

Java

Uvod



Pregled

- ✧ Što je Java ?
- ✧ Što Java nije ?
- ✧ Java - karakteristike
- ✧ Java povijest

Slide 2

Java Course 2001

Što je Java ?

- ✧ Objektno-orijentiran programski jezik (Java)
- ✧ Skup biblioteka (libraries) (*.java)
- ✧ Specifikacija virtualnog stroja (A virtual machine specification)
- ✧ Izvršna okolina (A runtime environment) (JVM)
- ✧ ...

Slide 3

Java Course 2001

Što Java nije

- ✧ Najbolji programski jezik
- ✧ Programski jezik za uljepšavanje web stranica

Slide 4

Java Course 2001

Karakteristike

- ❖ Objektno orijentiran programski jezik visokog nivoa
 - ❖ klase (classes)
 - ❖ instance klasa s metodama i poljimanasljedivanje (inheritance)
 - ❖ samo jednostruko nasljedivanje
 - ❖ Java koristi sučelja (interfaces) za podršku funkcionalnosti koju inače omogućava višestruko nasljedivanje
 - ❖ Java ne omogućava parametarski polimorfizam (parameterised polymorphism (generics)) - Matrix<float>, Matrix<double>
- ❖ Jednostavnost (pojednostavljeni C++)
 - ❖ Uklonjeno : preopterećenje operatora (overload of operators), višestruko nasljedivanje(multiple inheritance), pokazivači(pointer), goto
 - ❖ Dodano : garbage collection

Slide 5

Java Course 2001

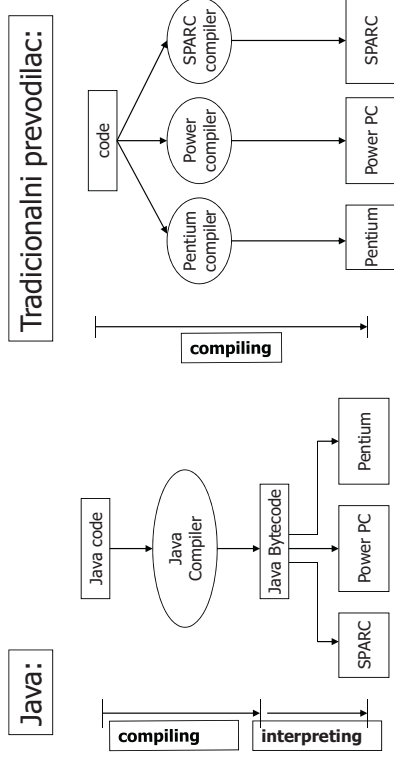
Karakteristike

- ❖ Java se prevodi (Compiled):
 - ❖ Java programi se prvo prevode u bytecode.
 - ❖ Source se prevodi u class datoteke i sadrže bytecode programa.
- ❖ Neovisnost o platformi
 - ❖ Bytecode koje proizvodi JAVA prevodilac je neovisan o platformi. Bytecod izvršava Java Virtual Machine
 - ❖ Java je multiplatformski jezik (PC, Mac, UNIX).

Slide 6

Java Course 2001

Neovisnost o platformi



Slide 7

Java Course 2001

Karakteristike

- ❖ Java je višenitna (Multi-Threaded):
 - ❖ podrška za multi-tasking ugrađena u jezik.
 - ❖ Java program može imati više niti izvršavanja.
 - ❖ Lightweight (lagani) procesi <-> niti
 - ❖ Dijeljeni adresni prostor (memorija)
 - ❖ Kontrola konkurentnosti (sinkronizacija)
 - ❖ Implementacija niti razlikuje se po platformama(will be fixed ?)

Slide 8

Java Course 2001

Karakteristike

- ✧ Java je robustna
 - ❖ Java aplikacije ne mogu srušiti cijelo računalo
 - ❖ Detaljne provjere u tijeku provođenja i izvršavanja.
 - ❖ Veoma razrađeno upravljanje iznimkama.
 - ❖ Striktna provjera grešaka tijekom provođenja i izvršavanja. Rana detekcija bugova poput provjere tipova i nepravilnog korištenja memorije.
- ✧ Java je sigurna:
 - ❖ Java je kreirana za verificiranje i izvršavanje binarnih programa u sigurnoj okolini.
 - ❖ Nema aritmetike pointera, provjera pristupa nizovima, garbage collection
 - ❖ "Sandbox" model

Slide 9

Java Course 2001

Karakteristika

- ✧ Jezik za mrežno računalstvo
 - ❖ Izgrađen s podrškom za mrežne komunikacije (bogata biblioteka).
 - ❖ Ugrađeno upravljanje za većinu mrežnih protokola (HTTP, FTP, MIME)
 - ❖ RMI (Remote Method Invocation) dozvoljava upravljanje s mrežnim objektima preko mreže
 - ❖ J2EE

Slide 11

Java Course 2001

Karakteristike

- ✧ Java je mala:
 - ❖ Kreirana da se izvodi na malim računalima i uređajima
 - ❖ Java je relativno mali jezik
 - ❖ Interpreter može stati u par stotina kilobajta
- ✧ Java is brza (?):
 - ❖ Mnogo brža od skriptnih jezika.
 - ❖ Prevedeni kod je 20 puta sporiji od C i C++
 - ❖ Brzina je dovoljna za mrežne i UI aplikacije
 - ❖ Nije pogodna za računski intenzivne aplikacije (performance computing – igre, znanstveni proračuni)
 - ❖ Just in Time (JIT) compilation, Java čipovi

Slide 10

Java Course 2001

Karakteristike

- ✧ Druge dobre stvari
 - ❖ Standardna biblioteka uvijek dostupna
 - ❖ Ugrađene napredne GUI/Graphics mogućnosti
 - ❖ Unicode svugdje
 - ❖ Mrežna distribucija aplikacija
 - ❖ Microsoft outside

Slide 12

Java Course 2001

Karakteristike

- ❖ Neke Java slabosti
- ❖ Različite Java verzije (1.0, 1.1, Java 2 1.2, 1.3, 1.4)
- ❖ Ekstremno jaka veza jezik <-> file/directory sustav
- ❖ Paketi(Packages) – nešto između biblioteke i direktorija

Slide 13

Java Course 2001

History

Jan 1996

- ❖ Netscape ugradio Javu
- ❖ Najava Sun-a o gradnji jeftinih Java čipova
- ❖ 1996
 - ❖ Svi uskaču u Java vagon
 - ❖ IBM, Borland, Symantec, Microsoft etc, etc
- ❖ 1997
 - ❖ JDK 1.1 (Java Development Kit)
 - ❖ Java Beans, JDBC, RMI, Security
 - ❖ JAVA OS
- ❖ 1999
 - ❖ JDK 1.2 = Java 2
- ❖ 1999. J2SE, J2EE, J2ME

Slide 15

Java Course 2001

Povijest

- ❖ 1991 - Sun Green Project
 - ❖ Potreban mali jezik za potrošačke uređaje: mobilne telefone, printere, itd. treba generirati mali, kompaktan kod, Oak jezik, prijenosan, OO jezik zasnovan na C++ , kasnije nazvan Java
- ❖ 1992 - 93
 - ❖ Neuspješan pokušaj prodaje tehnologije
- ❖ 1994
 - ❖ HotJava WWW browser napisan u Javi
- ❖ 1995
 - ❖ Java & HotJava (sposobnost izvršavanja Java bytecodea)

Slide 14

Java Course 2001

Tko je James Gosling ?



Java

Hello world !



Java Development Environment

- ✧ Za napisati prvi program potrebno je:
 - ❖ Java 2 platforma, Standard Edition
 - ❖ Tekst editor
- ✧ Java 2 SDK Instalacija
 - ❖ Pokreni Java 2 SDK instalaciju
 - ❖ Dodaj u PATH varijablu nešto poput C:\jdk1.3.1\bin
 - ❖ Provjeri (ukloni) CLASSPATH varijablu (-classpath command-line switch je preferirani način)

Slide 3

Java Course 2001

Aplikacija & Applet

Java programi mogu biti pisani i izvršavani na dva osnovna načina :

- ✧ Aplikacija
 - ❖ Samostalna aplikacija izvršavana iz linije naredbe
- ✧ Applet
 - ❖ Program koji se izvodi u okolini Web Browsera
 - ❖ Samostalni mod iz linije naredbe

Slide 2

Java Course 2001

Aplikacije

- ✧ Kreiramo jednu ili više datoteka s izvornim kodom
- ✧ Prevedemo svaku datoteku s izvornim kodom u class datoteku
- ✧ Aplikacija u Javi nije jedna izvršna datoteka. Ona je grupa class datoteka
- ✧ Pokretanje : pošaljete jednu class datoteku Java sustavu
- ✧ Ta class datoteka mora posjedovati metodu nazvanu **main**: `public static void main(String[] argv)`
- ✧ main metoda kontrolira tijek izvršavanja programa

Slide 4

Java Course 2001

Hello world aplikacija!

- ❖ Kreiraj izvornu datoteku s nazivom "HelloWorldApp.java"

```
/**
 * The HelloWorldApp class implements an application that
 * displays "Hello World!" to the standard output.
 */
public class HelloWorldApp {
    public static void main(String[] args) {
        // Display "Hello World !"
        System.out.println("Hello World !");
    }
}
```

Slide 5

Java Course 2001

Hello world aplikacija!

- ❖ Sačuvaj kod u datoteku:
 - ❖ HelloWorldApp.java (case-sensitive)
 - ❖ Naziv datoteke mora se podudarati s nazivom klase !!!
- ❖ Prevedi izvornu datoteku u bytecode datoteku
- ❖ U direktoriju gdje je datoteka izvornog koda:
 - ❖ javac HelloWorldApp.java (proizvodi HelloWorldApp.class)
- ❖ Izvršavanje s:
 - ❖ java HelloWorldApp (bez ekstenzije !)
 - ❖ HelloWorldApp nije naziv datoteke, već naziv klase !

Slide 6

Java Course 2001

Hello world aplikacija!

HelloWorldApp.java

```
/*
 * The HelloWorldApp class implements an
 * application that
 * displays "Hello World!" to the
 * standard output.
 */
public class HelloWorldApp {
    public static void
        main(String argv[]) {
        System.out.println
            ("Hello World!");
    }
}
```

HelloWorldApp.class

javac

0xCAFEBAFE
...

javac HelloWorld.java

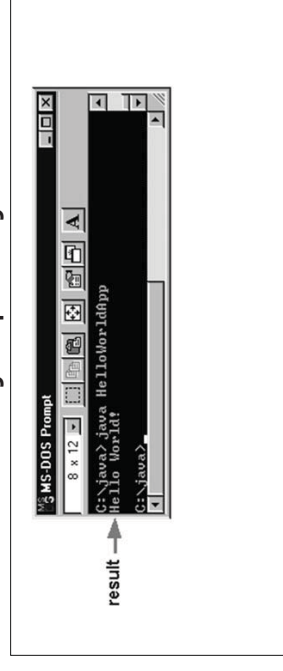
java HelloWorld

Slide 7

Java Course 2001

Hello world aplikacija

Rezultat izvršavanja aplikacije:



Slide 8

Java Course 2001

Hello world aplikacija

- ❖ Svi Java programi sastoje se od jedne ili više definicija klase
- ❖ HelloWorldApp je primarna ili kontrolna klasa(primary or controlling class)
- ❖ Samostalne aplikacije zahtijevaju metodu main u kontrolnoj klasi
- ❖ Applet ne zahtijeva main metodu

Slide 9

Java Course 2001

Kontrolna klasa

```
/**
 * The HelloWorldApp class implements an application that
 * displays "Hello World!" to the standard output.
 */
public class HelloWorldApp {
    public static void main(String[] args) {
        // Display "Hello World !"
        System.out.println("Hello World !");
    }
}
```

main metoda

definicija klase (kontrolna klasa)

Slide 11

Java Course 2001

Definiranje klase

- ❖ Podebljana linija počinje blok definicije klase u Javi

```
/**
 * The HelloWorldApp class implements an application that
 * displays "Hello World!" to the standard output.
 */
public class HelloWorldApp {
    public static void main(String[] args) {
        // Display "Hello World !"
        System.out.println("Hello World !");
    }
}
```

Slide 10

Java Course 2001

Main metoda

- ❖ Mora biti static -> metoda klase (class method)
- ❖ Metode klase možemo pozivati bez instanciranja objekta klase
- ❖ Kada pokrenemo Java aplikaciju Java interpreter pronalazi i poziva main metodu u klasi čiji je naziv dan u liniji naredbe

Slide 12

Java Course 2001

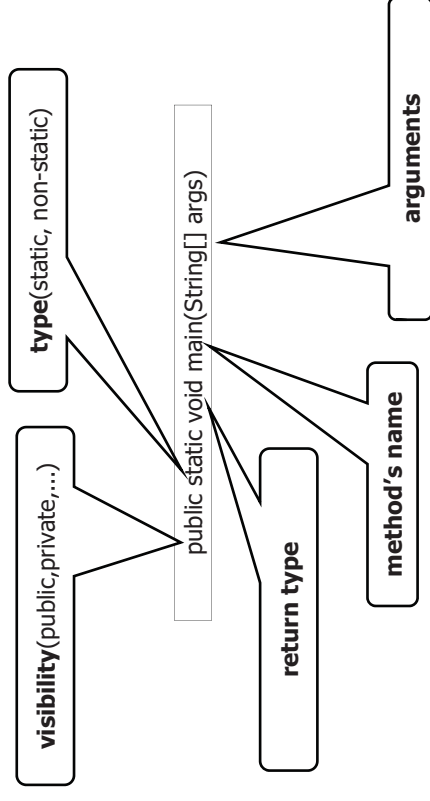
Komentari

- ❖ Tri vrste komentara:
 - ❖ `/* text */`
 - ❖ Prevodilac ignorira sve od `/*` do `*/`.
 - ❖ `/** documentation */`
 - ❖ Ovo označava komentar za dokumentaciju. Prevodilac ignorira tu vrstu komentara. JDK javadoc alat koristi te komentare za automatsko kreiranje dokumentacije.
 - ❖ `// text`
 - ❖ Prevodilac ignorira sve od `//` do kraja tekuće linije

Slide 13

Java Course 2001

Oznake metode



Slide 15

Java Course 2001

Oznake metode

Oznake metode je skup informacija o

metodi:

- ❖ naziv metode
- ❖ tip
- ❖ vidljivost
- ❖ argumenti
- ❖ tip return podatka

Slide 14

Java Course 2001

System i PrintStream klase

```
public class HelloWorldApp {  
    public static void main(String[] args) {  
        // Display "Hello World !"  
        System.out.println("Hello World !");  
    }  
}
```

Ova naredba poziva **println()** metodu klase **PrintStream** koja je referirana (pokazana) preko varijable **out** koja je varijabla klase **System** klase.

Slide 16

Java Course 2001

Java

Koncept objektno orijentiranog programiranja



Slide 3

Java

Softverski objekti

- ❖ modelirani po stvarnim objektima (bicikl, mobitel,...) ili abstraktnim konceptima (događaj, greška,...)
- ❖ održavaju svoj stanje u varijablama
- ❖ implementiraju djelovanje pomoću metoda.

Definicija: objekt je softverski paket koji se sastoji od varijabli i pripadnih metoda

Što je objekt?

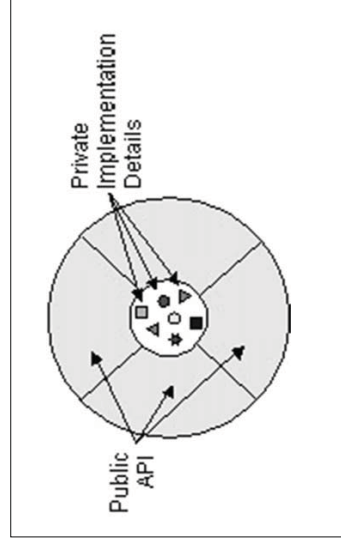
- ❖ objekti stvarnog svijeta: vaš mobitel, vaš stol, vaš bicikl, vaš televizor
- ❖ svi objekti imaju dvije karakteristike:
 - ❖ stanje (state)
(bicikl: trenutna brzina, trenutna brzina okretanja pedala, broj brzina, broj točkova)
 - ❖ djelovanje-funkcioniranje (behavior)
(bicikl: kočenje, ubrzavanje, usporavanje, promjena brzine)

Slide 2

Java

Vizualna reprezentacija

- ❖ uobičajena vizualna reprezentacija softverskog objekta

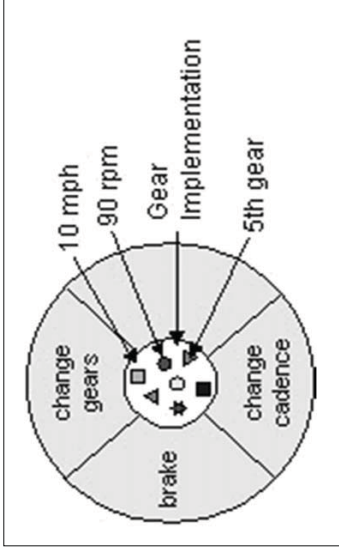


Slide 4

Java

Vizualna reprezentacija

- ❖ bicikl modeliran kao softverski objekt



Slide 5

Java

Prednosti enkapsulacije

- ❖ Modularnost

- ❖ Ako na početku definiramo metode koje čine sučelje s okolinom objekt će biti upotrebljiv bez obzira na unutarnju implementaciju

Možete voziti svaki bicikl jer svi imaju isto sučelje prema korisniku (vozaču)

- ❖ Skrivanje informacija

- ❖ Objekti posjeduju javno sučelje za komunikaciju s drugim objektima. Međutim objekt može sadržavati varijable i metode koje služe za njegov interni rad i nisu dostupne korisniku objekta.

Za promjenu brzine na biciklu potrebno je samo znati upravljati ručicom za promjenu brzine. Sve ostalo vozaču nije bitno (interni rad mehanizma).

Slide 7

Java

Enkapsulacija

- ❖ Varijable objekta čine centar ili nukleus objekta
- ❖ Metode okružuju i skrivaju nukleus objekta od ostalih bjekata u programu
- ❖ Pakiranje varijabli objekta unutar zaštitnog sloja metoda naziva se enkapsulacija

Slide 6

Java

Što su poruke (messages)?

- ❖ Interakcijom objekata postiže se funkcionalnost višeg reda

- ❖ Bicikl sam nije sposoban za bilo koju korisnu aktivnost. Bicikl je koristan kada drugi objekt (čovjek) počne interakcije (upravljanje volanom, vrćenje pedala)

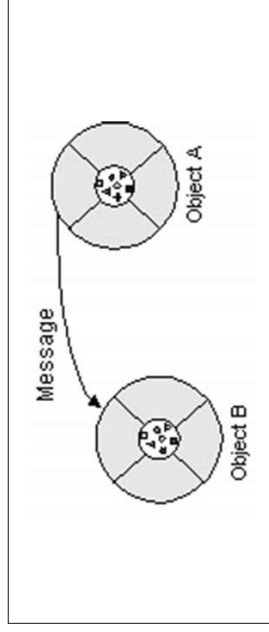
- ❖ Softverski objekti obavljaju interakciju (komunikaciju) s drugim objektima – međusobnim slanjem poruka

Slide 8

Java

Slanje poruka

- ✧ Kada objekt A želi da objekt B izvede neku od svojih metoda, tada objekt A šalje poruku objektu B.



Slide 9

Java

Komponente poruke

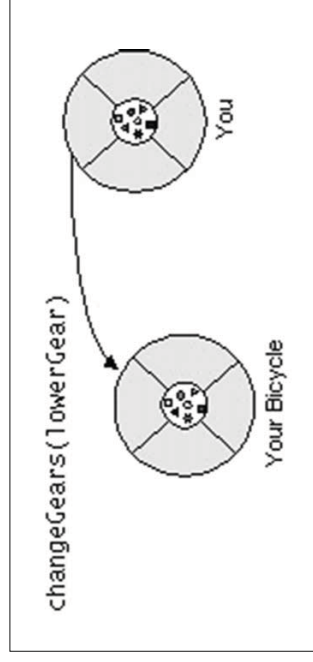
- ✧ Tri su sastavne komponente poruke:
 - ❖ Objekt kojem je poruka adresirana (vaš bicikl)
 - ❖ naziv metode koju treba izvesti (changeGears)
 - ❖ parametri proslijeđeni metodi (lower gear)

Slide 11

Java

Vi & vaš bicikl

- ✧ Promjena brzine



Slide 10

Java

Prednosti poruka

- ✧ Funkcioniranje objekta je izraženo preko metoda, tako da (osim nepoželjnog direktnog pristupa varijablama) slanje poruka bi trebalo pokriti sve interakcije između objekata.
- ✧ Objekti ne moraju biti unutar istog procesa ili čak ne moraju biti na istom stroju da bi međusobno izmjenjivali poruke.

Slide 12

Java

Što su klase?

❖ vaš bicikl je **instanca** od **klase** objekata bicikl

❖ Bicikle imaju stanja (trenutna brzina, trenutni broj okretaja, dva točka) i djelovanje (promjena brzine, kočenje) koja su zajednička

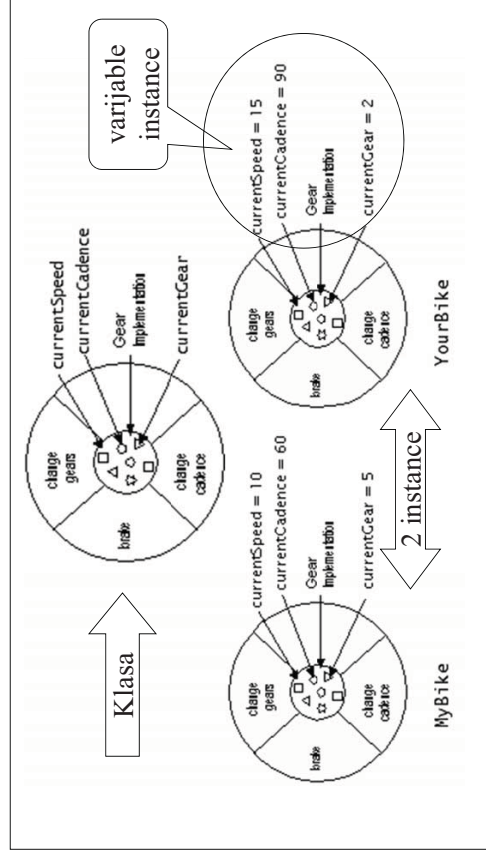
❖ Međutim, **svaki objekt bicikle ima jedinstveni skup stanja**.

Definicija: Klasa je plan ili kalup koji definira varijable i metode zajedničke za neki tip objekta.

Slide 13

Java

Klasa & instance



Slide 14

Java

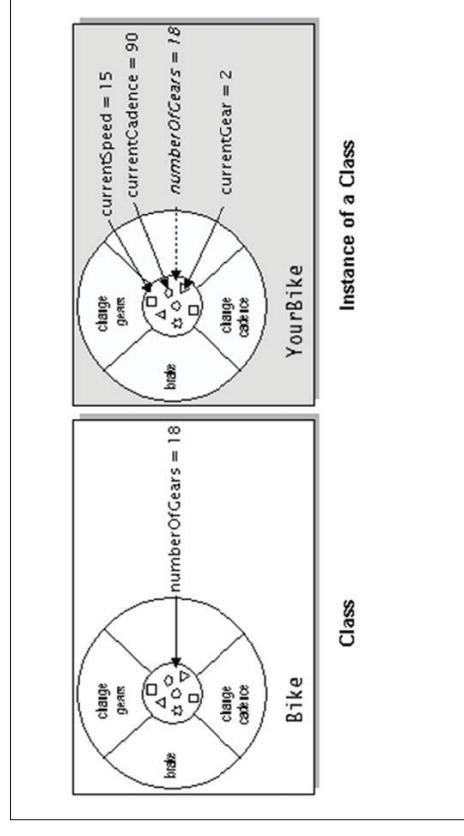
Varijable klase <-> varijable instance

- ❖ Varijabla klase sadržava informaciju koja se dijeli između svih instanci klase
- ❖ Pretpostavka: svi objekti bicikla imaju isti broj brzina, tada je
 - ❖ definiranje varijable instance neefikasno razbacivanje prostora !
 - ❖ možete definirati **varijablu klase**
 - ❖ Sve instance dijele tu varijablu

Slide 15

Java

Varijable klase



Slide 16

Java

Više o klasama

❖ Metode klase

- ❖ Metode koje je moguće pozvati bez postojanja objekta, poziv iz klase

❖ Ponovo - što su objekti ?

- ❖ Klasa je nacrt objekta
- ❖ Instanca je objekt

Slide 17

Java

Subclassing – izvođenje klasa

❖ Svaka subklasa naslijeđuje (inherits) stanja (deklarirane varijable) od superklase

- ❖ Mountain bikes, trkaće bicikle i tandemi imaju svi slijedeća stanja: broj okretaja pedale, brzinu kretanja i slično

❖ Svaka klasa naslijeđuje metode od superklase.

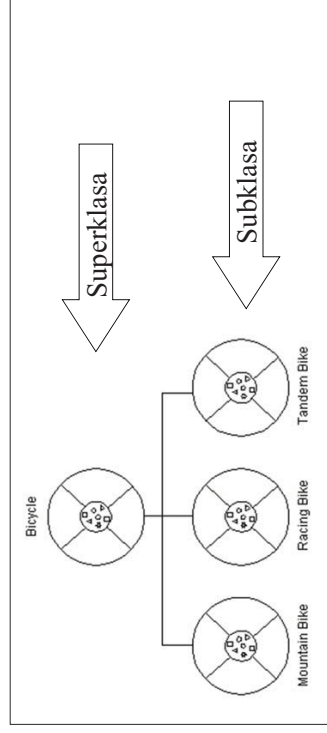
- ❖ Mountain bikes, trkaće bicikle i tandemi dijele istu funkcionalnost: kočenje i promjena brzine okretanja pedala, ...

Slide 19

Java

Što je nasljeđivanje?

- ❖ Objektno orijentirani sustavi dozvoljavaju klasama da budu definirane preko drugih klasa



Slide 18

Java

Subklase

- ❖ Subklase mogu uz naslijeđene dodati i nove varijable i metode.

- ❖ Tandem bicikle imaju dva sjedala i dva volana; mountain bikes imaju dodatni skup brzina, ...

- ❖ Subklase mogu i premostiti naslijeđene metode (override inherited methods) i realizirati specifične implementacije

- ❖ Npr. ako imate mountain bike s dodatnim skupom brzina možete premostiti "change gears" metodu tako da može upravljati dodatnim stanjima

Slide 20

Java

Koristi nasljeđivanja

- ❖ Subklase osiguravaju specijalizirane funkcije na osnovu zajedničkih elemenata koji su realizirani u superklasi. Nasljeđivanjem može se ponovo koristiti kod iz superklasa
- ❖ Mogu se implementirati superklase koje nazivamo apstraktne klase (abstract classes)
 - Definiraju generičku funkcionalnost
 - Apstraktna klasa definira, a može i parcijalno implementirati funkcionalnost.
 - veći dio implementacije je u naslijeđenim klasama.

Slide 21

Java

Java

Programi, Podaci, Varijable, Računanje- Uvod



Što je sučelje - Interface?

- ❖ Sučelje je uređaj koji objekti koriste za međusobnu komunikaciju (analogno protokolu)
- ❖ Protokol je definiran skupom konstanti i deklaracija metoda
- ❖ kada klasa implementira sučelje, klasa se obavezuje implementirati sve metode definirane u sučelju

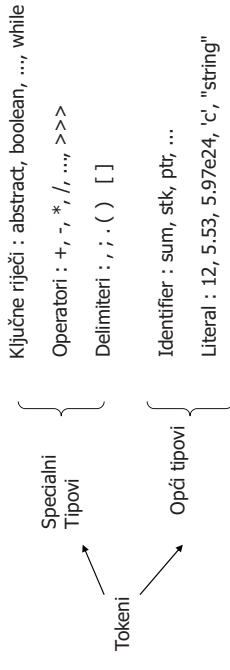
Slide 22

Java

Tokeni

- ❖ jedna od prvih faza prevođenja je skeniranje leksičkih elemenata tj. identifikacija tokena
- ❖ Token je primitivna jedinica koja ima sintaktično značenje
- ❖ **if (i<100) sum+=i;**

Tokeni : if, (, i, <, 100,), sum, +=, i, ;



Slide 2

Java © - Eugen Mudnić

Skup znakova (Character set)

- ❖ Java jezik je napisan korištenjem skupa znakova *Unicode* (16-bit skup znakova)
 - ❖ Prvih 256 znakova : Latin-1
 - ❖ Većina od prvih 128 znakova Latin-1 su ekvivalentni 7 bitnom ASCII skupu znakova
- ❖ Konverzija ASCII & Latin-1 <-> unicode obavlja se automatski
- ❖ Escape sekvence \uxxxx (x je heksadecimalno)

Slide 3

Java © - Eugen Mudnić

Komentari

- ❖ Komentari mogu uključivati bilo koji valjani unicode znak
- ❖ Komentari se ne mogu gnijezditi !

```
/* zakomentiraj – nije još implementirano
   /* Radi nešto*/
   Bicycle.changeGear();
*/
```



Slide 5

Java © - Eugen Mudnić

Komentari

- ❖ Tri vrste komentara:
 - ❖ `/* text */`
 - ❖ Prevodilac ignorira sve od `/*` do `*/`.
 - ❖ `/** documentation */`
 - ❖ Ovo označava komentar za dokumentaciju. Prevodilac ignorira tu vrstu komentara. JDK javadoc alat koristi te komentare za automatsko kreiranje dokumentacije.
 - ❖ `// text`
 - ❖ Prevodilac ignorira sve od `//` do kraja tekuće linije

Slide 4

Java © - Eugen Mudnić

Identifikatori

- ❖ Nazivi deklariranih entiteta
(Varijable, konstante, nizovi, klase, metode, labela)
- ❖ Mora početi sa slovom, slijede slova, ili znamenke

```
Ispravno  : box1, _box, $box, money_box, moneyBox
Neispravno : 1box, box1, #box, interface , money Box
```

Slide 6

Java © - Eugen Mudnić

Identifikatori – što je slovo ?

- ✧ Skoro svaki znak napisan u svijetu
 - ❖ Ispravno : cat, кошка , kokoš , ζορβα , αετόπτην
- ✧ Oznaka valute \$, £ , ...
- ✧ Spojna interpunkcija (" _ ")

Slide 7

Java © - Eugen Mudnić

Identifikatori

- ✧ Ne mogu se Java ključne riječi koristiti za identifikatore
- ✧ Ne mogu se nazivi predefiniranih konstanti koristiti kao identifikatori (true,false,...)
- ✧ Razlikovanje velika-mala slova (case significant)
- ✧ Dužina neograničena (ne zloupotrebjavati)
- ✧ Ne koristiti kriptične nazive (npr. Kvz78TG)
- ✧ Upotrebljavati konvencije za davanje baziva
- ✧ Koja je razlika između : topE topE ?

Slide 8

Java © - Eugen Mudnić

Ključne riječi

abstract	assert (java 1.4 !)	super
double	int	interface
boolean	else	long
break	extends	switch
byte	final	synchronized
case	finally	this
catch	float	throw
char	for	throws
class*	goto*	transient
const	if	try
continue	implements	void
default	import	volatile
do	instanceof	while
		static

null, true i false su formalno literali ! goto i const se ne koriste !

Slide 9

Java © - Eugen Mudnić

Java

Programi, Podaci, Varijable,
Računanje - 1



Varijable

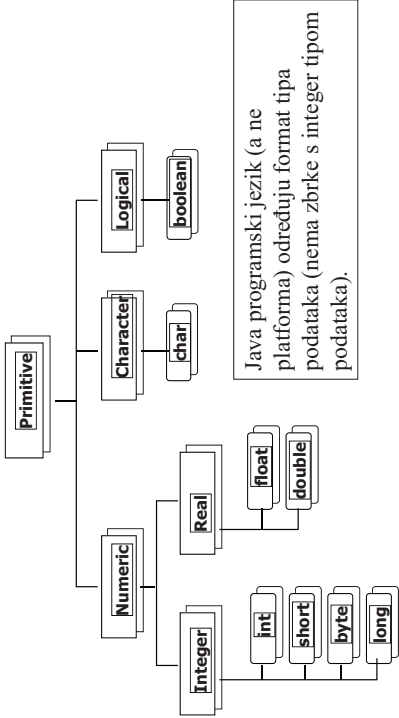
- ❖ Objekt čuva svoje stanje u varijabli.
- ❖ Varijabla je komad informacije imenovan identifikatorom.

❖ Deklaracija varijable:

syntax: <type> <name>



Primitivni tipovi podataka



Java programski jezik (a ne platforma) određuju format tipa podataka (nema zbrke s integer tipom podataka).

Data types

- ❖ Svaka varijabla ima točno određen tip podatka
- ❖ Tip određuje:
 - ❖ vrijednosti koje varijabla može sadržavati
 - ❖ operacije koje se mogu izvršavati nad tipom podatka
- ❖ Dvije kategorije tipova podataka u Javi:
 - ❖ Primitivni tip podataka
 - ❖ Reference (za referenciranje objekata)

Primitivni tipovi podataka

Ključna riječ	Opis	Veličina/Format
<i>(integers)</i>		
byte	Byte-length integer	8-bit two's complement
short	Short integer	16-bit two's complement
int	Integer	32-bit two's complement
long	Long integer	64-bit two's complement
<i>(real numbers)</i>		
float	Single-precision floating point	32-bit IEEE 754
double	Double-precision floating point	64-bit IEEE 754
<i>(other types)</i>		
char	A single character	16-bit Unicode character
boolean	A boolean value (true or false)	true or false

Cjelobrojni (Integer) tipovi podataka

Integer Type

Type	Size	Min. Value	Max. Value
byte	8bit	-128	127
short	16bit	-32768	32767
int	32bit	-2147483648	2147483647
long	64bit	-9223372036854775808	9223372036854775807

Unsigned are not supported !

```
byte smallerValue;  
short pageCount;  
int wordCount;  
long bigValue;
```

Slide 6

Java © - Eugen Mudnić

Cjelobrojni (Integer) literali

✧svaki integer literal je pretpostavljeno tipa **int** (by default)

✧1, -9999, 123456789 – literali tipa int

✧1L, -9999L, 123456789L – literali tipa long

✧ne mogu se specificirati byte i short literali

✧baza 16 - 0xFA15 ili 0XFA15

✧baza 8 – 035, 017 (pažljivo s vodećom nulom !)

Slide 8

Java © - Eugen Mudnić

Cjelobrojni (Integer) tipovi podataka

byte max	01111111	red color : sign bit
byte min	10000000	
short max	0111111111111111	
short min	1000000000000000	
int max	01111111111111111111111111111111	
int min	10000000000000000000000000000000	
long max	01111111.....	
long min	10000000.....	

Slide 7

Java © - Eugen Mudnić

Deklariranje cjelobrojnih varijabli

```
long bigOne; // declaration  
long bigOne=1024147L; // declaration and initialization  
long bigOne=99999999L, largeOne=254111L;  
int xCord=0, yCord=0; // Point coordinates  
int miles =0,  
yards =0,  
feet =0;  
byte luckyNumber = 7;  
byte smallNumber = 1234;
```

Before use variable must be declared and value must be assigned !

Slide 9

Java © - Eugen Mudnić

Floating Point (pokretni zarez) tipovi podataka

❖ Dva osnovna tipa :

- ❖ **float**
 - 3.4E38 to +3.4E38 , približno 7 točnih znamenki
- ❖ **double**
 - 1.7E308 to +1.7E308 , približno. 17 točnih znamenki (najmanja nenulta vrijednost je $\pm 4.9E-324$)

❖ Pridržavanje IEEE 754 standarda za operacije u pokretnom zarezu

Slide 10

Java © - Eugen Mudnić

Deklaracija Floating Point varijabli

```
double sunDistance= 1.496E8;  
float electronMass=9E-28F;  
float hisWeight=92.2F, herWeight=52.3F;  
float hisWeight=92.2;
```

prevodilac neće izvršiti
automatsku konverziju u tip float
!

Slide 12

Java © - Eugen Mudnić

Floating point literal

❖ pretpostavljeno da je floating point literal tipa **double**

- ❖ 1.0,345.768, 34E22 – literali tipa double
- ❖ 1.0f,345.768F, 34E22f – literali tipa float

Slide 11

Java © - Eugen Mudnić

Java

Programi, Podaci, Varijable, Računanje - 2



Operatori

Arithmetic Op. :	+	-	*	/	%								
Relational Op. :	>	>=	<	<=	==	!=							
Logical Op. :	&&		!										
Inc/Dec Op. :	++	--											
Bit Op. :	&		^	~	<<	>>							
Conditional Op. :	?:												
Assign Op. :	=	+=	-=	*=	/=	%=	&=	^=	=	>>=	<<=	>>>=	<<<=
Casting Op. :	(Data Type)												
Array Op. :	[]												
Method Op. :	()	.											
instanceof Op. :	instanceof												

Slide 2

Java © - Eugen Mudić

Aritmetiči izračuni – dodjeljivanja(assign)

operator dodjeljivanja

numFruit=numApples+numOranges;

izraz dodjeljivanja

numApples=numApples+1;
a=b=c=777;

short value = 0; // ovo je deklaracija & inicijalizacija
value =10; // **dodjeljivanje vrijednosti**

Slide 4

Java Course 2001

Operatori & operandi

- ❖ Unarni (++,--,+,*,...)
 - ❖ prefiks notacija : **operator op** (++i;)
 - ❖ Postfiks notacija : **op operator** (i++;)
- ❖ Binarni (=,*,+,*,...)
 - ❖ infix notacija : **op1 operator op2** (a+b)
- ❖ Ternarni (?:)
 - ❖ Infix notacija : **op1 ? op2 : op3**

❖ **Osim izvođenja operacije operator vraća i vrijednost**

Slide 3

Java Course 2001

Cjelobrojni izračuni

❖ Binarni operatori

Operator	Korištenje	Opis
+	op1 + op2	Zbraja op1 and op2
-	op1 – op2	Oduzima op2 od op1
*	op1 * op2	Množi op1 s op2
/	op1 / op2	Dijeli op1 s op2
%	op1 % op2	Računa ostatak dijeljenja op1 s op2

Slide 5

Java Course 2001

Cjelobrojni izračuni

❖ Unarni operatori

Operator	Korištenje	Opis
+	+op	Promovira op u tip int ako je tipa byte, short, ili char !
-	-op	Aritmetička negacija op

Slide 6

Java Course 2001

Cjelobrojni izračuni

- ❖ ++, --
- ❖ Prefiks operator

```
n = 1;
x = ++n;    // x=2, n=2
```
- ❖ Postfiks operator

```
n = 1;
x = n++;    // x=1, n=2
```
- ❖ Ne može se koristiti na izrazima, samo na varijablama

```
(a + b)++    // error
```

Slide 8

Java Course 2001

Cjelobrojni izračuni

❖ Inkrement i dekrement operatori

Operator	Korišt.	Opis
++	op++	Inkrementiraj op za 1; evaluiraj vrijednost op prije inkrementiranja
++	++op	Inkrementiraj op za 1; evaluiraj vrijednost op nakon inkrementiranja
--	op--	Dekrementiraj op za 1; evaluiraj vrijednost op prije inkrementiranja
--	--op	Dekrementiraj op za 1; evaluiraj vrijednost op nakon inkrementiranja


Slide 7

Java Course 2001

Kraći cjelobrojni tipovi

- ❖ aritmetičke operacije s varijablama tipa **byte** ili **short** se računaju korištenjem 32-bitne aritmetike te je rezultat 32-bit integer

```
short numOranges = 5;
short numApples = 10;
short numFruit = 0;
numFruit = numOranges + numApples;
```



```
short numOranges = 5;
short numApples = 10;
short numFruit = 0;
numFruit = (short) (numOranges + numApples);
```

explicit cast

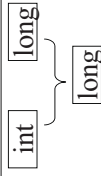
Slide 9

Java Course 2001

Tip long

- ❖ Aritmetička operacija koja uključuje vrijednost tipa long uvijek se računa korištenjem 64-bitnih vrijednosti

```
long result = 0;  
long factor = 10L;  
int number = 5;  
result = factor * number;
```



Slide 10

Java Course 2001

Greške u cjelobrojnoj aritmetici

- ❖ Operatori dijeljenja i ostatka

```
int numZero = 0;  
int numSpec = 10;  
int result=0;  
result=numSpec & numZero; // generira exception  
result=numSpec % numZero; // generira exception
```

Slide 12

Java Course 2001

Cjelobrojna aritmetika

- ❖ zapamti pravilo :

Sve cjelobrojne operacije koje uključuju bilo koji cjelobrojni tip **osim long** obavljaju se korištenjem 32-bitne aritmetike !

Slide 11

Java Course 2001

Primjeri cjelobrojnih izračuna

- ❖ analiziraj, prevedi i izvrši !
 - ❖ 01 TotalFruit\Fruit.java
 - ❖ 02 DelayedEnding\Fruit.java
 - ❖ 03 TotalFruitAndOranges\Fruit.java
 - ❖ 04 IncrementOranges\Fruit.java

Slide 13

Java Course 2001

Greške u cjelobrojnoj aritmetici

- ❖ Rezultati van opsega se režu bez upozorenja!

<pre>int num; num = (1000000*20000000)/500000; // result is -2909 !?</pre>
<pre>long num; num = (1000000*20000000)/500000; // result is -2909</pre>
<pre>long num; num = (1000000L*20000000L)/5000000L; // result is 4000000</pre>

Slide 14

Java Course 2001

Greške u floating point aritmetici

- ❖ Dva načina pojave greške:
 - ❖ vrijednost van opsega
 - ❖ matematički neodređen rezultat

```
double numOranges = 5.0;
double numApples = 10.0;
double fruitTypes = 0.0;
double averageFruit = 0.0;

//range overflow error
averageFruit = (numOranges + numApples) / fruitTypes;
// result mathematically indeterminate NaN (Not-a-Number)
averageFruit = (numOranges -5.0) / (numApples- 10.0);
```

Slide 16

Java Course 2001

Floating point izračuni

- ❖ Četiri osnovne operacije +, -, *, /
 - ❖ [Ch2\02 AverageFruit\01 AverageAverageFruit.java](#)
- ❖ Možete koristiti ++ i -- s floating point varijablama ! (inkrementiranje ili dekrementiranje za 1.0)
- ❖ Moguća primjena modulus operatora
 - ❖ floatOperand1 % floatOperand2

```
double result;
result = 12.6 % 5.1; // result is 2.4
```

Slide 15

Java Course 2001

Infinitivna aritmetika

- ❖ Zbroji, oduzmi

x	y	x+y	x-y
+1.F	+1.F	+1.F	NaN
+1.F	-1.F	NaN	+1.F
-1.F	+1.F	NaN	-1.F
-1.F	-1.F	-1.F	NaN
NaN	+1.F	NaN	NaN

- ❖ Množi, dijeli

x	y	x/y	x%y
Finite	±0.0	±1.F	NaN
Finite	±1.F	±0.0	Finite
±0.0	±0.0	NaN	NaN
±1.F	Finite	±1.F	NaN
±1.F	±1.F	NaN	NaN

Slide 17

Java Course 2001

Miješani aritmetički izrazi

- ❖ Pravila poredana kako se provjeravaju
 - ❖ Ako je bilo koji operand tipa **double**, drugi se konvertira u **double** prije izvršavanja operacije
 - ❖ Ako je bilo koji operand tipa **float**, drugi se konvertira u **float** prije izvršavanja operacije
 - ❖ Ako je bilo koji operand tipa **long**, drugi se konvertira u **long** prije izvršavanja operacije
 - ❖ Ako nijedan operand nije tipa double, float ili long onda su tipa int, short ili byte -> 32-bitna aritmetika

Slide 18

Java Course 2001

Casting u dodijeljivanju

- ❖ Automatski cast bit će obavljen samo ako nema mogućnosti gubljenja informacije
- ❖ byte->short->int->long->float->double
- ❖ za suprotan smjer potrebno je koristiti eksplicitni cast

Slide 20

Java Course 2001

Eksplicitni casting

- ❖ Može se izvršiti casting na bilo kojem osnovnom tipu (moguć gubitak informacije)

(Data Type) op

```
(int) 3.75      >>>> 3
(float) 3       >>>> 3.0
(float) (1 / 2) >>>> 0.0
(float) 1/2     >>>> 0.5
```

```
int three = 3;
int two = 2;
result = 1.5+three/two; // result is 2.5
result = 1.5+(double)three/two; // result is 3.0
```

Slide 19

Java Course 2001

op= operator

- ❖ Oblik **lhs op=rhs;**
- ❖ Operatori: +, -, %, ...
- ❖ skraćena reprezentacija: **lhs=lhs op (rhs);**
- ❖ **count +=5;** or **count = count + 5;**

```
double a=10.0,b=11.0;
int result=10000;
```

```
result/=a*b; // compiled (automatic cast !)
result=result/(a*b); // compiler error !
result=(int)(result/(a*b)); compiled (explicit cast)
```

Ipak postoji razlika !

Slide 21

Java Course 2001

Matematičke funkcije i konstante

- ❖ Dio standardne biblioteke, pohranjene u paketu `java.lang`
- ❖ primjer: `Ch2\03 MathCalc\MathCalc.java`

Slide 22

Java Course 2001

Znakovi(characters)

- ❖ Varijabla tipa `char` sadržava jedan znak, svaki zauzima 16 bita (Unicode)

```
char myCharacter = 'X';
```

❖ Escape sekvence

```
char myCharacter = '\u0058';
```

<code>\b</code>	Backspace
<code>\f</code>	Form feed
<code>\n</code>	New line
<code>\r</code>	Carriage return
<code>\t</code>	Tab

Slide 2

Java © - Eugen Mudić

Java

Programi, Podaci, Varijable,
Računanje - 3



Znakovna aritmetika(character arithmetic)

```
myCharacter+=1; // povećaj na slijedeći znak  
++myCharacter;
```

- ❖ U aritmetičkoj operaciji tip `char` se kovertira u tip **int**

```
char aChar = 0;  
char bChar = '\u0028';  
aChar=(char) (2*bChar+8); // result is 'X'
```

Slide 3

Java Course 2001

Bitwise and shift operacije

❖ Bitwise = operacije na bitovima

❖ Operator

❖ $\&$, $|$, $<<$, $>>$, $>>>$, \wedge , \sim

❖ Operand mora biti integer (32bit)

Operator	Precedence
$<<$ $>>$ $>>>$	(H) \updownarrow (L)
$\&$ \wedge $ $	

Slide 4

Java Course 2001

Primjer bitwise operacija

```
thirdBit=indicators & 0x4;    // izaberi 3. bit
indicators =indicators | 0x4;  // uključi 3. bit
indicators =indicators & ~0x4; // isključi 3. bit
```

```
indicators | = 0x4; // uključi 3. bit
indicators & = ~0x4; // isključi 3. bit
```

Slide 6

Java Course 2001

Bitwise operacije

❖ AND

❖ $1001_2 \& 0011_2 = 0001_2$

❖ OR

❖ $1001_2 | 0011_2 = 1011_2$

❖ Exclusive OR

❖ $1001_2 \wedge 0011_2 = 1010_2$

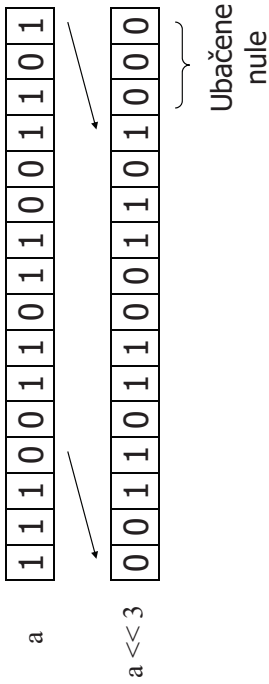
❖ 1's Complement

❖ $\sim 00001010_2 = 11110101_2$

Slide 5

Java Course 2001

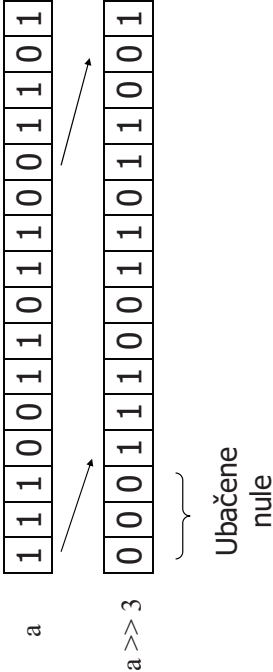
Shift lijevo <<



Slide 7

Java Course 2001

Unsigned shift desno >>>



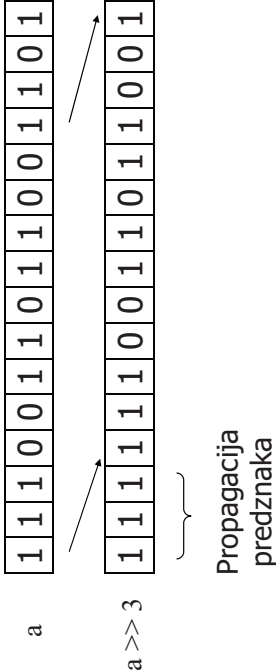
Boolean varijable

- može imati jednu od dvije vrijednosti
- true ili false

```
boolean state = true;  
state = false;
```

- boolean razlikuje se od ostalih osnovnih tipova podataka jer nije moguć cast u drugi tip, niti je za drugi tip moguće ivesti cast u boolean
- diskusija o boolean operatorima u slijedećem poglavlju

Shift desno >>



Prioritet operatora

Operator	Asocijacija	Prioritet
<div><div><div>() []</div><div>· postfix++ postfix--</div><div>unary + unary - prefix ++ prefix -- ~ !</div><div>(type) new</div><div>* / %</div><div>+ -</div><div><< >> >>></div><div>< <= > >= instanceof</div><div>= !=</div><div>&</div><div> </div><div>&&</div><div> </div><div>?:</div><div>= += -= *= /= %= &= ^= = <<= >>= >>>=</div></div></div>	<div><div>Left Assoc.</div><div>Right Assoc.</div><div>Left Assoc.</div><div>Left Assoc.</div><div>Left Assoc.</div><div>Left Assoc.</div><div>Left Assoc.</div><div>Left Assoc.</div><div>Left Assoc.</div><div>Left Assoc.</div><div>Left Assoc.</div><div>Left Assoc.</div><div>Left Assoc.</div><div>Left Assoc.</div><div>Right Assoc.</div></div>	<div><div>(High)</div><div></div><div>(Low)</div></div>

Ponovimo

- ✧ Cjelobrojni tipovi podataka su byte, short, int i long, koji zauzimaju 1, 2, 4 i 8 byte
- ✧ varijable tipa char zauzimaju 2 byte i mogu spremiti jedan Unicode znak
- ✧ Integer izrazi se računaju upotrebom 64-bitnih operacija za varijable tipa long, a upotrebom 32-bitnih operacija za sve ostale cjelobrojne tipove

Slide 12

Java Course 2001

Java

Petlje i logika - 1



Ponovimo

- ✧ Cast operator se po potrebi automatski umeće kod op= dodjeljivanja
- ✧ Floating point tipovi su float i double (4 i 8 bytes)
- ✧ vrijednosti koje su van opsega floating point tipova reprezentiraju se vrijednostima koje u prikazu su Infinity or -Infinity

Slide 13

Java Course 2001

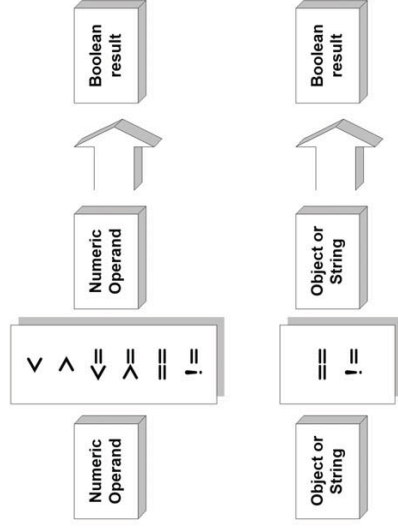
Relacijski operatori

Operator	Korištenje	Vraća true ako je:
>	op1 > op2	op1 je veći od op2
>=	op1 >= op2	op1 je veći ili jednak op2
<	op1 < op2	op1 je manji od op2
<=	op1 <= op2	op1 je manji ili jednak op2
==	op1 == op2	op1 i op2 su jednaki
!=	op1 != op2	op1 i op2 nisu jednaki

Slide 2

Java © - Eugen Mudnić

Izvođenje usporedbe



Slide 3

Java © - Eugen Mudnić

Relacijski operatori

- ❖ Svaki operator daje vrijednost **true** ili **false**
- ❖ Za spremanje rezultata usporedbe koristite varijablu tipa boolean

```
boolean state=false;
state= x-y < a+b;
```

Slide 4

Java © - Eugen Mudnić

if-else kontrolna struktura

- ❖ Opća sintaksa:

```
if (uvjetni izraz)
    statement or compound statement;
else //optional
    statement or compound statement;
```

```
if (a>10)
    c=1;
```

```
if (a>10)
    c=0;
else
    c=1;
```

Slide 5

Java © - Eugen Mudnić

Blokovi naredbi (Statement blocks)

```
if (conditional expression)
{
    statement1;
    statement2;
    ...
    statementn;
}
else
{
    statement1;
    statement2;
    ...
    statementn;
}
```

Slide 6

Java © - Eugen Mudnić

Ugnježdeni (nested) if-else

```
if (conditional expression)
    if (conditional expression)
        // ...
        statement;
```

```
if (conditional expression) statement;
else if (conditional expression) statement;
// ...
else if (conditional expression) statement;
else statement;
```

if-else primjeri

- ❖analiziraj, prevedi, izvrši
- ❖Ch3\01 If-Else\NumberCheck.java
- ❖Ch3\02 Nested-If\NumberCheck.java

Logički (boolean) operatori

Op.	Korištenje	Vraća true ako je:
&&	op1 && op2	op1 i op2 oba true, uvjetno(conditionally) računa op2 (uvjetni(conditional) AND)
	op1 op2	op1 ili op2 su true, uvjetno računa op2 (uvjetni OR)
!	! op	op je false (logička negacija)
&	op1 & op2	op1 i op2 su oba true, uvijek evaluiira i op1 i op2 (logički AND)
	op1 op2	op1 ili op2 su true, uvijek evaluiira op1 i op2 (logički OR)
^	op1 ^ op2	Ako su op1 i op2 različiti, tj ako je jedan od operanada true a drugi false

Logički (boolean) operatori

```
if(symbol >= 'A' && symbol <= 'Z') // Is it a capital letter
    System.out.println("You have the capital letter " + symbol);
```

```
if(age <16 || age>65)
    ticketPrice *=0.9;
```

```
if(!(age >=16 && age<=65))
    ticketPrice *=0.9;
```

```
if(count>0 && total/count >5)
    // Do something
```

Bez greške
ako je count
= 0

Logički (boolean) operatori

- ❖ U nekim slučajevima drugi operator u izrazu s uvjetnim operatorom neće biti evaluiran. Razmortimo slijedeći kod::

```
(numChars < LIMIT) && (...)
```

- ❖ desni operand može imati popratne efekte poput čitanja iz datoteke, ažuriranja neke vrijednosti ili izvođenja izračuna !!!

Slide 11

Java © - Eugen Mudnić

Logički (boolean) operatori

- ❖ primjeri:

- ❖ [Ch3\03 DecipheringCharacters\01_TheHardWay\LetterCheck.java](#)
- ❖ [Ch3\03 DecipheringCharacters\02_TheEasyWay\LetterCheck.java](#)
- ❖ [Ch3\03 DecipheringCharacters\03_Trivially\LetterCheck.java](#)

Slide 12

Java © - Eugen Mudnić

Uvjetni (ternarni) operator

Expr1 ? Expr2 : Expr3 (operator s 3 izraza)

Uvjetni
operator

```
max = x > y ? x : y;
```

if (x > y)
max = x;
else
max = y;

- ❖ primjer:

[Ch3\04 ConditionalPlurals\ConditionalOp.java](#)

Slide 13

Java © - Eugen Mudnić

Switch kontrolna struktura

```
switch ( test expression )  
{
```

```
    case value1 :  
        statement 1;  
        statement 2;  
        break;  
    optionalno
```

```
    case value1 :  
        statement 1;  
        statement 2;  
        break;  
    konstantna  
    vrijednost
```

```
    default:  
        statement;  
        break;
```

```
}
```

Slide 14

Java © - Eugen Mudnić

Mora proizvesti
rezultat tipa char,
byte, short ili int

Izvršava se kad
testni izraz nije ni
value1, niti value2

Switch kontrolna struktura

```
char yesNo = 'N' ;
...
switch(yesNo)
{
    case 'N' :
        case 'n' :
            System.out.println("No selected ");
            break;
    case 'Y' :
        case 'y' :
            System.out.println("Yes selected ");
            break;
}
```

Slide 15

Java © - Eugen Mudnić

Doseg varijabli

```
{
    int a = 1;
    // Reference to a is OK here
    // Reference to b here is an error
    {
        // Reference to a is OK here
        // Reference to b here is still an error
        int b = 2;

        // References to a and b are OK here - b exists now
    }

    // Reference to b here is an error - it doesn't exist
    // Reference to a is still OK
}
}
```

✧ primjer: Ch3\05 Scoping\Scope.java

Slide 17

Java © - Eugen Mudnić

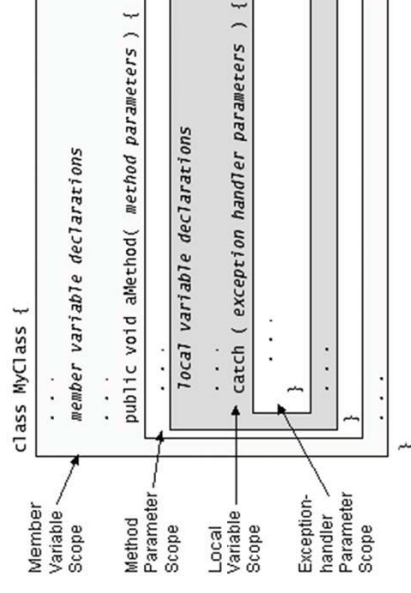
Doseg varijabli (Variable Scope)

- ✧ Varijable deklarirane unutar metoda nazivamo **lokalne varijable**
- ✧ **lokalne varijable** su dostupne jedino unutar metode
- ✧ nisu uvijek dostupne u svim dijelovima metode u kojoj su deklarirane

Slide 16

Java © - Eugen Mudnić

Kategorije dosega(scope categories)



Slide 18

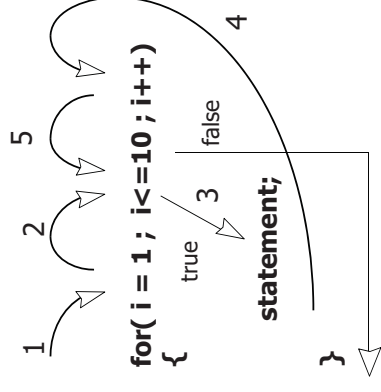
Java © - Eugen Mudnić

Java

Petlje i logika – 2



For petlja



Slide 3

Java Course 2001

Petlje – for petlja

✧ Višestruko izvršavanje izraza ili bloka izraza

```
for ( <exp1> ; < exp2> ; < exp3> )
```

```
{
    // statements
}
```

<exp1> : inicijaliziraj kontrolnu varijablu
<exp2> : provjeri kontrolnu varijablu
<exp3> : modificiraj kontrolnu varijablu

```
s = 0;
for (i=1; i<=10; ++i)
{
    s = s + i;
}
```

Slide 2

Java Course 2001

While petlja

```
while ( expression )
```

```
{
    // statements
}
```

petlja se izvršava dok je vrijednost logičkog izraza true

```
i = 1; s = 0;
while (i <= 10)
{
    s = s + i;
}
```

Slide 4

Java Course 2001

Do while petlja

```
do
{
    // statements
} while ( expression)
```

izvršava se bar jednom i dok je logički izraz true

```
i = 1; s = 0;
do
{
    s = s + i;
} while( i<=10 )
```

Slide 5

Java Course 2001

Petlje - primjeri

❖ primjeri:

- ❖ [Ch3\06_Loops\01_For\01_ForLoop\ForLoop.java](#)
- ❖ [Ch3\06_Loops\01_For\02_FloatingValues\ForLoop.java](#)
- ❖ [Ch3\06_Loops\02_While\WhileLoop.java](#)
- ❖ [Ch3\06_Loops\03_DoWhile\DoWhileLoop.java](#)

Slide 7

Java Course 2001

Floating point for petlja

- ❖ Moguće je u brojaču koristiti floating point vrijednosti

```
for(double radius = 1.0; radius <= 2.0; radius += 0.2)
{
    System.out.println("radius = " + radius + " area = "
        + Math.PI*radius*radius);
}

// slijedi beskonačna petlja

for(double radius = 1.0; radius != 2.0; radius += 0.2)
{
    System.out.println("radius = " + radius + " area = "
        + Math.PI*radius*radius);
}
```

Nikada ne koristi test
(== ili !=) koji ovisi o
egzaktnoj vrijednosti
floating point vrijednosti

Slide 6

Java Course 2001

Ugniježdene petlje

- ❖ Moguće je gnijezditi petlje jednu unutar druge do proizvoljne duljine

```
int k = 500;
for (int i=1; i<10 ; i++ )
{
    for (int j=1; j<5 ; j++ ){
        while( k>100 ){
            k--;
        }
    }
}
```

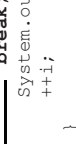
Slide 8

Java Course 2001

Break naredba

✧ Za iskakanje iz petlje (break out) (for, while ,do while). Oblik :**break [label]** ;

```
int i = 1;
while (true) {
    if (i == 3)
        break;
    System.out.println("This is a " + i + " iteration");
    ++i;
}
```



Slide 9

Java Course 2001

Continue naredba

✧ za preskočiti trenutnu iteraciju.
Oblik: **continue [label]** ;

```
for (i=0; i<=5; ++i) {
    if (i % 2 == 0)
        continue;
    System.out.println("This is a " + i + " iteration");
}
```

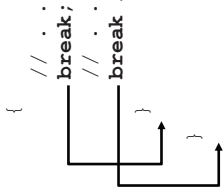


Slide 11

Java Course 2001

Break s labelom (labeled break statement)

```
labelBlock1 :
for { . . . } // Block1
{
    while( . . . ) // Block2
    {
        // . . .
        break;
        // . . .
        break labelBlock1;
    }
}
```



Slide 10

Java Course 2001

The continue statement

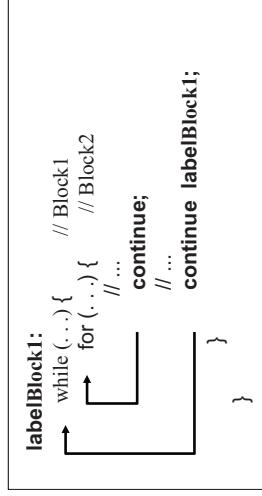
```
i = 0;
while (i <= 5) {
    ++i;
    if (i % 2 == 0)
        continue;
    System.out.println("This is a odd iteration - " + i);
}
```



Slide 12

Java Course 2001

Continue s labelom (labeled continue Statement)



Slide 13

Java Course 2001

Java

Arrays and strings -1 (nizovi i znakovni nizovi)



Petlje primjeri

❖ analiziraj, prevedi, izvrši !

- ❖ [Ch3\07_NestedLoops\01_Factorial\Factorial.java](#)
- ❖ [Ch3\07_NestedLoops\02_LabelledContinue\Factorial.java](#)
- ❖ [Ch3\08_Primes\I\Primes.java](#)
- ❖ [Ch3\08_Primes\II\Primes.java](#)
- ❖ [Ch3\08_Primes\III\Primes.java](#)

Slide 14

Java Course 2001

Nizovi-Arrays

❖ Što su Nizovi (Arrays)?

- ❖ Kolekcija referenci ili primitivnih vrijednosti
- ❖ Svaka referenca ili vrijednost mora biti istog tipa
- ❖ Cijela kolekcija ima jedan naziv
- ❖ Pojedine reference (vrijednosti) nazivamo elementimaniza
- ❖ Elementima pristupamo preko pozicije

Slide 2

Java Course 2001

Arrays

- ❖ Three step process
 - ❖ Declare an array variable
 - ❖ Create a new array "object" and assign the array to the array variable
 - ❖ Store values or objects in the array
- ❖ In Java, arrays are "object" or reference types in their own right, regardless of what they store

Slide 3

Java Course 2001

Nizovi: deklariranje varijable niza

- ❖ Za deklaraciju varijable niza potrebno je specificirati:
 - ❖ Tip elementa koji spremamo u niz
 - Može biti bilo koji tip reference(objekta) ili primitivne vrijednosti
 - ❖ Naziv cijele kolekcije
 - po pravilima za nazive varijabli
 - ❖ Prazne uglate zagrade poslije naziva ili tipa

```
double numbers[ ];  
Button[ ] buttonBar;  
int[] primes;
```

preferirano

Slide 4

Java Course 2001

Nizovi: kreiranje niza

- ❖ Niz kreiramo poput bilo kojeg drugog objekta
- ❖ Specijalna sintaksa za new:
 - `primes = new int[10];`
 - `buttonBar = new Button[10];`
- ❖ Koriste se uglate umjesto okruglih zagrada (brackets[], no parentheses)
- ❖ Ovo je konstruktor niza, nije konstruktor objekta
- ❖ `myArray` pohranjuje 10 integer brojeva
- ❖ `buttonBar` je niz od 10 referenci na `Button` objekte

Slide 5

Java Course 2001

Nizovi: pohranjivanje vrijednosti

- ❖ Elementi su numerirani od 0 do `length-1`
- ❖ Svaki niz ima `public` polje, `length`, u kojem je pohranjen broj elemenata u nizu
- ❖ Korištenje petlje za popunjavanje vrijednosti

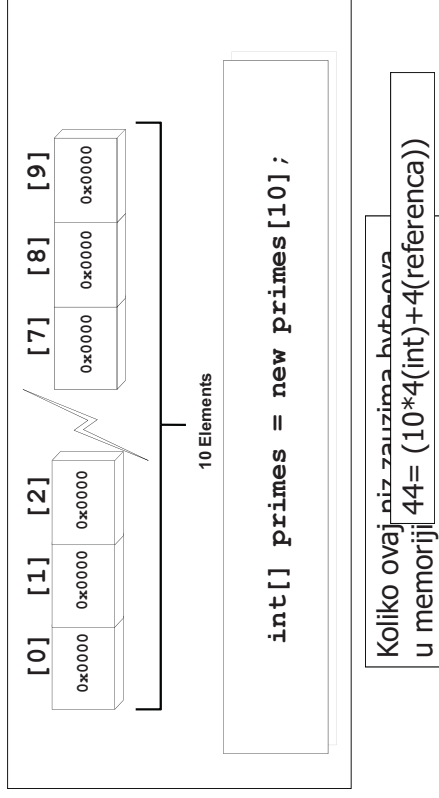
```
primes[ 3 ] = 10;
```

```
for (int i = 0; i < myArray.length; i++) {  
    myArray[ i ] = (i + 1) * 10;  
}
```

Slide 6

Java Course 2001

Deklaracija i definicija u jednom izrazu



Slide 7

Java Course 2001

Pristup elementu niza

```
num=primes[ 3 ];
```

- ❖ Vrijednost indeksa je pozitivna vrijednost tipa **int** ili izraz koji daje pozitivan rezultat tipa **int** (byte, short daju rezultat tipa int)
- ❖ Java provjerava vrijednost indeksa : Ako je indeks van dozvoljenih granica baca se exception tipa `IndexOutOfBoundsException`

Slide 9

Java Course 2001

Inicijalizacija & nizovi referenci

- ❖ Elementi u tek kreiranom nizu su:
 - ❖ Nula, ako su numerički (čak ako se radi o lokalnom nizu)
 - Za razliku od lokalnih varijabli koje je potrebno inicijalizirati
 - ❖ null, ako su elementi reference.
- ❖ Potrebno je objekte kreirati zasebno !

```
for (int i = 0; i < buttonBar.length; i++) {  
    buttonBar[ i ] = new Button("Button " + i);  
}
```

Slide 8

Java Course 2001

Ponovno korištenje (reusing) varijabli niza

- ❖ Varijabla niza i sam niz su odvojeni entiteti

```
int[] primes = new int[10]; Alociraj niz od 10 elemenata  
...  
primes = new int[50]; // Alociraj niz od 50 elemenata
```

Varijabla **primes** referira na novi niz tipa int koji je potpuno neovisan od prvog niza. Prethodni niz se odbacuje !

Slide 10

Java Course 2001

Inicijalizacija nizova

```
// initialize & declare
int[] primes= {2,3,5,7,11,13,17}; // niz od 7 elemenata
long[] even= {2,4,6,8,10}; // niz od 5 elemenata
```

```
int[] primes= new int[100];
primes[0]=2;
primes[3]=7;
```

Slide 11

Java Course 2001

Nizovi - primjer

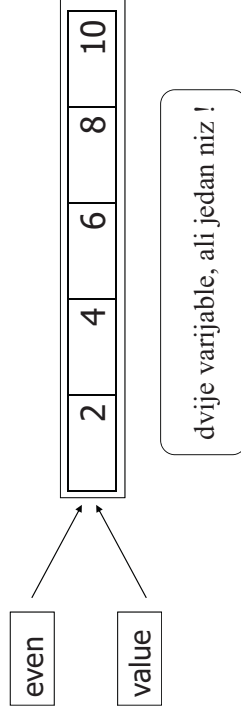
❖ [Ch4\01_MorePrimes\MorePrimes.java](#)

Slide 13

Java Course 2001

Inicijalizacija nizova

```
long[] even= {2,4,6,8,10};
long[] value = even;
```



Slide 12

Java Course 2001

Niz nizova (arrays of arrays)

❖ Deklaracija korištenjem višestrukih uglatih zagrada

```
double [][] yMatrix= new double[3][5];
❖ yMatrix has 3 rows, 5 columns
```

❖ Adresiranje pojedinog elementa pomoću dva subskripta

```
yMatrix[1][4] = 31.4
```

- ❖ Stavi vrijednost 31.4 u zadnji element drugog retka
- ❖ zapamti: nizovi su numerirani 0 do length-1

Slide 14

Java Course 2001

Niz nizova

```
int[][] scores = new int[3][2];
```

	[0]	[1]
[0]	50	100
[1]	0	735
[2]	12389	7

```
scores[0][0] = 50;  
scores[0][1] = 100;  
scores[1][0] = 0;  
scores[1][1] = 735;  
scores[2][0] = 12389;  
scores[2][1] = 7;
```

✧ primjer: Ch4\02 WeatherFan\WeatherFan.java

Slide 15

Java Course 2001

Niz nizova različite duljine

```
float[][] samples; //deklariraj niz nizova  
samples = new float[3][]; //definiraj tri elementa od kojih je svaki niz  
samples[0] = new float[2];  
samples[1] = new float[4];  
samples[2] = new float[10];
```

```
samples.length is 3  
samples[0].length is 2  
samples[1].length is 4  
samples[2].length is 10
```

samples[0]

--	--

samples[1]

--	--	--	--

samples[2]

--	--	--	--	--	--	--	--	--	--

Slide 16

Java Course 2001

Multi-dimenzionalni nizovi

```
long[][][] beans=new long[5][10][30];
```

```
long[][][] beans= new long [3][][];
```

```
beans[0]=new long[4][];  
beans[1]=new long[2][];  
beans[2]=new long[5][];  
for(int i=0; i<beans.length;i++)  
    for(int j=0; j<beans[i].length;j++)  
        beans[i][j]=new long[(int)(1.0+6.0*Math.random())];
```

```
int[][][] array= { { {2,1},{4,4},{1,2,3} },  
                   { {7} },{4,5},{7,4} },  
                   { {2,1},{1,4},{7,7,7} } };
```

Slide 17

Java Course 2001

Niz znakova

```
char[] message = char[50];  
char[] vowels ={'a','e','i','o','u'};  
char[] name= {'B','r','a','n','d',' ','N','e','w',' ','D','a','y'};
```

Slide 18

Java Course 2001

Prosljeđivanje nizova u metode

- ✧ Nizovi se mogu prosljeđivati kao argumenti metoda

```
public int addEmUp(int[ ] ar)
{
    int sum = 0;
    for(int i = 0; i < ar.length; i++)
        sum += ar[ i ];
    return sum;
}
```

Slide 19

Java Course 2001

Java

Arrays and strings -2 (nizovi i znakovni nizovi)



Prosljeđivanje nizova iz metode

- ✧ Nizove možemo i vratiti iz metode

```
public int[ ] makeArray(int howMany)
{
    int[ ] ar = new int[ howMany ];
    for(int i = 0; i < ar.length; i++)
        ar[ i ] =1;
    return ar;
}
```

Slide 20

Java Course 2001

Korištenje stringova

- ✧ Ne mislimo na niz znakova !
- ✧ U Javi stringovi su objekti klase String koja je standardno uključena
- ✧ "Ovo je string literal !"

\n is new line character

```
System.out.println("This is \n a string constant");
System.out.println("PI or \U003C0 ");
```

Unicode π

Slide 2

Java Course 2001

Kreiranje String konstanti

Deklaracija poput
varijabli osnovnog tipa

```
String myString;  
String hisString= "His string";  
myString="My inaugural string";
```

String objekt

variable koja pohranjuje
referencu na String objekt

Slide 3

Java Course 2001

Kreiranje String konstanti

- ❖ Varijabla tipa String je referenca na String objekt
- ❖ String objekti su nepromjenjivi – sadržaj im se ne može mijenjati !!!
- ❖ znakovi u String objektu su Unicode znakovi (svaki zauzima 2 bytes !)

Slide 5

Java Course 2001

Kreiranje String konstanti

```
String myString;  
myString= "My inaugural string";  
myString= "My second string";
```

myString →

M	y		i	n	a	u	g	u	r	a	l	s	t	r	i	n	g
---	---	--	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

odbačeno

~~myString →

M	y		i	n	a	u	g	u	r	a	l	s	t	r	i	n	g
---	---	--	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

myString →

M	y		s	e	c	o	n	d	s	t	r	i	n	g
---	---	--	---	---	---	---	---	---	---	---	---	---	---	---

Slide 4

Java Course 2001

neinicijalizirana String varijabla

```
String anyString; // Neinicijalizirana varijabla
```

Ako je poslije ne
inicijaliziramo dobit ćemo
grešku prevodioca

```
String anyString=null; // Ne referencira nijedan String  
// ali je inicijalizirana
```

```
if(anyString == null)  
System.out.println("anyString ne referira ni na što");
```

Slide 6

Java Course 2001

Niz Stringova

- ❖ Isti mehanizam koji je korišten za osnovne tipove

```
String[] names = new String[5];
```

- ❖ primjer:

[Ch4\03 LuckyStars\LuckyStars.java](#)

Slide 7

Java Course 2001

usporedba String varijabli

```
string1 == string2
```

- ❖ će provjeriti da li dvije varijable referiraju na isti string
- ❖ ovo nije usporedba sadržaja stringova
- ❖ primjer (dva stringa koja su identična ali ne ista):
 - ❖ [Ch4\05 MatchStrings\01_NotTheSame\MatchStrings.java](#)

Slide 9

Java Course 2001

Spajanje stringova

Kompletno novi
String objekt

```
hisString = "First part" + " and second part";
```

```
myString = "Too many"
```

```
myString += " cooks spoil the broth";
```

Ovo ne
modificira
String
"Too many"

- ❖ primjer:

[Ch4\04 Concatenation\JoinStrings.java](#)

Slide 8

Java Course 2001

usporedba String varijabli

- ❖ korištenjem String metode equals()

```
string2.equals(string1)
```

- ❖ primjer:

❖ [Ch4\05 MatchStrings\02_Identity\MatchStrings.java](#)

Slide 10

Java Course 2001

Provjera početka i kraja stringa

```
String[] myString="Too many cooks spoil the broth"
```

- ❖ `myString.startsWith("Too")` vraća `true`
- ❖ `myString.startsWith("TOO")` vraća `false`
- ❖ `myString.endsWith("oth")` vraća `true`

Slide 11

Java Course 2001

Pristup znakovima stringa

- ❖ Koristi `string1.charAt(position)`
- ❖ pozicija je indeks tipa **int**
- ❖ indeks manji od 0 ili veći od zadnje pozicije – baca se exception
- ❖ primjer:
 - ❖ [Ch4\07 StringCharacters\StringCharacters.java](#)

Slide 13

Java Course 2001

Usporedba stringova

```
String[] string1="mad dog";  
String[] string2="mad cat";
```

- ❖ **`string1.compareTo(string2)`**
 - ❖ uspoređuje `string1` sa `string2`.
 - ❖ Vraća 0 ako su jednaki, 1 ako je `string1` veći od `string2`, -1 ako je manji
- ❖ `string1.compareTo(string2)` vraća `true`
- ❖ primjer:
 - ❖ [Ch4\06 OrderingStrings\SequenceStrings.java](#)

Slide 12

Java Course 2001

Pretraživanje stringa

```
String text="Mad dog again";  
int index = 0;  
  
index =text.indexOf('a'); // 1  
index =text.lastIndexOf('a'); // 10  
index =text.indexOf('a',4); // 8  
index ="Good cat".indexOf('o'); // 1  
  
index =text.indexOf('c'); // -1 !!!
```

Slide 14

Java Course 2001

Traženje substringa

```
String text="Mad dog again";
int index = 0;
index =text.indexOf("ad"); // 1
index =text.lastIndexOf("ga"); // 9
index =text.indexOf("og",2); // 5
index =text.indexOf("cat"); // -1 !!!
```

❖ primjer:

❖ [Ch4\08 FindCharacters\FindCharacters.java](#)

Slide 15

Java Course 2001

Modificiranje objekta tipa String

❖ Metoda za kreiranje novog string objekta koji je modificirana verzija postojećeg string objekta

```
String text=" banking account ";
String newText=text.replace('b','B');
newText=newText.trim();
```

Slide 17

Java Course 2001

Ekstrakcija substrings

```
String place = "Palm Springs";
String lastWord = place.substring(5); // Springs
String segment + place.substring(7,11); // ring !
```

❖ indeks van granica - exception

❖ primjer:

❖ [Ch4\09 ExtractSubstring\ExtractSubstring.java](#)

Slide 16

Java Course 2001

Niz znakova od String objekta

```
String text = "To be or not to be";
char[] textArray = text.toCharArray();
// kopira znakove iz text u textArray
text.getChars(9,12,textArray,0);
```

Slide 18

Java Course 2001

String objekt iz niza znakova

```
char[] textArray = {'T', 'o', ' ', 'b', 'e', ' ', 'o', 'r', ' ',  
                    'n', 'o', 't', ' ', 'o', 'b', 'e'};  
String text1 = String.valueOf(textArray);
```

✧ . . .

Slide 19

Java Course 2001

Kreiranje StringBuffer objekta

```
StringBuffer aString = new StringBuffer("Nice string");  
  
StringBuffer myString= null;  
myString = new StringBuffer("Second nice string");  
  
myString=aString;
```

Slide 21

Java Course 2001

StringBuffer objekt

- ✧ String objekte ne možemo mijenjati (immutable strings)
- ✧ StringBuffer objekt možemo direktno mijenjati (mutable strings)
- ✧ Koristimo StringBuffer kada intenzivno modificiramo stringove – dodavanje, brisanje i zamjena substringova u stringu

Slide 20

Java Course 2001

Kapacitet StringBuffer objekta

```
StringBuffer aString = new StringBuffer("Nice string again");
```

aString.length() je 17 (Unicode characters)

N	i	c	e		s	t	r	i	n	g	a	g	a	i	n					
---	---	---	---	--	---	---	---	---	---	---	---	---	---	---	---	--	--	--	--	--

aString.capacity() je 33
(string lenght +16)

memorijski
spremnik

Slide 22

Java Course 2001

Kapacitet StringBuffer objekta

```
StringBuffer aString = new StringBuffer(50);  
int theCapacity = aString.capacity();  
aString.ensureCapacity(500);
```

Slide 23

Java Course 2001

Dodavanje - StringBuffer object

```
StringBuffer aString = new StringBuffer("Nice string again");  
aString.append(" and again");  
// aString is "Nice string again and again"
```

Capacity is managed automatically

```
StringBuffer aString = new StringBuffer(); // Capacity is 16  
aString.append("To").append(" be").append(" or").append(" not").append(" to").append(" be");
```

Slide 25

Java Course 2001

Promjena duljine stringa

```
StringBuffer aString = new StringBuffer("Nice string again");  
int theLength = aString.length(); // 17  
aString.setLength(11); // "Nice string"  
aString.setLength(17); // "Nice string\u0000\u0000... \u0000"
```

$2 * 11(\text{original}) + 2 = 33$
 $17(\text{new length}) < 33$
new capacity = 33

$\backslash u0000$ are appended

Slide 24

Java Course 2001

Doadavanje

```
StringBuffer buf = new StringBuffer("The number is ");  
long number = 999 ;  
buf.append(number); // buf is "The number is 999"
```

```
StringBuffer buf = new StringBuffer("Value is ");  
boolean isOpen = false;  
buf.append(isOpen); // buf is "value is false"
```

Slide 26

Java Course 2001

Dodavanje

```
buf.append(12.34);  
buf.append("Text");
```

✧ Možete dodati bilo koji od sljedećih tipova

:

boolean, char, String, Object, int, long,
float, double, byte, short, char[]

Slide 27

Java Course 2001

Još neke StringBuffer metode

```
char ch = buf.charAt(2);  
buf.setCharAt(3,'Z');  
buf.reverse();  
...  
...  
...
```

Slide 29

Java Course 2001

Umetanje

```
buf.insert(index, "text");
```

✧ Umetnuti možete bilo koji od sljedećih tipova :

boolean, char, String, Object, int, long,
float, double, byte, short, char[]

Slide 28

Java Course 2001

Kreiranje String objekta iz StringBuffer objekta

```
StringBuffer buf = new StringBuffer("Many hands make  
light work");
```

```
String saying = buf.toString();
```

```
String saying = "Many" + " hands";
```

prevodilac prethodno
implementira kao:

```
String saying = new StringBuffer().append("Many").  
append(" hands");
```

Slide 30

Java Course 2001

Java

Klase (Classes)



Definicija klase

❖ Polja(fields)

```
xCenter = 8.4, yCenter = 12
zCenter = 2.0, radius = 1.41
-----
area()
circumference()
display()
```

❖ Metode(methods)

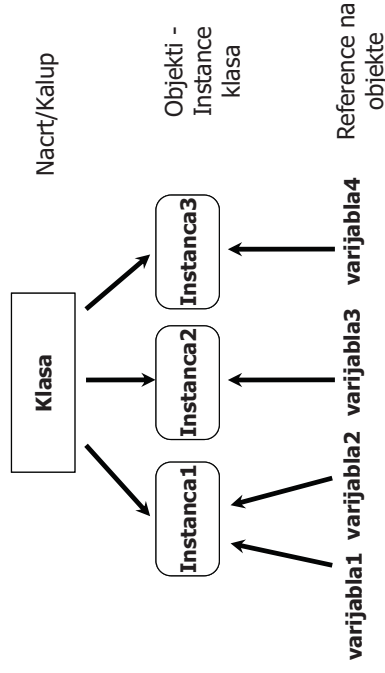
- ❖ sadržavaju izvršni kod klase i definiraju djelovanje objekata

A Sphere objekt

Slide 3

Java © - Eugen Mudnić

Što je klasa ?



Slide 2

Java © - Eugen Mudnić

Klasa – kreacija objekta

❖ Za kreaciju objekta:

- ❖ deklariraj varijablu koja će referirati na objekt:

```
Sphere mySphere;
```

- ❖ kreiraj objekt korištenjem new

```
mySphere= new Sphere();
```

```
xcenter = 0.0, ycenter = 0.0
zcenter = 0.0, radius = 0.0
-----
area()
circumference()
display()
```

mySphere →

Slide 4

Java © - Eugen Mudnić

Varijable u definiciji klase

- ❖ Dvije vrste varijabli
- ❖ Varijable instance (instance variables)
 - ❖ svaki objekt klase (instance) ima svoj primjerak varijable
- ❖ Varijable klase (class variables)
 - ❖ Klasa ima samo jedan primjerak varijable koja je dijeljena između svih instanci klase

```
class Sphere
{
    // class variable
    static double PI=3.14;

    // instance variables
    double xCenter;
    double yCenter;
    double zCenter;
    double radius;
}
```

Slide 5

Java © - Eugen Mudnić

Metode u definiciji klase

- ❖ Metode instance (instance methods)
 - ❖ izvršenje u relaciji s određenim objektom
- ❖ Metode klase (class methods)
 - ❖ deklaracija upotrebom ključne riječi static
 - ❖ mogu biti izvršavane bez postojanja objekta
 - ❖ ne mogu referirati na varijable instance

```
class Sphere
{
    static double PI=3.14;
    static int count=0;
    double radius;

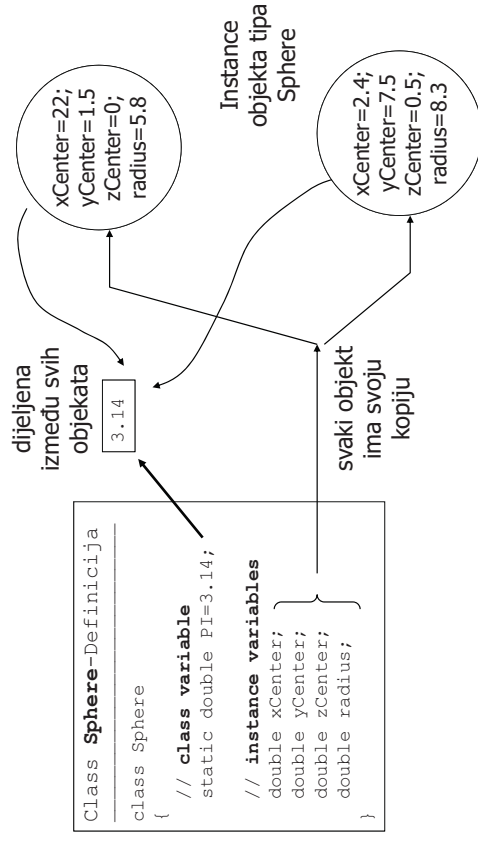
    // instance method
    double area()
    {
        return (radius*radius*PI);
    }

    // class method
    static int getCount()
    {
        return count;
    }
}
```

Slide 7

Java © - Eugen Mudnić

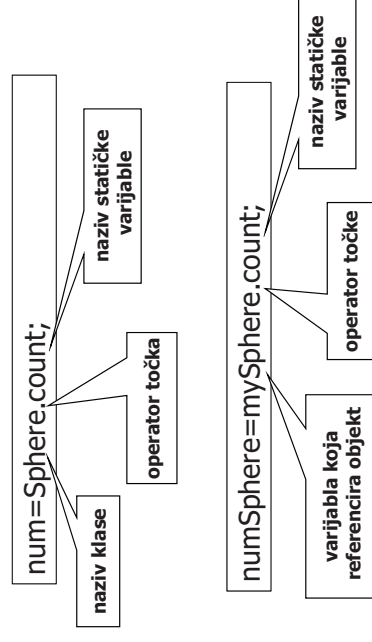
Instance i varijable klase



Slide 6

Java © - Eugen Mudnić

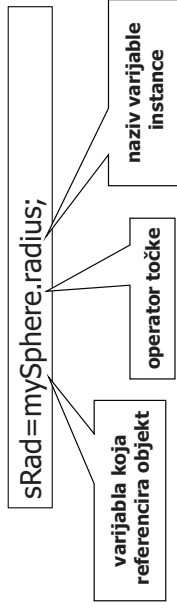
Pristup statičkim varijablama



Slide 8

Java © - Eugen Mudnić

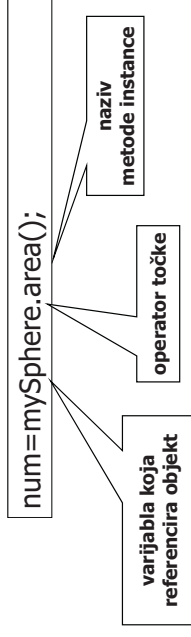
Pristup varijabli instance



Slide 9

Java © - Eugen Mudnić

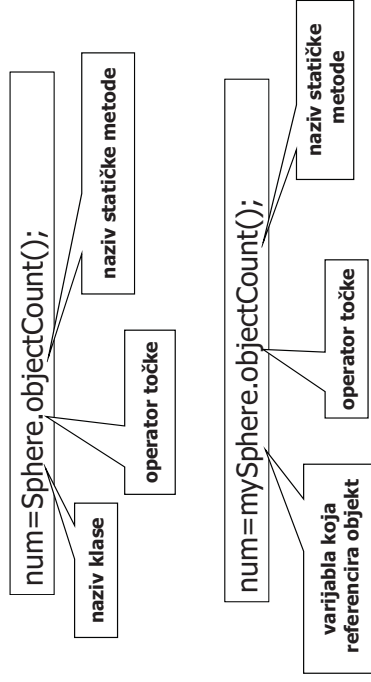
Pristup metodama instance



Slide 11

Java © - Eugen Mudnić

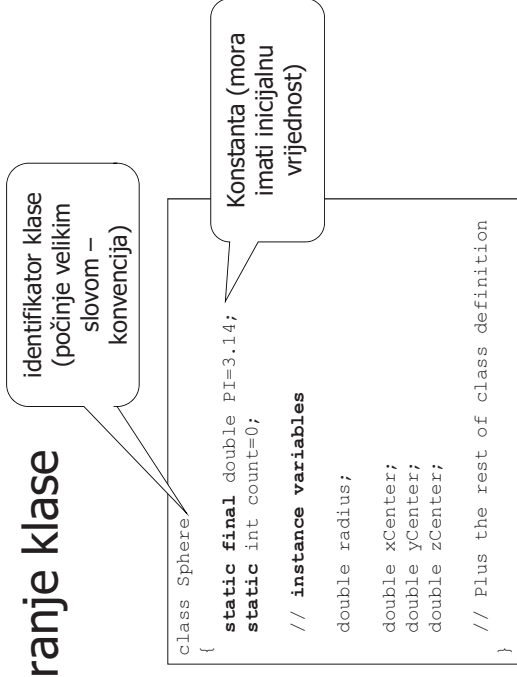
Pristup statičkim metodama



Slide 10

Java © - Eugen Mudnić

Definiranje klase

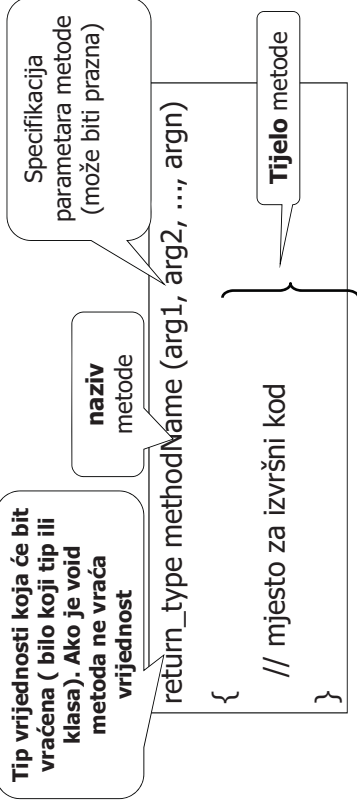


Slide 12

Java © - Eugen Mudnić

Definiranje metoda – osnovna struktura

✧ imenovani blok koda



Slide 13

Java © - Eugen Mudnić

Povratak iz metode

```
double area()
{
    return (radius*radius*PI) ;
}

void doSomething()
{
    // code ...
    return;
}

void doSomething()
{
    // code ...
}
```

Slide 14

Java © - Eugen Mudnić

Argumenti i lista parametara

- ✧ **Parametri** se pojavljuju u listi parametara u definiciji metode.
- Parametar definira tip vrijednosti i naziv preko koje je referenciramo
- ✧ **Argument** je vrijednost koju proslijeđujemo u pozivu metode.

Slide 15

Java © - Eugen Mudnić

Argumenti – lista parametara

```
public static void main (String[] args)
{
    ...
    x = obj.mean(3.0,5.0);
    ...
}

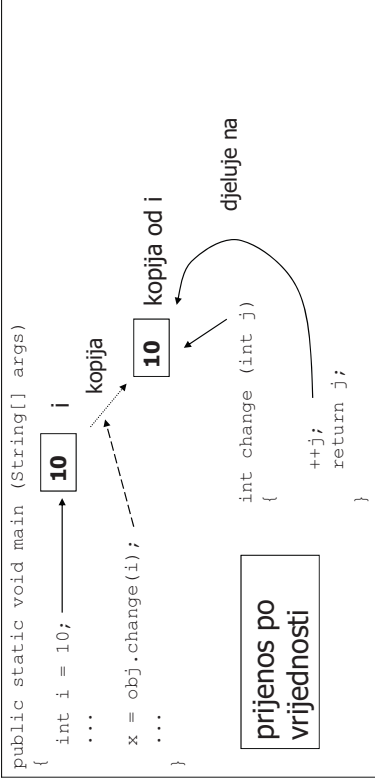
double mean ( double value1, double value2)
{
    double result = (value1+value2)/2.0;
    return result;
}
```

The diagram shows the flow of arguments. Arrows point from the values `3.0` and `5.0` in the `main` method's call to `obj.mean(3.0,5.0)` to the parameters `value1` and `value2` in the `mean` method's signature.

Slide 16

Java © - Eugen Mudnić

Prijenos vrijednosti argumenata



Slide 17

Java © - Eugen Mudnić

Definiranje metoda klase

❖ Ključna riječ

static

❖ Metode klase ne mogu referirati varijable instance

```
class Sphere
{
    // Class definition as before
    // Static method
    static int getCount
    {
        return count;
    }
}
```

Slide 19

Java © - Eugen Mudnić

Prijenos vrijednosti argumanata

❖ Kopija reference se prenosi u metodu (ne kopija objekta)

❖ Final parametri:

```
double mean ( double final value1, double final value2)
```

```
{
```

❖ prevencija modifikacije argumenta (compiler checked)

❖ koristi se i za reference na objekte

Slide 18

Java © - Eugen Mudnić

Pristupanje varijablama u metodama

```
class Sphere
{
    static final double PI = 3.14; // Varijabla klase , konst. vrijed.
    static int count = 0;         // Varijabla klase - broj objekata

    // Varijabla instance
    double radius;                // Radijus kugle(sphere)
    double xCenter;               // 3D kordinate
    double yCenter;               // centra
    double zCenter;               // kugle

    // Statička metoda koja vraća broj kreiranih objekata
    static int getCount ()
    {
        return count;
    }
    //Metoda instance za proračun volumena
    double volume ()
    {
        return 4.0/3.0*PI*radius*radius*radius;
    }
}
```

Metode klase mogu koristiti samo varijable klase

Metode instance mogu koristiti i varijable klase i varijable instance

Varijabla **this**

- ✧ Svaka metoda **instance** posjeduje varijablu **this** koja referira **objekt** nad kojim je ta metoda pozvana

```
// prevodilac ubacuje this referencu na objekt umjesto vas
return 4.0/3.0*PI*radius* radius* radius;
return 4.0/3.0*PI*this.radius* this.radius* this.radius;
```

Slide 21

Java © - Eugen Mudnić

Inicijalizacija varijabli

```
class Sphere
{
    static final double PI=3.14; // Varijabla klase - konst. vrijed.
    static int count=0;         // Varijabla klase - broj objekata

    // Varijable instance

    double radius=11.2;

    double xCenter=5.0;
    double yCenter=1.0;
    double zCenter=22.3;

    // Ostatak definicije klase
}
```

Svaki objekt tipa Sphere
će biti kreiran s navedenim
vrijednostima.

Slide 23

Java © - Eugen Mudnić

Varijabla **this**

- ✧ Ako za parametar ili lokalnu varijablu metode koristite naziv isti kao i člana klase onda za razlikovanje koristite **this**

```
void changeRadius (double radius)
{
    // promijeni varijablu instance u vrijednost prenesenu
    // preko parametra
    this.radius = radius;
}
```

Slide 22

Java © - Eugen Mudnić

Korištenje inicijalizacijskih blokova

- ✧ Blok koda koji se izvršava prije nego je kreiran i jedan objekt klase
- ✧ Dva tipa inicijalizacijskih blokova:
 - ❖ **static** inicijalizacijski blok – izvršava se samo jednom pri učitavanju klase i u njemu se mogu inicijalizirati samo statičke varijable
 - ❖ **non-static** inicijalizacijski blok – izvršava se prilikom kreiranja svakog objekta (instance klase) i stoga može inicijalizirati varijable instance
- ✧ primjeri:
 - ❖ `ch5\01_TryInitialization\01_StaticTryInitialization.java`
 - ❖ `ch5\01_TryInitialization\02_NonStaticTryInitialization.java`

Slide 24

Java © - Eugen Mudnić

Konstruktori (constructors)

- ❖ Svaki put kada se iz klase kreira objekt, poziva se specijalna metoda koju nazivamo **konstruktor**
- ❖ Ako ne definirate konstruktor za klasu, prevodilac će sam kreirati **default konstruktor** koji je prazan (nema koda)
- ❖ primarna svrha: zasebna inicijalizacija varijabli instance za objekt koji kreiramo
- ❖ Bilo koji inicijalizacijski blok izvršava se **prije** konstruktora

Slide 25

Java © - Eugen Mudnić

Konstruktori

- ❖ Dvije specijalne karakteristike
 - ❖ Konstruktor nikada ne vraća vrijednost i ne smije se u definiciji navesti povratna vrijednost (čak ni tip void)
 - ❖ Konstruktor ima uvijek isti naziv kao i klasa

Slide 26

Java © - Eugen Mudnić

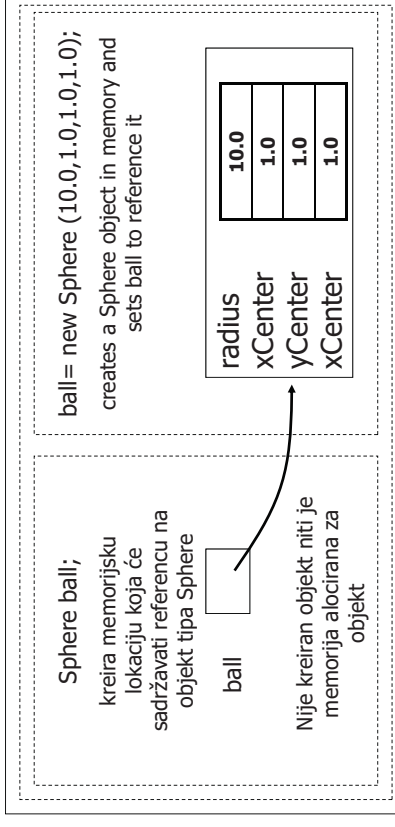
```
class Sphere
{
    static final double PI = 3.14; // Varijabla klase , konst. vrijed.
    static int count = 0;         // Varijabla klase - broj objekata

    // Varijable instance
    double radius;               // Radijus kugle(sphere)
    double xCenter;              // 3D kordinate
    double yCenter;              // centra
    double zCenter;              // kugle

    // Konstruktor klase
    Sphere(double theRadius, double x, double y, double z)
    {
        radius = theRadius;      // Postavi radijus
        // Postavi koordinate centra
        xCenter = x;
        yCenter = y;
        zCenter = z;
        ++count;                 // Povećaj brojač broja objekata
    }

    // Statička metoda koja vraća broj kreiranih objekata
    static int getCount()
    {
        return count; }
    . . . . .
}
```

Kreiranje objekta iz klase



Sphere ball=new Sphere (10.0,1.0,1.0,1.0);

Slide 28

Java © - Eugen Mudnić

Kreiranje objekta iz klase

```
Sphere ball;  
ball=new Sphere (10.0,1.0,1.0,1.0);  
newball =ball;
```

- ❖ newball referira na isti objekt kao i varijabla ball !
- ❖ **separacija varijable i objekta !**

Slide 29

Java © - Eugen Mudnić

Životni vijek objekta

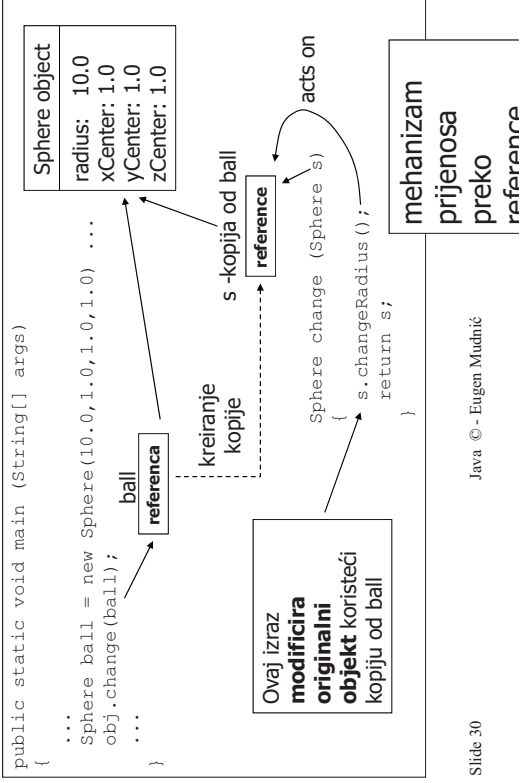
```
Sphere ball=new Sphere (10.0,1.0,1.0,1.0);
```

- ❖ Objekt postoji sve dok postoji bar jedna varijabla koja ga referencira !
- ❖ Moguće je postaviti varijablu da ne pokazuje ni na što : ball=null;
- ❖ Garbage collection – proces uklanjanja objekata
- ❖ Objekti se ne brišu u istom trenutku kad više nema referenci na njih

Slide 31

Java © - Eugen Mudnić

Prenošenje reference na objekt u metodu



Slide 30

Java © - Eugen Mudnić

Definiranje i korištenje klase

- ❖ Two source files:
 - ❖ Both files are in the same directory !
 - ❖ Directory Ch5\02_SphereClass
 - ❖ CreateSpheres.java – has main() method
 - ❖ Sphere.java – definition of the class Sphere
- ❖ compile by: javac CreateSpheres.java
- ❖ Sphere.java will be compiled automatically
- ❖ .class files are stored in the current directory (or in directory specified with -d option)

Slide 32

Java © - Eugen Mudnić

Preopterećenje metoda (overloading)

- ❖ Java dozvoljava metode s istim nazivom sve dok svaka metoda ima jedinstven skup parametara
- ❖ signatura metode : **naziv metode + tip i broj parametara**
- ❖ povratna vrijednost nije dio signature !

```
int round(double dnum);  
int round(float fnum);  
long round(float numb);  
k=round(float num);
```

Slide 33

Java © - Eugen Mudnić

Poziv konstruktora iz konstruktora

- ❖ Konstruktor može zvati konstruktor , ali **samo u prvoj izvršnoj naredbi** konstruktora
- ❖ Za poziv konstruktora iste klase koristimo sljedeći poziv :
this(...);

Slide 35

Java © - Eugen Mudnić

Višestruki konstruktori

- ❖ Konstruktori mogu biti preopterećeni baš kao i svaka metoda
- ❖ Omogućava se kreiranje objekta iz različitog skupa podataka
- ❖ Primjer:
 - ❖ directory Ch5\03_MultipleConstructors
 - ❖ [CreateSpheres.java](#)
 - ❖ [Sphere.java](#)

Slide 34

Java © - Eugen Mudnić

```
class Sphere  
{  
    // Konstruktori klase
```

```
//Konstruiraj kuglu u ishodištu  
Sphere()  
{  
    radius = 1.0;  
    // ostala polja bit će nula (pretpostavljena vrijednost)  
    ++count;  
    // Povećaj brojčak objekata  
}
```

```
//Konstruiraj kuglu u određenoj točki  
Sphere(double x, double y, double z)  
{  
    this(); //Poziv konstruktora bez argumanata  
    xCenter = x;  
    yCenter = y;  
    zCenter = z;  
}
```

```
Sphere(double theRadius, double x, double y, double z)  
{  
    this(x, y, z); // Pozovi konstruktor s tri argumenta  
    radius = theRadius; // Postavi radius  
}
```

Dupliciranje objekata korištenjem konstruktora

Java posjeduje clone() metodu (detalji u sljedećem poglavlju)

Kreiraj objekt tipa Sphere

```
Sphere eightBall = new Sphere(10.0, 10.0 ,0.0);
```

Slijedeći izraz ne kreira novi objekt !

```
Sphere newBall =eightBall;
```

Slide 37

Java © - Eugen Mudnić

Korištenje objekata

❖ primjeri:

- ❖ direktorij: Ch5\05_TryGeometry
- ❖ [TryGeometry.java](#)
- ❖ [Line.java](#)
- ❖ [Point.java](#)

Slide 39

Java © - Eugen Mudnić

Dupliciranje objekata korištenjem konstruktora

Recept: kreiramo novi konstruktor

```
Sphere(final Sphere oldSphere)
{
    radius = oldSphere.radius;
    xCenter = oldSphere.xCenter;
    yCenter = oldSphere.yCenter;
    zCenter = oldSphere.zCenter;
}
```

Sad kreiramo newBall kao odvojeni duplicirani objekt

```
Sphere newBall =new Sphere(eightBall);
```

Slide 38

Java © - Eugen Mudnić

Rekurzija

- ❖ Metoda može pozvati samu sebe – rekurzija
- ❖ Potrebno je uključiti neku logiku po kojoj će metoda prestati zvati samu sebe
- ❖ primjer:
[Ch5\06 Recursion\PowerCalc.java](#)

Slide 40

Java © - Eugen Mudnić

Java

Klase – 2



Slide 2

Java Course 2001

Učitavanje klase

- ❖ Kada kreiramo novi objekt
 - ❖ JVM provjerava da li je klasa već učitana
 - ako nije, čita odgovarajuću class datoteku
 - ❖ zatim se kreira objekt
- ❖ Ovo se naziva **dynamic loading/linking**
 - ❖ klase se učitavaju samo ako su potrebne
 - ❖ C/C++ učitava cijeli program

Slide 3

Java Course 2001

Učitavanje klase

- ❖ Svaka Java klasa ima vlastitu .class datoteku koja sadržava
 - ❖ nazive i tipove varijabli
 - ❖ nazive metoda i tipove
 - ❖ byte code metoda
- ❖ Kada pozivamo JVM interpreter navodimo naziv klase (javac HelloWorldApp)
 - ❖ JVM čita odgovarajuću class datoteku i poziva njenu *main* metodu

Slide 2

Java Course 2001

Pronalaženje klase

- ❖ kada JVM treba učitati klasu, potrebna mu je informacija gdje da počne s traženjem
- ❖ počinje traženje relativno na:
 - ❖ direktorije navedene u -classpath opciji
 - ❖ direktorije navedene u CLASSPATH environment varijabli
 - ❖ u default lokaciji

Slide 4

Java Course 2001

Name space

- ✧ Java ima hijerarhijski prostor naziva (name space)
 - ❖ onemogućeni sukobi naziva
 - ❖ svaka varijabla i metoda je dio klase
 - ❖ svaka klasa je dio paketa
 - ❖ nazivi paketa su hijerarhijski
 - java.lang
 - java.io
 - java.awt
 - java.awt.image
 - myclasses.graphics.3D
 - myclasses.games.chess

Slide 5

Java Course 2001

Što su to paketi (packages)

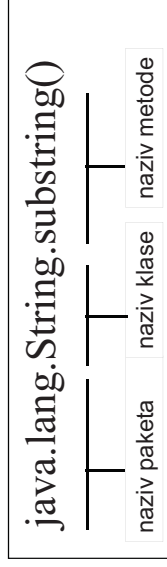
- ✧ Paketi – temeljni dio Java programa
- ✧ Paket je skup funkcionalno povezanih klasa i sučelja koji osigurava zaštitu pristupa i upravljanje prostorom naziva. Svaka klasa u Javi je sastavni dio nekog paketa
- ✧ Za dosad napisane klase implicitno smo koristili **default package** (paket bez naziva)
- ✧ **java.lang** – standardne klase, automatski dostupne programu

Slide 7

Java Course 2001

Name space

- ❖ klase, metode i varijable mogu biti pozivani korištenjem punog naziva
 - ❖ **naziv paketa**, nakon njega
 - ❖ **naziv klase**, nakon njega
 - ❖ **naziv varijable (metode)**



Slide 6

Java Course 2001

```
import naredba
```

```
class Test {
    public static void main(String args[]) {
        java.util.Vector v;
        v = new java.util.Vector();
    }
}
```

- ✧ uvijek je mogući poziv klasa, metoda, itd. korištenjem punih naziva
- ✧ koristimo import za izbjegavanje punih naziva

```
import java.util.Vector;

class Test {
    public static void main(String args[])
    {
        Vector v;
        v = new Vector();
    }
}
```

Slide 8

Java Course 2001

import naredba

- ❖ import može biti korišten za import cijelog paketa

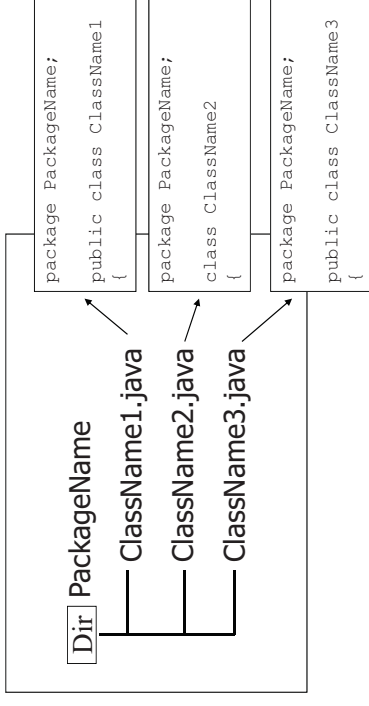
```
import java.util.*;
class Import {
    public static void main(String args[]) {
        Vector v;
        BitSet bs; //isto dio od java.util
        v = new Vector();
        bs = new BitSet();
    }
}
```

Slide 9

Java Course 2001

Paketi i struktura direktorija

- ❖ Paketi su usko povezani s strukturom direktorija u koji su pohranjeni



Slide 11

Java Course 2001

Pakiranje vaših klasa

- ❖ Dodaj package naredbu kao prvu naredbu u datoteci izvornog koda koja sadrži definiciju klase

public

- vidljiva van paketa.
- samo jedna klasa može biti public
- naziv klase mora biti jednak nazivu datoteke

package Geometry;
public class Sphere

{
//detalji definicije klase

}
naziv datoteke:
Sphere.java

Slide 10

Java Course 2001

Pakiranje vaših klasa

- ❖ **Svaka klasa** koju želite uključiti u paket (Geometry) mora sadržavati **istu package naredbu** na početku koda
- ❖ Sve datoteke za klase u paketu moraju biti snimljene u **direktorij s istim nazivom kao i naziv paketa** (Geometry)
- ❖ Sve klase koje nisu deklarirane kao **public** neće biti dostupne izvan paketa

Slide 12

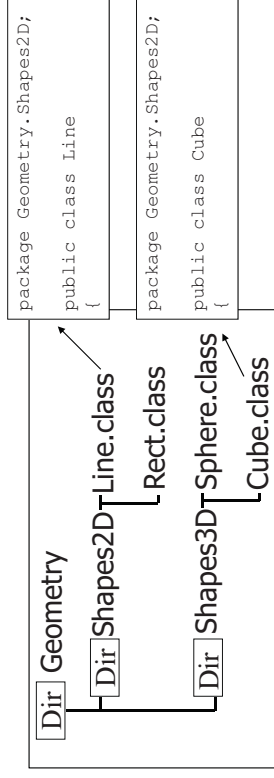
Java Course 2001

Nazivi paketa

- ❖ Paketi mogu imati višesložan naziv

```
package Geometry.Shapes3D;
```

```
package Geometry.Shapes2D;
```



Slide 13

Java Course 2001

Globalni prostor naziva

- ❖ programiranje za internet zahtijeva **globalni prostor naziva (global name space)**

- ❖ shema zasnovana na nazivima internet domena

```
hr.fesb.giga.jproject.Geometry.Shapes2d.Line.display()
```

- ❖ prvi dio je rezervirana internet domena
- ❖ drugi dio je odabran na nivou organizacije - projekta
- ❖ treći dio je individualno odabran

Slide 15

Java Course 2001

Prevođenje paketa

- ❖ Direktorij paketa mora biti poznat

prevodiocu !

- ❖ c:\JavaStuff\Geometry

- ❖ Line.java

- ❖ Point.java

Prevedi s:

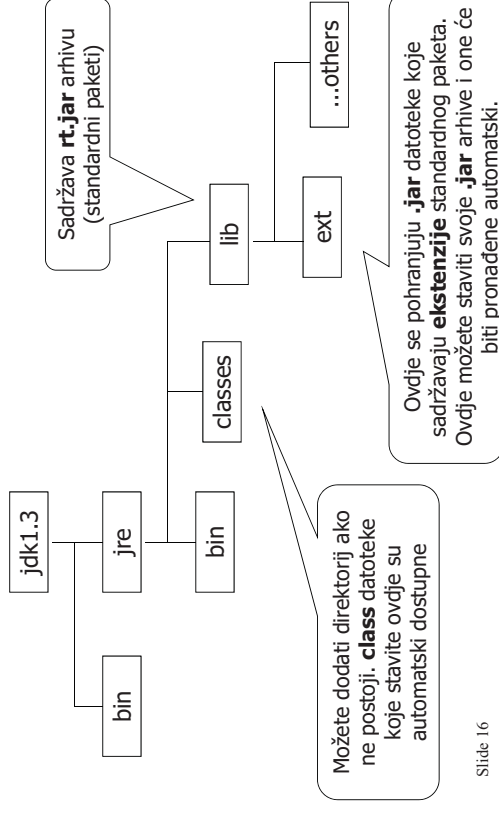
- ❖ c:\JavaStuff\Geometry>javac -classpath C:\JavaStuff Line.java

- ❖ ili c:\JavaStuff>javac Geometry\Line.java

Slide 14

Java Course 2001

Korištenje ekstenzija



Možete dodati direktorij ako ne postoji. **class** datoteke koje stavite ovdje su automatski dostupne

Ovdje se pohranjuju **.jar** datoteke koje sadržavaju **ekstenzije** standardnog paketa. Ovdje možete staviti svoje **.jar** arhive i one će biti pronađene automatski.

Slide 16

Kreiranje arhive (.jar)

- ❖ Paketi su već prevedeni
- ❖ Naredba:
 - ❖ c:\JavaStuff\>jar -cvf Geometry.jar Geometry*.*
- ❖ Ovo će kreirati komprimiranu Geometry.jar arhivu
- ❖ Da bi paket bio dostupan bilo kojem programu potrebno ga je kopirati u **ext** direktorij

Slide 17

Java Course 2001

Standardne klase koje enkapsuliraju osnovne tipove podataka

- ❖ Boolean, Character, Byte, Short, Integer, Long, Float, Double
- ❖ nalaze se u paketu java.lang
- ❖ Svaka klasa enkapsulira odgovarajući osnovni tip
 - ❖ metode: static toString(), non-static toString()
 - ❖ static final konstante MAX_VALUE, MIN_VALUE
 - ❖ POSITIVE_INFINITY, NEGATIVE_INFINITY, NaN (Double and Float)
 - ❖ static parseInt(), parseLong(),

Slide 19

Java Course 2001

Standardni paketi

- ❖ java.lang
- ❖ java.io
- ❖ java.awt
- ❖ java.swing
- ❖ java.applet
- ❖ java.util
- ❖ java.sql

Slide 18

Java Course 2001

Java

Klase – 3



Kontrola pristupa članovima klase

❖ Pristup unutar klase

- ❖ Unutar statičke metode klase može se pristupati samo statičkim članovima klase
- ❖ Unutar ne-statičke klase možete pristupati bilo kojem članu klase

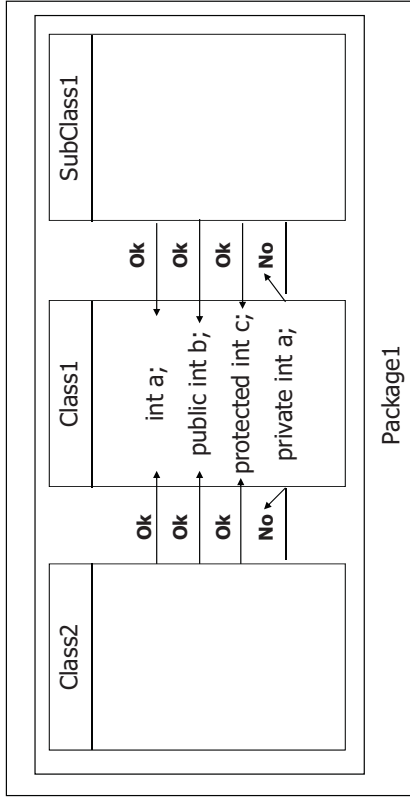
❖ Pristup metodama i varijablama klase iz druge klase ovisi o::

- ❖ atributima pristupa(access attributes)
- ❖ da li se klase nalaze u istom paketu

Slide 2

Java Course 2001

Klase unutar istog paketa



Slide 4

Java Course 2001

Korištenje atributa pristupa

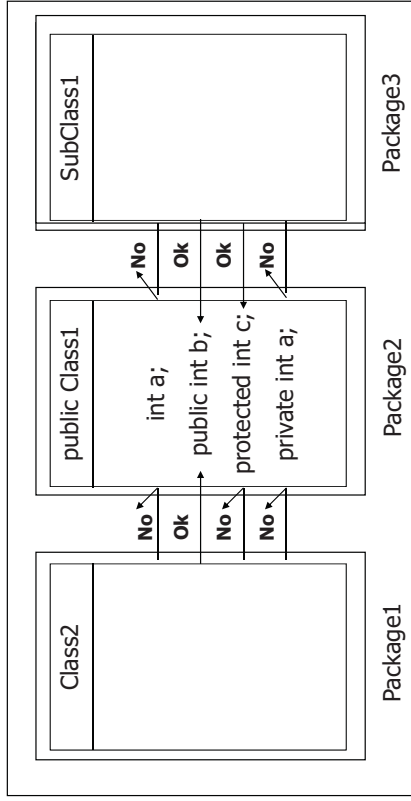
- ❖ Metode i varijable druge klase nisu uvijek dostupne

Atribut pristupa	Dozvoljeni pristup
Bez prava pristupa (default)	Iz bilo koje klase u istom paketu
public	Iz bilo koje druge klase
private	Bez prava pristupa iz bilo koje druge klase
protected	Iz bilo koje klase unutar istog paketa i iz bilo koje sub klase

Slide 3

Java Course 2001

Pristup van paketa



Slide 5

Java Course 2001

Specificiranje atributa pristupa

- ❖ Primjer:
- ❖ Ch5\07_TryPackage\Geometry\
 - ❖ Point.java
 - ❖ Line.java

Slide 6

Java Course 2001

Odabir atributa pristupa

- ❖ enkapsulirajte podatkovne članove (data members) –> private (koristi accessor i mutator metode)
- ❖ metode koje trebaju biti dostupne van klase -> public
- ❖ uvijek postoje iznimke

Slide 7

Java Course 2001

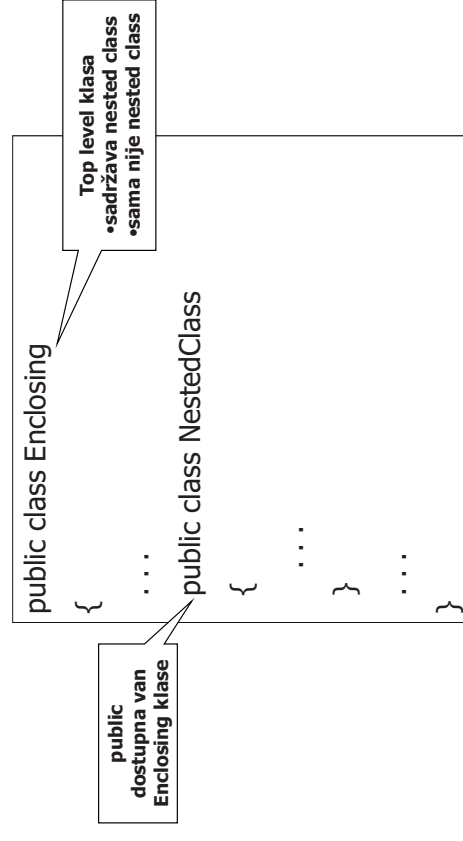
Ugniježdene klase(nested classes)

- ❖ Moguće je staviti definiciju jedne klase unutar druge klase -> **nested class**
- ❖ **nested class** je klasa koja je **član** druge klase
- ❖ nested class can itself have another class nested inside it
- ❖ nested class može imati **atribute pristupa** kao i svaki drugi član klase

Slide 8

Java Course 2001

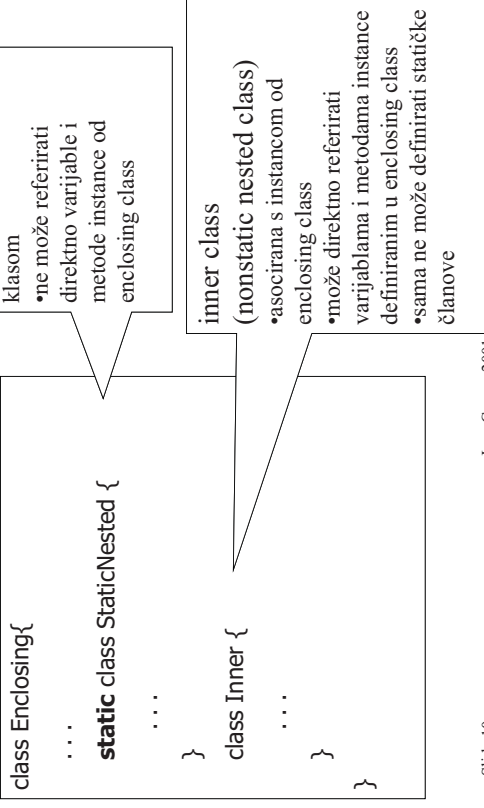
Nested classes



Slide 9

Java Course 2001

Nested classes



Slide 10

Java Course 2001

Nested classes – kreacija objekta

Enclosing enclosing = new Enclosing;

Za kreaciju jednog objekta inner klase

Enclosing.Inner inner = enclosing.new Inner();

//ili unutar metode instance

Inner inner = new Inner();

// this.Inner inner = this.new Inner();

Za kreaciju objekta static nested class

Enclosing.StaticNested example = new Enclosing.StaticNested();

Slide 12

Java Course 2001

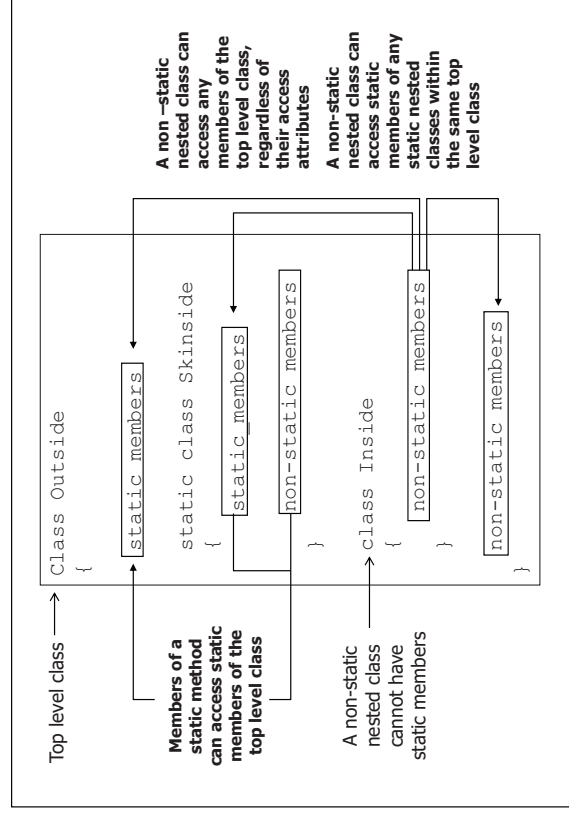
Pojmovi : Nested class , Inner class

- ❖ "Nested class" reflektira **sintaktičku relaciju** između dvije klase (sintaktički kod jedne klase pojavljuje se u kodu druge klase)
- ❖ "Inner class" reflektira **relaciju između instanci** dviju klasa (instancija od Inner klase može egzistirati samo unutar instance od Enclosing klase)

Slide 11

Java Course 2001

Više od riječi



Nested classes -primjeri

- ❖ Static nested class:
 - ❖ Ch5\08_MagicHat\01_OutOfHats
 - MagicHat.java
 - TryNestedClass.java
- ❖ Accessing top level class members
 - ❖ Ch5\08_MagicHat\02_AccessTopMembers
 - MagicHat.java
 - TryNestedClass.java
- ❖ Using a nested class outside the top level class
 - ❖ Ch5\08_MagicHat\03_FreeRange
 - MagicHat.java
 - TryNestedClass.java

Slide 14

Java Course 2001

The finalize() method

- ❖ U definiciju klase moguće je uključiti finalize() metodu
- ❖ finalize() metoda poziva se neposredno prije nego što je objekt uništen i prostor koji je zauzimao bude oslobođen (ne kada objekt izlazi izvan dosega)
- ❖ Ne pouzdajte se u finalize() metodu !

Slide 16

Java Course 2001

Lokalne nested classes

- ❖ Klasa definirana unutar metode – local nested class (local inner class)
- ❖ Moguće je kreirati objekte lokalne inner klase samo lokalno – unutar metode
- ❖ local inner class može referirati samo final varijable metode

Slide 15

Java Course 2001

Native methods

- ❖ Moguće je u klasu uključiti metodu koja je implementirana u nekom drugom programskom jeziku poput C ili C++ jezika

```
// Declare method that is not in Java
public native long getData();
```

- ❖ Metoda neće imati tijelo u Javi
- ❖ Standardni API za implementaciju native metoda u C je JNI – Java Native Interface
- ❖ Program nije više prenosiv
- ❖ Korištenje native metoda nije moguće unutar appleta (sigurnosni zahtjevi)

Slide 17

Java Course 2001