PT&SCHE

Introduction of part-time and short cycle studies in Serbia

Funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

UNIVERZITET
METROPOLITAN
BEOGRAD

KI103 - JAVA 1: OSNOVE PROGRAMIRANJA U JAVI

Matematičke funkcije, znaci i nizovi znakova

Lekcija 05

PRIRUČNIK ZA STUDENTE

KI103 - JAVA 1: OSNOVE PROGRAMIRANJA U JAVI

Lekcija 05

MATEMATIČKE FUNKCIJE, ZNACI I NIZOVI ZNAKOVA

- Matematičke funkcije, znaci i nizovi znakova
- Poglavlje 1: Aritmetički operatori
- Poglavlje 2: Matematičke funkcije
- Poglavlje 3: Znaci i rad sa njima
- Poglavlje 4: Nizovi sa znacima i rad sa njima
- Poglavlje 5: Domaći zadaci
- Zakljucak

UVOD

U ovoj lekciji naučićete kako da rešavate uobičajene matematičke probleme primenom nekoliko Javinih klasa, kao što je Math.

Cilj ove lekcije je da naučite kako da:

- Rešavate matematičke probleme upotrebom metoda klase ***Math.***
- Koristite znake primenom tipa ***char***,
- Predstavljate znake upotrebaom ASCII i Unicode šifarnika,
- Koristite specijalne znake upotrebom izlaznih sekvenci,
- Izvršite konverziju brojčanih vrednosti u znake tipa char, a znake tipa char i cele brojeve.
- Upoređujete i testirate znake upotrebom statičkih metoda u klasi ***Character***,
- Koristite nizove znakova upotreboma klase ***String,***
- Određujete dužinu teksta upotrebom metoda ***length()*** klase ***String***,
- Nalazite neki znak u nizu znakova, upotrebom metoda ***charAt(i***),
- Koristite operator + da bi povezali odvojene nizove znakova,
- Dobijate nizove sa velikim ili malim znacima, i da radite druge operacije sa nizovima,
- Upoređujete nizove znakova upotrebom metoda ***equal()*** i ***compareTo()***,
- Dobijate pod-nizove znakova, da primenom metoda ***indexOf()*** nalazite podnizove u nekom nizu,
- Konvertujete heksadecimalne oznake u decimalne brojeve upotrebom metoda ***HexDig2Dec().***

Poglavlje 1

Aritmetički operatori

RASPOLOŽIVI ARITMETIČKI OPERATORI

Aritmetički operator je simbol koji označava potrebu da se uradi neka aritmetička operacija, a Operandi su vrednosti na koje se operator primenjuje.

Aritmetički operator je simbol koji označava potrebu da se uradi neka aritmetička operacija. Ako se u istom izrazu koristi više operatora, onda mora postojati redosled kojim se operacije odvijaju. U tom smislu se može govoriti o prioritetu operatora. Prvo se izvršavaju operatori sa višim prioritetom. U Javi postoji mnogo operatora. Neki od njih su dati u sledećoj tabeli:

Operatori iz iste grupe u ovoj tabeli, imaju isti prioritet. Tako na primer, *operatori plus i minus imaju isti prioritet.*

Izračunavanje izraza ide sa leva udesno.

U izrazu učestvuju **operator**i i **operandi**. Operatore smo već pomenuli, a operandi su vrednosti na koje se operator primenjuje. Na primer, u iskazu:

13 - 5

13 i 5 su operandi, a minus **(-) je operator.**

Operacija u kojoj učestvuju samo celi brojevi se uvek odvija sa 32 bit-a, ili više. Ako je jedan od operanada 64-bitni (tip***long***), onda se i operacija radi sa 64 bit-a. U suprotnom se operacija uvek odvija sa 32 bit-a, čak i ako su oba operanda manja.

Ako se u izrazu koristi promenljiva tipa short (16 bit-ova), procesor tu vrednosti ubacuje u svoju 32-bitnu aritmetičku :

jedinicu za cele brojeve. Veličina promenljive ne mora da odgovara broju bit-ova koje procesor koristi za aritmetiku. Na prirmer:

```
short x = 12;       // 16 bitni short
int   rezultat;         // 32 bitni int
rezultat = x / 3;   // aritmetika se radi sa 32 bita
```

Operator	Značenje	Prioritet
-	unarni minus	visok
+	unarni plus	visok
*	množenje	srednji
/	deljenje	srednji
%	deljenje sa ostatkom	srednji
+	sabiranje	nizak
-	oduzimanje	nizak

Slika 1.1 Aritmetički operatori

PRIMER PRIMENE OPERATORA DELJENJA

Operator deljenja "/" znači celobrojno deljenje, ako su oba operanda celi brojevi. Ako je jedan od operanada realan broj, onda je u pitanju deljenje realnih brojeva.

Kod izraza x/3 računar će podeliti 32 bit-ni broj 12 sa 32 bit-nim brojem 3 i sve to staviti u 32 bit-nu promenljivu rezultat. Literal 3 automatski predstavlja 32 bit-nu vrednost. Evo još jednog primera:

```
short x = 12;
short y = 3;
short rezultat;
rezultat = x / y;
```

Kod izraza x/y će računar podeliti 32 bit-ni broj 12, sa 32 bit-nim brojem 3, čak i ako promenljive x i y imaju po 16 bit-ova. Rezultat će biti skraćen na 16 bit-ova i onda postavljen u promenljivu rezultat.

Operator deljenja "/" znači celobrojno deljenje, ako su oba operanda celi brojevi. Ako je jedan od operanada realan broj, onda je u pitanju deljenje realnih brojeva. Rezultat celobrojnog deljenja je uvek ceo broj. Ako je rezultat deljenja realan broj, decimalni deo se odbacuje (ne zaokružuje se).

U tom smislu postoji razlika između onog što radi Java, i onoga što bi uradio digitron. Kalkulator će za izraz 7/2 uraditi deljenje realnih brojeva.

Java će na isti izraz primeniti aritmetiku celih brojeva, tako da će rezultat ovog izraza biti 3. Iako se ovo može učiniti čudnim, ipak većina programskih jezika, pa i Java – funkcionišu na ovaj način.

Ako pogledamo operatore koje smo do sada pomenuli, zapazićemo da neki od njih rade sa jednim, a neki sa dva operanda. U tom smislu se operatori mogu podeliti na unarne (sa jednim operandom) i binarne (sa dva operanda). Binarni operatori uvek moraju da imaju dva operanda, iako se ponekad može učiniti da ih ima više. U takvim slučajevima su jedan, ili oba operanda – podizraz. Na primer:

(12.0 * 31) / 12

Prvi operand za operator deljenja je u zagradama i predstavlja poseban izraz.

Ako su kod binarnih operatora oba operanda celi brojevi, onda je reč o aritmetici celih brojeva. Ako je makar jedan od operanada realan broj, onda je u pitanju operacija sa realnim brojevima.

PRIMER DELJENJA REALNIH I CELIH BROJEVA

Ako su a i b celobrojne promenljive, onda prikazani izrazi predstavljaju celobrojno deljenje

Ako su a i b celobrojne promenljive, onda sledeći izrazi predstavljaju celobrojno deljenje:

12 * b

a - 2

56%a

Ako su a i b celi brojevi, a x i y realni brojevi, onda sledeći izrazi predstavljaju deljenje realnih brojeva:

x * b

(a - 2.0)

56*y

Pravilo o aritmetici celih brojeva ili aritmetici realnih brojeva se primenjuje i na složenije izraze, korak po korak. Pogledajmo sledeći izraz:

(1/2 + 3.5) / 2.0

Kakav je rezultat? Primenićemo pravilo: prvo unutrašnje zagrade, a onda operatori sa najvišim operatorima:

(1/2 + 3.5) / 2.0

---

prvo ovo

Pošto su oba operanda celi brojevi, operacija je celobrojno deljenje, što daje rezultat:

(0 + 3.5) / 2.0

Sada nastavljamo sa izrazom unutar zagrada. Operator plus je sabiranje realnih brojeva, pošto je jedan od operanada realan broj (3.5).

3.5 / 2.0

Na kraju se obavlja i poslednja operacija:

1.75

OPERATOR DELJENJA PO MODULU

Simbol za operaciju deljenja po modulu je kao što ste videli, oznaka procenta %.

Simbol za operaciju deljenja po modulu je kao što ste videli, oznaka procenta %. Tako je na primer, rezultat sledećeg izraza:

13 % 5

broj 2. To je ostatak posle celobrojnog deljenja ova dva broja.

```
class DeljenjePoModulu{
  public static void main ( String() args){
    Int kolicnik, ostatak;
    kolicnik = 17 / 3;
    ostatak = 17 % 3;
    System.out.println(“Kolicnik je: “ = kolicnik);
    System.out.println(“Ostatak je: “ + ostatak);
    System.out.println(“Original je: “ = (kolicnik * 3 + ostatak) );
  }
}
```

Slika 2 Primer deljenja po modulu

Iz ovog primera se vidi kako se radi sa ovim operatorom.

Ovaj operator može da se koristi i sa negativnim brojevima. U tom slučaju, važe sledeća pravila:

Ako su oba operanda pozitivni, operacija se obavlja onako kako je predstavljeno.

1. Ako je levi operand negativan, onda je i rezultat negativan.
2. Ako je levi operand pozitivan, onda je i rezultat pozitivan.
3. Znak desnog operanda se zanemaruje u svim slučajevima.

Evo kako to izgleda na nekoliko primera:

17 % 3 == 2 -17 % 3 == -2

17 % -3 == 2 -17 % -3 == -2

ARITMETIČKI OPERATORI (VIDEO)

Video daje rayne primere primene osnovnih aritmetičkih operatora

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIMER 1

Rešavanje jednačine na osnovu unete vrednosti za x.

Potrebno je dati rešenje za y za vrednost x koju korisnik definiše putem JOptionPane-a. Jednačina je:

$$y = \frac{(x^2 - 2)}{\sqrt{2x}}$$

Slika 1.2 - Jednačina za prvi zadatak

Objašnjenje:

Kod učitavanja x-a naredbom JOptionPane.showInputDialog("Unestite X:")

dobijamo X kao String, te je potrebno izvršiti konverziju u double tip.

Za to koristimo metodu Double.parseDouble()

Kod pisanja formula u Javi jako je važno obratiti pažnju na zagrade.

Za koren možete koristiti Sqrt metodu Math biblioteke.

x na kvadrat se može pisati kao x*x ili kao Math.pow(x,2)

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package primer1;

import javax.swing.JOptionPane;

/**
 *
 * @author Aleksandra
 */
public class Primer1 {

    /**
     *
```

```
     * @param args
     */
    public static void main(String[] args) {
        new Primer1();
    }

    public Primer1() {
        double x = Double.parseDouble(JOptionPane.showInputDialog("Unestite X:"));
        double rezultat = (x * x - 2) / (Math.sqrt(2 * x));
        System.out.println("Rešenje zadatka je: " + rezultat);

    }

}
```

ZADATAK 1

Vežbanje primene operatora u matematičkim izrazima

- Kreirajte program koji pronalazi rešenje kvadratne jednačine;
- Jednačina je data u opštem obliku:

$$ax^2+bx+c=0$$

Slika 1.3 Opšt oblik kvadratne jednačine

- Rešenje se trži po formuli:

$$x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

Slika 1.4 Formula za rešavanje kvadratne jednačine

- Pre pronalaženja rešenja neophodno je testirati veličinu D, koja se naziva diskriminanta:

$$b^2-4ac$$

Slika 1.5 Diskriminanta kvadratne jednačine

- Vrednosti a, b i c se unose sa tastature;
- Ukoliko je D = 0, jednačina ima dva ista rešenja x1=x2;
- Ukoliko je D>0, jednačina ima dva različita realna rešenja;
- Ukoliko je D<o, kao rešenje prikazati komentar "Jednačina nema realnih rešenja":

Za operaciju kvadratni koren, koristiti metodu Math.sqrt() (o njoj će biti više reči u narednom izlaganju) kaou prethodnom primeru.

OPERATORI INKREMENTIRANJA

Operator inkrementiranja ++ dodaje jedinicu promenljivoj.

Operacija koja se u programima najčešće izvodi je najverovatnije, povećavanje vrednosti promenljive za 1. Većina programa u sebi ima petlje. Većina petlji se kontroliše preko promenljive koja broji prolaze kroz petlju. Svaki put kada se izvrši petlja, promenljivoj se dodaje jedan.

Postoje i druge situacije kada se promenljivoj dodaje broj jedan. Istraživanja su pokazala da je dodavanje jedinice nekoj promenljivoj, operacija koja se najčešće izvršava. Sa druge strane, i procesori su tako napravljeni da ovu operaciju izvršavaju vrlo brzo.

Već znamo kako se promenljivoj dodaje 1:

brojac = brojac + 1 ; // dodavanje jedinice brojacu

Ovo je prilično jednostavno, ali postoji i kraći način da se to isto uradi:

brojac++ ; // dodaje jedan brojacu

U ovom iskazu se koristi operator inkrementiranja ++. On vrednosti promenljive brojac dodaje 1.

Operator inkrementiranja ++ dodaje jedinicu promenljivoj. Obično je promenljiva celobrojnog tipa (byte, short, int, long), ali može biti i realan broj (float, double). Između dva znaka plus ne sme da se nađe nijedan karakter. Obično se znaci plus stavljaju odmah iza promenljive, premda to nije obavezno.

Operator inkrementiranja se može koristiti i kao deo aritmetičkog izraza.

```
int sum = 0;
int brojac = 10;
sum = brojac ++ ;
System.out.println("sum: "+ sum " + brojac: " + brojac);
```

NAČIN IZVRŠAVANJA OPERATORA INKREMENTIRANJA

Ako se operatori inkrementiranja koriste samostalno, onda nije bitno da li se koristi prefiks ili postfiks verzija

Kako radi iskaz: sum = brojac++;

U ovom iskazu se vrednost promenljive brojac povećava nakon što se promenljiva upotrebi.

Iskaz dodele se izvršava na sledeći način:

1. Izračunava se vrednost izraza sa desne strane znaka = . Vrednost je 10 (pošto se brojač još uvek nije povećao)
2. Vrednost se dodeljuje promenljivoj sa leve strane znaka = . Promenljiva sum dobija vrednost 10.
3. Sada se primenjuje operator ++: brojac se povećava na 11
4. U poslednjem iskazu se štampa: sum: 10 brojac: 11

Operator inkrementiranja treba koristiti pažljivo i samo kada je to potrebno. Ponekad izraz može biti duži, ali i jasniji bez ovog operatora.

Operator inkrementiranja koji smo do sada koristili se stavljao iza promenljive. On se može staviti i ispred promenljive. U tom smislu razlikujemo prefiks i postfiks operatore inkrementiranja. U oba slučaja se vrednost promenljive povećava za 1, ali postoji razlika u tome kako se to radi:

++ brojac - znači povećanje pre upotrebe

brojac ++ - znači povećanje nakon upotrebe

Ako se ovi operatori koriste samostalno, onda nije bitno da li se koristi prefiks ili postfiks verzija. Pogledajmo prethodni primer:

```
int sum = 0;
int brojac = 10;
sum = ++brojac ;
System.out.println("sum: "+ sum " + brojac: " + brojac);
```

Operator ++ je sada prefiks operator.

Sada se vrednost promenljive brojac povećava pre nego što se u širem izrazu upotrebi njena vrednost.

Iskaz dodele se izvršava u sledećim koracima:

1. Izračunava se izraz sa desne strane znaka =. Vrednost je sada 11 (pošto se brojac povećava pre upotrebe)
2. Promenljivoj sa leve strane znaka = se dodeljuje vrednost. sum dobija vrednost 11
3. Poslednji iskaz štampa sum: 11 brojac: 11

OPERATOR DEKREMENTIRANJA

Ovaj operator smanjuje vrednost promenljive, na koju se primeni, za 1

Pored operatora inkrementiranja, postoje i – operatori dekrementiranja. To je operator --. Ovaj operator smanjuje vrednost promenljive, na koju se primeni, za 1. Takođe postoje prefiks i postfiks verzija.

Sledeći kod:

```
int x = 99;
int y = 10;
y = --x ;
System.out.println("x: " + x + "  y: " + y );
```

Štampa sledeći izlaz:

x: 98 y: 98

Izraz	Operacija	Primer	Rezultat
x++	vrednost se upotrebi, pa se doda 1	int x = 10; int y; y = x++ ;	x je 11; y je10
++x	dodaje 1, pa se onda koristi vrednost	int x = 10; int y; y = ++x ;	x je 11; y je 11
x--	koristi se vrednost, pa onda oduzima 1	int x = 10; int y; y = x-- ;	x je 9; y je 10
--x	oduzima se 1, pa se onda koristi vrednost	int x = 10; int y; y = --x ;	x je 9; y je 9

Slika 1.6 Primer načina računjanja inkrementa i dekrementa

OPERATOR INKREMENTIRANJA I DEKREMENTIRANJA (VIDEO)

Video objašnjava primenu operatora inkrementiranja

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

OPERATORI INKREMENTIRANJA I DEKREMENTIRANJA (VIDEO)

Video objašnjava primenu operatora inkrementiranja (++) i dekrementiranja (--)

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

JOŠ NEKI OPERATORI DODELE

*Operatori +, -, *, / (i drugi) se mogu koristiti zajedno sa znakom jednakosti, čime se dobijaju složeni operatori*

Operatori +, -, *, / (i drugi) se mogu koristiti zajedno sa znakom jednakosti, čime se dobijaju složeni operatori. Na primer, sledeći iskaz će promenljivoj sum dodati 5.

sum += 5;

Ovaj iskaz ima isti efekat kao:

sum = sum + 5; // dodaje 5 vrednosti za sum

Operator	Operacija	Primer	Efekat
=	dodela	sum = 5;	sum = 5;
+=	sabiranje sa dodelom	Sum += 5;	Sum = sum + 5;
-=	oduzimanje sa dodelom	sum -= 5;	sum = sum - 5;
*=	množenje sa dodelom	sum *= 5;	sum = sum * 5;
/=	deljenje sa dodelom	sum /= 5;	sum = sum/5;

Slika 1.7 Drugi operatori dodele

OPERATORI DODELE (VIDEO)

Video daje primere raynih operatora dedele i aritmetičkih operatora u Javi.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

KONSTANTE

Rezervisana reč final saopštava kompajleru da se vrednost neće menjati

Pri izradi programa često je potrebno da konstantnoj vrednosti date ime. Možete na primer na taj način definisati poresku stopu od 18% i poresku stopu od 8%, koje trenutno kod nas

postoje. To su konstante zato što njihova vrednost, tokom izvršavanja programa, ne bi smela da se menja. Ovo se može uraditi na sledeći način:

```
class RacunanjePoreza{
  public static void main ( String[] arg )  {
    final double visaStopa = 0.18;
    final double nizaStopa = 0.08;
    . . . . . .
  }
}
```

Slika 1.8 Primer primene konstante

U programu možete uočiti ključnu reč final. Ta rezervisana reč saopštava kompajleru da se vrednost neće menjati. Ovaj kontakt je poput ugovora između Vas i kompajlera: Vi se obavezujete da tu vrednost nećete menjati, a kompajler je tu da proverava da li se tog obećanja držite.

Imena konstanti podležu istim pravilima kao i imena promenljivih. (Programeri ponekad za imena konstanti koriste velika slova, ali je to stvar stila, a ne deo specifikacije jezika). Konstante mogu da se koriste i u izrazima tipa:

porez = ukupanIznos * visaStopa ;

Sa druge strane, sledeći iskaz sadrži sintaksičku grešku:

visaStopa = 0.20; // pokušaj promene poreske stope (neuspešan)

Zašto se koriste konstante? Dva su osnovna razloga za to: One povećavaju čitljivost programa i olakšavaju proveru njegove ispravnosti.

Ako konstanta treba da se promeni (na primer, novi poreski zakon promeni poreske stope), onda jedino treba da promenite deklaraciju. Ne morate da pretražujete ceo program i da menjate svako pojavljivanje konkretnog broja.

PRIMER 2

Kombinovanje rezultata različitih jednačina

Na osnovu zadatih formula sa slike pronaći x1 i x2 a zatim izračunati rezultat. X korisnik treba da unese preko JOptionPane-a.

$$x1 = \frac{x^2 - 2x + 3}{x^3} \qquad x2 = \frac{x^3 - 2x^2 - 2x + 2}{\sqrt{x}}$$

$$rezultat = \frac{x1 * x2}{\sqrt{x}}$$

Slika 1.9 - Jednačine za X1, X2 i rezultat

Kada korisnik unese vrednost za X

x = Double.parseDouble(JOptionPane.showInputDialog("Unesite X:"));

potrebno je prvo izračunati x1,

double x1 = (Math.pow(x, 2) - 2 * x + 3) / Math.pow(x, 3);

zatim x2 double x2 = (Math.pow(x, 3) - 2 * Math.pow(x, 2) - 2 * x + 2) / Math.sqrt(x);

a onda i rezultat kada su x1 i x2 već izračunati.

double rezultat = (x1 * x2) / Math.sqrt(x);

Rezultat možemo ispisati ili preko JOptionPane ili preko System.out.println

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package primer2;

import javax.swing.JOptionPane;


/**
 *
 * @author Aleksandra
 */
public class Primer2
    public static void main(String[] args) {
        new Primer2();
    }

    public Primer2() {
        double x = Double.parseDouble(JOptionPane.showInputDialog("Unesite X:"));
        double x1 = (Math.pow(x, 2) - 2 * x + 3) / Math.pow(x, 3);
        double x2 = (Math.pow(x, 3) - 2 * Math.pow(x, 2) - 2 * x + 2) /
Math.sqrt(x);
        double rezultat = (x1 * x2) / Math.sqrt(x);
        System.out.println("Rezultat je: " + rezultat);

    }
}
```

ZADATAK 2

Samostalno vežbanje kombinovanja rezultata različitih jednačina

Modifikovati Primer 2 na sledeći način:

- Kreirati promenljivu rezultat2;
- Ako je x1 < x2, rezultat2 = (x1 + x2) / rezultat;

- U suprotnom, rezultat2 = (x1 - x2) / rezultat.

DEFINISANJE KONSTATI SA FINAL (VIDEO)

Video daje objašnjenje i primere definisanja konstantnih veličina pomoću reyervisane reči u Javi: final.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

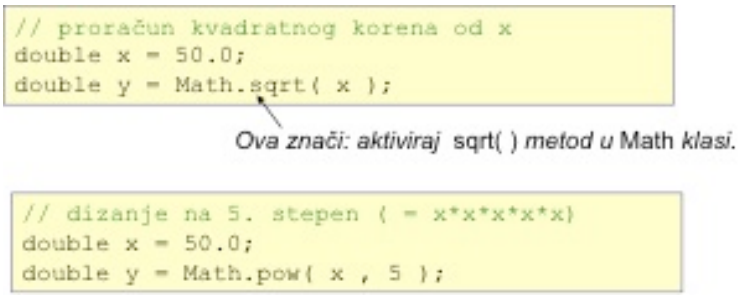
Poglavlje 2

Matematičke funkcije

KLASA MATH

Klasa Math je deo Javine biblioteke koja sadrži veliki broj metoda koji obezbeđuju izračunavanje različitih matematičkih funkcija.

Za rešavanje matematičkih problema uz pomoć nekog programskog jezika, potrebno je da on ne samo podržava aritmetičke operatore, već i izračunavanje uobičajenih standardnih operatora i funkcija. Tako je podrška u Javi izražena kroz mogućnost korišćenja jedne klase koja je deo Javine biblioteke klase. To je klasa***Math*.** Java ima mnogo funkcija u klasi Math. Ona za svaku funkciju obezbeđuje po jedan metod. To su ***static*** metodi, što znači da se mogu aktivirati upotrebom imena klase***Math*** (kasnije će biti više reči o značenju "static"). Na slici 1 dat je primer



Slika 2.1 Primer korišćenja klase Math

Izrazi:

double y = Math.sqrt(x);

double y = Math.pow(x , 5);

pozivaju metode klase ***Math*** koje su potrebne za izračunavanje kvadratnog korena nekog broja x, odnosno, za dizanje nekog broja x na 5. stepen.

KLASA MATH (VIDEO)

Video najpre objašnjava šta je API i njegovo korišćenj ekod klase Math

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIMERI KORIŠĆENJA KLASE MATH

Pozivom odgovarajućeg metoda klase Math i definisanjem odgovarajućih vrednosti argumenata tih metoda, dobija se vrednost tih matematičkih funkcija.

Na slici 2 prikazani su i ostali metodi koje sadrži klasa Math, i koji se mogu na analogni način koristiti. Na slici 3 prikazani su primeri korišćenja ovih metoda

abs(x)	apsloutna vrednost za x
cos(x), sin(x), tan(x)	kosinus, sinus i nagens u *radijanima*
acos(y), asin(y), atan(y), ...	arkus kosinusa, akrkus sinusa itd..
toDegrees(radian)	pretvara radijane u stepene
toRadians(degree)	pretvara stepene u radijane
ceil(x)	gornje zaokruživanje na najbliži *int*
floor(x)	donje zaokurivanje na najbliži *int*
round(x)	zaokružuje na najbliži ceo broj
exp(x)	eksponencijalna funkcijal: $y = e^x$
log(y)	prirodni logaritam od y ($y = e^x$)
pow(a, b)	a^b (a na b)
max(a, b)	max od a i b
min(a, b)	min od a i b

Slika 2.2 Jedan deo metoda klase Math

Izraz	Rezultat	Tip rezultata
`Math.sqrt( 25.0 );`	5.0	double
`Math.sqrt( 25 );`	5.0	double
`Math.log( 100 );`	4.60517018	double
`Math.log10( 100.0 );`	2.0	double
`Math.sin( Math.PI/2 );`	1.0	double
`Math.cos( Math.PI/4 );`	0.70710678	double
`Math.abs( -2.5 );`	2.5	double
`Math.abs( 12 );`	12	int
`Math.max( 8, -14);`	8	int
`Math.min( 8L, -14L);`	-14L	long
`Math.max( 8.0F, 15);`	15F	float
`Math.pow( 2, 10 );`	1024.0	double
`Math.toRadians( 90 );`	1.5707963	double
`Math.E;`	2.7182818...	double
`Math.PI;`	3.1415926...	double

Slika 2.3 Primeri korišćenja metoda klase Math

ODREĐIVANJE MAKSIMALNE I MINIMALNE VREDNOSTI (VIDEO)

Video daje primere koriđćenje metoda max() o min() klase Math

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

NEKI METODI KLASE MATH (VIDEO)

Video objašnjava primenu nekih matematičkih funkcija (metoda) klase Math

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

POLIMORFIZAM POJEDINIH METODA KLASE MATH

Pojedini metodi klase Math su polimorfni, jer mogu dati različite rezultate, u zavisnosti od parametara koji se koriste u pozivu ovih metoda

Kada smo govorili o objektno-orijentisanom modeliranju, pomenuli smo pojam „polimorfizam “. Da se podsetimo: naziv je grčkog porekla i označava „mnogo formi; više formi“, tj. njime označavamo metode koji u različitim kontekstima daju različite rezultate, mada se pozivaju na isti način. I pojedini metodi klase Math su polimorfni, jer mogu dati različite rezultate, u zavisnosti od parametara koji se koriste u pozivu ovih metoda (slika 4).

`abs(x)`	vraća "int" ako je x "int"; vraća "long" ako je x long; vraća "float" ako je x float; vraća "double" ako je x "double".
`max(a,b)`	vraća "int" ako su i a *i* b "int"; vraća "long" ako su a *i* b "long"; itd.
`round(x)`	vraća "float" ako je x float; vraća "double" ako je x "double".
`ali...`	
`sqrt(x)`	*uvek pretvara* x u double i vraća vrednost tipa double. Najveći broj Math funkcija tako radi (sin, cos, tan, log, log10, ...).

Slika 2.4 Poliformni metodi klase Math

Radi ilustracije polimorfizma, na slici 5 navedeni su primeri korišćenja metoda max korišćenjem različitih tipova njegovih parametara.

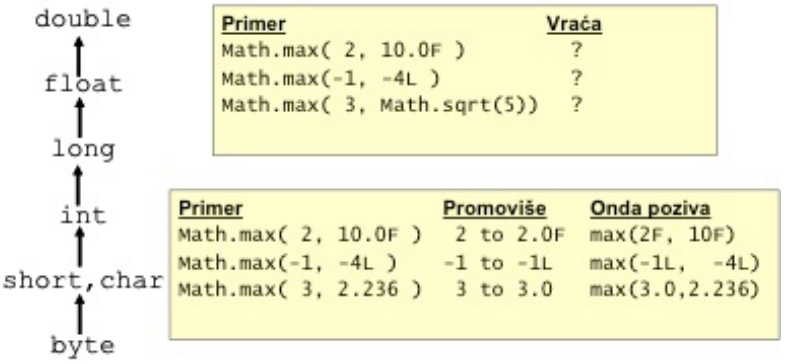
Primer	Vraća
`Math.max( 2, 10 )`	`(int) 10`
`Math.max( -1L, -4L )`	`(long) -1L`
`Math.max( 2F, 10.0F )`	`(float) 10.0F`
`Math.max( -4.0, 0.5 )`	`(double) 0.5`

Slika 2.5 Polimorfizam kod primene metoda max klase Math

AUTOMATSKA KONVERZIJA TIPA PARAMETARA

Java prvo promoviše jedan argument (menja mu tip), a onda traži funkciju koja daje isti tip.

Šta se dešava ako su u istom pozivu metoda parametri različitog tipa? Koji je onda tip vrednosti koja se vraća? Odgovor je sledeći: Java prvo promoviše jedan argument (menja mu tip), a onda traži funkciju koja daje isti tip. Na slici 6 dati su primeri automatske promene tipa unetih parametara u viši tip podataka, da bi onda određivala, shodno tom tipu, tip izlaza iz metoda sa tipom mogućeg rezultata koji daje metod max klase Math



Slika 2.6 Primer automatske konverzije tipa argumenata u skladu sa tipom rezultata koji daje metod max klase Math

Ovo pravilo se primenjuje na sve Math funkcije, a ne samo na polimorfne funkcije. Na slici 7 ovo je ilustrovano na primeru metoda***sqrt***. Kada metod nema rešenje za određene tipove parametara, onda ih on automatski prevodi u drugi, viši tip podataka za koji ima rešenje, tj. izlazni rezultat tipa u koji konvertuje unete parametre metoda

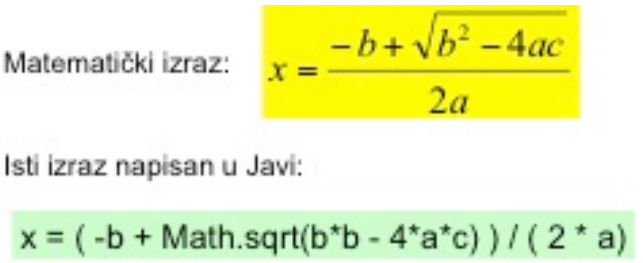
Primer	Akcija
Math.sqrt(2)	nema funkcije sqrt(int). Promoviše drugi tip.
Math.sqrt(2L)	nema funkcije sqrt(long). Promoviše drugi tip.
Math.sqrt(2F	nema funkcije sqrt(float). Promoviše drugi tip.
Math.sqrt(2.0)	Da! sqrt(double) - ovaj tip podržava

Slika 2.7 Automatska konverzija tipa parametara i kod metoda koji nisu polimorfni

REDOSLED IZVRŠAVANJA OPERACIJA U MATEMATIČKIM IZRAZIMA

Primenjuje se redosled izvršavanja koji je u zavisnosti od vrste operacija koje treba izvršiti i od zagrada koje se koriste.

Jedan matematički izraz se definiše kombinacijom potrebnih aritmetičkih operatora i metoda koje obezbeđuje klasa Math. Ovo ćemo ilustrovati na primeru izraza koji označava rešenje neke kvadratne jednačine (slika 8).

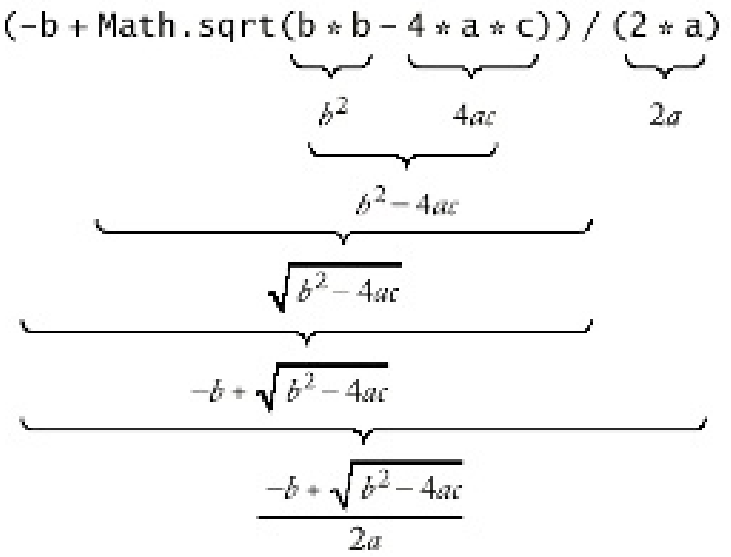


Slika 2.8 Primer pisanja matematičkog izraza u Javi

Sada se postavlja pitanje: Kojim redosledom Java izvršava operacije navedene u matematičkom izrazu opisanom u Javi? Da li koristi redosled, poštujući pravilo „s leva na desno“ ili nešto drugo?

Odgovor je – nešto drugo. Kao i kod drugih programskih jezika (na primer, Fortran), primenjuje se redosled izvršavanja koji je u zavisnosti od vrste operacija koje treba izvršiti i od zagrada koje se koriste.

Prvo se koriste operacije množenja i deljenja, a zatim operacije sabiranja i oduzimanja. Pri ovome, poštuje se i redosled operacija koji definišu primenjene zagrade. Prvo se izračunavaju operacije u okviru unutrašnjih zagrada, pa onda operacije u okviru spoljašnjih zagrada. Ovo je ilustrovano na primeru izraza sa slike 8 , a koji je dat na slici 9 .



Slika 2.9 Redosled izvršavanja matematičkog izraza sa slike 8

GENERISANJE SLUČAJNOG BROJA

Metod random() klase Math generiše slučajni broj tipa double koji je veći ili jednak 0.0, a manji od 1.0.

Metod ***random()*** klase ***Math*** generiše slučajni broj tipa double koji je veći ili jednak 0.0, a manji od 1.0.

Dajemo jednostavan primer generisanja slučajnog broja primenom jednostavnog izraza:

```
(int)(Math.random() * 10);  // Vraća slučajan ceo broj
                            // između 0 i 9
50 + (int)(Math.random() * 50)  //Vraća slučajan ceo broj
                            // između 50 i 99
```

Uopšteno, izraz:

a + (int)Math..random() * b;

Vraća slučajan ceo broj između a i a+b.

RAD SA SLUČAJNO GENERISANIM BROJEVIMA (VIDEO)

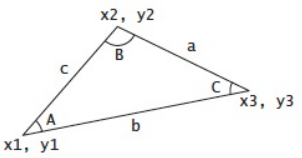
Primena metoda random() klase Math

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIMER 3: PRORAČUN UGLOVA TROUGLA

Me daje metod Match.sqrt() daje kvadratni koren, a metod Math.acos() daje ugao sa dtimi kosinusom, tj. Arkus kosinusa.

Ako je trougao dat dužinama svojih stranica, njegovi uglovi se mogu izračunati pomoću izraza datih na slici 1 .




```
A = acos((a * a - b * b - c * c) / (-2 * b * c))
B = acos((b * b - a * a - c * c) / (-2 * a * c))
C = acos((c * c - b * b - a * a) / (-2 * a * b))
```

Slika 2.10 Proračun uglova trougla sa zadatim stranicama

Unos podataka i prikaz rezultata:

Unesite tri tačke: 1 1 6.5 1 6.5 2.5

Tri ugla su 15.26 90.0 74.74

```
import java.util.Scanner;

  public class ComputeAngles {
    public static void main(String[] args) {
     Scanner input = new Scanner(System.in);

     // Unos koordinata tri temena trougla
     System.out.print("Unesite tri tacke: ");
```

```
    double x1 = input.nextDouble();
    double y1 = input.nextDouble();
    double x2 = input.nextDouble();
    double y2 = input.nextDouble();
    double x3 = input.nextDouble();
    double y3 = input.nextDouble();

    //  Proracun tri strance trougla
    double a = Math.sqrt((x2 - x3) * (x2 - x3)
      + (y2 - y3) * (y2 - y3));
    double b = Math.sqrt((x1 - x3) * (x1 - x3)
      + (y1 - y3) * (y1 - y3));
    double c = Math.sqrt((x1 - x2) * (x1 - x2)
      + (y1 - y2) * (y1 - y2));

    // Proracun tri ugla trougla
    double A = Math.toDegrees(Math.acos((a * a - b * b - c * c)
       / (-2 * b * c)));
    double B = Math.toDegrees(Math.acos((b * b - a * a - c * c)
      / (-2 * a * c)));
    double C = Math.toDegrees(Math.acos((c * c - b * b - a * a)
      / (-2 * a * b)));

    // Prikaz rezultata
    System.out.println("Tri ugla su " +
      Math.round(A * 100) / 100.0 + " " +
      Math.round(B * 100) / 100.0 + " " +
      Math.round(C * 100) / 100.0);
  }
}
```

PRIMER 4: POVRŠINA VALJKA

Metod Math.pow() saje stepe neke vrednosti, a Math.PI daje vrendost konstante PI, tj. broja π.

Napisati program kojim se računa površina valjka na osnovu poluprečnika i visine valjka koji korisnik unosi kao celobrojnu vrednost sa standardnog ulaza. Vrednost broja Pi treba učitati iz klase ***Math***.. Za učitavanje podataka od korisnika treba koristiti objekat klase ***Scanner***, a za ispis podataka objekat ***System.out***.

Analiza problema:

Nakon kreiranja objekta ulaz klase ***Scanne***r i ispisivanja poruke korisniku da unese poluprečnik valjka kreiraćemo promenljivu poluprečnik koja čuva veličinu poluprečnika valjka koju je korisnik uneo i dodeliti joj vrednost koja je unešena sa standardnog ulaza pozivajući objekat ulaz klase ***Scanner*** i njegovu metodu nextInt() koja vraća ceo broj. Isti postupak ćemo ponoviti za visinu valjka. Na osnovu poluprečnika i visine valjka ćemo izračunati površinu valjka po formuli:

2 * poluprecnik * poluprecnik * Math.PI + 2 * poluprecnik * visina * Math.PI

gde je Math.PI statička konstanta klase Math.

Na kraju ćemo ispisati poruku korisniku sa izračunatom površinom valjka koristeći metodu println() objekta System.out.

```
package l02;

 import java.util.Scanner;

 public class Primer4 {
    public static void main(String[] args) {
        Scanner ulaz = new Scanner(System.in);
        System.out.println("Unesite poluprecnik valjka: ");
        int poluprecnikValjka = ulaz.nextInt();
        System.out.println("Unesite visinu valjka: ");
        int visinaValjka = ulaz.nextInt();
        double povrsinaBaza = 2 * Math.pow(poluprecnikValjka, 2) * Math.PI;
        double povrsinaOmotaca = 2 * poluprecnikValjka * Math.PI * visinaValjka;
        double povrsinaValjka = povrsinaBaza + povrsinaOmotaca;
        System.out.println("Povrsina valjka je: " + povrsinaValjka);
    }
 }
```

PRIMER 5: RAČUNANJE PO FORMULI

Metod Math.log() određuje vrednost logaritma, a metod Math.abs() daje apsolutnu vrenost nekog broja. Metod Math.max() daje veži broj od data dva broja.

Napisati program kojim se računa **f po sledećoj formuli, pri čemu korisnik unosi ceo broj a sa standardnog ulaza.**

Ako je **a < 10,**

f = a * sqrt(10 + E) + sin(Pi * 4)

Ako je **50 > a >= 10,**

f = max(Pi * 10, E * 12) + 2^a

Ako je **a >= 50,**

f = a * abs(log(E+4) – Pi)

Program treba da bude deo NetBeans projekta pod nazivom lo2. Naziv klase treba da bude Primer5.

Za učitavanje podataka od korisnika treba koristiti objekat klase Scanner, a za ispis podataka objekat System.out.

Analiza problema: Nakon smeštanja broja koji je učitan sa standardnog ulaza u promenljivu broj A, potrebno je napraviti realnu promenljivu koja u kojoj će biti smešten rezultat računanja po formuli, a zatim koristeći naredbe if i else ispitati uslove na sledeći način i izračunati vrednost rezultata po formuli

```
package l02;

 import java.util.Scanner;

 public class Primer5 {
    public static void main(String[] args) {
        Scanner ulaz = new Scanner(System.in);
        System.out.println("Unesite broj a: ");
        int brojA = ulaz.nextInt();
        double brojF;
        if (brojA < 10){
            brojF = brojA * Math.sqrt(10 + Math.E) + Math.sin(Math.PI * 4);
        }
        else if (brojA < 50){
            brojF = Math.max(Math.PI * 10, Math.E * 12) + Math.pow(2, brojA);
        }
        else{
            brojF = brojA * Math.abs(Math.log(Math.E + 4) - Math.PI);
        }
        System.out.println("Rezultat je: " + brojF);
    }
 }
```

PRIMER 6

Računanje minimuma, maksimuma, apsolutne vrednosti i korena.

Treba napraviti program koji traži od korisnika da unese

String (abs, max, min ili sqrt).

U zavisnosti od unosa korisnik dalje treba da unese vrednost ili dve vrednosti u zavisnosti od funkcije.

Treba ispisati rezultat funckije. Za proveru šta je korisnik uneo koristite if ili switch.

OBJAŠNJENJE:

Inicijalizujemo promenljive za prvuVrednost, druguVrednost i rezultat.

Učitavamo od korisnika String operacija.

Proveravamo case-om koji je od slučaja i u zavisnosti od toga učitavamo jednu ili dve vrednosti i računamo rezultat.

Važno je na kraju svakog CASE-a staviti naredbu break.
Proveriti šta se dešava ako se BREAK izostavi.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package l02.primer6;

import javax.swing.JOptionPane;

/**
 *
 * @author Aleksandra
 */
public class Primer6 {

    /**
     *
     * @param args
     */
    public static void main(String[] args) {
        new Primer6();
    }

    public Primer6() {
        double rezultat = 0;
        double prvaVrednost = 0;
        double drugaVrednost = 0;
        String operacija = JOptionPane.showInputDialog("Unesite operaciju \r\n Za
apsolutnu vrednost: abs \r\n "
                + "Za maximum dva broja: max \r\n  Za minimum dva broja: min \r\n
Za koren vrednosti: sqrt \r\n");
        switch (operacija) {
            case "max":
                prvaVrednost =
Double.parseDouble(JOptionPane.showInputDialog("Unesite prvu vrednost"));
                drugaVrednost =
Double.parseDouble(JOptionPane.showInputDialog("Unesite drugu vrednost"));
                rezultat = Math.max(prvaVrednost, drugaVrednost);
                break;
            case "min":
                prvaVrednost =
Double.parseDouble(JOptionPane.showInputDialog("Unesite prvu vrednost"));
                drugaVrednost =
Double.parseDouble(JOptionPane.showInputDialog("Unesite drugu vrednost"));
                rezultat = Math.min(prvaVrednost, drugaVrednost);
                break;
            case "abs":
```

```
                prvaVrednost =
Double.parseDouble(JOptionPane.showInputDialog("Unesite vrednost"));
                rezultat = Math.abs(prvaVrednost);
                break;
            case "sqrt":
                prvaVrednost =
Double.parseDouble(JOptionPane.showInputDialog("Unesite vrednost"));
                rezultat = Math.sqrt(prvaVrednost);
                break;
        }
        System.out.println("Rešenje je: " + rezultat);
    }

}
```

ZADATAK 3

Samostalno vežbanje primene klase Math i njenih metoda

Na osnovu zadatih formula sa sledeće slike pronaći x1 i x2 a zatim izračunati rezultat. X korisnik treba da unese preko JOptionPane-a.

$$x1 = \frac{x^3 - 4}{\sqrt{CeoDeo(5x)}} \qquad x2 = \frac{x^3 - 5x^3 - 6x + 2}{\sqrt{2x}}$$

$$rezultat = \frac{x1 * x2}{\sqrt{x}}$$

Slika 2.11 Jednačine za zadatak broj 3

Poglavlje 3

Znaci i rad sa njima

ŠTA JE CHAR?

Znakovni tip koji se naziva char, može da predstavlja mala i velika slova, cifre, znakove interpunkcije i kontrolne znake.

Imajući u vidu da se različiti znakovi kod računara često koriste u programskom jeziku Java za predstavljanje pojedinačnih znakova postoji **znakovni tip** koji se naziva ***char***.

Tip ***char*** u memoriji zauzima **2 bajta**, tj. za kodiranje svakog znaka se koristi 16-bitni **Unicode**, to omogućava, korišćenje većine alfabeta koji se koriste u svetu.

Napomena:

Tip char spada u celobrojne tipove u smislu da takvi podaci mogu da se koriste kao operandi u aritmetičkim izrazima. Računaće se s brojčanom vrednošću koda odgovarajućeg znaka (ceo broj). Ova osobina će nam biti neophodna kad budemo kreirali programe koji će generisati slučajna slova, reči i slično.

Tip podataka char sadrži pojedinačne znakove alfabeta kao što su:

- mala i velika slova (A,a,..., Z, z)
- cifre (0,1,...)
- znakove interpunkcije (*,!, ?,...)
- kontrolne znake (razmak, novi red, tabulator...)

Znakovni literali se u programu uokviruju jednim apostrofom:

```
'm'
'y'
'A'
```

UNICODE I ASCII KOD

Unicode je šema za šifriranje znakova u bitne brojeve koju podržava Java.

Računari interno koriste binarne brojeve. Znaci se čuvaju u računaru kao sekvence nula i jedinica (0 i 1). Mapiranje znaka u njegov binarni prikaz se naziva “Encoding sheme” ili šema

šifrovanja znaka. Kako su znakovi šifrovani u niz nula i jedinica nam govori upravo ova šema za znakova. Java podržava *Unicode* šemu šifrovanja znakova, koja je kao standard donešena od strane Unicode konzorcijuma da bi se podržale sve razlike u jezicima na svetu.

Prvobitno je*Unicode* šifrovanje zamišljeno da radi 16-bitno, međutim ispotavilo se da to nije dovoljno pa je uvedeno i 32-bitno šifrovanja kako bi se podržali i jezici poput kineskog. Pošto sve što je predstavljeno sa 16 bita, tj. 2 bajta, može se predstaviti i sa četiri heksadecimalne vrednosti. Na primer, engleska reč Welcome, na kineskom se piše sa dva znaka koji se heksadecimalno mogu predstaviti kao: *\u6B22\u8FCE*. \u predstavlja UNICODE vrednost koja označava 16-bitniu Unicode šifru. Opseg Unicode šifara se kreće od \u0000 do \uFFFF.

Računari najčešće koriste ASCII (American Standard Code for Information Interchange) 8-bitnu šemu šifriranih znakova, sa kojim su mogli da predstave sva velika i mala slova, znake interpunkcije i kontrolne znake.

U donjoj tabeli, prikazani su ASCII šifrirane oznake za najčešće korišćene znake (tip char).

znak	Decimalna šifrirana vrednost	Unicode vrednost
‘o’ - ‘9’	48 - 57	\u0030 - \u0039
‘A’ - ’Z’	65 - 90	\u0041 - \u005A
‘a’ – ’z’	97 / 122	\u0061 - \u007A

Slika 3.1 ASCII decimalne i heksadecimalne šifre za najčešće znake

Unicode koristi ASCII šifru sa \u0000 do \u007F koji odgovaraju 127 ASCII znacima.

Možete koristiti ASCII znake kao što su ‘X’, ‘1’I ‘$’ u Java programu , kao I u Unicode-u. U tom smislu, sledeći iskazi su ekvivalentni:

char letter = 'A';

char letter = '\u0041'; // Character A's Unicode is 0041

Oba dodeljuju znaku A promenljivu letter tipa ***char.***

KONVERZIJA ZNAKOVNIH U NUMERIČKE TIPOVE I OBRNUTO

Znakovni tip –char - se može pretvoriti u broj kao što se i broj može pretvoriti u znakovni tip

Znakovni tip (***char***) se može pretvoriti u broj kao što se i broj može pretvoriti u znakovni tip. Ovo može zvučati čudno ali pošto svaki broj koji je između 0 i 65536 (16 bita) predstavlja određeni znak po UNICODE-u možemo izvršiti konverziju i broj pretvoriti u znak.

Primer 1: Heksadecimalni broj 41 predstavlja slovo A

```
char karakter = (char)0XAB0041;
System.out.println(karakter); // ovaj kod će ispisati veliko slovo A.
```

Ukoliko pokušamo da pretvorimo broj sa decimalnim zarezom u znakovni tip prvo će se otpisati decimalni deo a zatim će se broj konvertovati u karakter.

Primer 2: Decimalan broj 65.25 predstavlja slovo A

```
char karakter = (char)65.25;
System.out.println(karakter); // Ispisuje slovo A
```

Ukoliko uradimo obrnuto konverziju pa karkater A pretvorimo u broj takođe ćemo dobiti broj 65.

```
int i = (int)'A';
System.out.println(i); // Ispisuje 65
```

TIP PODATAKA CHAR (VIDEO)

Video objašnjava tip podataka i njegove konverzije

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

KONTROLNI ZNACI

Kontrolni znaci formatiraju prikaz stringova i drugih znakova na izlaznom uređaju računara (npr. Da monitoru ili štampaču).

Kontrolni znaci se u programima predstavljaju sa nekoliko znakova koji se nalaze unutar apostrofa:

```
'\n'
'\t'
```

Svaki od prethodnih primera predstavlja po jedan znak. Prvi je 16-bitni znakovni tip koji označava **novi red**, a drugi znakovni tip **tabulator**. Na sledećoj slici prikazani su kontrolni (specijalni) znakovni tipovi sa svojim nazivima na engleskom, ASCII oznakama i objašnjenjima.

Spec. karakter	Naziv	ASCII vrednost i značenje
\a	Bell (beep)	007 zvuk bipa
\b	Backspace	008 pomera pokazivač ulevo
\t	Horizontal Tab	009 pomeranje do sledećeg tab-a
\n	New line	010 počinje novu liniju
\v	Vertical Tab	011 vrši vertikalno tabuliranje
\f	Form feed	012 pomera se za jednu liniju dole
\r	Carriage return	013 pomeraj na početak linije
\0	Null	000 karakter nule
\"	Double Quote	034 za upotrebu unutar stringa sa "
\'	Single Quote	039 za upotrebu unutra char sa '
\\	Backslash	092 obratna dupla kosa crta

Slika 3.2 Kontrolni znaci sa nazivima, ASCII oznakama i objašnjenjima.

Bilo koja (celobrojna) vrednost tipa***char*** može da se izrazi u obliku oktalnog ili heksadecimalnog literala. Na taj način mogu da se izraze i vrednosti koje ne predstavljaju nijedan od štampajućih ili upravljačkih znakova na datom računaru.

Heksadecimalne vrednosti pišu se nizom od tačno četiri heksadecimalne cifre (0, 1, 2,..., 9, **A,** B, ...,**F;** mala slova su dozvoljena) iza znaka **\u**. Na taj način može da se predstavlja bilo koji **Unicode** znak, čak i ako to ne podržava okruženje.

Važna napomena:

- Tip podataka ***char*** ne sadrži nikakve informacije o fontu.
- Tip podataka***char*** ne može da sadrži celu reč, već samo jedan znak!
- **Stringovi** (tj. nizovi znakova), koji su jako često korišćena vrsta podataka u programiranju, nemaju odgovarajući primitivni tip u Javi.
- Skup stringova se u Javi smatra klasnim tipom, odnosno stringovi su objekti unapred definisane klase **String**.
- String literali se uokviruju duplim apostrofom - primer: "CS101", dok se znakovni literali uokviruju jednim apostrofom - primer 'ž'

METODI CHARACTER KLASE

Metodi klase Character obezbeđuju testiranje oznaka.

Java obezbeđuje sledeće metod klase Character za testiranje znakova:

Metod	Opis
isDigit(ch)	tačno ako je oznaka ch cifra
isLetter(ch)	Tačno ako je oznaka ch slovo
isLetterOfDigit(ch)	Tačno ako je oznaka ch slovo ili cifra
isLowerCase(ch)	Tačno ako je oznaka ch malo slovo
isUpperCase(ch)	Tačno ako je oznaka ch veliko slovo
toLowerCase(ch)	Daje malu veličinu označene ch oznake
toUpperCase(ch)	Daje veliku veličinu označene ch oznake

Slika 3.3 Metodi klase Character

Na primer,

```
System.out.println("isDigit('a') is " + Character.isDigit('a'));
 System.out.println("isLetter('a') is " + Character.isLetter('a'));
 System.out.println("isLowerCase('a') is "
    + Character.isLowerCase('a'));
 System.out.println("isUpperCase('a') is "
    + Character.isUpperCase('a'));
 System.out.println("toLowerCase('T') is "
    + Character.toLowerCase('T'));
 System.out.println("toUpperCase('q') is "      + Character.toUpperCase('q'));
```

Daje sledeći rezultat:

isDigit('a') je pogrešno

isLetter('a') ie tačno

isLowerCase('a') je tačno

isUpperCase('a') je pogrešno

toLowerCase('T') je t

toUpperCase('q') je Q

PRIMER 7 - POREĐENJE ZNAKOVA I NJIHOVO TESTIRANJE

Dva znaka (tip char) se upoređuju pomoću relacionih operatora kao što se upoređuju dva broja

Dva znaka (tip char) se upoređuju pomoću relacionih operatora kao što se upoređuju dva broja. To se radi tako što se upoređuju dva Unicode broja. Na primer:

'a' < 'b‘ je tačno jer je Unicode za 'a' (97) manji nego Unicode za 'b' (98).

'a' < 'A' je netačno jer je Unicode za 'a' (97) veći od Unicode za 'A' (65).

'1' < '8‘ je tačno jer je Unicode za '1' (49) manji nego Unicode za '8' (56).

Često, u programima treba da proverimo da li je neki znak broj, slovo, veliko slovo ili malo slovo. Imajući u vidu Unicode brojeve za ove oznake, ovo se može utvrditi na sledeći način:

```
if (ch >= 'A' &amp;amp;&amp;amp; ch <= 'Z')
    System.out.println(ch + "je veliko slovo");
else if (ch >= 'a' &amp;amp;&amp;amp; ch <= 'z')
    System.out.println(ch + " je malo slovo");
else if (ch >= '0' &amp;amp;&amp;amp;  ch <= '9')
    System.out.println(ch + " je cifra");
```

ZADATAK 4

Samostalno vežbanje poređenja znakova i njihovog testiranja

Iskoristite prethodni primer:

- Karakter se unosi sa tastature;
- Proverava se kojoj grupi karaktera pripada (videti prethodni primer);
- Štampati na konzoli odgovarajući komentar;

METODI KLASE CHARACTER (VIDEO)

Objašnjavaju se metodi klase Character

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 4

Nizovi sa znacima i rad sa njima

SADRŽAJ

Klasa String sadrži metode koji omogućuju operacije sa nizovima oznaka, tj. teksta definisanim između dva apostrofa.

U ovom delu lekcije, izložiće se deo koji se ondosi na predstavljanje nizova sa znacima i načini kako se radi sa njima, a preko sledećih nastavnih jedinica:

- Klasa String
- Upoređivanje dva niza sa znacima
- Učitavanje znakova sa tastature
- Format prikaza izlaza na monitoru

4.1 Klasa String

ŠTA JE KLASA STRING?

Klasa String je predefinisana klasa u Java biblioteci koja omogućava formiranje teksta koji čini skup znakovnih tipova, tj. char.

Tip***char*** predstavlja samo jedan znakovni tip. Da bi predstavili ceo tekst postoji tip podataka koji se zove ***String***. Na primer, sledeći kod deklariše poruku koja ima vrednost “Dobro došli u Javu”

String poruka = “Dobro došli u Javu”;

String je predefinisana klasa u Java biblioteci kao i klase poput ***Scanner***, ***System*** i ***JOptionPan***e. ***String*** nije primitivni tip podataka. ***String*** je ustvari referentni tip. Svaka klasa može da bude referentni tip nekog tipa ali o tome ćemo pričati u kasnijim lekcijama. Kao što se kod brojeva + za sabiranje kod ***String*** tipa on omogućuje da se dva teksta spoje.

Primer:

```
//Spajanje više poruka
String poruka  = “Dobro došli” + “ u ” + “ Javu”;
//Spajanje poruke i broja u poruku
String poruka2 = “Poglavlje: ” + 2;
```

```
//Spajanje poruke i karaktera u poruku
String poruka3 = “Sektor ” + ‘B’;
```

Pored mogućnosti spajanja vrednosti ***String*** tipa postoji i mogućnost dodavanja na postojeći ***String***. Ovo se radi sa +=.

Primer:

```
String poruka4 = “Java je”;
poruka+=” programski jezik”;
```

Ukoliko želite da izčitate sa konzole reč možete koristiti Scanner javinu klasu.

Primer:

```
Scanner ulaz = new Scanner(System.in);
System.out.println(“Unesite rec:”);
String s1 = ulaz.next();
System.out.println(“Upisali ste reč ” + ulaz);
```

OBJEKAT STRING

Objekat String nije primitivni tip podataka, već predstavlja niz znakova (tipa char) kojim se definiše neki tekst.

U svakom programskom jeziku često radimo sa nizovima znakova, recimo - "CS101" ili "Uvod u programiranje". Ovakvi nizovi karaktera se nazivaju **stringovima**, s obzirom da nemamo adekvatniju reč u srpskom jeziku. Pre dvadesetak godina se koristio termin **niska** kao sinonim za string, ali nije zaživeo.

Treba na kraju primetiti da **stringovi** (tj. nizovi znakova), koji su često korišćena vrsta podataka u programiranju, nemaju odgovarajući primitivni tip u Javi.

Kao što smo rekli - kad za određeni tip podataka nemamo odgovarajući primitivni tip - koristimo neki klasni tip, tj. klasu, koja je već ugrađena u programski jezik Java ili definišemo sopstveni tip - recimo: Student, Igrač, Kladionica...

Skup stringova se u Javi smatra klasnim tipom, odnosno stringovi se tretiraju kao objekti koji pripadaju unapred definisanoj klasi **String**. Po sličnom principu ćemo raditi i sa datotekama, slikama, ikonama, bazama podataka i slično.

Važna napomena:

Naš primarni cilj je da, u početku, ovladamo radom sa primitivnim tipovima i (samo) korišćenjem funkcionalnosti klasa koje su već ugrađene u programski jezik Java, a tek kasnije ćemo praviti sopstvene klase. Ovaj pristup se pokazao kao lakši za razumevanje kod početnika, imajući u vidu da težimo da se u početku uči što manje teorije, a da se radi što više zadataka.

METODI KLASE STRING

Statički metod se može pozvati bez upotrebe nekog objekta. Metodi instance su metodi koji se pozivaju samo od strane nekog objekta. To su metodi klase String.

Tip ***char*** predstavlja samo jednu oznaku. Niz oznaka se predstavlja tipom ***String***. Na primer, sledeći iskaz deklariše veličinu tipa ***String***:

String poruka = “Dobrodošli u Javu!”

String je klasa u Javi koja sadrži metode koji omogućavaju manipulacije sa nizovima oznaka. Promenljiva tipa String je referentnog tipa jer ukazuje na objekat. U gornjem primeru, taj objekat je niz oznaka “Dobrodošli u Javu!”

Metod	Opis
isDigit(ch)	tačno ako je oznaka ch cifra
isLetter(ch)	Tačno ako je oznaka ch slovo
isLetterOfDigit(ch)	Tačno ako je oznaka ch slovo ili cifra
isLowerCase(ch)	Tačno ako je oznaka ch malo slovo
isUpperCase(ch)	Tačno ako je oznaka ch veliko slovo
toLowerCase(ch)	Daje malu veličinu označene ch oznake
toUpperCase(ch)	Daje veliku veličinu označene ch oznake

Slika 4.1.1 Neki metodi klase String

Nizovi oznaka (Strings) u Javi su objekti. Zbog toga, metodi prikazani u tabeli na slici 3 mogu se pozivati samo iz specijalnih primeraka, tj. instanci klasa. Zato se ovi metodi nazivaju i metodi instanci. Metodi koji nisu metodi instanci, nazivaju se statičkim metodima. ***Statički metod*** *se može pozvati bez upotrebe nekog objekta.* Svi metodi klase Math su statički metodi, jer nisu povezani ni sa jednim objektom, tj. primerkom ili instancom neke klase.

Sintaksa za poziv nekog metoda instance je:

referenca-promenljive.imeMetoda(argumenti)

Metod može imati i puno argumanata, a i ne mora da ima argumente. Na primer, metod ***charAt(index)*** ima jedan argument, a metod ***length()*** nema ni jedan.

Sintaksa za statičke metode je:

ImeKlase.imeMetoda(argumenti)

Na primer, metod pow klase Math se poziva na sledeći način:

Math.pow(2, 2.5)

ODREĐIVANJE DUŽINE NIZA ZNAKOVA

Metod length() klase String vraća broj oznaka uzetog niza oznaka

Metod ***length()*** klase String vraća broj oznaka uzetog niza oznaka. To je prikazano u sledećem primeru (daje se samo karakteristični deo koda).

Deo koda:

```
String message = "Dobrodošli u Javu";
 System.out.println("Dužina niza  " + message + " je "
    + message.length());
```

Dobijeni rezultat:

Dužina niza Dobro došli u Javu je 17

Broj oznaka nekog niza možemo dobiti i na sledeći način:

“Dobrodošli u Javu”.length()

Dobija se rezultat: 17

METODI KLASE STRING 1 (VIDEO)

Video prikayuje više primera korišćenje nekih metoda klase String

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

METODI KLASE STRING 2 (VIDEO)

Video pokayuje primrenu još nekih metoda klase String

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

DOBIJANJE ZNAKA IZ NIZA ZNAKOVA

Metod chatAt(index) daje oznaku na dajoj lokaciji niza oznaka.

Ako želimo da dobijemo oznaku koja se nalazi na određenom mestu datog niza znakova (npr. na 5. mestu), onda treba koristiti metod chatAt(index).

Deo koda:

```
String message = "Dobrodošli u Javu";
 System.out.printl("Oznaka na 5. mestu niza " + message + " je "
    + message.charAt(5));
```

Dobijen rezultat:

Znak na 5. mestu niza Dobrodošli u Javu je o

PRIMER 8 - SPAJANJE DVA NIZA ZNAKOVA

Metod concat() klase String spaja dva niza oznaka. Isto se postiže i korišćenjem operatora +.

Metod concat() klase String spaja dva niza oznaka. Na primer, spajanjem nizova s1 i s2 dobija se niz oznaka s3:

String s3 = s1.concat(s2);

Kako je spajanje nizova oznaka, tj. delova tekstova, čest slučaj u programima pisanim u Javi, Java omogućava i jednostavnije spajanje nizova oznaka. Prethodni primer se može i ovako napisati:

String s3 = s1 + s2;

Operator + može spojiti i niz oznaka sa nekim brojem. U tom slučaju broj se konvertuje u niz oznaka i onda se spaja. U ovakvim slučajevima neophodno je da sadrži bar jedan niz oznaka. Ako nijedan od operanda nije niz oznaka (String) onda operator + postaje operator sabiranja, umesto operator spajanja nizova oznaka.

Ovde se daje nekoliko primera spajanja nizova oznaka:

```
// Spajanje tri niza
 String message = "Dobrodošli " + "u " + "Javu";
 // Niz Poglavlje se spaja sa brojem 2
 String s = "Poglavlje" + 2; // s postaje Poglavlje2
 // String Dodatak se spaja sa oznakom B
 String s1 = "Dodatak" + 'B'; // s1 becomes DodatakB
```

Može se koristiti i operator += za spajanje nizova. Na primer, spajanje niza “Dobrodošli u Javu” i niza “Java je zabavna” se može izvršiti ovako:

String message = “Dobrodošli u Javu”:

message += “ i Java je zabavna”;

Dobija se sada sledeći rezultat za objekat message: “Dobrodošli u Javu i Java je zabavna”

ZADATAK 5

Samostalno vežbanje spajanja stringova

- Stringovi se unose sa tastature;
- Unos stringova se vrši sve dok se ne unese string "kraj" ili "Kraj";
- Spojiti sve unete stringove dodajući jednu prazninu između njih;
- U rezultujući string ne ulazi poslednji uneti;
- Kreirati program koji pripada paketu lo2.spajanjeStringovaSaPetljom;
- Klasa se naziva Zadatak5;
- Prevesti program i demonstrirati rezultate izvršavanja (rezultujući string).

KONVERZIJA NIZA ZNAKOVA

Metod toLowerCase() vraća novi niz koji sadrži samo male oznake, a metod toUpperCase() vraća niz samo sa velikim oznakama.

Metod ***toLowerCase()*** vraća novi niz koji sadrži samo male oznake, a metod ***toUpperCase()*** vraća niz samo sa velikim oznakama. Na primer,

Niz “Dobrodošli” sa ***toLowerCase()*** postaje “dobrodošli”

Niz “Dobrodošli” sa ***toUpperCase()*** postaje “DOBRODOŠLI”

Metod ***trim()*** vraća novi niz oznaka (String) uklanjajući prazna polja na krajevima niza.

PRIMER 9

Generisanje slučajnog niza oznaka.

Napisati program koji generiše slučajni niz od 6 oznaka.

Prva oznaka niza treba da bude veliko slovo.

Objašnjenje:

Generisanje slučajnog niza oznaka zahteva korišćenje klase Random. Najpre se primenom klase Random dobija slučajno generisan neki broj.

r.nextInt(26) - dobijamo slučajan broj od 0 do 26

nama treba neko slovo od A do Z, kao su ASCII tablici sva slova redom, jedno za drugim, to znači da slovo B dobijamo dodavanjem 1 na ASCII kod slova 'A'. Za ostala slova važi isto - na slovo A dodajemo redni broj slova koje želimo da dobijemo (od 0 do 26). (char) (r.nextInt(26) + 'a');

Primenom metoda ***valueOf()*** klase String dobija se odgovarajuća oznaka koja odgovara generisanom broju. Prvu generisanu oznaku (slovo) pretvaramo u veliko slovo. Zatim se dodaju ostali znaci niza ***rec***.

Potrebno je vršiti konverziju u char za svaku random generisanu vrednost. Nadovezivanje niske reč vršimo operatorom +

rec += (char) (r.nextInt(26) + 'a');

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package primer9;

import java.util.Random;

/**
 *
 * @author Aleksandra
 */
public class Primer9{

    public static void main(String[] args) {
        new Primer9();
    }

    public Primer9() {
        Random r = new Random();
        String rec = "";
        rec += String.valueOf((char) (r.nextInt(26) + 'a')).toUpperCase();
        rec += (char) (r.nextInt(26) + 'a');
        rec += (char) (r.nextInt(26) + 'a');
        rec += (char) (r.nextInt(26) + 'a');
        rec += (char) (r.nextInt(26) + 'a');
        rec += (char) (r.nextInt(26) + 'a');
        System.out.println("Rec je: " + rec);
    }
}
```

ZADATAK 6

Samostalno vežbanje konverzije iz char u int

Pokušajte da uradite suprotno od prethodnog primera - konvertujte znake u brojeve (broj koji odgovara znaku po ASCII tablici):

- zadati su (naizmenično mala i velika slova): a, B, c, D i e;
- izvršiti konverziju zadatih karaktera u tip int;
- sabrati dobijene brojeve i prikazati rezultat.

DOBIJANJE PODNIZA ZNAKOVA

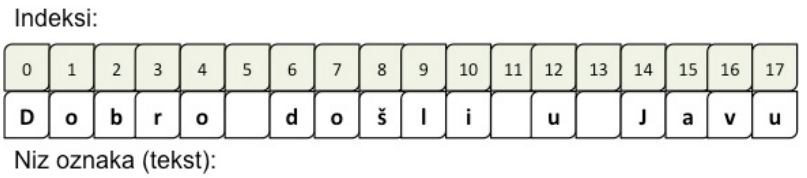
Metod substring() klase String izdvaja deo niya oznaka iz datog niza oznaka.

Primenom metoda ***substring()*** klase String, može se dobiti podniz datog niza oznaka. Donja tabela prikazuje metode koje omogućavaju dobijanje podnizova.

Metod	Opis
index(ch)	Vraća indeks prvog javljanja oznake ch u nizu.Vraća -1 ako je nije našao.
indexOf(ch, fromIndex)	Vraća indeks prvog javljanja oznake ch posle fromIndex u nizu. Vraća -1 ako nije pronađena.
indexOf(s)	Vraća indeks prvog javljanja niza s u ovom nizu. Vraća -1 ako podniz nije pronađen.
lastIndexOf(s, fromIndex)	Vraća indeks prvog javljanja niza s u ovom nizu, posle fromIndex. Vraća -1 ako podniz nije pronađen.
lastIndexOf(ch)	Vraća indeks poslednjeg javljanja oznake ch u nizu.Vraća -1 ako je nije našao.
lastIndexOf(ch, fromIndex)	Vraća indeks poslednjeg javljanja oznake ch posle fromIndex u nizu. Vraća -1 ako nije pronađena.
lastIndexOf(s)	Vraća indeks poslednjeg javljanja niza s u ovom nizu. Vraća -1 ako podniz nije pronađen.
lastIndexOd(s, fromIndex)	Vraća indeks poslednjeg javljanja niza s u ovom nizu, posle fromIndex. Vraća -1 ako podniz nije pronađen.

Slika 4.1.2 Metodi klase String za nalaženje podnizova

Primeri:



Slika 4.1.3 Položaj oznaka u nizu "Dobro došli u Javu"

"Dobrodošli u Javu".indexOf(‘D') vraća 0.

"Dobrodošli u Javu".indexOf('o') vraća 1.

"Dobrodošli u Javu".indexOf('o', 5) vraća 7.

"Dobrodošli u Javu".indexOf("došli") vraća 6.

"Dobrodošli u Javu".indexOf("Javu", 5) vraća 14.

"Dobrodošli u Javu".indexOf("javu", 5) vraća -1.

"Dobrodošli u Javu".lastIndexOf('W') vraća 0.

"Dobrodošli u Javu".lastIndexOf('o') vraća 7.

"Dobrodošli u Javu".lastIndexOf('o', 5) vraća 4.

"Dobrodošli u Javu".lastIndexOf(“došli") vraća 6.

"Dobrodošli u Javu".lastIndexOf("Javu", 5) vraća -1.

"Dobrodošli u Javu".lastIndexOf("Javu") vraća 12.

KONVERZIJA IZMEĐU NIZA ZNAKOVA I BROJEVA

Primenom metoda parsInt(), parseDouble() i dr. može se izvršiti konverzija nizova oznaka u brojeve odgovarajućeg tipa.

Može izvršiti konverzija niza brojeva u neki broj. Da bi se od jednog niza dobio broj tipa***int***, potrebno je koristiti:

```
int  inVlaue = Integer.parseInt(int String)
```

gde je inString uneti niz oznaka, na primer, “123”. Dobijen je ceo broj inValue=123

Da bi se neki niz oznaka konvertovao u broj tip double, koristi se metod: Double.parseDouble() na sledeći način:

```
double doubleValue = Double.parseDouble(doubleString);
```

gde je uneti niz brojeva “123.45”, a dobijeni broj tipa double je 123.45.

Neki broj možete konvertovati u niz oznaka spajanjem operatorom na sledeći način:

```
String s = number + "";
```

KONVERZIJU STRINGOVA U CELE BROJEVE (VIDEO)

Video pokazuje kako se stringovi mogi konvertovati u cele brojeve.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIMER 10

Isecanje String promenjivih.

Napisati program koji od korisnika zahteva da unese svoje ime, prezime i državu rođenja. Program treba da na osnovu unetih parametara uzme prva tri slova imena, zadnja tri slova prezimena, prva tri slova države i spoji u jedan String. String prikazati u konzoli.

Objašnjenje:

substring metoda String-a nam omogućava da isecamo String po volji.

Prvi parametar substring-a je početan broj karaktera za isecanje, a drugi je broj karaktera do koga se String seče. String se iseca uključujući prvi indeks, a ne uključujući drugi. Ako metodu pozovemo od 0 do 3, dobićemo 0, 1 i 2. karakter Stringa nad kojim je pozivamo.

Vrlo je bitno da substring metoda vraća novi String i ne menja postojeći.

ime.substring(0, 3) - dobijamo prva tri karaktera niske IME

prezime.substring(prezime.length() - 3, prezime.length()) - dobijamo poslednja tri karaktera niske PREZIME (da bismo to odredili moramo unazad da brojimo, tj. da koristimo dužinu stringa koju dobijamo length()

drzavaRodjenja.substring(0, 3) - dobijamo prva tri slova niske DRŽAVA

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package primer10;

import javax.swing.JOptionPane;

/**
 *
 * @author Aleksandra
 */
public class Primer10 {

    public static void main(String[] args) {
        new Primer10();
    }

    public Primer10() {
        String ime = JOptionPane.showInputDialog("Unesite ime:");
        String prezime = JOptionPane.showInputDialog("Unesite prezime:");
        String drzavaRodjenja = JOptionPane.showInputDialog("Unesite drzavu
rodjenja:");
        String obrada = ime.substring(0, 3) + prezime.substring(prezime.length() -
3, prezime.length()) + drzavaRodjenja.substring(0, 3);
        System.out.println("Obrađen tekst je = " + obrada);
    }
}
```

ZADATAK 7

Samostalno vežbanje da li se znak nalazi u reči i koliko puta

- Sa tastature se unose znak i string;
- Proveriti da li se uneti znak nalazi u unetom stringu;
- Ukoliko se nalazi, prebrojati koliko se puta pojavljuje;
- Prikazati rezultate na konzoli.

4.2 Upoređivanje dva niza sa znacima

METOD EQUALS()

Metod equals() klase String utvrđuje da li su sadržaji dva niza znakova jednaki ili nisu.

Klasa ***String*** sadrži metode potrebne za upoređivanje nizova sa znacima. Oni su dati u donjoj tabeli.

Metod	Opis
equals(s1)	Vraća tačno ako je ovaj niz jednak sa niyom s1
equalsIgnoreCase(s1)	Vraća tačno ako je ovaj niz jednak sa nizom s1, a veličina oznaka je nebitna
compareTo(s1)	Vraća neki ceo broj veći od 0, jednak 0 ili manji od 0, da bi ukazao da je ovaj niz veći, jednak ili manji od niza s1.
compareToIgnoreCase(s1)	Isto kao compareTo, samo što ne reaguje na veličinu oznaka
startWith(prefix)	Vraća tačno ako ovaj niz počinje sa unetim prefiksom
endsWith(suffix)	Vraća tačno ako se ovaj niz završava sa unetim sufiksom
contains(s1)	Vraća tačno ako je s1 podniz ovog niza

Slika 4.2.1 Metod klase String za upoređenje nizova oznaka

Za upređivanje dva niza sa znakovima ne možemo koristiti operator == jer on daje samo informaciju da li su dva niza, kao objekti, na istoj adresi. Da bi se utvrdilo da li dva niza imaju isti sadržaj, treba koristiti metod ***equals()***.

Sledeći primer pokazuje kako se on može koristiti.

```
if (string1.equals(string2))
 System.out.println("string1 i imaju isti sadržaj");
 else
 System.out.println("string1 and string2 nisu jednaki");
```

U sledećem primeru se upoređuju tri data niza znakova.

```
String s1 = "Dobrodišli u Javu";
 String s2 = "Dobrodošli u Javu";
 String s3 = "Dobrodošli u C++";
 System.out.println(s1.equals(s2)); // tačno
 System.out.println(s1.equals(s3)); // pogrešno
```

METOD ZA UPOREĐENJE DVA NIZA

Metod CompareTo() se koristi za upređenja dva niza znakova. Daje 0, ako su dva niza jednaka, <0 ako je niz s1 leksikografski manji od niza s2,

Metod ***CompareTo()*** se koristi za upređenja dva niza sa znacima. On daje 0, ako su dva niza jednaka, <0 ako je niz s1 leksikografski manji od niza s2, a >0 ako je niz s1 leksikografski veći od niza s2.

Upređivanje dva niza s1 = “abc” i s2 = “abg”

s1.comareTo(s2);

pokazuje sledeće:

- Ako su prve dve oznake jednake, upoređuje se treća oznaka u oba niza.
- Kako je po Unicode-u, oznaka **c** za 4 manja od Unicode oznake za oznaku **g**, to se dobija vrednost -4.

Klasa String sadrži i metode ***equalsIgnoreCase()*** i ***compareToIgnoreCase()*** za upoređivanje dva niza oznaka. Ovi metodi ne reaguju na veličinu oznaka u ovim nizovima.

Klasa ***String*** sadrži i metode ***startsWith(prefix) i endsWith(suffix)*** tako da iskaz: **str.startsWith(prefix)**

proverava da li niz str počinje sa navedenim prefiksom.

Iskaz: **str.endsWith(suffux)**

proverava da li se niz str završava sa navedenim sufiksom).

Iskaz : ***str.contains(s1)***

proverava da li niz str sadrži u sebi niz s1.

Evo nekoliko primera:

“Dobrodošli u Javu".startsWith(“Dobro") vraća true.

" Dobrodošli u Javu a".startsWith(“dobro") vraća false.

." Dobrodošli u Javu ".endsWith("vu") vraća true

" Dobrodošli u Javu ".endsWith("v") vraća false

" Dobrodošli u Javu ".contains(“u") vraća true.

" Dobrodošli u Javu” .contains(“U") vraća false

PRIMER UPOREĐIVANJA DVA NIZA

Medod nextLine() unosi niz oznaka, zajedno sa praznim poljima, sve dok se ne pritisne tipka Enter.

Program treba da pita korisnika da unese nazive dva grada i da ih onda prikaže na monitoru poređane po alfabetskom redosledu.

Dobijeni rezultat:

Unesite naziv prvog grada: Niš

Unesite naziv drugog grada: Beograd

Alfabetski redosled unetih gradova je Beograd Niš

Programski kod:

```
import java.util.Scanner;

 public class OrderTwoCities {
   public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    // Pita korisnika da unese nazive dva grada
    System.out.print("Unesite naziv prvog grada: ");
    String city1 = input.nextLine();
    System.out.print("Unesite naziv drugog grada: ");
    String city2 = input.nextLine();

    if (city1.compareTo(city2) < 0)
        System.out.println("Alfabetski redosled unetih gradova je " +
            city1 + " " + city2);
    else
        System.out.println("Alfabetski redosled unetih gradova je " +
           city2 + " " + city1);
  }
 }
```

UPOREĐENJE STRINGOVA (VIDEO)

Video pokazuje primere upoeđenja stringova

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIMER 11

Poredjenje stringova - equals i ==

Napisati java program koji od korisnika zahteva da unese Ime I Prezime kao jedan string.

Zatim metodomsubstring podeliti ime I prezime u dva stringa (od I do beline).

Potom napraviti novi string ime I prezime koje je razdvojeno belinom tako što ćete konkatenirati nisku ime i nisku prezime.

Uporediti ime_prezime sa novo_ime_prezime metodom equals I operatorom == I ispisati rezultate u vidu logičkih vrednosti u konzolu.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package primer11;

import javax.swing.JOptionPane;

/**
 *
 * @author Aleksandra
 */
public class Primer11{

    public static void main(String[] args) {

        new Primer11();
    }

    public Primer11() {
        // == "ne cita" sta pise u String-ovima
        String ime_prezime = JOptionPane.showInputDialog("Unesite ime i prezime");
        //pravimo nisku ime od 0. indeksa do indeksa gde se pojavljuje ""
        String ime = ime_prezime.substring(0, ime_prezime.indexOf(" "));
        // pravimo nisku prezime od  pozicije na kojoj se pojavilo "" do kraja
indeksa
        //važno je da prezime ne počne karakterom "", pa zbog toga je početni
indeks uvećan za 1
        String prezime = ime_prezime.substring(ime_prezime.indexOf(" ") + 1);


        String novo_ime_prezime = ime + " " + prezime;

        if (ime_prezime == novo_ime_prezime) {
            System.out.println("== vratio true");
        } else {
            System.out.println("== vratio false");
```

```
        }

        if (ime_prezime.equals(novo_ime_prezime)) {
            System.out.println("equals() vratio true");
        } else {
            System.out.println("equals() vratio false");
        }

    }
}
```

PRIMER 12

Porednjenje stringova compareTo() metodom

Napisati java program koji od korisnika zahteva da unese Ime I Prezime kao jedan string.

Zatim metodomsubstring podeliti ime I prezime u dva stringa (od I do beline).

Potom napraviti novi string ime I prezime koje je razdvojeno belinom tako što ćete konkatenirati nisku ime i nisku prezime.

Uporediti ime_prezime sa novo_ime_prezime metodom compareTo.

```
Opis metode compareTo:
* @param anotherString the {@code String} to be compared.
* @return the value {@code 0} if the argument string is equal to
* this string; a value less than {@code 0} if this string
* is lexicographically less than the string argument; and a
* value greater than {@code 0} if this string is
* lexicographically greater than the string argument.
*/
```

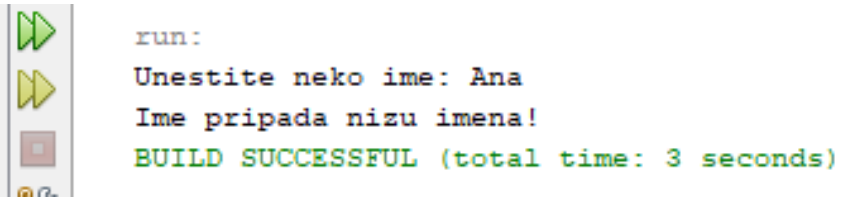
```
package primer12;

//import zadatak8.*;
import javax.swing.JOptionPane;

/**
 *
 * @author Aleksandra
 */
public class Primer12 {

    public static void main(String[] args) {

        new Primer12();
    }
```

```
    public Primer12() {
        // == "ne cita" sta pise u String-ovima
        String ime_prezime = JOptionPane.showInputDialog("Unesite ime i prezime");
        //pravimo nisku ime od 0. indeksa do indeksa gde se pojavljuje ""
        String ime = ime_prezime.substring(0, ime_prezime.indexOf(" "));
        // pravimo nisku prezime od  pozicije na kojoj se pojavilo "" do kraja
indeksa
        //važno je da prezime ne počne karakterom "", pa zbog toga je početni
indeks uvećan za 1
        String prezime = ime_prezime.substring(ime_prezime.indexOf(" ") + 1);

        String novo_ime_prezime = ime + " " + prezime;

        int rez = ime_prezime.compareTo(novo_ime_prezime);
        if (rez == 0) {
            System.out.println(" leksikografski su jednaki");
        } else if (rez < 0) {
            System.out.println(ime_prezime + " < " + novo_ime_prezime);
        } else {
            System.out.println(ime_prezime + " > " + novo_ime_prezime);
        }
    }
}
```

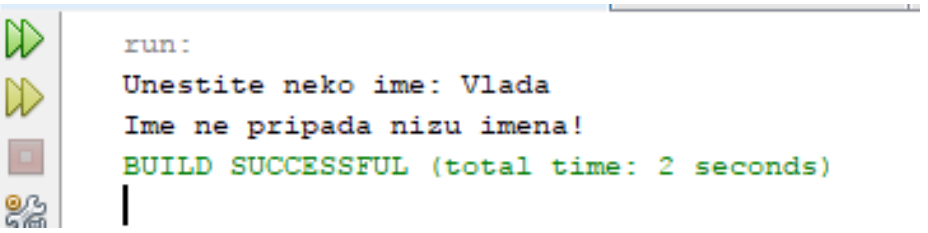
ZADATAK 8

Samostalno vežbanje upoređivanja stringova

- Dat je niz stringova: String [] imena = {"Petar", "Milan", "Jovana", "Ana"};
- Unosi se string ime sa tastature;
- Primenom metoda equals() ili compareTo() proveriti da li se uneto ime nalazi u zadatom nizu imena;
- Koristiti naredbu for koja prolazi kroz niz i uzima jedan po jedan string;
- Koristiti logičku promenljivu pronadjen za utvrđivanje da li ime pripada nizu;
- Izlaz neka bude kao na sledećim slikama.



Slika 4.2.2 Ime pripada nizu imena



Slika 4.2.3 Ime ne pripada nizu imena

4.3 Učitavanje znakova sa tastature

UČITAVANJE NIZA ZNAKOVA SA TASTATURE

Metod next() klase Scanner omogućava učitavanje niza oznaka sa konzole

Metod ***next()*** klase ***Scanner*** omogućava učitavanje niza oznaka sa konzole. Metod ***next()*** čita jedan niz sve do javljanja praznog (blanko) polja.

Ako se želi pročitati cela linija koja sadrži više nizova odvojenih praznim poljem, onda se koristi metod ***nextLine().*** On učitava sve upisane oznake na liniji konzole do pritiska tipke Enter, uključujući i prazna polja.

Metod ***next()*** se poziva iz objekta klase Scanner, kao što je to pokazano u sledećem primeru.

Sledeći primer pokazuje primenu metoda ***nextLine()*** klase Scanner.

```
Scanner ulaz = new Scanner(System.in);
 System.out.print("Učitaj tri reči odvojene praznim poljem: ");
 String s1 = ulaz.next();
 String s2 = ulaz.next();
 String s3 = ulaz.next();
 System.out.println("s1 je " + s1);
 System.out.println("s2 je " + s2);
 System.out.println("s3 je " + s3);
```

Dobijeni rezultat – prikaz na monitoru:

Učitaj tri reči odvojene praznim poljem: Dobrodošli u Javu

s1 je Dobrodošli

s2 je u

s3 je Javu

```
Scanner ulaz = new Scanner(System.in);
 System.out.println("Učitaj liniju: ");
```

```
String s = ulaz.nextLine();
System.out.println("Učitana linija je " + s);
```

Dobijeni rezultat – prikaz na monitoru:

Učitaj linju: Dobrodošli u Javu

Učitana linija je Dobrošli u Javu

UČITAVANJE JEDNOG ZNAKA SA TASTATURE

Metod chatAt() daje oznaku na datoj poziciji datog niza oznaka.

Da bi se pročitala samo jedna oznaka (char) sa konzole, potrebno je koristiti metod ***nextLine()*** i onda pozvati metod ***chatAt(0).***

Sledeći primer pokazuje kako se učitava samo jedna oznaka (tipa char) preko konzole.

Deo koda:

```
Scanner ulaz = new Scanner(System.in);
 System.out.print("Unesi oznaku: ");
 String s = ulaz.nextLine();
 char ch = s.charAt(0);
 System.out.println("Uneta oznaka je " + ch);
```

PRIMER 13

Primena UpperCase i toLowerCase metoda String-a na učitani String

Napisati program koji od korisnika zahteva da unese tekst: upper ili lower. Zatim korisnik unosi određeni tekst. Nakon unosa u zavisnosti od toga da li je prvo upisao upper ili lower sve unete karaktere pretvoriti u velika slova ili mala.

Objašnjenje:

toLowerCase metoda String-a omogućava smanjivanje svih slova na mala slova, dok metoda toUpperCase omogućava povećavanje svih slova na velika slova.

Metoda unos.equals("lower") proverava da li su po vrednosti jednake niska UNOS i STRING "lower".

```
package primer13;

import javax.swing.JOptionPane;

/*
```

```
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author Aleksandra
 */
public class Primer13 {

    public static void main(String[] args) {
        new Primer13();
    }

    public Primer13() {
        String unos = JOptionPane.showInputDialog("Unesite nešto od sledećih
opcija:"
                + "\r\n Da napravite sva uneta slova malim upišite: lower \r\n"
                + "Da napravite sva uneta slova velikim upišite: upper");

        String tekst = JOptionPane.showInputDialog("Unesite tekst");
        if (unos.equals("lower")) {
            tekst = tekst.toLowerCase();
        } else if (unos.equals("upper")) {
            tekst = tekst.toUpperCase();
        }
        System.out.println("Tekst je: " + tekst);
    }

}
```

PRIMER 14

Korišćenje StringBuilder-a

Napraviti Java program koji od korisnika zahteva unos jednog stringa i zatim proverava da li je taj string palindrom.

String je palindom ako se isto čita s leva i s desna.

NPR: RATAR, ANA...

Napraviti StringBuilder koji će okrenutni nisku.

detaljno objašnjenje koda je u komentarima.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package primer14;
```

```
import java.util.Scanner;

/**
 *
 * @author Aleksandra
 */
public class Primer14 {

    public static void main(String[] args) {
        new Primer14();
    }
    /* Metod ispituje da li je String zadat kao argument palindrom */

    public static boolean palindrom(String kandidat) {
        // U klasi StringBuilder postoji metod reverse()
        // za obrtanje stringa (u klasi String ne postoji)
        //kako string nije promenljiv, kada bismo uradili
        // kandidat.reverse() ne bi došlo do suštinske promene niske KANDIDAT.
        //Preporuka je da student pokuša da ispiše nisku kandidat pre i nakon
poziva metode reverse()

        //zato koristimo StringBuilder koji nam pravi novi string koji je moguće
menjati
        StringBuilder pom = new StringBuilder(kandidat);

        // Obrtanje sadrzaja (sadrzaj StringBuilder objekta na koji
        // referise promenljiva pom je promenjen)
        pom.reverse();
        // Vracamo se nazad na objekat klase String
        String obrnuti = pom.toString();

        return kandidat.equals(obrnuti);

        /* kraci nacin (umesto svega prethodnog) */
        // return new StringBuilder(kandidat).reverse().toString().equals(kandidat);
    }

    public Primer14() {

        Scanner sc = new Scanner(System.in);
        System.out.println("Unesite string za proveru: ");
        String kandidat = sc.next();
        /* npr. String "anavolimilovana" jeste palindrom */
        if (palindrom(kandidat)) {
            System.out.println("String je palindrom");
        } else {
            System.out.println("String nije palindrom");
        }
        sc.close();

    }
}
```

ZADATAK 9

Samostalno vežbanje učitavanje znakova sa tastature

Unesite string str1 sa tastature preko konzole;

Unesite string str2 sa tastature preko InputDialog-a;

Proverite :

- koji sting ima veću dužinu;
- da li se kraći string nalazi u dužem stringu kao podstring;
- da li prvi string sadrži slovo "a"
- da li drugi string sadrži slovo "B";
- konvertujete oba stringa tako da su sačinjeni od velikih slova;
- spojite stringove, sa jednim praznim mestom između njih;
- prikazati na konzoli sve rezultate izvršavanja programa.

4.4 Format prikaza izlaza na monitoru

FORMATIRANJE PRIKAZA IZLAZA NA MONITORU

Metod System.out.printf() se koristi za definisanje formata prikaza izlaznih rezultata iz programa na konzoli, tj. monitoru.

Metod ***System.out.printf()*** se koristi za definisanje formata prikaza izlaznih rezultata iz programa na monitoru.

Primenom metoda ***System.out.pringln()*** može se prikazati izlazni rezultati programa na monitoru. Međutim, ako želimo da izlazne rezultate prikažemo vrlo precizno, kao što se to često radi, na primer, sa oznakama valuta, onda treba koristiti metod: ***Syste.out.printf().***

Sledeći primer prikazuje upotrebu pintf() metoda:

```
double amount = 12618.98;
 double interestRate = 0.0013;
 double interest = amount * interestRate;
 System.out.printf("Interest is $%4.2f",
    interest);
```

Dobijen rezultat:

Interest is $16.40

Sintaksa za metod printf() je:

System.out.printf(format, item1, item2, ..., itemk)

Argument format definiše kako će nešto da bude prikazano, a ostali argumenti pokazuju to što treba da bude prikazano (brojevi slova, i dr.).

Donja tabela prikazuje argument format u metodu printf(). .

format	Izlaz	Primer
%b	Logičke veličine	True ili false
%c	Neki znak	'a'
%d	Decimalni ceo broj	200
%f	Realan broj tipa float	45.460000
%e	Broj u standardnoj naućnoj notaciji	4.556000e +01
%s	Niz oznaka	"Java je laka za učenje"

Slika 4.3.1 Specifikacija argumenta format u metodu printf()

Primer:

```
int count = 5;

double amount = 45.56;

System.out.printf("count is %d and amount is %f", count,   amount);
```

Prikaz na monitoru:

count is 5 and amount is 45.560000

AUTOMATSKO PODEŠAVANJE PRAZNOG PROSTORA

Ako veličina koja se prikazuje zahteva više prostora nego što je specificirano, onda se prostor automatski povećava

Ako veličina koja se prikazuje zahteva više prostora nego što je specificirano, onda se prostor automatski povećava. Evo primera:

System.out.printf("%3d#%2s#%4.2f\n", 1234, "Java", 51.6653);

prikazuje:

234#Java#51.67

Veličine se formatiraju s desna na levo. Formatiranje s leva na desno se vrši ako se ubaci -. .

Na primer, sledeći izrazi:

System.out.printf("%8d%8s%8.1f\n", 1234, "Java", 5.63);

System.out.printf("%-8d%-8s%-8.1f \n", 1234, "Java", 5.63);

daju sledeći prikaz:

Slika- 2 Prikaz rezultata na konzoli (monitoru)

PRIMER 15 - PRIMENA METODA PRINTF()

Metodt printf() obezbeđuje prikaz rezulata u definisanom obliku.

Ovde se daje primer programa koji koristi metod printf(). Takođe, dat je i prikaz rezultata koje daje program.

```
public class FormatDemo {
   public static void main(String[] args) {
    // Prikay naslova tabele
    System.out.printf("%-10s%-10s%-10s%-10s%-10s\n", "Stepeni",
       "Poluprečnici", "Sinus", "Cosinus", "Tangens");

    // Prikaz vrednosti za ugao od 30 stepeni
    int degrees = 30;
    double radians = Math.toRadians(degrees);
    System.out.printf("%-10d%-10.4f%-10.4f%-10.4f%-10.4f\n", degrees,
        radians, Math.sin(radians), Math.cos(radians),
        Math.tan(radians));

    // Prikaz vrednosti za ugao od 60 stepen
    degrees = 60;
    radians = Math.toRadians(degrees);
    System.out.printf("%-10d%-10.4f%-10.4f%-10.4f%-10.4f\n", degrees,
        radians, Math.sin(radians), Math.cos(radians),
        Math.tan(radians));
   }
 }
```

Stepeni Poluprečnici Sinus Cosinus Tangens

30 0.5236 0.5000 0.8660 0.5773

60 1.0472 0.8660 0.5000 1.7320

ZADATAK 10

Format prikaza izlaza na monitoru - samostalna vežba

- Vratite te se na primere Primer1 i Zadatak1;
- Koristeći naredbu prinff() formatirajte rezultate tako da se prikazuju sa dve decimale;
- Ponovo prevedite i pokrenite program.

- Dokumentujte rezultate izvršavanja programa.

FORMATIRANJE STRINGOVA PRIMENOM METODA PRINTF() (VIDEO)

Video objašnjava primenu metoda printf() za formatiranje stringova

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 5

Domaći zadaci

ZADACI 1 - 3

Zadaci 1 - 3 su zadaci za samostalan rad bez rešenja

Zadatak 1

Napraviti program koji daje rešenje jednačine na sledećoj slici. Napomena: Za ceo deo koristiti Math.floor.

$$y = \frac{x^3 - 4}{\sqrt{CeoDeo(5x)}}$$

Zadatak 2

Treba napraviti program koji traži od korisnika da unese String (exp, round, floor ili ceil). U zavisnosti od unosa korisnik dalje treba da unese vrednost ili dve vrednosti u zavisnosti od funkcije. Koristiti istoimene funkcije iz Math biblioteke. Treba ispisati rezultat funkcije. Za proveru šta je korisnik uneo koristite if ili switch.

ZADACI 4 - 6

Zadaci 4 - 6 su zadaci za samostalan rad bez rešenja

Zadatak 4

Napraviti program koji menja svaku reč Jabuka u tekstu u Kruška i svaku reč Put u tekstu u Autoput. Ispisati prva 3 i zadnja 3 karaktera izmenjenog teksta. Napomena za zamenu reči u tekstu koristiti. Replace(“stara reč”,”nova reč”);

Zadatak 5

Napraviti program koji ispisuje 10 random karaktera na ekranu sa prvim velikim slovom. Napomena: Koristiti random klasu i pretvaranje brojeva u karaktere.

Zadatak 6

Napraviti program koji na tekst koji korisnik unese doda 10 random karaktera a potom slovo a svuda zameni sa slovom b.

Zakljucak

POUKE LEKCIJE

Rezime naučenog u ovoj lekciji

1. Aritmetički operatori omogućavaju izvršenje jednostavnih matematičkih operacija (sabiranje, oduzimanje, deljenje, množenje i dr.) nad operandima, tj. nad datim vrednostima (promenljive primitivnog tipa).
2. Java obezbeđuje metode klase ***Math*** koje obezbeđuju matematičke finkcije: ***sin, cos, tan, asin, acos, atan, toRadians,toDegree, exp, log, log10, pow, sqrt, cell, floor, rint, round, min, max,abs, i random***
3. ***Tip znaka char predstavlja jedan znak.***
4. ***Izlazne sekvence čine kosa crta \ iza koje se nalazi jedan znak ili kombinacija digitalnih veličina.***
5. ***Znak \ se naziva izlaznom sekvenciom, ili kontrolnim kodom.***
6. ***Znaci ' ', \t, \f, \r, i \n su oznake razmaka.***
7. ***Znaci se mogu upređivati korišćenjem njihovih Unicode brojeva, a upotrebom relacionih operatora.***
8. Klasa ***Character*** sadrći metode***: isDigit, isLetter, isLetterOrDigit, isLowerCase, i isUpperCase*** za testiranje da li je znak cifra, slovo, malo slovo i veliko slovo. Klasa takođe sadrži metode ***toLowerCase*** i ***toUpperCase*** koje daje mala, odn. velika slova.
9. String je niz znakova. Vrednost objekta String je definisana nizom znakova između duplih apostrofa “. Vrednost jednog znaka (tip char) defisana jednim znakom između jednostrukih apostrofa ‘.
10. Stringovi su objekti u Javi. Metod koje se može pozvati samo od strane specifičnog objekta naziva metod instance. Metod koji nije metod instance, je statički metod koji može da se pozove bez upotrebe nekog objekta.
11. Dužina striga, tj. teksta, se određuje metodom ***length()*** . Metodom ***charAt(index)*** daje znak na mestu u nizu koji je definisan indeksom. Metodima***indexOf*** i ***lastIndexOf*** nalaze oznaku ili podniz (deo stringa).

NASTAVAK ZAKLJUČAKA

Još 4 zaključka

1. Metodom ***concat()*** ili upotrebom operatora plus (+) vrši se spajanje dva ili više nizova znakova (stringova).
2. Upotrebom ***substring()*** dobija se podniz (deo stringa) nekog niza (stringa).
3. Primenom metoda ***equals() i compareTo()*** upoređuju se nizovi znakova, tj. stringova. Metod ***equals()*** vraća ***true*** ako su dva niza znakova jednaki, a ***false***, ako nisu. Metod ***compareTo()*** vraća 0, pozitivan ceo broja, ili negativni ceo broj, zavisno

od toga jedan niz znakova (string), jednak, manji ili veći od drugog niza znakova (stringa).

4. Metod ***printf()*** se koristi za dobijanje formatiranog izlaza, tj. prikaza na monitoru, a zavisno od specifikatora formata.