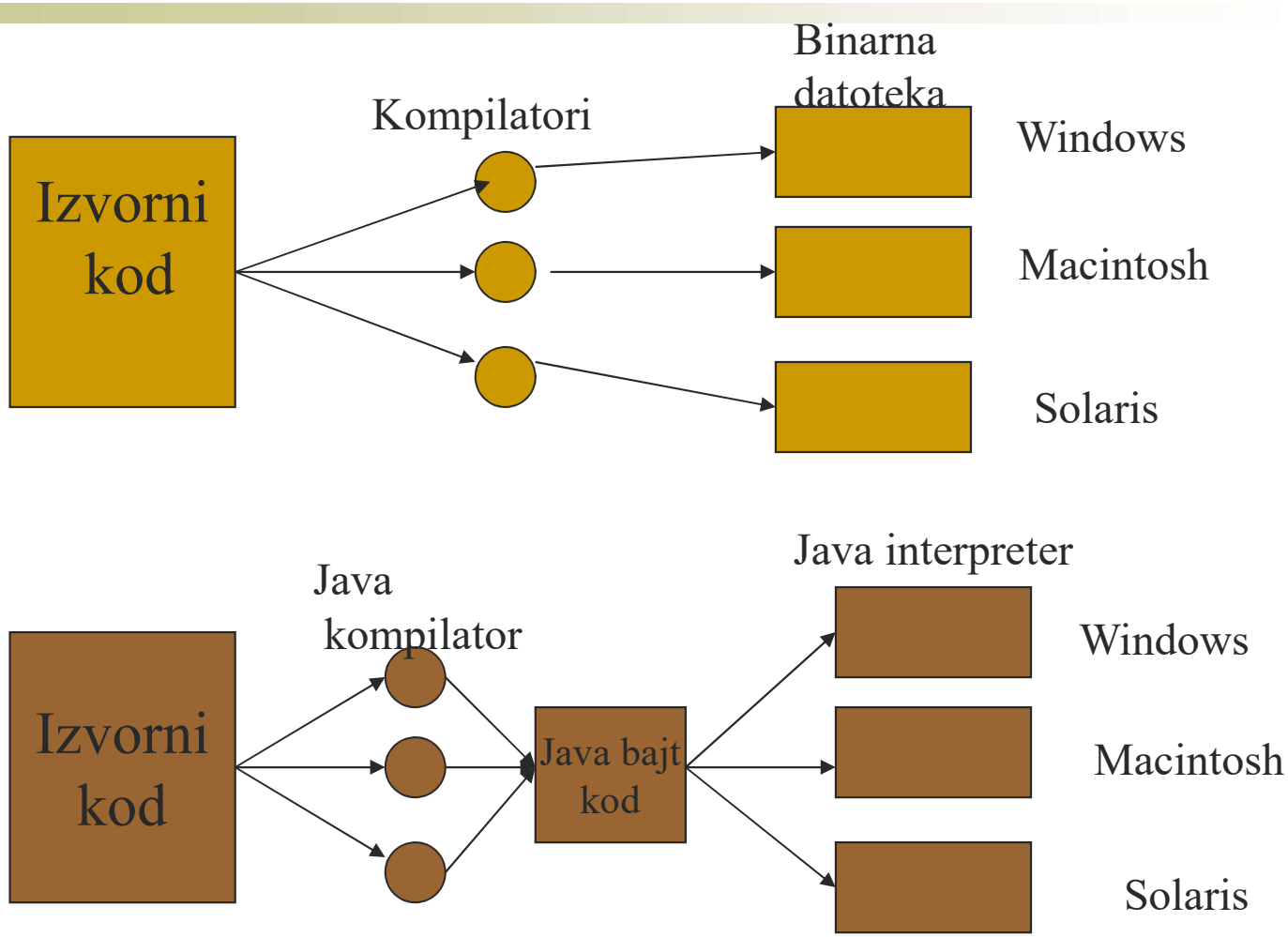
Interpretiranje

- **Prevođenje i izvršavanje Java aplikacije podijeljeno je u dvije faze:**

  1. Java kompilator iz izvornog koda programa stvara bajt-kod (tzv. J-kod)

  2. Java izvršni sustav koji interpretira bajt-kod i prevodi njegove naredbe u naredbe specifične za računalo na kojemu se izvršava aplikacija.

Interpretiranje … nastavak



Programske paradigme

- imperativno programiranje

- objektno-orijentirano programiranje

- funkcionalno programiranje

- logičko programiranje

Programske paradigme – Objektno orijentirano programiranje

Objektno orijentirano programiranje polazi od toga da se računalni program može promatrati kao skup pojedinačnih dijelova, ili objekata, koji surađuju (rade) jedan s drugim, dok se pak prema tradicionalnom shvaćanju program promatra kao skup funkcija, ili pojednostavljeno kao lista računalnih instrukcija.

Svaki objekt ima sposobnost primanja poruke, obrade podataka i slanja poruke drugim objektima, iz čega proizlazi da se svaki objekt može promatrati kao mali neovisni stroj ili glumac s različitom ulogom, odnosno odgovornošću.

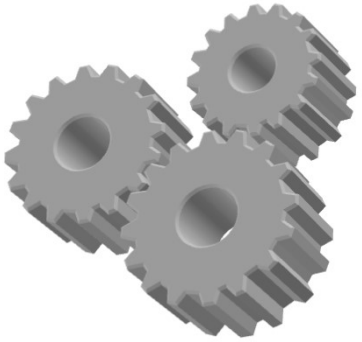
Programske paradigme – Objektno orijentirano programiranje

Osnovni koncepti:

- Objekt
- Klasa
- Nasljeđivanje
- Enkapsulacija (omatanje)
- Polimorfizam

Što je OBJEKT?

Koje su osnovne značajke OBJEKTA?



Programske paradigme – Objektno orijentirano programiranje

Objekt može biti bilo što sposobno za omogućavanje (pribavljanje) ograničenog skupa korisnih servisa (usluga). Svaki objekt ima stanje, ponašanje i identitet.

Svaki objekt ima pristup bilo kojem znanju potrebnom za izvršavanje njegovih usluga, što ne znači da objekt sadrži tu informaciju (znanje), već da joj može prići pitajući neki drugi objekt.

Uz objekt se vežu slijedeći pojmovi:

- odgovornost
- poruke
- protokol.

Programske paradigme – Objektno orijentirano programiranje

Odgovornost objekta - servis za koji se objekt “složio” ili mu je dodijeljen za izvršavanje. Objekti su zaduženi za izvršavanje specifičnih zadataka, a odgovornost se koristi kako bi se otkrilo tko (koji objekt) je zadužen za zadatak, bez potrebe da se razmišlja o strukturi objekta

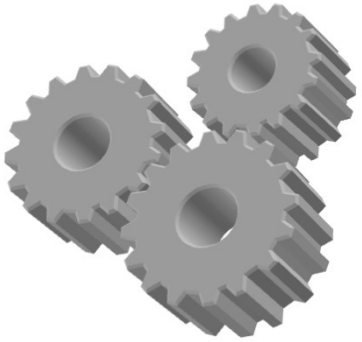
Poruka je formalna komunikacija poslana od strane jednog objekta drugom a koja zahtjeva neki servis (uslugu).

Pojam protokol se koristi za onaj dio sučelja koji prikazuje poruke na koje je objekt spreman odgovoriti.

Skup poruka na koje objekt odgovara i mijenjanje stanja objekta koje se bilježi – naziva se sučelje (engl. interface).

Što je KLASA?

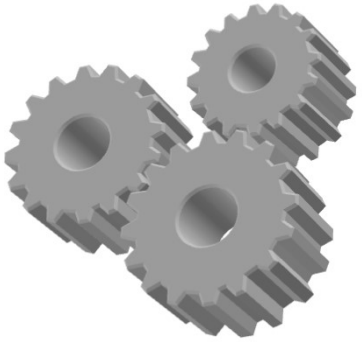
Razlika između KLASE I objekta?



Klasa

je apstrakcija (generički opis) za skup objekata s istim atributima i operacijama, odnosno to je predložak (*engl. template*) za kreiranje objekata. Zbog toga se kaže da je objekt instanca (pojavni – konkretni oblik) klase

Što se podrazumijeva pod NASLJEĐIVANJEM?



Što se nasljeđuje?

Programske paradigme – Objektno orijentirano programiranje

Nasljeđivanje

mehanizam koji omogućuje da klasa (podklasa) redefinira (promijeni) ponašanje i osobine neke druge klase (nadklase). Podklasa ima sve što i nadklasa, ali i još nešto što je specifično samo za nju.

Nasljeđivanje se može definirati i kao nadređena-podređena relacija između klasa u kojoj podređena (dijete) klasa ima isto ponašanje (odgovornosti) kao i nadređena (roditelj) klasa i plus najmanje jedno dodatno

Programske paradigme – Objektno orijentirano programiranje

Enkapsulacija (omatanje)

osigurava da kod izvan klase (tj. Druga klasa) vidi samo funkcionalne, ali ne i implementacijske detalje klase.

“skriva” ponašanje objekta od njegove implementacije, odnosno odvaja kako objekt izgleda od toga kako implementira svoje ponašanje.

jamči da nitko osim objekta ne može znati kako “iznutra” taj objekt izgleda, tj. na koji način izvršava svoj zadatak. Rezultat enkapsulacije je da svaki objekt prema „vanjskom svijetu“, tj ostalim klasama pokazuje svoje sučelje tj. Skup poruka (metoda) na koje odgovara.

Programske paradigme – Objektno orijentirano programiranje

Polimorfizam

potječe od grčkih riječi

poly – mnogo i morph-oblik,

što znači mnogo oblika.

U objektno orijentiranom programiranju to znači da jedna poruka može uzrokovati različite oblike odgovora. Polimorfizam je ponašanje koje se mijenja ovisno od toga koja klasa ga uzrokuje, što znači da dvije ili više klasa mogu reagirati potpuno različito na istu poruku, odnosno da je **primatelj poruke odgovoran za njenu interpretaciju.**

Pripremiti za sljedeće predavanje

- Sljedeće predavanje: 13.03.2018.
- Pripremiti:
  - Instalirati razvojno okruženje na svojoj opremi

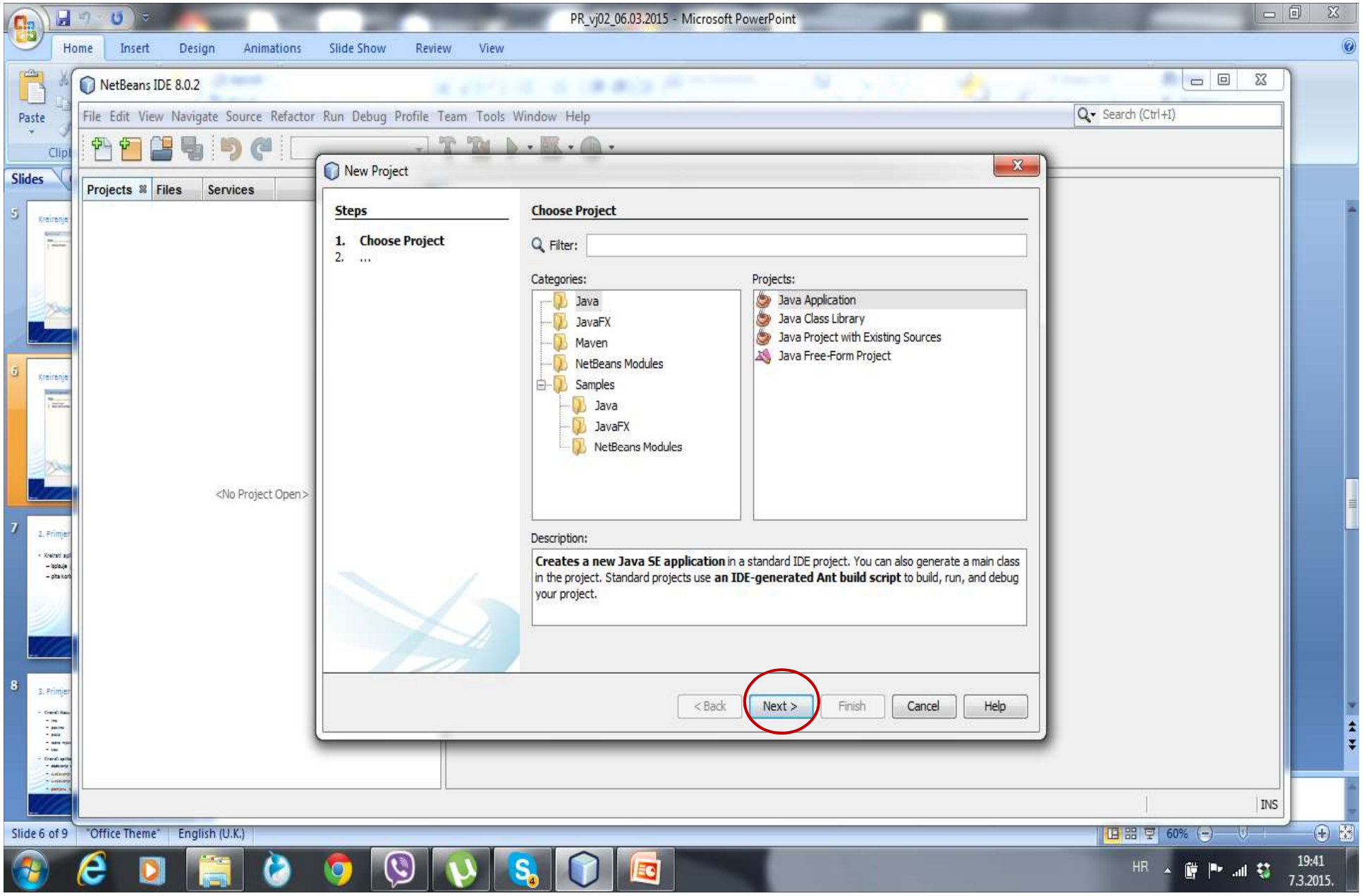
PONIJETI SVOJA RAČUNALA NA NASTAVU !!!

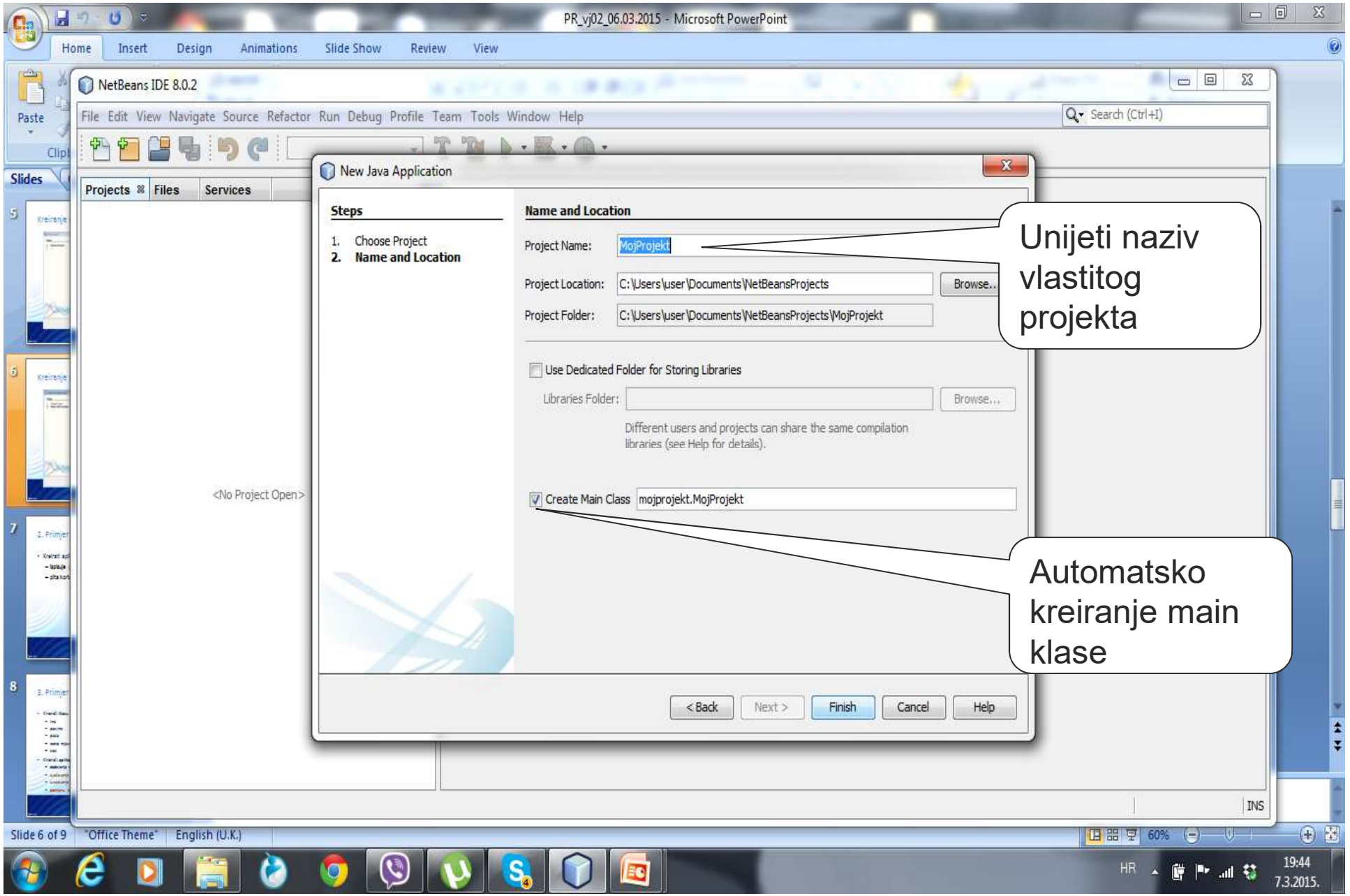
Instaliranje Jave (JDK i Netbeans)

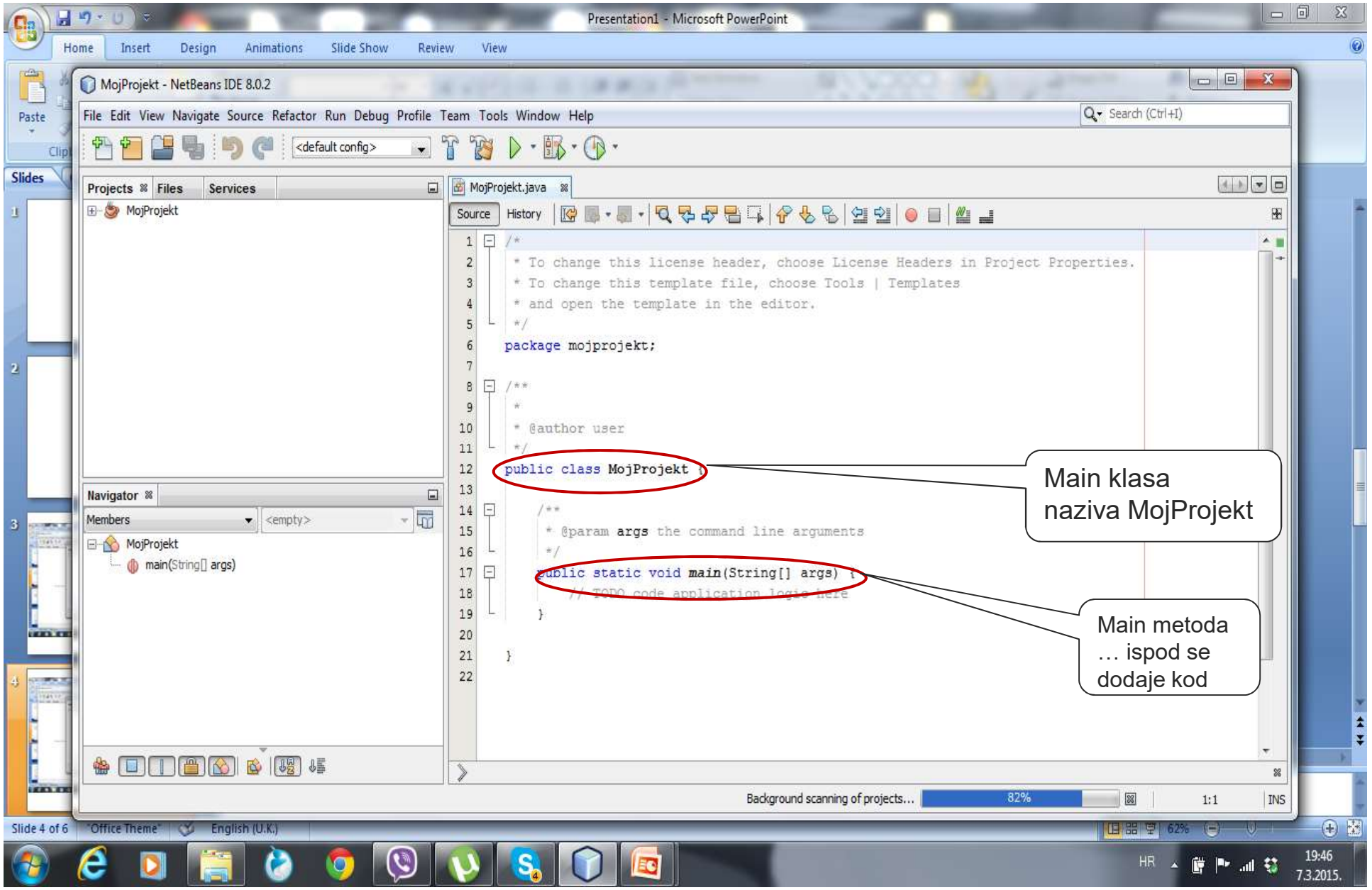
Postavljanje putanje (path-a)

- Desni klik na My Computer, Advanced System Settings -> Environment Variables
- Otvori se prozor s dva dijela. U gornjem dijelu (ako već ne postoji varijabla PATH) kliknuti na New i kad se otvori novi prozor, utipkati u gornji dio PATH, a u donji dio C:\Program Files\Java**jdk1.8.0_161**\bin

  odgovarajuća verzija jave
- Ako je već postojala varijabla PATH, onda je treba modificirati klikom na Edit.


PR_vj02_06.03.2015 - Microsoft PowerPoint
Home
Insert
Design
Animations
Slide Show
Review
View
NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects
Files
Services
<No Project Open>
New Project
Steps
1. Choose Project
2. ...
Choose Project
Filter:
Categories:
Java
JavaFX
Maven
NetBeans Modules
Samples
Java
JavaFX
NetBeans Modules
Projects:
Java Application
Java Class Library
Java Project with Existing Sources
Java Free-Form Project
Description:
Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use an IDE-generated Ant build script to build, run, and debug your project.
< Back
Next >
Finish
Cancel
Help
INS
Slide 6 of 9
"Office Theme"
English (U.K.)
60%
HR
19:41
7.3.2015.


PR_vj02_06.03.2015 - Microsoft PowerPoint
NetBeans IDE 8.0.2
New Java Application
Steps
1. Choose Project
2. Name and Location
Name and Location
Project Name: MojProjekt
Project Location: C:\Users\user\Documents\NetBeansProjects
Project Folder: C:\Users\user\Documents\NetBeansProjects\MojProjekt
Use Dedicated Folder for Storing Libraries
Libraries Folder:
Different users and projects can share the same compilation libraries (see Help for details).
Create Main Class mojprojekt.MojProjekt
< Back
Next >
Finish
Cancel
Help
Unijeti naziv vlastitog projekta
Automatsko kreiranje main klase


Presentation1 - Microsoft PowerPoint
MojProjekt - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Files Services
MojProjekt
Navigator
Members
MojProjekt
main(String[] args)
MojProjekt.java
Source History
/*
* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.
*/
package mojprojekt;
/**
*
* @author user
*/
public class MojProjekt {
/**
* @param args the command line arguments
*/
public static void main(String[] args) {
// TODO code application logic here
}
}
Main klasa naziva MojProjekt
Main metoda … ispod se dodaje kod
Background scanning of projects...
82%
1:1
INS
Slide 4 of 6
Office Theme
English (U.K.)
62%
HR
19:46
7.3.2015.

Pitanja..