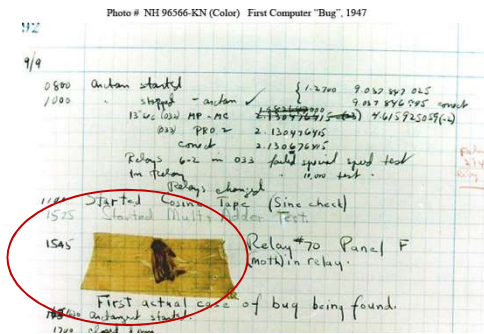


# [ Programiranje ]



Parser

Web Page:  
+ <img src=  
• <a href=  
+ <script src=



Parser

Web Page:  
• <img src=  
• <a href=  
• <script src=

Nastava: prof.dr.sc. Dražena Gašpar

Datum: 13.03.2018.



# Tema: Tipovi podataka

# Pojam podatka

- Svojstva objekata i njihovih odnosa u prostoru i vremenu izražavamo pomoću podataka.
- Podatak je nematerijalne prirode
- Primarno postoji u našim mislima
- Pridružen je nekom konceptu, odnosno značenju kojim opisujemo svojstva
- Svojstva (objekt, odnosi i koncept) su promjenjivi u vremenu.

# Pojam podatka

## PODATAK

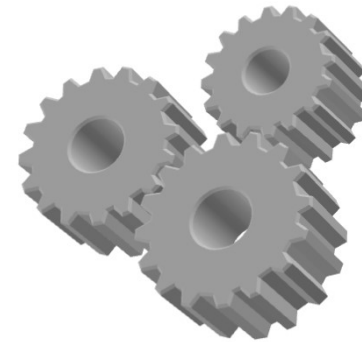
Apstraktna struktura sastavljena od:

- značenja (naziv i opis značenja određenog svojstva)
- Vrijednosti (mjera i iznos)
- Vremena



Što je  
VARIJABLA?

Čemu služi?



# [ Domena: tip varijable ]

---

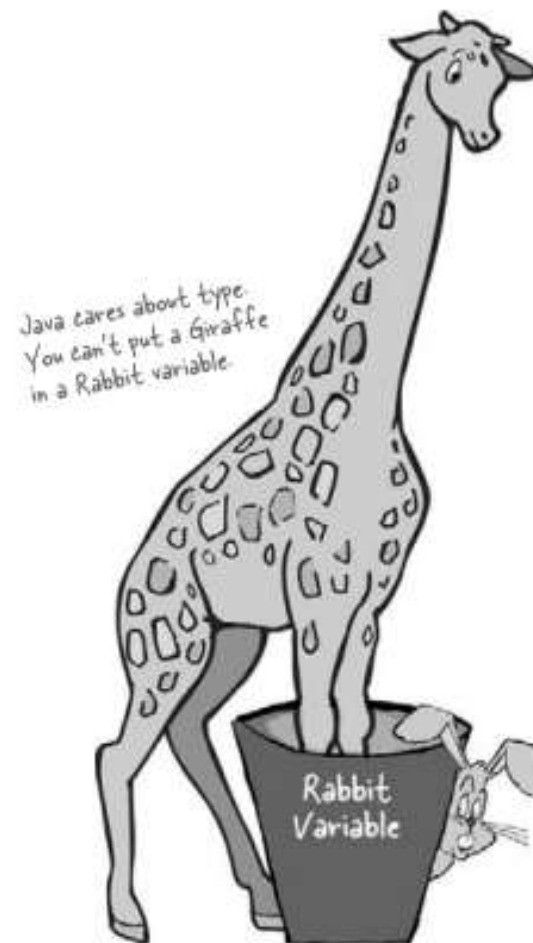
Pored imena i lokacije, svaka varijabla je definirana i tipom varijable. Tip varijable je imenovani skup (npr. cjelobrojni, znakovni i sl.) svih vrijednosti koje varijabla može poprimiti i za koji su dozvoljene odgovarajuće operacije.

Svaki kompilator mora znati:

- koji izrazi su valjani (validni)
- Kojeg je tipa rezultat svakog valjanog izraza
- Ukupan broj tipova je ZATVOREN SKUP



Što se podrazumijeva  
pod  
TIPOM PODATAKA?



# [ Definiranje ]

---

Procesor rezerviranje memorije realizira preko tipova podataka i varijabli.

Tip podataka označava potrebnu količinu memorije za pohranu podatka, kao i vrstu podatka koja će biti pohranjena na toj memorijskoj lokaciji.

Broj byte-ova rezerviranih za tip podataka ovisi o programskom jeziku koji se koristi za pisanje programa i vrsti računala na kojoj je program kompiliran



## Definicija: Tip podataka

Neka je zadan skup  $T$  sačinjen od  $n$  proizvoljnih apstraktnih objekata:

$$T := \{v_1, \dots, v_n\}, n > 1$$

Ukoliko su svi objekti istorodni u smislu da se u okviru nekog programskog jezika na njih može primjenjivati jedan određeni skup operatora, onda se  $T$  naziva tip podataka.

# [ Definicija ]

---

Dodjeljivanje tipova podataka ima osnovnu namjenu dati određeno semantičko značenje skupu bitova koji inače nemaju nikakvo značenje. Tip podataka je obično povezan bilo sa vrijednostima u memoriji ili s objektima kao što su varijable.

Kako se bilo koja vrijednost jednostavno sastoji od skupa bitova u računalu, hardver ne pravi razliku čak ni između memorijskih adresa, koda instrukcija, znakova, cjelobrojnih i brojeva s pomičnim zarezom.

Tipovi podataka informiraju programe i programere kako bi trebali tretirati te puke bitove.

# [ Tipovi podataka ]

---

Sustav tipova podataka je specifičan za svaki programski jezik i određuje načine na koji se programi pisani u tom programskom jeziku mogu ponašati, čineći istovremeno ponašanje izvan tih pravila ilegalnim i označavajući ga kao programsku pogrešku.

# Tipovi podataka – osnovna podjela

1. Elementarni ili primitivni
2. Složeni ili korisnički definirani

Elementarni ili primitivni tipovi podataka su osnovni tipovi koji su već «ugrađeni» u, odnosno predstavljaju neodvojivi dio određenog programskog jezika, a koriste se u kao temelj za formiranje složenih ili korisnički definiranih tipova podataka.

Složeni ili korisnički definirani tipovi podataka su takvi tipovi koji mogu biti konstruirani u konkretnom programskom jeziku na bazi elementarnih tipova podataka tog jezika i drugih složenih tipova podataka.

# [ Elementarni tipovi podataka ]

Danas se najviše koriste 32-bitna računala i memorijski zahtjevi u odnosu na elementarne tipove podataka uobičajeno se izražavaju u slijedećim memorijskim jedinicama:

- byte – 8 uzastopnih bitova memorije
- polu-riječ (engl. *half-word*) – 16 uzastopnih bitova = 2 byte-a
- riječ (engl. *word*) – 32 uzastopna bita = 4 byte-a
- dvostruka riječ (engl. *double word*) – 64 uzastopna bita = 8 byte-ova
- četverostruka riječ (engl. *quad word*) – 128 uzastopnih bitova = 16 byte-ova.

# [ Elementarni tipovi podataka ]

---

Osnovna podjela:

- a. Znakovni tip podataka
- b. Numerički tipovi podataka
- c. Logički tip podataka

# Znakovni (character) tip podataka

Osnovni znakovni tip podataka jeste znak (engl. character). U računalnoj terminologiji znak je jedinica informacije koja odgovara slovu ili simbolu pisanog oblika prirodnog jezika.

Primjer za znak je slovo, broj ili oznaka interpukcije. Koncept znaka također uključuje i kontrolne znakove koji ne odgovaraju simbolima prirodnog jezika već drugim bitovima informacija koji se koriste za obradu teksta, kao što su instrukcije za pisače i druge uređaje koji prikazuju takav tekst.

# Znakovni tip podataka

## Tip CHARACTER

**T := { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9',  
'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',  
'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W',  
'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j',  
'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w',  
'x', 'y', 'z', '+', '-', '=', '/', ...kontrolni znaci }**



# Znakovni tip podataka

## Različite metode za binarno kodiranje ASCII (American Standard Code for Information Interchange)

### Kontrolni znaci:

BEL (Bell) – zvučni signal (007 oktalno)

LF (Line Feed) – prijelaz u slijedeći red (012 oktalno)

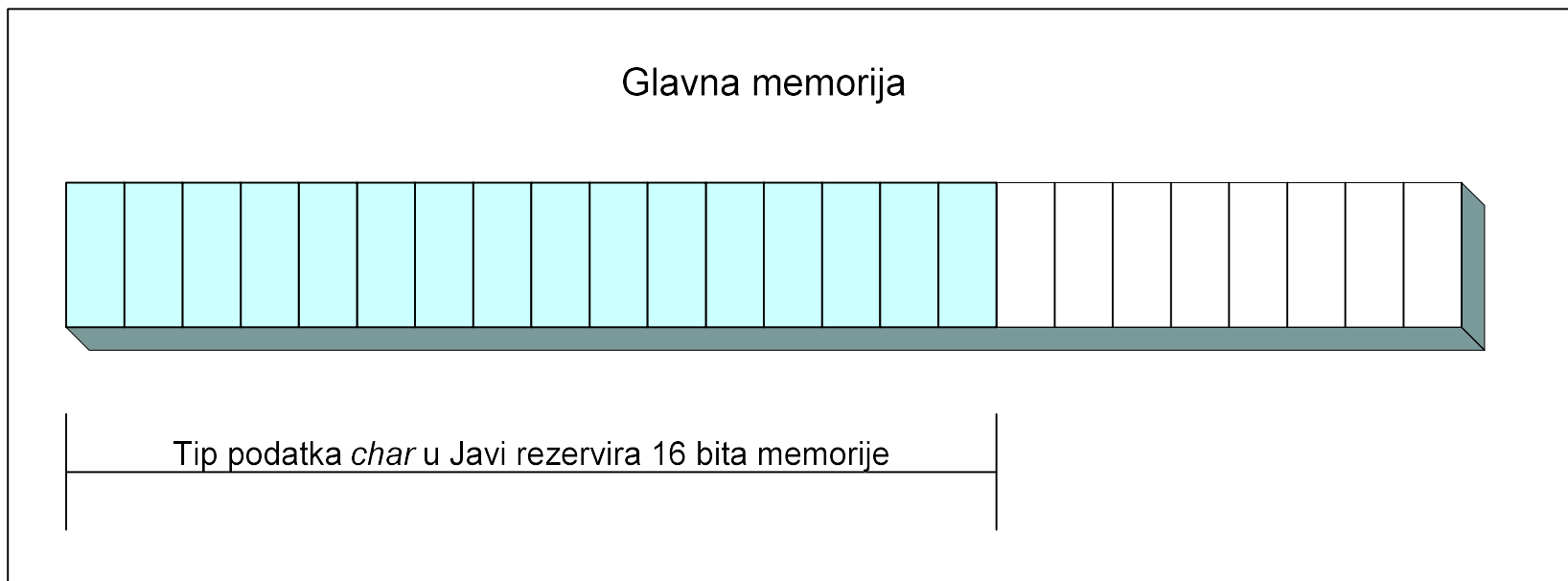
FF (Form Fed) – prijelaz na narednu stranu (014 okt.)

CR (Carriage Return) – povratak na početak reda (015)

ESC (Escape) – prelazak u naredbeni rad (033 okt.)

# [ Znakovni tip podataka ]

Rezerviranje memorije:



# Numerički tipovi podataka

Numerički tipovi podataka mogu se podijeliti na dvije opće skupine:

- cjelobrojni tipovi podataka
- tip podataka s pomičnim zarezom

# Numerički tipovi podataka

Pojam cjelobrojni (engl. Integer) tip podataka odnosi se na bilo koji tip podatka koji može predstaviti neki podskup matematičkih cjelobrojnih vrijednosti.

Vrijednost podatka koji je cjelobrojnog tipa jeste matematička cjelobrojna vrijednost koja odgovara tom tipu. Predstavljanje ovakvog podatka je u biti način na koji se vrijednost pohranjuje u računalnoj memoriji. Cjelobrojni tip podatka može biti “neoznačeni” (engl. Unsigned) tj. Sposoban za predstavljanje samo pozitivnih cjelobrojnih vrijednosti, ili “označen” tj. sposoban za predstavljanje i negativnih i pozitivnih cjelobrojnih vrijednosti.

# Numerički tipovi podataka

Uobičajeno predstavljanje pozitivnih cjelobrojnih vrijednosti je niz bitova, tj. uporaba binarnog numeričkog sustava.

Duljina ili preciznost cjelobrojnog tipa podataka je broj bitova koji se koristi u njegovom predstavljanju. Cjelobrojni tip podatka s  $n$  bitova može kodirati  $2^n$  brojeva, tako na primjer neoznačeni cjelobrojni tip obično predstavlja pozitivne vrijednosti od 0 do  $2^n - 1$ .

# Numerički tipovi podataka

Tip podatka s pomičnim zarezom (engl. *floating-point type*) je digitalno predstavljanje broja koji pripada nekom podskupu racionalnih brojeva i često se koristi za aproksimaciju proizvoljnih realnih brojeva u računalu.

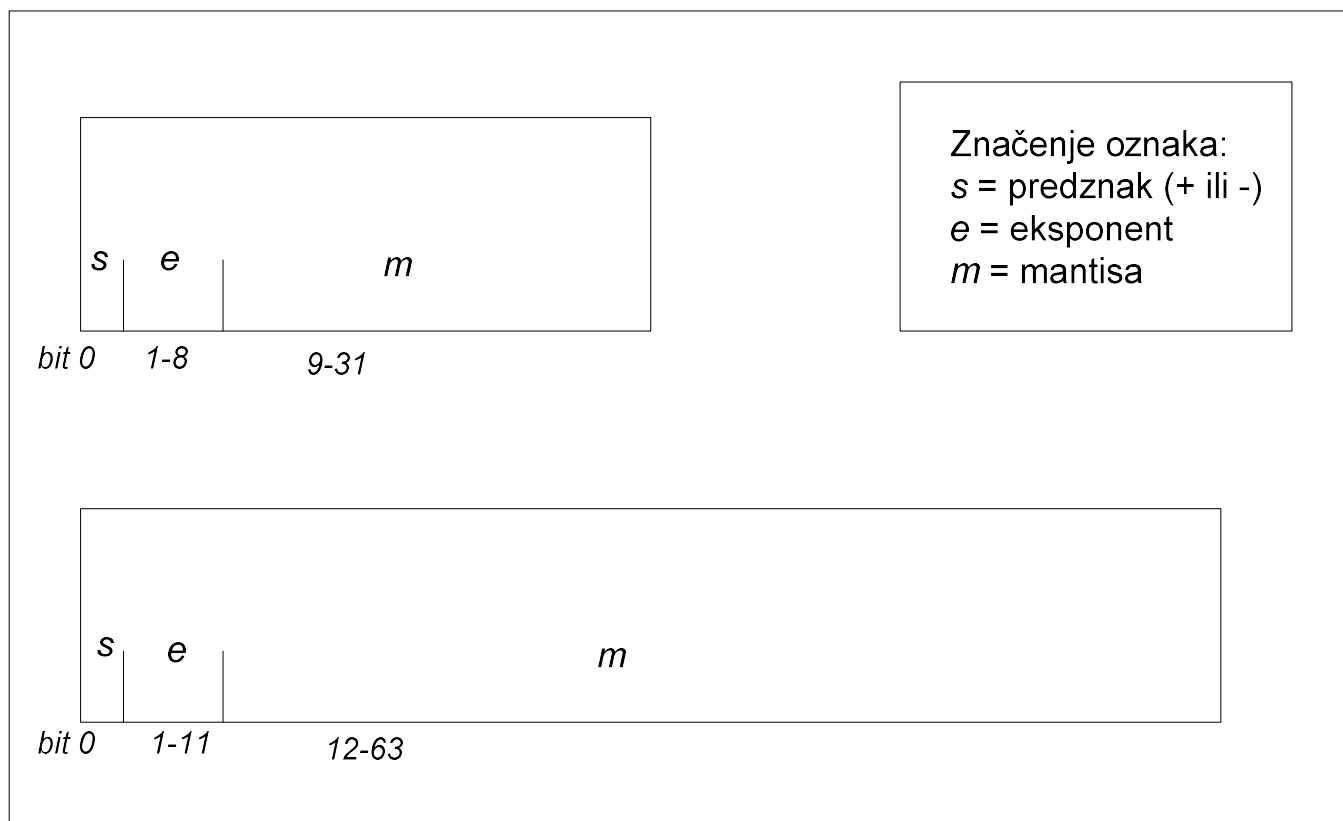
Predstavlja realni broj kao proizvod binarnog broja s jednim brojem koji nije nula lijevo od decimalne točke ili zareza, i odgovarajućeg eksponenta od 2.

Na primjer, realni broj 123.45 se predstavlja kao

$$1.2345 \times 10^2$$

# Numerički tipovi podataka

Suvremena računala koriste standard IEEE 754 za predstavljanje vrijednosti s pomičnim zarezom.



# Numerički tipovi podataka

Izračun s pomičnim zarezom je aritmetički izračun urađen s brojevima u pomičnom zarezu i često uključuje neku vrstu aproksimacije ili zaokruživanja pošto rezultat operacije možda nije moguće egzaktno predstaviti.

Bitno je primijetiti da nemogućnost predstavljanja postoji zbog činjenice da sustav pomičnog zareza definira diskretni brojni sustav čija je preciznost određena veličinom signifikanta i mantise. Čak i izračuni kod kojih su operandi potpuno predstavljivi često dovodi do rezultata koji nisu predstavljivi. U takvim slučajevima, sustav pomičnog zareza se referira na svoje postojeće diskretne brojevi i strateški odabire odgovarajuće predstavljanje kako bi osigurao najbolju moguću razinu točnosti.



# Logički tip podataka

- Tip podataka koji obuhvaća samo dvije vrijednosti  $T := \{v_1, v_2\}$

$v_1 = \text{false}$  (ili  $v_1 = F$ ),  $v_2 = \text{true}$  (ili  $v_2 = T$ )

ili

$v_1 = 0$ ,  $v_2 = 1$

# Logički tip podataka

**Logički ili boolean tip podataka koriste operatori Boolean algebre kao što su:**

- **konjukcija (AND)**
- **disjunkcija (OR)**
- **jednakost (=)**
- **negacija (NOT)**
- **i dr.**



# JAVA TIPOVI PODATAKA

# Java tipovi podataka

2 osnovne skupine:

- **jednostavni**

- cjelobrojni (byte, short, int, long)
- za brojeve s pomičnim zarezom (float, double)
- znakovni (char)
- logički (boolean)

- **složeni**

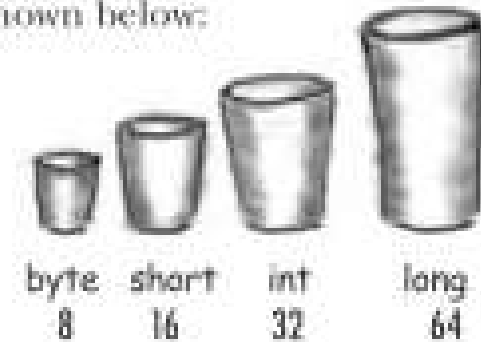
- polja
- klase
- Sučelja

**Osnovna osobina JAVA tipova: fiksna domena  
(područje)**

**Zbog osiguranja portabilnosti.**

# Java tipovi podataka

number of bits (cup size). The sizes are shown below:



## Cjelobrojni tipovi

Koriste se za prikaz predznačenih cijelih brojeva (i pozitivnih i negativnih)

Tip	Duljina	Najmanja vrijednost	Najveća vrijednost
long	64	-9223372036854775808	9223372036854775807
int	32	-2147483648	2147483647
short	16	-32768	32767
byte	8	-128	127

# Java tipovi podataka

## Tip byte

### uporaba:

- za čitanje iz datoteka,
- za komuniciranje preko mreže ili
- za rad sa sirovim binarnim podacima koji možda nisu izravno kompatibilni s drugim Java tipovima podataka.

### Deklariranje:

**byte b, c;**

# Java tipovi podataka

Tip short

**uporaba:**

- vrlo rijetka, danas su računala 32-bitna

**Deklariranje:**

**short s;**

# Java tipovi podataka

## Tip int

### uporaba:

- najčešće korišten tip podataka ( za brojače petlji, indekse nizova i sl.)
- važno: ukoliko se u izrazu miješaju tipovi byte, short i int, sve se vrijednosti prvo pretvaraju u int prije konačnog izračunavanja izraza

### Deklariranje:

**int i, j, k;**



# Java tipovi podataka

Tip long

**uporaba:**

- za prikaz dugačkih cijelih brojeva

**Deklariranje:**

**long l;**

# Java tipovi podataka

Brojevi s pomičnim zarezom (realni brojevi)

Koriste se za sva izračunavanja kod kojih treba povećati preciznost tako da se prikaže i decimalni dio.

Tip	Duljina	Najmanja vrijednost	Najveća vrijednost
<b>double</b>	<b>64</b>	<b>1.7E-308</b>	<b>1.7E+308</b>
<b>float</b>	<b>32</b>	<b>3.4E-038</b>	<b>3.4E+038</b>

# Java tipovi podataka

## Tip float

- osigurava tzv, jednostruku preciznost
- pogodan za 32 bitna računala
- nedovoljno precizan kada je riječ o vrlo velikim ili vrlo malim vrijednostima

**Deklariranje:**

```
float visokatemperatura;
```

**Brojevi s pomičnim zarezom:**

**154.88**

**1.5488e2**

# Java tipovi podataka

## Tip double

- osigurava tzv, dvostruku preciznost
- pogodan za 64 bitna računala
- precizan kada je riječ o vrlo velikim ili vrlo malim vrijednostima

**Deklariranje:**

```
double pi;
```

# Java tipovi podataka

Tip char (znakovni tip)

- baziran na Unicode standardu (16 bita – 2 byte)
- [www.unicode.org](http://www.unicode.org)

**Deklariranje:**

**char znak;**

# Java tipovi podataka

Tip boolean (logički tip)

- true ili false
- ovaj tip vraćaju svi operatori usporedbe i to je obvezni tip podataka u uvjetnim izrazima (if, for)

**Deklariranje:**

**boolean b;**

# [ Izrazi i operatori ]

Izraz (engl. *expression*) je kombinacija literala, varijabli, operatora, poziva funkcijama koji kao rezultat daju određenu vrijednost koja se onda može koristiti u bilo kojem drugom kontekstu unutar programa.

Izlazni rezultat izraza je vrijednost.

Primjer:

**a + b**

a, b → operandi

+ → operator

# [ Izrazi i operatori ]

---

Operator - simbol koji označava operaciju koja će se izvršiti na jednoj ili više vrijednosti, a rezultat te operacije je neka druga vrijednost.

OSNOVNO grupiranje:

- Aritmetički operatori
- Operatori usporedbe
- Logički operatori



# Aritmetički operatori

Operator	Namjena	Primjer
+	Zbrajanje	$x=x+12$
-	Oduzimanje	$x=x-12$
*	Množenje	$x=x*12$
/	Dijeljenje	$x=x/12$
%	Modulo (ostatak cjelobrojnog dijeljenja)	$x=x\%12$
++	Inkrementiranje	$x++$
+=	Zbrajanje i dodjela vrijednosti	$x += 12$
-=	Oduzimanje i dodjela vrijednosti	$x -= 12$

# Aritmetički operatori

Operator	Namjena	Primjer
<code>*=</code>	Množenje i dodjela vrijednosti	<code>x *= 12</code>
<code>/=</code>	Dijeljenje i dodjela vrijednosti	<code>x /= 12</code>
<code>%=</code>	Modulo i dodjela vrijednosti	<code>x %= 12</code>
<code>--</code>	<b>Dekrementiranje</b>	<code>x--</code>

## Aritmetički operatori

$i = i + 8;$  ili  $i += 8;$

$i = i \% 2;$  ili  $i \% = 2;$

$i = i * 7;$  ili  $i *= 7;$

$i = i - 4;$  ili  $i -= 4;$

$i = i / 5;$  ili  $i /= 5;$

# Aritmetički operatori

Operatori povećanja/smanjenja    ++ / --

Ako je  $a = 40$ , slijedi

1.)  $a = a + 1$  → rezultat je 41, a to se može napisati i pomoću operatora uvećavanja kao

$a++$     → rezultat je isto 41    ..... kao postfiks

$++a$     → rezultat je isto 41    .... kao prefiks

2.)  $a = a - 1$  → rezultat je 39, a to se može pomoću operatora umanjenja napisati kao

$a--$  → rezultat je isto 39    ... kao postfiks

$--a$  → rezultat je isto 39    ... kao prefiks

# Aritmetički operatori

Operatori povećanja/smanjenja

++ / --

Dvojaki način pisanja: kao prefiks i kao sufiks

→ `x=50; y=+++x; => y=51,x=51`

ili

`x=x+1; y=x;`

→ `x=50; y=x++; => y=50, x=51`

ili

`y=x; x=x+1;`

# Aritmetički operatori

→ Razlika u izrazima

→ `int m=7`

→ `int a = 2 * ++m` rezultat je ???

→ `int b = 2 * m++` rezultat je ???

# Aritmetički operatori

→ Razlika u izrazima

→ `int m=7`

Prvo se m uveća za 1

→ `int a = 2 * ++m` rezultat je 16

→ `int b = 2 * m++` rezultat je 14

Prvo se 2 pomnoži s m

# Aritmetički operatori - dodjeljivanja

*varijabla = varijabla operator izraz;*

*|||*

*Varijabla operator=izraz;*



# Operatori usporedbe

Operator	Namjena	Primjer
==	Jednako	$A == 0$
!=	Različito	$A != 0$
>	Veće	$A > 0$
<	Manje	$A < 0$
>=	Veće od ili jednako	$A >= 0$
<=	Manje od ili jednako	$A <= 0$

# Logički operatori

Operator	Namjena
&	Logička konjunkcija (AND – i)
	Logička disjunkcija (OR – ili)
^	Logička isključiva disjunkcija (isključivo OR tj.XOR – isključivo ili)
	Uvjetna disjunkcija
&&	Uvjetna konjunkcija
!	Logička unarna negacija (NOT – ne)
&=	Dodjeljivanje uz logičku konjunkciju
=	Dodjeljivanje uz logičku disjunkciju

# Logički operatori

Operator	Namjena
$\wedge=$	Dodjeljivanje uz logičku isključivu disjunkciju
$==$	Jednako
$!=$	Različito
$?:$	Trojni operator uvjetne dodjele ili ternarni operator

# Logički operatori

A	B	A B	A&B	A^B	!A
False	False	False	False	False	True
True	False	True	False	True	False
False	True	True	False	True	True
True	True	True	True	False	False

OR      AND      XOR

# [ Uvjetni logički operatori ]

- Uporaba `&&` i `||` znači da Java neće izračunavati desnu vrijednost ako se rezultat može dobiti samo iz lijeve (operator OR daje kao rezultat true kada A ima vrijednost true, bez obzira na vrijednost B, odnosno operator AND daje kao rezultat false kada A ima vrijednost false bez obzira na vrijednost B).

Primjer:

```
if (djeljitelj !=0 && broj/djeljitelj >10)
```

# [ Prioriteti izvršavanja operatora ]

() []  
++ -- !  
\* / %  
+ -  
> >= < <=  
== !=  
&  
&  
|  
&&  
||  
?:  
=

# [ Pridruživanje ]

- Opći oblik:

*variabla = vrijednost*

Na lijevoj strani - jednostavna varijabla ili element polja, odnosno bilo koja *l-vrijednost* (engl. *l-value*, gdje *l* dolazi od engl. *left tj. lijevo*). *L-vrijednost* mora označavati memorijsku lokaciju određenog tipa upisivu u vrijeme izvođenja (engl. *writable at run-time*). S desne strane operatora pridruživanja nalazi se bilo koji izraz odgovarajućeg tipa, ili tipa svodivog na tip *l-vrijednosti*. Taj izraz se, suprotno od *l-vrijednosti* naziva *r-vrijednost* (engl. *r-value*, *r* dolazi od engl. *right tj. desno*). Svaka *l-vrijednost* može imati ulogu *r-vrijednosti*, ali obrnuto ne vrijedi.

# [ Pridruživanje ]

- Java operator pridruživanja je simbol =
- Vraća vrijednost sa svoje lijeve strane, tj. vrijednost koja se pridružuje u izrazu pridruživanja, npr.:

$z = a + b/c ;$

- Pridruživanje se obavlja s desna na lijevo.
- Najprije se izračuna r-vrijednost  $a + b/c$  , potom se ona pridruži l-vrijednosti z (varijabla).
- Primjer prikazuje model pridruživanja po vrijednosti koji se uglavnom koristi za elementarne tipove podataka (Java, C#), dok se za složene tipove podataka koristi model pridruživanja po referenci.



# [ Pridruživanje ]


Za lokalne varijable metoda, Java (i C#) definira notaciju ***definitivnog pridruživanja*** (engl. *definite assignment*) koje onemogućava uporabu neinicijaliziranih varijabli.

Temelji se na kontrolnom toku programa i može biti statički provjeravana od strane kompilatora.

- Svaki mogući put do izraza mora dodijeliti vrijednost svakoj varijabli i tom izrazu.
- Konzervativno pravilo i može ponekad proglasiti pogrešnim neke programe, iako oni nikada ne bi stvarno koristili neinicijalizirane varijable

# [ 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\_40\bin  
  
odgovarajuća verzija jave
- Ako je već postojala varijabla PATH, onda je treba modificirati klikom na Edit.

# PRVI Java program

```
class prviProgram {  
    public static void main (String args[]) {  
        System.out.println("Moj prvi program!!!");  
    }  
}
```



# PRVI Java program nastavak

## 1. Kompiliranje

```
javac prviProgram.java
```

**javac -> java kompilator koji kompilira java izvorni kod u java byte kod (formira klasu).**

## 2. Pokretanje

```
java prviProgram
```

**java -> java byte kode interpreter koji izvršava java program i starta JVM (Java Virtual Machine) tako što pokušava pozvati main() metodu**

# PRVI Java program

klasa

```
/* ovo je moj prvi JAVA program koji sam
   napisao/la na satu Programiranja I*/
class prviProgram {
    //Program započinje pozivom metodi main
    public static void main (String args[]) {
        System.out.println("Moj prvi program!!!");
    }
}
```

# PRVI Java program

Klase se učitavaju u izvršno okruženje i koriste kao predložak za kreiranje instanci objekata.

Iako je njihova definicija statična, moraju biti na raspolaganju u vrijeme izvršavanja programa kako bi bile u stanju “proizvesti” objekte sa svim osobinama definiranim u klasi.

Ne mora postojati nikakav objekt za određenu klasu u određenom vremenu.

# PRVI Java program

Klasa je enkapsulirana (omotana) kolekcija podataka i metoda koje se izvršavaju nad podacima. Definicija klase, podataka i metoda služi kao nacrt (engl. blueprint) koji se koristi za kreiranje novih objekata te klase.

Definicija klase se obično sastoji od (redosljed nije bitan):

- Modifikatora pristupa – definira raspoloživost klase iz drugih klasa
- Ključne riječi Class – znak Javi da slijedeći blok definira klasu
- Instanci polja – Sadrži varijable i konstante koje koriste objekti klase
- Konstruktora – metoda koje imaju isti naziv kao i klasa, a koji se koriste za kontrolu inicijalnog stanja bilo kojeg kreiranog objekta klase
- Polja klase – sadrže varijable i konstante koje pripadaju klasi i koje dijele svi objekti klase
- Metoda klase – metode koje se koriste za kontrolu vrijednosti polja klase

# PRVI Java program

```
/* Ovo je moj prvi JAVA program koji sam napisao/la na satu Programiranja I*/  
public class prviProgram {  
    //Program započinje pozivom metodi main  
    public static void main (String args[]) {  
        System.out.println("Moj prvi program!!!");  
    }  
}
```

Modifikator pristupa

Deklaracija klase

Početak bloka

Kraj bloka



# PRVI Java program

```
/* ovo je moj prvi JAVA program koji sam napisao/la na
   satu Programiranja I*/
```

→ **Višeredni komentar koji počinje s /\* i završava s \*/**

```
class JavaAplikacija {
```

```
//Program započinje pozivom metodi main
```

→ **Jednoredni komentar koji počinje s //**

```
→ public static void main (String args[]) {
    System.out.println("Moj prvi program!!!");
```

```
}
```

```
}
```

# PRVI Java program

```
/* ovo je moj prvi JAVA program koji sam napisao/la na
   satu Programiranja I*/
```

```
class JavaAplikacija {
```

```
//Program započinje pozivom metodi mai
```

→ **Jednoredni komentar koji počinje s //**

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

→ **Izvršenje započinje pozivom metode**

**main()**

**static → dozvoljava da metoda main() bude  
pozvana bez pravljenja posebne instance klase**

**public → modifikator pristupa**

**void → metoda ne vraća nikakvu**

**vrijednost**

```
System.out.println("Moj prvi program!!!");
```

```
}}
```

Metoda  
klase

# PRVI Java program

Kada se definira klasa u objektno orijentiranom programiranju, ponašanje te klase se implementira pomoću jedne ili više metoda.

Metoda se sastoji od:

- Modifikatora pristupa – definira raspoloživost metode:
  - public (javna) – mogu biti poznate i korištene od vanjskih korisnika tj. metoda iz drugih klasa
  - private (privatne) – mogu ih vidjeti i koristiti samo metode u okviru iste klase
  - protected (zaštićene) – dozvoljen pristup iz bilo koje klase koja “proširuje” (extends) ili nasljeđuje klasu
- Ključne riječi “static” – obično pojedinačni objekti pristupaju varijablama i metodama. Ako je metoda ili varijabla definirana kao “static” (statična-nepromjenjiva), ona postaje metoda ili varijabla klase, što znači da se primjenjuje na klasu objekata kao cjelinu, a ne na pojedinačne objekte. Svi objekti jedne klase dijele statičke varijable
- Argumenata – parametara potrebnih da bi metoda izvršila svoju zadaću (funkciju)
- Vrste podatka kojeg metoda vraća – obavezan za svaku definiciju metode. Nema default (podrazumijevanog) tipa vraćanja jer određuje tip objekta koji će biti vraćen kada metoda završi svoju zadaću. Može biti integer, string, bilo koji definirani tip objekta ili void (ne vraća nikakav podatak)

# PRVI Java program

```
/* ovo je moj prvi JAVA program koji sam napisao/la na
   satu Programiranja I*/
```

```
class JavaAplikacija {
```

```
//Program započinje pozivom metodi main
```

```
public static void
```

```
main (String args[]) {
```

Argumenti naredbene  
(komandne) linije

→ *parametar metode main(), niz instanci klase*

***String***

***{ } → Početak i kraj tijela metode***

```
System.out.println("Moj prvi program!!!");
```

```
}  
}
```

Gotove klase iz java  
biblioteke

# PRVI Java program

## GOTOVE JAVA klase

Java programski jezik sadrži skup klasa koje su organizirane u pakete, ovisno o funkcionalnosti grupe. Na primjer, skup klasa koje pomažu u kreiranju i korištenju mreže smješteno je u `java.net` paketu.

**UGRAĐENE (built-in) klase koje se isporučuju s Java jezikom:**

- `java.lang` -> osnovne funkcije Java jezika
- `javax.swing` -> osnovni paket za –swing GUI sučelje
- `java.util` -> olakšava potporu za interfejse (sučelja), implementaciju, sortiranje i pretraživanja
- `java.awt` -> upravlja podlogom (layout), upravlja događajima, grafikom
- `java.io` -> opće sučelje za sve ulazno/izlazne operacije

# [Pripremiti za sljedeće predavanje]

- Sljedeće predavanje: 20.03.2018.
- Napraviti program koji koristi sve tipove podataka, osnovne operatore (+, -, /, \* i modulo) i prikazuje rezultate izraza.
- Pripremiti:
  - Pripremiti se za prvi test iz teorije:
    - Poglavlje 1 – Osnovni pojmovi
    - Poglavlje 2 – Povijesni razvoj programskih jezika
    - Poglavlje 3 – Programske paradigme



**Pitanja..**