



CS101 - UVOD U OBJEKTNO- ORIJENTISANO PROGRAMIRANJE

Elementarno programiranje u Javi

Lekcija 02

PRIRUČNIK ZA STUDENTE

CS101 - UVOD U OBJEKTNO-ORIJENTISANO PROGRAMIRANJE

Lekcija 02

ELEMENTARNO PROGRAMIRANJE U JAVI

- ✓ Elementarno programiranje u Javi
- ✓ Poglavlje 1: Pisanje jednostavnog programa
- ✓ Poglavlje 2: Čitanje unosa sa konzole i identifikatori
- ✓ Poglavlje 3: Promeljive i dodeljivanje iskaza i izraza
- ✓ Poglavlje 4: Konstante i konvencije imenovanja
- ✓ Poglavlje 5: Numerički tipovi podataka i operacije
- ✓ Poglavlje 6: Literali i izračunavanje izraza
- ✓ Poglavlje 7: Studija slučaja: Prikaz trenutnog vremena
- ✓ Poglavlje 8: Operatori
- ✓ Poglavlje 9: Konverzije numeričkog tipa
- ✓ Poglavlje 10: Studija slučaja: Brojanje novčanih jedinica
- ✓ Poglavlje 11: Uobičajene greške i zamke
- ✓ Poglavlje 12: Video snimci i test za samotestiranje 1 i 2
- ✓ Poglavlje 13: Pokazna vežba
- ✓ Poglavlje 14: Individualna vežba
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

▼ Uvod

CILJEVI

Ciljevi ove lekcije

- Pisanje Java programa za obavljanje jednostavnih proračuna (§2.2).
- Dobijanje unosa sa konzole koristeći klasu Scanner (§2.3).
- Koristiti identifikatore za imenovanje promenljivih, konstanti, metoda i klasa (§2.4).
- Korišćenje promenljivih za čuvanje podataka (§2.5 i 2.6).
- Programirati sa iskazima dodele i izrazima dodele (§2.6).
- Koristiti konstante za čuvanje trajnih podataka (§2.7).
- Određivanje naziva klasa, metoda, promenljivih i konstanti prateći njihove konvencije o imenovanju (§2.8).
- Java numerički primitivni tipovi podataka: bajt, kratki, int, dugački, plutajući i dvostruki (§2.9).
- Učitavanje bajt, kratku, int, dugu, plutajuću ili dvostruku vrednost iz tastatura (§2.9.1).
- Izvođenje operacija pomoću operatora +, -, *, / i % (§2.9.2).
- Izvođenje eksponentnih operacija koristeći Math.pow(a, b) (§2.9.3).
- Pisanje celobrojnih literala, literala sa pomičnim zarezom i literala u naučnom notacija (§2.10).
- Pisanje i procenu numeričkih izraza (§2.12).
- Dobijanje trenutno sistemskog vremena koristeći System.currentTimeMillis() (§2.13).
- Korišćenje proširenih operatora dodele (§2.14).
- Razlika između postinkrementa i preinkrementa i između postdekrement i preddekrement (§2.15).
- Prebacivanje vrednosti jednog tipa na drugi tip (§2.16).
- Opis procesa razvoja softvera i njegova primena na razvoj program otplate kredita (§2.17).
- Pisanje programa koji pretvara veliku količinu novca u manji jedinice (§2.18).
- Izbegavanje uobičajenih grešaka i zamki u osnovnom programiranju (§2.19).

UVOD U LEKCIJU

Fokus ove lekcije je na učenju elementarnih tehnika programiranja za rešavanje problema.

U lekciji 1 naučili ste kako da kreirate, kompajlirate i pokrećete veoma osnovne Java programe. U ovoj lekciji naučićete kako da rešavate probleme pisanjem programa. Kroz ove probleme naučićete elementarno programiranje koristeći primitivne tipove podataka, promenljive, konstante, operatore, izraze i ulaz i izlaz. Naučićete i osnovne korake koji

se odnose na analizu problema, projektovanje rešenja i implementaciju rešenja kreiranjem programa.

▼ Poglavlje 1

Pisanje jednostvanog programa

PROGRAM ZA IZRAČUNAVANJE POVRŠINE KRUGA

Pisanje programa uključuje postavljanje strategije za rešavanje problema, a zatim korišćenje programskog jezika za implementaciju te strategije.

Pisanje programa uključuje projektovanje algoritama i prevođenje algoritama u uputstva za programiranje, ili kod. Algoritam navodi korake koje možete da pratite da biste rešili problem. Algoritmi mogu pomoći programeru da isplanira program pre nego što ga napiše u programskom jeziku. Algoritmi se mogu opisati na prirodnim jezicima ili u pseudokodu (prirodni jezik pomešan sa nekim programskim kodom).

Algoritam za izračunavanje površine kruga može se opisati na sledeći način:

1. Učitaj poluprečnik kruga. (radius)
2. Izračunajte površinu (area) koristeći sledeću formulu: $area = radius * radius$
* Program deklarise poluprečnik i površinu kao promenljive. Ključna reč double označava da su radius i površina vrednosti sa pomičnim zarezom uskladištene u računaru
3. Prikažite rezultat.

Savet

Uvek je dobra praksa da ocrtate svoj program (ili njegov osnovni problem) u obliku algoritma pre nego što počnete da kodirate.

Kada kodirate – tj, kada pišete program – vi prevodite algoritam u program. Već znate da svaki Java program počinje definicijom klase u kojoj ključnu reč `class` prati ime klase. Pretpostavimo da ste izabrali **ComputeArea** kao naziv klase. Nacrt programa bi izgledao ovako:

```
public class ComputeArea {  
    // Details to be given later  
}
```

Slika 1.1 Određivanje imena klase

ODREĐIVANJE NAZIVA KLAŠE I METODA MAIN()

Svaki Java program počinje svoje u glavnom metodu main() koji mora da bude u nekoj klasi (class) te program počinje da se izvršava iz klase main()

Kao što znate, svaki program pisan u Javi (zvaćemo ga kraće: "Java program") sadrži glavni metod programa (ili aplikacije) pod nazivom main() u kome počinje izvršenje programa:

```
public class ComputeArea {  
    public static void main(String[] args) {  
        // Step 1: Read in radius  
  
        // Step 2: Compute area  
  
        // Step 3: Display the area  
    }  
}
```

Slika 1.2 Određivanje metoda main ()

Program treba da pročita poluprečnik koji je korisnik uneo sa tastature. Ovo otvara dva važna pitanja:

- Učitavanje veličine poluprečnika (radius)
- Čuvanje veličine poluprečnik u programu

Da bi sačuvao unetu vrednost poluprečnika, program treba da deklarise simbol koji se zove promenljiva. Promenljiva predstavlja vrednost sačuvanu u memoriji računara.

DEKLARISANJE PROMENLJIVIH

Svaka promenljiva treba da bude ofređenog tipa, i to se deklarise (objavljuje). Realni brojevi su brojevi sa pomerljivom decimalnom tačkom i pripadaju tipu double.

Umesto da koristite x i y kao imena promenljivih, izaberite opisna imena: u ovom slučaju, **radius** za poluprečnik i **area** za površinu. Da bi kompajler znao šta su radius i area , navedite njihove tipove podataka. To je vrsta podataka uskladištenih u promenljivoj, bilo da je ceo broj, realan broj, ili nešto drugo. Ovo je poznato kao **deklarisanje promenljivih**. Java pruža jednostavne tipove podataka za predstavljanje celih, realnih brojeva, znakova i Bulovih tipova. Ovi tipovi su poznati kao primitivni tipovi podataka ili **fundamentalni tipovi**.

Realni brojevi (tj. brojevi sa decimalnom tačkom- u SAD, odn. zapetom - u Evropi) su predstavljeni korišćenjem metode poznate kao pokretna tačka (**floating point**) u računarima. Pošto se decimalni brojevi u računarstvu koriste sa tačkom, a ne sa zapetom (tj. zarezmom), to ćemo mi ovde realne brojeve nazivati kao brojeve sa "pokretnom decimalnom tačkom". U Javi možete koristiti ključnu reč double da biste deklarovali promenljivu sa pomičnom decimalnom

tačkom. U slučaju našeg programa, potrebno je sada da objavimo da za vrednost promenljivih **radius** (poluprečnik) i **area** (površina) očekujemo da budu tipa **double**. Program deklarise poluprečnik i površinu kao promenljive. Ključna reč **double** označava da su promenljive **radius** i **area** uskladištene u računaru kao tip **double**. Program se može proširiti na sledeći način:

```
public class ComputeArea {  
    public static void main(String[] args) {  
        double radius;  
        double area;  
  
        // Step 1: Read in radius  
  
        // Step 2: Compute area  
  
        // Step 3: Display the area  
    }  
}
```

Deklarisanje promenljivih

Slika 1.3 Deklarisanje promenljivih: radius i area

UNOS VREDNOSTI POLUPREČNIKA I OBRAČUN POVRŠINE KRUGA

Unos vrednosti poluprečnika, proračun i prikaz površine kruga .

Promeljivim veličinama potrebno je odrediti inicijalne vrednosti (koje se kasnije mogu menjati). U našem primeru, to znači treba odrediti incijalnu vrednost za promenljivu **radius**, a zatim uneti formulu za proračun površine kruga i time odrediti vrednost promenljive **area**. Na kraju, treba dobijeni rezultat prikayati na monitor računara. Zato, naš program sada izgleda ovako:

LISTING 2.1 ComputeArea.java

```
1 public class ComputeArea {  
2     public static void main(String[] args) {  
3         double radius; // Declare radius  
4         double area; // Declare area  
5  
6         // Assign a radius  
7         radius = 20; // radius is now 20  
8  
9         // Compute area  
10        area = radius * radius * 3.14159;  
11  
12        // Display results  
13        System.out.println("The area for the circle of radius " +  
14            radius + " is " + area);  
15    }  
16 }
```

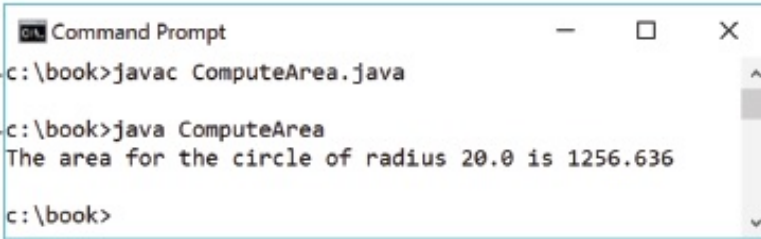
Slika 1.4 Listing klase ComuteArea posle dodeljivanja vrednosti promenljivoj radius, obračuna porvšine kruga area i prikazivanja rezultata na monitoru

Znak plus (+) ima dva značenja: jedno za sabiranje, a drugo za spajanje (kombinovanje) nizova. Znak plus (+) u redovima 13-14 naziva se operator spajanja stringova. Kombinuje dva stringa u jedan. Ako je niz kombinovan sa brojem, broj je pretvoren u string i spojen sa drugim nizom. Prema tome, znaci plus (+) u redovima 13-14 spajaju nizove u duži niz

KOMPAJLIRANJE, IZVRŠENJE I PRAĆENJE IZVRŠENJA PROGRAMA

Pored kompajliranja i izvršenja programa, praćenjem rezultata po linijama programa može biti korisno programerima pri analizi rada programa.

Na slici 5 je prikazan prikaz postupka kompajliranja i izvršenja programa **ComputeArea.java** u slučaju da se koristi konzola (tastatura računara), a na slici 6 je prikazana tabela sa rezultatom koji daje svaka linija programa. To se zove praćenje (**tracing**) programa. **Programi za praćenje (tracing programs)** su korisni za razumevanje kako programi funkcionišu i korisni su alati za pronalaženje grešaka u programima.



```
Command Prompt
c:\book>javac ComputeArea.java
c:\book>java ComputeArea
The area for the circle of radius 20.0 is 1256.636
c:\book>
```

Slika 1.5 Kompajliranje i izvršenje programa ComputeArea sa konzole, tj. tastature.

line#	radius	area
3	no value	
4		no value
7	20	
10		1256.636

Slika 1.6 Praćenje rezultata po programskim linijama programa ComputeArea (slika 4)

▼ Poglavlje 2

Čitanje unosa sa konzole i identifikatori

UČITAVANJE VREDNOSTI POLUPREČNIKA SA KONZOLE

Čitanje unosa sa konzole (tastature) omogućava programu da prihvati unos od korisnika.

U **Listingu 2.1**, poluprečnik (radius) je fiksiran u izvornom kodu. Da biste koristili drugačiji vrednost poluprečnika, morate da izmenite izvorni kod i ponovo ga kompajlirate. Očigledno, ovo nije zgodno, pa umesto toga možete koristiti klasu Scanner za unos na konzoli.

Java koristi **System.out** za označavanje standardnog izlaznog uređaja, a **System.in** za standardni ulazni uređaj. *Podrazumevas da da izlazni uređaj je monitor a da je ulazni uređaj je tastatura.* Da biste izvršili izlaz na konzoli, jednostavno koristite metod **println()** za prikaz nekog broja ili stringa (niza slova) na konzoli, tj. na monitoru.

Da biste izvršili unos sa konzole, potrebno je da koristite klasu **Scanner** da kreirate objekat za čitanje unosa sa System.in, na sledeći način:

```
Scanner input = new Scanner(System.in);
```

Sintaksa **new Scanner(System.in)** kreira objekat tipa **Scanner**. Sintaksa **Scanner input** deklarise da je ulaz promenljiva čiji je tip **Scanner**. Cela linija **Scanner input = new Scanner(System.in)** kreira objekat Scanner i dodeljuje njegovu referencu ulazu promenljive. Objekat može pozvati svoje metode. Pozvati metod na objektu znači tražiti od objekta da izvrši zadatak. Možete da pozovete metod **nextDouble()** da pročitate duplu vrednost na sledeći način:

```
double radius = input.nextDouble();
```

Ova izraz čita broj sa tastature i dodeljuje brojčanu vrednost poluprečniku (radius).

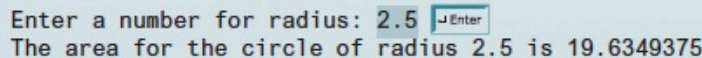
U Javi, nazivi klasa počinju velikim slovom (npr, ComputeArea) , a metoda - malim. Da bi se metodi posebno istakli, mi ćemo uvek u tekstu, pored naziva metoda, dodavati i otvorenu i zatvorenu malu zagradu, npr. main(), println(),

UNOS POUPREČNIKA SA KONZOLE U PROGRAMU COMPUTEAREA

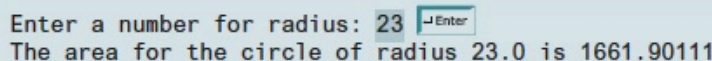
Listing programa ComputeAreaWithConsoleInput i prikaz unosa radijusa i dobijene površine

LISTING 2.2 ComputeAreaWithConsoleInput.java

```
1 import java.util.Scanner; // Scanner is in the java.util package
2
3 public class ComputeAreaWithConsoleInput {
4     public static void main(String[] args) {
5         // Create a Scanner object
6         Scanner input = new Scanner(System.in);
7
8         // Prompt the user to enter a radius
9         System.out.print("Enter a number for radius: ");
10        double radius = input.nextDouble();
11
12        // Compute area
13        double area = radius * radius * 3.14159;
14
15        // Display results
16        System.out.println("The area for the circle of radius " +
17            radius + " is " + area);
18    }
19 }
```



Enter a number for radius: 2.5 Enter
The area for the circle of radius 2.5 is 19.6349375



Enter a number for radius: 23 Enter
The area for the circle of radius 23.0 is 1661.90111

Slika 2.1 Listing programa ComputeAreaWithConsoleInput i prikaz unosa radijusa i dobijene površine

Klasa **Scanner** je u paketu **java.util** u liniji 1. Linija 6 kreira objekat **Scanner**. Imajte na umu da naredba za unos može biti izostavljena ako je zamenite **Scanner** sa **java.util.Scanner** u redu 6.

Red 9 prikazuje niz "Unesite broj za radijus: " na konzoli. Ovo je poznato kao prompt, jer upućuje korisnika da unese unos. Vaš program uvek treba da kaže korisniku šta da unese kada očekuje unos sa tastature.

Podsetimo se da je metoda **print()** u redu 9 identična metodi **println()**, osim što se **println()** pomera na početak sledećeg reda posle prikaza stringa, ali **print()** ne prelazi na sledeći red kada se završi.

Linija 6 kreira objekat **Scanner**. Izjava u redu 10 čita unos sa tastature.

double radius = input.nextDouble();

Pošto što korisnik unese broj i pritisne taster Enter, program čita broj i dodeljuje ga radijusu. Više detalja o objektima biće predstavljeno u lekciji 9. .

UVOZ KLASSE SCANNER IY PAKETA JAVA.UTIL

Iskaz import jednostavno govori kompajleru gde da locira klase

Klasa **Scanner** je u paketu **java.util**. je u program uvezena u liniji 1. Postoje dve vrste **import** iskaza:

- specifičan uvoz import i
- uvoz džoker znakova.

Specifični uvoz specificira jednu klasu u iskazu za uvoz. Na primer, sledeći iskaz uvoyi uvozi klasu **Scanner** iz pakteta **java,util** :

import java.util.Scanner;

Uvoz džoker znakova uvozi sve klase u paketu koristeći zvezdicu kao džoker znak.

Na primer, sledeća izjava uvozi sve klase iz paketa java.util;

import java.util.*;


Informacije o klasama u uvezenom paketu se ne čitaju u vreme kompajliranja ili izvođenja osim ako se klasa ne koristi u programu. Iskaz **import** jednostavno govori kompajleru gde da locira klase. Ne postoji razlika u performansama između specifičnog uvoza i deklaracije uvoza sa džoker znakom

PROGRAM COMPUTE AVERAGE

Primer čitanja višestrukih unosa sa tastature. Program čita tri broja i prikazuje njihov prosek.

LISTING 2.3 ComputeAverage.java

```
1 import java.util.Scanner; // Scanner is in the java.util package
2
3 public class ComputeAverage {
4     public static void main(String[] args) {
5         // Create a Scanner object
6         Scanner input = new Scanner(System.in);
7
8         // Prompt the user to enter three numbers
9         System.out.print("Enter three numbers: ");
10        double number1 = input.nextDouble();
11        double number2 = input.nextDouble();
12        double number3 = input.nextDouble();
13
14        // Compute average
15        double average = (number1 + number2 + number3) / 3;
16
17        // Display results
18        System.out.println("The average of " + number1 + " " + number2
19            + " " + number3 + " is " + average);
20    }
21 }
```



Enter three numbers: 1 2 3 Enter
The average of 1.0 2.0 3.0 is 2.0



Enter three numbers: 10.5 Enter
11 Enter
11.5 Enter
The average of 10.5 11.0 11.5 is 11.0

Slika 2.2 Proračun srednje vrednosti za tri uneta broja preko tastature

Kodovi za uvoz klase **Scanner** (red 1) i kreiranje **Scanner** objekta (red 6) su isti kao u prethodnom primeru, kao i u svim novim programima koje ćete pisati za čitanje unosa sa tastature.

Red 9 traži od korisnika da unese tri broja. Brojevi se čitaju u redovima 10–12. Možete uneti tri broja razdvojena razmacima, zatim pritisnuti taster **Enter** ili uneti svaki broj nakon čega sledi pritisak na taster Enter, kao što je prikazano u primerima pokretanja ovog programa.

Ako ste uneli unos koji nije numerička vrednost, pojavila bi se greška tokom izvršavanja.

U 12. lekciji ćete naučiti kako da rukujete izuzetkom kako bi program mogao da nastavi da radi

IDENTIFIKATORI

Identifikatori su imena koja identifikuju elemente kao što su klase, metode i promenljive u programu

Kao što vidite u Listingu 2.3, **ComputeAverage**, **main ()**, **input**, **number1**, **number2**, **number3** i tako dalje su imena stvari koje se pojavljuju u programu. U programskoj terminologiji, takva imena se nazivaju identifikatori. Svi identifikatori moraju poštovati sledeća pravila:

- Identifikator je niz znakova koji se sastoji od slova, cifara, donjih crta (_) i znakova dolara (\$)
- Identifikator *mora da počinje slovom, donjom crtom (_) ili znakom dolara (\$)*. Ne može početi cifrom.
- Identifikator *ne može biti rezervisana reč*. Pogledajte Dodatak A za listu rezervisanih reči. Rezervisane reči imaju specifično značenje u jeziku Java. Ključne reči su rezervisane reči.
- Identifikator *može biti bilo koje dužine*.

Na primer, **\$ 2**, **ComputeArea**, **area**, **radius**, i **println()** su legalni identifikatori, dok **2A** i **d+4** nisu, jer ne poštuju pravila. Java kompajler otkriva nedozvoljene identifikatore i prijavljuje greške u sintaksi.

▼ Poglavlje 3

Promeljive i dodeljivanje iskaza i izraza

ŠTA SU PROMENLJIVE?

Promeljive se koriste za predstavljanje vrednosti koje se mogu menjati u programu

Promeljive (variables) se koriste za čuvanje vrednosti koje će se kasnije koristiti u programu. Zovu se promenljive jer se njihove vrednosti mogu menjati. U programu u Listingu 2.2, radijus i površina su promenljive tipa double. Radijusu i površini možete dodeliti bilo koju numeričku vrednost, a vrednosti poluprečnika i površine se mogu ponovo dodeliti. Na primer, u sledećem kodu (slika 3), radijus je inicijalno 1,0 (red 2), a zatim promenjen u 2,0 (red 7), a oblast je podešena na 3,14159 (red 3), a zatim resetovana na 12,56636 (red 8).

```
1 // Compute the first area
2 radius = 1.0;                                radius: 1.0
3 area = radius * radius * 3.14159;             area: 3.14159
4 System.out.println("The area is " + area + " for radius " + radius);
5
6 // Compute the second area
7 radius = 2.0;                                radius: 2.0
8 area = radius * radius * 3.14159;             area: 12.56636
9 System.out.println("The area is " + area + " for radius " + radius);
```

Slika 3.1 Unošenje vrednosti poluprečnika kruga dva puta

Promeljive služe za predstavljanje podataka određenog tipa. Da biste koristili promenljivu, morate je deklarirati tako što ćete kompajleru reći njeno ime, kao i koju vrstu podataka može da skladišti. Deklaracija promenljive govori kompajleru da dodeli odgovarajući memorijski prostor za promenljivu na osnovu njenog tipa podataka. Sintaksa za deklarisanje promenljive je:

datatype variableName;

PRIMERI DEKLARACIJA PROMENLJIVIH

Kako promenljiva se deklarira sa tipom podataka koji koriste.

Evo nekoliko primera deklaracija promenljivih:

```
int count;           // Declare count to be an integer variable
double radius;       // Declare radius to be a double variable
double interestRate; // Declare interestRate to be a double variable
```

Slika 3.2 Primeri deklaracija promenljivih

Ovi primeri koriste tipove podataka **int** i **double**. Kasnije ćete se upoznati sa dodatnim tipovima podataka,

Ako su promenljive istog tipa, mogu se deklarirati zajedno, na sledeći način:

datatype variable1, variable2, . . . , variablen;

Promenljive su odvojene zarezima. Na primer,

int i, j, k; // Declare i, j, and k as int variable

Promenljive često imaju početne vrednosti. Možete deklarirati promenljivu i inicijalizovati je u jednom koraku. Razmotrite, na primer, sledeći kod:

int count = 1;

Ovo je ekvivalentno sledeće dve izjave:

int count;

count = 1;

Takođe možete koristiti skraćeni obrazac da zajedno deklarirate i inicijalizujete promenljive istog tipa.

Na primer, **int i = 1, j = 2;**

PRIMERI PRIMENE ISKAZA DODELE

Iskaz dodele promenljivoj dodeljuje vrednost. Iskaz dodele se takođe može koristiti kao izraz u Javi.

Nakon što je promenljiva deklarirana, možete joj dodeliti vrednost pomoću naredbe za dodelu. U Javi, znak jednakosti (=) se koristi kao operator dodele. Sintaksa za naredbe dodele je kao što sledi:

variable = expression;

Izraz predstavlja izračunavanje koje uključuje vrednosti, promenljive i operatore koje, uzimajući ih zajedno, daje vrednost. U iskazu dodele, izraz na desnoj strani operatora dodele se vrednuje, a zatim se vrednost dodeljuje promenljivoj na levoj strani operatora dodele. Na primer, razmotrite sledeći kod:

```
int y = 1;           // Assign 1 to variable y
double radius = 1.0; // Assign 1.0 to variable radius
int x = 5 * (3 / 2);  // Assign the value of the expression to x
x = y + 1;           // Assign the addition of y and 1 to x
double area = radius * radius * 3.14159; // Compute area
```

Slika 3.3 Upotreba operator dodelivanja (=)

Možete koristiti promenljivu u izrazu. Promenljiva se takođe može koristiti sa obe strane operatora =. Na primer,

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

U ovom iskazu o dodeljivanju, rezultat $x + 1$ se dodeljuje x . Ako je x 1 pre izvršenja naredbe, onda postaje 2 nakon što se naredba izvrši.

Da biste dodelili vrednost promenljivoj, morate da postavite ime promenljive levo od operatora dodeljivanja. Dakle, sledeća izjava je pogrešna:

```
1 = x; // Pogrešno
```

PRIMENA OPERATORA DODELE

U Javi, izraz dodele je u suštini izraz koji vrednuje vrednost koja se dodeljuje promenljivoj na levoj strani operatora dodele

U Javi, izraz dodele je u suštini izraz koji vrednuje vrednost koja se dodeljuje promenljivoj na levoj strani operatora dodele. Iz tog razloga, izraz dodele je poznat i kao izraz dodele. Na primer, sledeća izjava je tačna:

```
System.out.println(x = 1);
```

što je ekvivalentno

```
x = x + 1
```

```
System.out.println(x);
```

Ako je vrednost dodeljena više promenljivih, možete da koristite lančane dodele poput ove:

```
i = j = k = 1;
```

što je ekvivalentno

```
x = 1;
```

```
j = k;
```

```
i = j;
```


▼ Poglavlje 4

Konstante i konvencije imenovanja

IMENOVANJE KONSTANTE

Imenovana konstanta je identifikator koji predstavlja trajnu vrednost.

LISTING 2.4 ComputeAreaWithConstant.java

```
1 import java.util.Scanner; // Scanner is in the java.util package
2
3 public class ComputeAreaWithConstant {
4     public static void main(String[] args) {
5         final double PI = 3.14159; // Declare a constant
6
7         // Create a Scanner object
8         Scanner input = new Scanner(System.in);
9
10        // Prompt the user to enter a radius
11        System.out.print("Enter a number for radius: ");
12        double radius = input.nextDouble();
13
14        // Compute area
15        double area = radius * radius * PI;
16
17        // Display result
18        System.out.println("The area for the circle of radius " +
19            radius + " is " + area);
20    }
21 }
```

Slika 4.1 BrojPI je definisan ka konstanta i uneta vrednost se više ne može menjati

Vrednost promenljive može da se promeni tokom izvršavanja programa, ali imenovana konstanta, ili jednostavno konstanta, predstavlja trajne podatke koji se nikada ne menjaju. Poznata je i konstanta kao konačna varijabla u Javi. U našem **ComputeArea** programu, konstanta PI, ako je često koristite, ne želite da nastavite da kucate 3.14159; umesto toga, možete deklarirati konstantu za PI. Evo sintakse za deklarisanje konstante:

final datatype CONSTANTNAME = value;

Konstanta mora biti deklarirana i inicijalizovana u istoj naredbi. Reč final je Java ključna reč za deklarisanje konstante. Po konvenciji, sva slova u konstanti su velika. Na primer, možete deklarirati broj pi kao konstantu i ponovo napisati Listing 2.2, kao u Listingu 2.4.

Tri prednosti primene konstanti: (1) Vrednost se samo jedanput unosi, (2) Ako treba promeniti vrednost, mewa sasmo na jednom mestu, gde je defisana, i (3) Lako se čita program koji ima konstante

KONVENCIJE IMENOVANJA

Držanje Java konvencija o imenovanju čini vaše programe lakim za čitanje i izbegava greške.

Uverite se da ste odabrali deskriptivna imena sa direktnim značenjima za promenljive, konstante, klase i metode u vašem programu. Kao što je ranije pomenuto, **imena razlikuju velika i mala slova**. Dole su navedene konvencije za imenovanje promenljivih, metoda i klasa.

- **Koristite mala slova za promenljive i metode**—na primer, promenljive **radius** i **area**, i metoda **print**.
- **Ako se ime sastoji od nekoliko reči, spojite ih u jednu**, čineći prvu reč malim i velikim početnim slovom svakog sledeće, .na primer, promenljiva , **numberOfStudents**. Ovaj stil imenovanja je poznat kao CamelCase jer velika slova u imenu podsećaju na grbu kamile.
- **Napišite veliko početno slovo svake reči u imenu klase**—na primer, imena klasa **ComputeArea** ili **System**
- Napišite veliko slovo u konstanti i koristite donje crte između reči—na primer, konstante **PI** i **MAKS_VALUE**.

Važno je da se pridržavate konvencija o imenovanju da bi vaši programi bili laki za čitanje.

Oprez

Nemojte birati imena klasa koja se već koriste u Java biblioteci. Na primer, pošto je klasa **System** definisana u Javi, ne bi trebalo da imenujete svoju klasu **System**.

▼ Poglavlje 5

Numerički tipovi podataka i operacije

PREGLED NUMERIČKIH TIPOVA I OPERACIJA

*Java ima šest numeričkih tipova za cele brojeve i brojeve sa pokretnim zarezom sa operatorima +, -, *, / i %.*

Svaki tip podataka ima opseg vrednosti. Kompajler dodeljuje memorijski prostor za svaku promenljivu ili konstantu prema njenom tipu podataka. Java pruža osam primitivnih tipova podataka za numeričke vrednosti, znakove i logičke vrednosti. Ovaj odeljak predstavlja numeričke tipove podataka i operatore.

Tabela 2.1 navodi šest numeričkih tipova podataka, njihovi opsezi vrednosti za skladišćenje

Tip	Opis	Opseg vrednosti
byte	Celi broj, 8-bitni	-2^7 do 2^7-1 (-128 do 127)
short	Celi broj, 16-bitni	-2^{15} do $2^{15}-1$ (-32,768 do +32,767)
int	Celi broj, 32-bitni	-2^{31} do $2^{31}-1$ (-2147483648 do 2147483647)
long	Celi broj, 64-bitni	-2^{63} do $2^{63}-1$ (-10E18 do +10E18)
float	Broj sa pokretnim zarezom jednostruke tačnosti, 32-bitni	-3.4E+38 do +3.4E+38
double	Broj sa pokretnim zarezom dvostruke tačnosti, 64-bitni	-1.7E+308 do 1.7E+308
char	Znak, 16-bitni, može se tretirati kao neoznačeni 16-bitni celi broj	'\u0000' do '\uFFFF',
boolean	Logička vrednost	true ili false

Slika 5.1 Numerički tipovi podataka, njihovi opisi i opsezi vrednosti za skladišćenje

ČITANJE BROJEVA SA TASTATURE

*Učitavanje vrednosti podataka sa tastature se vrše metodima klase **Scanner***

Znate kako da koristite metod **nextDouble()** u klasi **Scanner** za čitanje vrednosti tipa **double** sa tastature. Takođe možete da koristite metode navedene u tabeli 2.2 da biste pročitali broj tipa **byte**, **short**, **int**, **long**, i **float** tip podatka.

Tabela 2.2:MethodsFor Scanner Objects

Metod	Opis
nextByte()	Čita ceo broj tipa byte
nextShort()	Čita ceo broj tipa short
nextInt()	Čita ceo broj tipa int
nextLong()	Čita ceo broj tipa long
nextFloat()	Čita decimalni broj tip float
nextDouble()	Čita decimalni broj tip double

Slika 5.2 Metodi Scanner objekta za unos primitivnih vrednosti

Evo primera učitavanja vrednosti različitih tipova podataka preko tastature:

```
1 Scanner input = new Scanner(System.in);
2 System.out.print("Enter a byte value: ");
3 byte byteValue = input.nextByte();
4
5 System.out.print("Enter a short value: ");
6 short shortValue = input.nextShort();
7
8 System.out.print("Enter an int value: ");
9 int intValue = input.nextInt();
10
11 System.out.print("Enter a long value: ");
12 long longValue = input.nextLong();
13
14 System.out.print("Enter a float value: ");
15 float floatValue = input.nextFloat();
```

Slika 5.3 Učitavanje vrednosti različitih tipova podataka sa tastature

Ako unesete vrednost sa netačnim opsegom ili formatom, pojaviće se greška tokom izvršavanja. Na primer, ako unesete vrednost 128 za red 3, došlo bi do greške jer je 128 van opsega za ceo broj tipa byte

NUMERIČKI OPERATORI

Operandi su vrednosti kojima upravlja operator.

Operatori za numeričke tipove podataka uključuju standardne aritmetičke operatore: sabiranje (+), oduzimanje (-), množenje (*), deljenje (/) i ostatak (%), kao što je navedeno u tabeli 2.3. Operandi su vrednosti kojima upravlja operator.

Tabela 2.3 Numerički operatori

Naziv	Značenje	Primer	Rezultat
+	Sabiranje	$34 + 1$	35
-	Oduzimanje	$34.0 - 0.1$	33.9
*	Množenje	$300 * 30$	9000
/	Delenje	$1.0 / 2.0$	0.5
%	Ostatak	$20 \% 3$	2

Slika 5.4 Numerički operatori

Kada su oba operanda deljenja celi brojevi, rezultat deljenja je količnik, a razlomak je skraćen. Na primer, $5 / 2$ daje 2, a ne 2,5, a $-5 / 2$ daje -2, a ne -2,5. Da biste izvršili deljenje sa pomičnim zarezom, jedan od operanada mora biti broj sa pomičnim zarezom. Na primer, $5,0 / 2$ daje 2,5.

Operator %, poznat kao ostatak, daje ostatak nakon deljenja. Operand sa leve strane je dividenda, a operand sa desne strane je delilac. Dakle, $7 \% 3$ daje 1, $3 \% 7$ daje 3, $12 \% 4$ daje 0, $26 \% 8$ daje 2, a $20 \% 13$ daje 7

Operator % se često koristi za pozitivne cele brojeve, ali se može koristiti i sa negativnim celim brojevima i vrednostima sa pokretnim zarezom. Ostatak je negativan samo ako je dividenda negativna. Na primer, $-7 \% 3$ daje -1, $-12 \% 4$ daje 0, $-26 \% -8$ daje -2, a $20 \% -13$ daje 7.

Diagram illustrating division with remainder. It shows four small division problems: $3 \overline{)7}$, $7 \overline{)3}$, $4 \overline{)12}$, and $8 \overline{)26}$. To the right, a larger example shows $13 \overline{)20}$ with labels: 'Količnik' (Quotient) pointing to 1, 'Deljenik' (Dividend) pointing to 20, and 'Ostatak' (Remainder) pointing to 7. The label 'Delilac' (Divisor) points to 13.

Slika 5.5 Deljenje sa ostakom

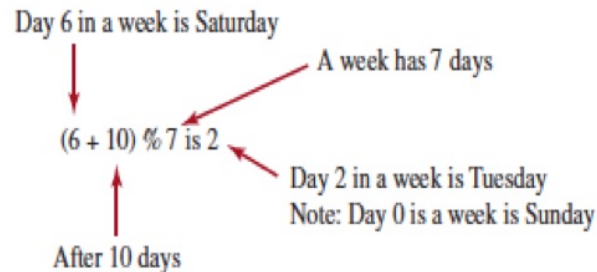
Operator %, poznat kao ostatak, daje ostatak nakon deljenja. Operand sa leve strane je dividenda, a operand sa desne strane je delilac. Dakle, $7 \% 3$ daje 1, $3 \% 7$ daje 3, $12 \% 4$ daje 0, $26 \% 8$ daje 2, a $20 \% 13$ daje 7.

Operator % se često koristi za pozitivne cele brojeve, ali se može koristiti i sa negativnim celim brojevima i vrednostima sa pokretnim zarezom. Ostatak je negativan samo ako je dividenda negativna. Na primer, $-7 \% 3$ daje -1, $-12 \% 4$ daje 0, $-26 \% -8$ daje -2, a $20 \% -13$ daje 7.

OPERATOR DELENJA SA OSTATKOM

Ostatak je veoma koristan u programiranju. Na primer, paran broj % 2 je uvek 0,

Ostatak je veoma koristan u programiranju. Na primer, paran broj % 2 je uvek 0, a pozitivan neparan broj % 2 je uvek 1. Dakle, možete koristiti ovo svojstvo da odredite da li je broj paran ili neparan. Ako je danas subota, biće ponovo subota za 7 dana. Pretpostavimo da ćete se vi i vaši prijatelji sastati za 10 dana. Koji će biti dan za 10 dana? Možete saznati da je dan utorak koristeći sledeći izraz



Slika 5.6 Objašnjenje izraza sa ostatkom

Metod **nextInt()** (red 8) čita ceo broj u sekundama. Linija 10 dobija minute koristeći sekunde / 60. Linija 11 (sekunde % 60) dobija preostale sekunde nakon oduzimanja minuta.

LISTING 2.5 DisplayTime.java

```
1 import java.util.Scanner;
2
3 public class DisplayTime {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6         // Prompt the user for input
7         System.out.print("Enter an integer for seconds: ");
8         int seconds = input.nextInt();
9
10        int minutes = seconds / 60; // Find minutes in seconds
11        int remainingSeconds = seconds % 60; // Seconds remaining
12        System.out.println(seconds + " seconds is " + minutes +
13            " minutes and " + remainingSeconds + " seconds");
14    }
15 }
```

Enter an integer for seconds: 500
500 seconds is 8 minutes and 20 seconds

line#	seconds	minutes	remainingSeconds
8	500		
10		8	
11			20

Slika 5.7 Listingu 2.5 dobija minute i preostale sekunde iz količine vremena u sekundama.

EKSPONENTNE OPERACIJE

Izračunavanje eksponentne vrednosti ab

Metod **Math.pow(a, b)** se može koristiti za izračunavanje a^b . Metod **pow()** je definisana u klasi **Math** u Java API-ju. Metod pozivate koristeći sintaksu **Math.pow(a, b)** (npr. `Math.pow(2, 3)`), koja vraća rezultat a^b (23). Ovde su a i b parametri za **pow()** metod, a brojevi 2 i 3 su stvarne vrednosti koje se koriste za pozivanje metode. Na primer,

```
System.out.println(Math.pow(2, 3)); // Displays 8.0
System.out.println(Math.pow(4, 0.5)); // Displays 2.0
System.out.println(Math.pow(2.5, 2)); // Displays 6.25
System.out.println(Math.pow(2.5, -2)); // Displays 0.16
```

Slika 5.8 Primeri primene metod `pow()` za ekponentne operacije

▼ Poglavlje 6

Literali i izračunavanje izraza

LITERALI

Literal je konstantna vrednost koja se pojavljuje direktno u programu.

Celobrojni Literali (Integer Literals)

Celobrojni literal se može dodeliti celobrojnoj promenljivoj sve dok može da stane u promenljivu. Greška pri kompajliranju će se desiti ako je literal prevelik da bi se promenljiva mogla zadržati. Naredba `bajt b = 128`, na primer, će izazvati grešku pri kompajliranju, jer 128 ne može da se uskladišti u promenljivoj tipa bajta. (Imajte na umu da je opseg za vrednost bajta od -128 do 127.)

Pretpostavlja se da je celobrojni literal tipa int, (Integer Literal) čija je vrednost između -2^{31} (-2147483648) i $2^{31}-1$ (2147483647). Da biste označili celobrojni literal dugog tipa, dodajte mu slovo L ili l. Na primer, da biste napisali ceo broj 2147483648 u Java programu, morate da ga zapišete kao 2147483648L ili 2147483648l, jer 2147483648 premašuje opseg vrednosti int. L je poželjniji jer se l (malo L) lako može pomešati sa 1 (cifrenom).

Primer: 2147483648L

Literali sa pomičnom tačkom (Floating-Point Literals)

Literali sa pomičnom tačkom (Floating-Point Literals na engleskom se koristi tačka umesto zapete) se pišu sa decimalnom tačkom. Podrazumevano, literal sa pomičnom tačkom se tretira kao vrednost dvostrukog tipa. Na primer, 5.0 se smatra dvostrukom vrednošću, a ne float vrednošću. Možete napraviti broj sa plutajućim dodavanjem slova f ili F, a broj možete napraviti duplim dodavanjem slova d ili D. Na primer, možete koristiti 100.2f ili 100.2F za broj sa plutajućim brojem, a 100.2d ili 100.2D za dvostruki broj.

Float vrednost ima 6-9 brojeva značajnih cifara, a double tip ima 15-17 brojeva značajnih cifara.

Naučna notacija

Literali sa pomičnom tačkom se mogu napisati u naučnoj notaciji u obliku $a * 10^b$. Na primer, naučna notacija za 123,456 je $1,23456 * 10^2$, a za 0,0123456 je $1,23456 * 10^{-2}$. Za pisanje naučnih brojeva koristi se posebna sintaksa. Na primer, $1,23456 * 10^2$ je zapisano kao 1,23456E2 ili 1,23456E+2 i $1,23456 * 10^{-2}$ kao 1,23456E-2. E (ili e) predstavlja eksponent i može biti napisan malim ili velikim slovima.

IZRAČUNAVANJE MATEMATIČKIH IZRAZA

Java izrazi se vrednuju na isti način kao i aritmetički izrazi.

Pisanje numeričkog izraza u Javi uključuje direktan prevod aritmetičkog izraza pomoću Java operatora. Na primer, aritmetički izraz

$$\frac{3 + 4x}{5} - \frac{10(y - 5)(a + b + c)}{x} + 9\left(\frac{4}{x} + \frac{9 + x}{y}\right)$$

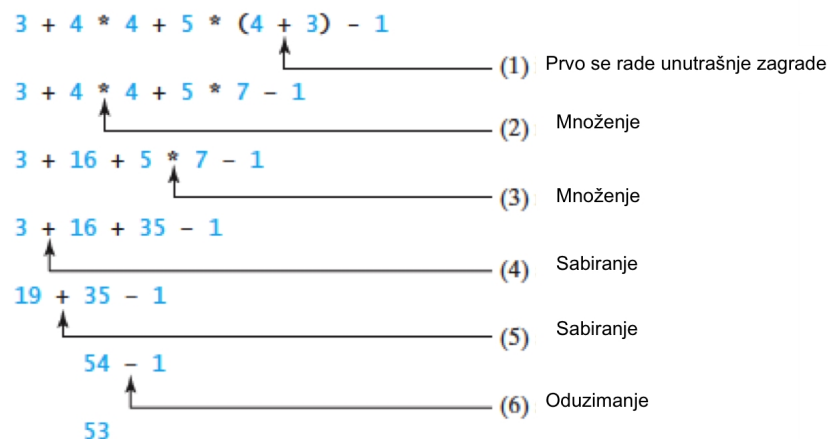
može se prevesti u Java izraz na sledeći način:

$$(3 + 4 * x) / 5 - 10 * (y - 5) * (a + b + c) / x + 9 * (4 / x + (9 + x) / y)$$

Slika 6.1 Prevođenje aritmetičkog izraza u Java izraz

Iako Java ima svoj način da proceni izraz iza scene, rezultat Java izraza i njegovog odgovarajućeg aritmetičkog izraza je isti. Stoga, možete bezbedno primeniti aritmetičko pravilo za procenu Java izraza. Operatori sadržani u parovima zagrada se prvo procenjuju. Zagrade mogu biti ugneždene, u kom slučaju se prvo procenjuje izraz u unutrašnjim zagradama. Kada se u izrazu koristi više od jednog operatora, za određivanje redosleda evaluacije koristi se sledeće pravilo prioriteta operatora:

- Operatori množenja, deljenja i ostatka se prvo primenjuju. Ako izraz sadrži nekoliko operatora množenja, deljenja i ostatka, oni se primenjuju s leva na desno.
 - Operatori sabiranja i oduzimanja se primenjuju poslednji. Ako izraz sadrži nekoliko operatora sabiranja i oduzimanja, oni se primenjuju s leva na desno.
- Evo primera kako se izraz izračunava:



Slika 6.2 Prioriteti u izvršavanju operatora u Java izrazima

PRIMER: KONVERZIJA FAREHAJTOVIH STEPENA U CELYIJUSOVE STEPENE

Program prikazuje broj milisekundi od 1.1.20270. godine (početka upotrebe UNIX-a)

LISTING 2.6 FahrenheitToCelsius.java

```

1  import java.util.Scanner;
2
3  public class FahrenheitToCelsius {
4      public static void main(String[] args) {
5          Scanner input = new Scanner(System.in);
6
7          System.out.print("Enter a degree in Fahrenheit: ");
8          double fahrenheit = input.nextDouble();
9
10         // Convert Fahrenheit to Celsius
11         double celsius = (5.0 / 9) * (fahrenheit - 32);
12         System.out.println("Fahrenheit " + fahrenheit + " is " +
13             celsius + " in Celsius");
14     }
15 }

```

Enter a degree in Fahrenheit: 100
Fahrenheit 100.0 is 37.7777777777778 in Celsius

line#	fahrenheit	celsius
8	100	
11		37.7777777777778

Slika 6.3 Pretvaranje Farenhajtovih stepeni u Celyiusove stepene

Budite pažljivi kada primenjujete podelu. Deljenjem dva cela broja dobija se ceo broj u Javi. 5/9 je kodirano 5.0/9 umesto 5/9 u redu 11, jer 5/9 daje 0 u Javi.

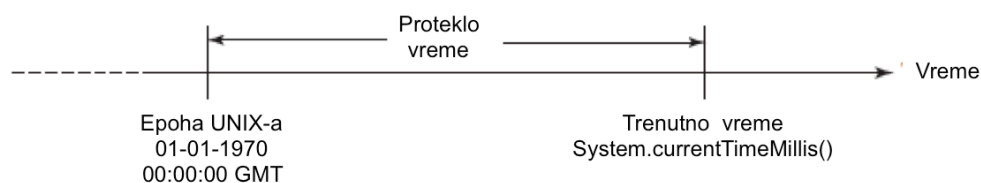
▼ Poglavlje 7

Studija slučaja: Prikaz trenutnog vremena

CURRENTTIMEMILLIS METOD

Možete da pozovete `System.currentTimeMillis()` da vratite trenutno vreme

currentTimeMillis metoda u klasi `System` vraća trenutno vreme u milisekundama proteklo od vremena u ponoć, 1. januara 1970. GMT, kao što je prikazano na slici 1. Ovo vreme je poznato kao UNIX epoha. Epoha je tačka kada vreme počinje, a 1970. je bila godina kada je UNIX operativni sistem formalno uveden.



Slika 7.1 Metod `currentTimeMillis` račun vreme od 1.1.1970 u milisekundama

Možete koristiti ovaj metod da dobijete trenutno vreme, a zatim izračunajte trenutnu sekundu, minut i sat na sledeći način:

1. Dobijte ukupan broj milisekundi od ponoći, 1. januara 1970, u **totalMilliseconds** pozivanjem **System.currentTimeMillis()** (npr. 1203183068328 milisekundi).
2. Dobijte ukupan broj sekundi **totalSeconds** deljenjem **totalMilliseconds** sa 1000 (npr. 1203183068328 milisekundi / 1000 = 1203183068 sekundi).
3. Izračunajte trenutnu sekundu iz **totalSeconds % 60** (npr. 1203183068 sekundi % 60 = 8, što je trenutna sekunda).
4. Dobijte ukupan broj minuta **totalMinutes** tako što ćete **totalSeconds** podeliti sa 60 (npr. 1203183068 sekundi / 60 = 20053051 minuta).
5. Izračunajte trenutni minut iz **totalMinutes % 60** (npr. 20053051 minuta % 60 = 31, što je trenutni minut).
6. Dobijte ukupan broj sati **totalHours** tako što ćete **total Minutes** podeliti sa 60 (npr. 20053051 minuta / 60 = 334217 sati).
7. Izračunajte trenutni sat iz **totalHours % 24** (npr. 334217 sati % 24 = 17, što je trenutni sat)

PROGRAM SHOWCURRENTTIME.JAVA

Ovaj metod određuje trenutno vreme, a zatim izračunava trenutnu sekundu, minut i sat

LISTING 2.7 ShowCurrentTime.java

```

1  public class ShowCurrentTime {
2      public static void main(String[] args) {
3          // Obtain the total milliseconds since midnight, Jan 1, 1970
4          long totalMilliseconds = System.currentTimeMillis();
5
6          // Obtain the total seconds since midnight, Jan 1, 1970
7          long totalSeconds = totalMilliseconds / 1000;
8
9          // Compute the current second in the minute in the hour
10         long currentSecond = totalSeconds % 60;
11
12         // Obtain the total minutes
13         long totalMinutes = totalSeconds / 60;
14
15         // Compute the current minute in the hour
16         long currentMinute = totalMinutes % 60;
17
18         // Obtain the total hours
19         long totalHours = totalMinutes / 60;
20
21         // Compute the current hour
22         long currentHour = totalHours % 24;
23
24         // Display results
25         System.out.println("Current time is " + currentHour + ":"
26             + currentMinute + ":" + currentSecond + " GMT");
27     }
28 }

```

Current time is 17:31:8 GMT

Slika 7.2 Obračun trenutnog vremena

Red 4 poziva `System.currentTimeMillis()` da bi dobio trenutno vreme u milisekundama kao dugu vrednost. Dakle, sve promenljive su deklarisanе kao dugi tip u ovom programu. Sekunde, minute i sati se izdvajaju iz trenutnog vremena pomoću operatora `/` i `%` (linije 6–22).

	line#	4	7	10	13	16	19	22
variables								
totalMilliseconds		1203183068328						
totalSeconds			1203183068					
currentSecond				8				
totalMinutes					20053051			
currentMinute						31		
totalHours							334217	
currentHour								17

Slika 7.3 Prikaz rezultata po linijama programa

U seriji uzorka, za drugu se prikazuje jedna cifra 8. Poželjni izlaz bi bio 08. Ovo se može popraviti korišćenjem metode koja formatira jednu cifru sa prefiksom 0 (pogledajte vežbu programiranja 6.37).

Sat prikazan u ovom programu je GMT. Pokaybi primer 2.8 omogućava prikaz sata u bilo kojoj vremenskoj zoni..

Java takođe obezbeđuje metod **System.nanoTime()** koji vraća proteklo vreme u nanosekundama. **nanotime()** je precizniji i tačniji od **currentTimeMillis()**.

▼ Poglavlje 8

Operatori

OPERATORI PROŠIRENOG DODELJIVANJA

*Operatori +, -, *, / i % mogu se kombinovati sa operatorom dodele da bi se formirali prošireni operatori*

Vrlo često se trenutna vrednost promenljive koristi, modifikuje, a zatim ponovo dodeljuje istoj promenljivoj. Na primer, sledeća izjava povećava promenljivu count za 1:

count = count + 1;

Java vam omogućava da kombinujete operatore dodele i sabiranja pomoću proširenog (ili složenog) operatora dodele. Na primer, prethodna izjava se može napisati kao

count += 1;

+= se naziva operator dodele sabiranja. Tabela 2.4 prikazuje druge proširene operatore dodele

Operator	Ime	Primer	Ekvivalent
+=	Dodela sabiranja	i += 8	i = i + 8
-=	Dodela oduzimanja	i -= 8	i = i - 8
*=	Dodela množenja	i *= 8	i = i * 8
/=	Dodela deljenja	i /= 8	i = i / 8
%=	Dodela ostatka	i %= 8	i = i % 8

Slika 8.1 Operatori proširenog dodeljivanja

Operator proširene dodele se izvodi poslednji nakon što se procene svi ostali operatori u izrazu.

Na primer,

x /= 4 + 5,5 * 1,5;

je isto kao

x = x / (4 + 5,5 * 1,5)

OPERATORI POVEĆANJA I SMANJENJA

Operator povećanja (+ +) i operator smanjenja (- -) služe za povećanje i smanjenje promenljive za 1

Dva skraćena operatora ++ i -- su za povećanje i smanjenje promenljive za 1. Ovo je zgodno jer je to često koliko vrednost treba da se promeni u mnogim programskim zadacima. Na primer, sledeći kod povećava i za 1 i smanjuje j za 1.

```
int i = 3, j = 3;
i++; // i postaje 4
j--; // j postaje 2
```

i++ se izgovara kao „i plus plus“, a i-- kao „i minus minus“. Ovi operatori su poznati kao **postfiksni inkrement** (ili postinkrement) i **postfiksni dekrement** (ili postdekrement), jer se operatori ++ i -- postavljaju iza promenljive. Ovi operatori se takođe mogu postaviti pre promenljive. Na primer,

```
int i = 3, j = 3;
++i; // i postaje 4
--j; // j postaje 2
```

++i povećava i za 1 i —j smanjuje j za 1. Ovi operatori su poznati kao prefiks inkrement (ili preinkrement) i prefiks dekrement (ili predekrement).

Kao što vidite, efekat i++ i ++i ili i-- i --i je isti u prethodnim primerima. Međutim, njihovi efekti su drugačiji kada se koriste u izjavama koje ne čine samo povećanje i smanjenje. Tabela 2.5 opisuje njihove razlike i daje primere.

Operator	Ime	Opis	Primer
++var	preinkrement	Povećava var za 1 i koristi novi var u iskazu	int j = ++i; // j je 2, i je 2
var++	postinkrement	Povećava var za 1 ali koristi originalni var u iskazu	int j = i++; // j je 1, i je 2
--var	predekrement	Smanjuje var za 1 i koristi novi var u iskazu	int j = --i; // j je 0, i je 0
var--	postdekrement	Smanjuje var za 1 ali koristi originalni var u iskazu	int j = i--; // j je 1, i je 0

Slika 8.2 Operatori povećanja i smanjenja

PRIMERI RADA SA OPERATORIMA POVEĆANJA I SMANJENJA

Operandi se izvršavaju s leva na desno u Javi.

Evo dodatnih primera koji ilustruju razlike između prefiksnog oblika ++ (ili - -) i postfiksnog oblika ++ (ili - -). Razmotrite sledeći kod:

```
int i = 10;
int newNum = 10 * i++;
```

Same effect as

```
int newNum = 10 * i;
i = i + 1;
```

```
System.out.print("i is " + i
    + ", newNum is " + newNum);
```

↓ Output is

i is 11, newNum is 100

U ovom slučaju, i se povećava za 1, tada se u množenju koristi stara vrednost i. Dakle, **newNum** postaje 100. Ako se i++ zameni sa ++i, onda postaje sledeće:

```
int i = 10;
int newNum = 10 * (++i);
```

Same effect as

```
i = i + 1;
int newNum = 10 * i;
```

```
System.out.print("i is " + i
    + ", newNum is " + newNum);
```

↓ Output is

i is 11, newNum is 110

i se povećava za 1, a nova vrednost i se koristi u množenju. Dakle, **newNum** postaje 110

Slika 8.3 Ovi primeri ilustruju efekte primene operatora povećanja i smanjenja

Evo još jednog primera:

```
double x = 1.0;
double y = 5.0;
double z = x- - + (++y);
```

Pošto se se sve tri linije izvrše, **i** postaje 6.0, **z** postaje 7.0, a **x** postaje 0.0.

Operandi se izvršavaju s leva na desno u Javi. Levi operand binarnog operatora se izračunava pre nego što se izračunava bilo koji deo desnog operanda. Ovo pravilo ima prednost nad svim drugim pravilima koja regulišu izraze. Evo primera:

```
int i = 1;
int x = ++i + i * 3;
```

++i se procenjuje i vraća 2. Kada se ocenjuje **i * 3**, **i** je sada 2. Prema tome, **x** postaje 8.

▼ Poglavlje 9

Konverzije numeričkog tipa

KONVERZIJA TIPA PODATKA

Konverzija tipa je operacija koja pretvara vrednost jednog tipa podataka u vrednost drugog tipa podataka

Možete li da izvršite binarne operacije sa dva operanda različitog tipa? Da. Ako su ceo broj i broj sa pokretnom decimalnom tačkom uključeni u binarnu operaciju, Java automatski konvertuje ceo broj u vrednost sa pokretnom decimalnom tačkom. Dakle, $3 * 4,5$ je isto što i $3,0 * 4,5$.

Uvek možete dodeliti vrednost numeričkoj promenljivoj čiji tip podržava veći opseg vrednosti; tako, na primer, možete dodeliti **long** vrednost promenljivoj sa **float** brojem. Ne možete, međutim, da dodelite vrednost promenljivoj sa manjim opsegom osim ako ne koristite konverzija tipa (type casting). **Konverzija tipa** je operacija koja pretvara vrednost jednog tipa podataka u vrednost drugog tipa podataka. Konverzija tipa sa malim opsegom na tip sa većim opsegom poznato je kao **proširenje tipa**. Prebacivanje tipa sa velikim opsegom na tip sa manjim opsegom poznato je kao **sužavanje tipa**. Java će automatski proširiti tip, ali morate eksplicitno sužiti tip.

Sintaksa za konverziju tipa je da navedete ciljni tip u zagradama, posle čega sledi ime promenljive ili vrednost koju treba konvertovati. Na primer, sledeća iskaz

```
System.out.println((int)1.7);
```

prikazuje 1. Kada se **double** vrednost ubaci u vrednost **int**, razlomak se skraćuje. Sledeći iskaz

```
System.out.println((double)1 / 2);
```

prikazuje 0,5, jer se 1 prvo konvertuje na 1,0, a zatim se 1,0 deli sa 2. Međutim, iskaz

```
System.out.println(1 / 2);
```

prikazuje 0, jer su i 1 i 2 celi brojevi i rezultujuća vrednost takođe treba da bude ceo broj. Program u Listingu 2.8 prikazuje porez na promet sa dve cifre iza decimalnog zareza.

PROGRAM SALESTAX.JAVA

*Porez se može zaokružiti na dve decimale koristeći (int)(tax * 100 + 0,5) / 100,0.*

LISTING 2.8 SalesTax.java

```

1 import java.util.Scanner;
2
3 public class SalesTax {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6
7         System.out.print("Enter purchase amount: ");
8         double purchaseAmount = input.nextDouble();
9
10        double tax = purchaseAmount * 0.06;
11        System.out.println("Sales tax is $" + (int)(tax * 100) / 100.0);
12    }
13 }

```

Enter purchase amount: 197.55 <input type="button" value="Enter"/> Sales tax is \$11.85			
line#	purchaseAmount	tax	Output
8	197.55		
10		11.853	
11			11.85

Slika 9.1

Korišćenjem unosa u uzorku, promenljiva **purchaseAmount** je 197,55 (red 8). Porez na promet iznosi 6% od kupovine, tako da se porez procenjuje kao 11.853 (red 10). Pogledajte:
porez * 100 je 1185,3
(int)(porez * 100) je 1185
(int)(porez * 100) / 100,0 je 11,85

Tako se na izvodu u redu 11 prikazuje porez 11,85 sa dve cifre iza decimalne tačke. Obratite pažnju da izraz **(int)(tax * 100) / 100,0** zaokružuje porez na dve decimale.

Ako je porez 3,456, **(int)(tax * 100) / 100,0** bi bilo 3,45. Može li se zaokružiti na dve decimale

Imajte na umu da se svaka dvostruka vrednost **x** može zaokružiti na ceo broj koristeći **(int)(x + 0,5)**.

Dakle, porez se može zaokružiti na dve decimale koristeći **(int)(tax * 100 + 0,5) / 100,0**.

▼ Poglavlje 10

Studija slučaja: Brojanje novčanih jedinica

BROJANJE NOVČANIH JEDINICA

Program menja datu količinu novca u manje novčane jedinice

Pretpostavimo da želite da razvijete program koji menja datu količinu novca u manje novčane jedinice. Program dozvoljava korisniku da unese iznos kao dvostruku vrednost koja predstavlja ukupnu vrednost u dolarima i centima, i daje izveštaj u kojem se navodi novčani ekvivalent u maksimalnom broju dolara, četvrtine, dima, nikla i penija, ovim redosledom, kako bi rezultat bio u minimalnom broju novčića.

Evo koraka u razvoju programa:

1. Zatražite od korisnika da unese iznos kao decimalni broj, kao što je 11,56.
2. Pretvorite iznos (npr. 11,56) u cente (1156).
3. Podelite cente sa 100 da biste pronašli broj dolara. Dobijte preostale cente koristeći ostatak od 100 centi
4. Preostale cente podelite sa 25 da biste pronašli broj četvrtina. Dobijte preostale cente koristeći preostali ostatak centi 25.
5. Podelite preostale cente sa 10 da biste pronašli broj novčića. Dobijte preostale cente koristeći preostalih 10 centi.
6. Podelite preostale cente sa 5 da biste pronašli broj novčića. Dobijte preostale cente koristeći preostali ostatak centi 5.
7. Preostali centi su peni.
8. Prikažite rezultat.

Kompletan program je dat u Lisingu 2.10

LISTING PROGRAMA COMPUTINGCHANGE

Na unetu sumu dolara, program računa odgovarajuću stukturu po novčanim jedinicama.

LISTING 2.10 ComputeChange.java

```

1  import java.util.Scanner;
2
3  public class ComputeChange {
4      public static void main(String[] args) {
5          // Create a Scanner
6          Scanner input = new Scanner(System.in);
7
8          // Receive the amount
9          System.out.print(
10             "Enter an amount in double, for example 11.56: ");
11         double amount = input.nextDouble();
12
13         int remainingAmount = (int)(amount * 100);
14
15         // Find the number of one dollars
16         int numberOfOneDollars = remainingAmount / 100;
17         remainingAmount = remainingAmount % 100;
18
19         // Find the number of quarters in the remaining amount
20         int numberOfQuarters = remainingAmount / 25;
21         remainingAmount = remainingAmount % 25;
22
23         // Find the number of dimes in the remaining amount
24         int numberOfDimes = remainingAmount / 10;
25         remainingAmount = remainingAmount % 10;
26
27         // Find the number of nickels in the remaining amount
28         int numberOfNickels = remainingAmount / 5;
29         remainingAmount = remainingAmount % 5;
30
31         // Find the number of pennies in the remaining amount
32         int numberOfPennies = remainingAmount;
33
34         // Display results
35         System.out.println("Your amount " + amount + " consists of");
36         System.out.println(" " + numberOfOneDollars + " dollars");
37         System.out.println(" " + numberOfQuarters + " quarters ");
38         System.out.println(" " + numberOfDimes + " dimes");
39         System.out.println(" " + numberOfNickels + " nickels");
40         System.out.println(" " + numberOfPennies + " pennies");
41     }
42 }

```

Slika 10.1

```

27         // Find the number of nickels in the remaining amount
28         int numberOfNickels = remainingAmount / 5;
29         remainingAmount = remainingAmount % 5;
30
31         // Find the number of pennies in the remaining amount
32         int numberOfPennies = remainingAmount;
33
34         // Display results
35         System.out.println("Your amount " + amount + " consists of");
36         System.out.println(" " + numberOfOneDollars + " dollars");
37         System.out.println(" " + numberOfQuarters + " quarters ");
38         System.out.println(" " + numberOfDimes + " dimes");
39         System.out.println(" " + numberOfNickels + " nickels");
40         System.out.println(" " + numberOfPennies + " pennies");
41     }
42 }

```

Enter an amount in double, for example, 11.56: 11.56

Your amount 11.56 consists of

- 11 dollars
- 2 quarters
- 0 dimes
- 1 nickels
- 1 pennies

Slika 10.2

ANALIZA PROGRAMA

Objašnjava se šta rade određeni delovi programskog koda.

line#	11	13	16	17	20	21	24	25	28	29	32
variables											
amount	11.56										
remainingAmount		1156		56		6		6		1	
numberOfOneDollars			11								
numberOfQuarters					2						
numberOfDimes							0				
numberOfNickels									1		
numberOfPennies											1

Slika 10.3 Dobijene vrednosti u linijama programa

Promenljivi iznos čuva iznos unet sa konzole (red 11). Ova varijabla se ne menja, jer se iznos mora koristiti na kraju programa za prikaz rezultata. Program uvodi promenljivu `remainingAmount` (red 13) za čuvanje promenljivog preostalog iznosa.

Promenljivi iznos je dvostruka decimala koja predstavlja dolare i cente. Konvertuje se u int promenljivu `remainingAmount`, koja predstavlja sve cente. Na primer, ako je iznos 11,56, tada je početni preostali iznos 1156. Operator deljenja daje ceo deo deljenja, tako da je $1156 / 100$ 11. Operator ostatka dobija ostatak deljenja, tako da je $1156 \% 100$ 56.

Program izdvaja maksimalan broj singlova iz preostalog iznosa i dobija novi preostali iznos u promenljivoj `remainingAmount` (redovi 16–17). Zatim izdvaja maksimalan broj četvrtina iz preostalog iznosa i dobija novi preostali iznos (redovi 20–21).

Nastavljajući isti proces, program pronalazi maksimalan broj novčića, nikla i penija u preostalom iznosu. Jedan ozbiljan problem sa ovim primerom je mogući gubitak preciznosti pri bacanju dvostrukog iznosa na int `remainingAmount`.

Ovo može dovesti do netačnog rezultata. Ako pokušate da unesete iznos 10,03, $10,03 * 100$ postaje 1002,9999999999999. Videćete da program prikazuje 10 dolara i 2 penija.

Da biste rešili problem, unesite iznos kao celobrojnu vrednost koja predstavlja cente (pogledajte Vežbu programiranja 2.22).

▼ Poglavlje 11

Uobičajene greške i zamke

UOBIČAJENE GREŠKE 1 I 2

Ukazuje se na česte greške i daju se saveti kako da se izbegnu

Uobičajena greška 1: Nedeklarisane/neinicijalizovane varijable i neiskorišćene promenljive

Promenljiva mora biti deklarisan sa tipom i dodeljena joj vrednost pre upotrebe. Česta greška nije deklarisanje promenljive ili inicijalizacijapromenljive. Razmotrite sledeći kod:

```
double interestRate = 0.05;  
double interest = interestrate * 45;
```

Ovaj kod je pogrešan, jer je kamatnoj stopi dodeljena vrednost 0,05; ali kamata nije deklarisan i inicijalizovana. Java je osetljiva na velika i mala slova, pa smatra da su kamatna stopa i kamatna stopa dve različite varijable.

Ako je promenljiva deklarisan, ali se ne koristi u programu, to može biti potencijalna programska greška. Zbog toga bi trebalo da uklonite neiskorišćenu promenljivu iz svog programa. Na primer, u sledećem kodu, takRate se nikada ne koristi. Treba ga ukloniti iz koda.

```
double interestRate = 0.05;  
double taxRate = 0.05;  
double interest = interestRate * 45;  
System.out.println("Interest is " + interest);
```

Ako koristite IDE kao što su Eclipse i NetBeans, dobićete upozorenje o neiskorišćenim promenljivim.

Uobičajena greška 2: prekoračenje celog broja

Brojevi se čuvaju sa ograničenim brojem cifara. Kada se promenljivoj dodeli vrednost koja je prevelika (po veličini) da bi se sačuvala, to izaziva prelivanje. Na primer, izvršavanje sledeće izjave izaziva prelivanje, jer najveća vrednost koja se može uskladištiti u promenljivoj int tip je 2147483647. 2147483648 će biti prevelik za int vrednost:

```
int vrednost = 2147483647 + 1;  
// vrednost će zapravo biti -2147483648
```

Slično tome, izvršavanje sledeće naredbe takođe izaziva prelivanje, jer je najmanja vrednost koja se može sačuvati u promenljivoj tipa int -2147483648. -2147483649 je preveliko u veličini koja se čuva u int promenljivoj.

```
int value = -2147483648 - 1;  
// value will actually be 2147483647
```

Java ne prijavljuje upozorenja ili greške prilikom koverzije tipa , pa budite pažljivi kada radite sa celim brojevima blizu maksimalnog ili minimalnog opsega datog tipa.

Kada je broj sa pomičnim zarezom suviše mali (tj. preblizu nuli) da bi se uskladištio, to uzrokuje nedovoljan protok. Java ga približava nuli, tako da obično ne morate da brinete o tim mailb vrednostima.

UOBIČJENE GEŠKE 3 I 4

Analiziraju se greške zaokruživanja i nenamerno deljenje celog broja.

Uobičajena greška 3: Greške zaokruživanja

Greška zaokruživanja, koja se takođe naziva greška zaokruživanja, je razlika između izračunate aproksimacije broja i njegove tačne matematičke vrednosti. Na primer, $1/3$ je približno 0,333 ako zadržite tri decimale, a 0,3333333 ako zadržite sedam decimala.

Pošto je broj cifara koji se mogu uskladištiti u promenljivoj ograničen, greške zaokruživanja su neizbežne.

Proračuni koji uključuju brojeve sa pokretnim zarezom su aproksimirani jer se ovi brojevi ne čuvaju sa potpunom tačnošću. Na primer,

`System.out.println(1.0 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1);`

prikazuje 0,50000000000000001, a ne 0,5, i

`System.out.println(1.0 - 0.9);`

prikazuje 0,09999999999999998, a ne 0,1. Celi brojevi se čuvaju precizno. Dakle, proračuni sa celim brojevima daju precizan celobrojni rezultat.

Uobičajena greška 4: Nenamerno deljenje celog broja

Java koristi isti operator deljenja, odnosno `/`, da izvrši i ceobroj i deljenje sa pokretnim zarezom. Kada su dva operanda celi brojevi, operator `/` vrši celobrojno deljenje. Rezultat operacije je ceo broj. Razlomni deo je skraćen.

Da biste naterali dva cela broja da izvrše deljenje sa pokretnom tačkom, pretvorite jedan od celih brojeva u broj sa pokretnom tačkom. Na primer, kod u (a) prikazuje taj prosek kao 1, a kod u (b) prikazuje taj prosek kao 1,5.

```
int number1 = 1;  
int number2 = 2;  
double average = (number1 + number2) / 2;  
System.out.println(average);
```

(a)

```
int number1 = 1;  
int number2 = 2;  
double average = (number1 + number2) / 2.0;  
System.out.println(average);
```

(b)

Slika 11.1 Konverzija dva cela broja u brojeve sa pokretnom decimalnom tačkom.

▼ Poglavlje 12

Video snimci i test za samotestiranje 1 i 2

VIDEO SNIMCI

Opciono vam se budi pet video snimaka koji pokrivaju neke delove ove lekcije

Izdavač našeg udžbenika (Pearson), obezbeđuje na posebnoj URL, video snimke relevantne za svako poglavlje udžbenika. Za poglavlje 2, na sledećem linku : https://media.pearsoncmg.com/ph/esm/ecs_liang_iip_12/cw/#videonotes nude se video snimci za sva poglavlja udžbenika. Za lekciju 2 su relevantni video snimci za Chapter 2 Elementary Programming, koji izgleda ovako :

Chapter 2: Elementary Programming	
Obtain Input	
Use Operators / and %	
Software Development Process	
Compute Loan Payments	
Compute BMI	

Slika 12.1 Ovo je samo slika dela spiska video materijal akoje vam prikayuje dati line izdavača:

Ovo je sam snimak dela preikaya video snimaka na sajtu izdavača. Izbor vršite preko vašeg pretraživača korićenjme, gore datog linka. Možete birati sve, ili pojedine video snimke, ili ni jedan. Mislimo da su vrlo korisni za vaše učenje. Video snimak Software Development Pro možete pogledati, ali mi taj deo udžbenika ovde nismo koristili, jer se taj deo programa izučava na predmetima SE101 i SE201.

Posle preglada odabranih video snimaka (na engleskom), možete preći na sledeću sekciju, koja vam daje link ka testu za samotestiranje. Ovaj test nije obavezan, ali smatramo ga takođe vrlo korisnim za vas, jer vam omogućava proveru vašeg razumevanja izloženog gradiva u ovoj lekciji. Posle odgovora na pitanje, dobijate odmah odgovor da li ste tačno odgovorili, i na dodatni klik, možete dobiti tačan odgovor.

TEST ZA SAMOTESTIRANJE

Test pokriva prvih 11. poglavlja u ovom onlajn materijalu

mDita-LAMS dozvoljava najviše 14 objekata učenja (poglavlja), ne računajući Uvod i Zaključak. Kako imamo 13 poglavlja, tj. objekata učenja u materijalu, možemo da imamo samo još jedno poglavlje za video snimke i za test samotestiranja, te on objedinjava dva testa, koje normalno dodeljujemo svakoj lekciji. Zato on sadrži i veći broj pitanja (ne morate na sva da odgovarate)

Koristimo originalni test koji je autor udžbenika pripremio za svako poglavlje. Ovi testovi (u knjizi ih nazivaju kvizovima) su zgodni u fazi vašeg učenja, jer posle svakog pitanja, kada unesete odgovor, dobijete odmah informaciju da li ste odgovorili tačno ili ne, i na jedan klik, dobijete tačan odgovor. Zato smo odabrali ove testove, jer su pogodni i za učenje, a daju veliki broj pitanja i odgovora.

Ako imate problem sa engleskim jezikom, možete se koristiti i sa Google Translate aplikacijom.

U ovom testu, odgovorite na pitanja od broja 2.1 do 2.34, od 2.37 do 2.62 i od 2.65 do 2.67 (možete preskakati pitanja, ako ih ne želite).

Ovo će vam pomoći da bolje savladate predeno gradivo ove lekcije, jer vam test posle svakog pitanja i vašeg odgovora, daje informaciju da li ste dali tačan odgovor i mogućnost da vam prikaže tačan odgovor.

Da bi aktivirali test, kliknite na sledeći link: <https://liveexample-ppe.pearsoncmg.com/selftest/selftest12e?chapter=2&username=liang12e>

▼ Poglavlje 13

Pokazna vežba

PRIMER 1C

Proračun zapremine cilindra

Ovde ćemo definisati zadatak, i dati i rešenje, uz eventualno kratko objašnjenje. a na vežbama će se opisati i sikovati postupak rešavanja

Zadatak: (Izračunajte zapreminu cilindra) Napišite program koji čita poluprečnik i dužinu cilindra i izračunava površinu i zapreminu koristeći sledeće formule:

$površina = poluprečnik * poluprečnik * \pi$

$zapremina = površina * dužina$

Rešenje:

```
import java.util.Scanner;

public class Exercise02_02 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        // Enter radius of the cylinder
        System.out.print("Enter the radius and length of a cylinder: ");
        double radius = input.nextDouble();
        double length = input.nextDouble();

        // Compute area and volume
        double area = radius * radius * 3.14159;
        double volume = area * length;

        // Display result
        System.out.println("The area is " + area);
        System.out.println("The volume of the cylinder is " + volume);
    }
}
```

PRIMER 2C

Pretvaranje funte u kilograme.

Zadatak:

(Pretvorite funte u kilograme) Napišite program koji pretvara funte u kilograme. Program traži od korisnika da unese broj u funtama, pretvara ga u kilograme i prikazuje rezultat. Jedna funta je 0,454 kilograma. Evo primera rada

Rešenje:

```
public class Exercise02_04 {
    public static void main(String[] args) {
        // Prompt the input
        java.util.Scanner input = new java.util.Scanner(System.in);
        System.out.print("Enter a number in pounds: ");
        double pounds = input.nextDouble();
        double kilograms = pounds * 0.454;

        System.out.println(pounds + " pounds is " + kilograms + " kilograms");
    }
}
```

PRIMER 3A

Sabiranje zbira cifara svih celih brojeva do zadatog nivoa.

Zadatak: 3a: Zbroj cifara u celom broju) Napišite program koji čita ceo broj između 0 i 1000 i sabira sve cifre celog broja. Na primer, ako je ceo broj 932, zbir svih njegovih cifara je 14.

Savet: Koristite operator % da biste izdvojili cifre, a operator / da biste uklonili izvučenu

```
// Exercise02_06.java: Summarize all digits in an integer < 1000
public class Exercise02_06 {
    // Main method
    public static void main(String[] args) {
        java.util.Scanner input = new java.util.Scanner(System.in);
        // Read a number
        System.out.print("Enter an integer between 0 and 1000: ");
        int number = input.nextInt();

        // Find all digits in number
        int lastDigit = number % 10;
        int remainingNumber = number / 10;
        int secondLastDigit = remainingNumber % 10;
        remainingNumber = remainingNumber / 10;
        int thirdLastDigit = remainingNumber % 10;

        // Obtain the sum of all digits
        int sum = lastDigit + secondLastDigit + thirdLastDigit;

        // Display results
        System.out.println("The sum of all digits in " + number
            + " is " + sum);
    }
}
```

PRIMER 4B

Proračun trenutnog vremena

Zadatak: (Trenutno vreme) Listing 2.7, ShowCurrentTime.java, daje program koji prikazuje trenutno vreme u GMT. Revidirajte program tako da traži od korisnika da unese pomak vremenske zone na GMT i prikazuje vreme u navedenoj vremenskoj zoni. Evo primera rada:

```
import java.util.Scanner;

public class Exercise02_08 {
    public static void main(String[] args) {
        // Prompt the user to enter the time zone offset to GMT
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the time zone offset to GMT: ");
        long timeZoneOffset = input.nextInt();

        // Obtain the total milliseconds since the midnight, Jan 1, 1970
        long totalMilliseconds = System.currentTimeMillis();

        // Obtain the total seconds since the midnight, Jan 1, 1970
        long totalSeconds = totalMilliseconds / 1000;

        // Compute the current second in the minute in the hour
        long currentSecond = totalSeconds % 60;

        // Obtain the total minutes
        long totalMinutes = totalSeconds / 60;

        // Compute the current minute in the hour
        long currentMinute = totalMinutes % 60;

        // Obtain the total hours
        long totalHours = totalMinutes / 60;

        // Compute the current hour
        long currentHour = (totalHours + timeZoneOffset) % 24;

        // Display results
        System.out.println("Current time is " + currentHour + ":"
            + currentMinute + ":" + currentSecond);
    }
}
```

PRIMER 5C:

Proračun energije potrebne za zagrevanje vode

Zadatak: (Nauka: izračunavanje energije) Napišite program koji izračunava energiju potrebnu za zagrevanje vode od početne do konačne temperature. Vaš program treba da zatraži od korisnika da unese količinu vode u kilogramima i početnu i konačnu temperaturu vode. Formula za izračunavanje energije je $K = M * (finalTemperature - početnaTemperature) * 4184$ gde je M težina vode u kilogramima, početna i konačna temperatura su u stepenima Celzijusa, a energija K se meri u džulima. Evo primera rada:

```
public class Exercise02_10 {
    public static void main(String[] args) {
        java.util.Scanner input = new java.util.Scanner(System.in);

        System.out.print(
            "Enter the amount of water in kilograms: ");
        double mass = input.nextDouble();

        System.out.print("Enter the initial temperature: ");
        double initialTemperature = input.nextDouble();

        System.out.print(
            "Enter the final temperature: ");
        double finalTemperature = input.nextDouble();

        double energy =
            mass * (finalTemperature - initialTemperature) * 4184;

        System.out.print("The energy needed is " + energy);
    }
}
```

PRIMER 6B

Obračun godišnje kamate i kamate ya sledeći mesec

(Financial application: calculate interest) If you know the balance and the annual percentage interest rate, you can compute the interest on the next monthly payment using the following formula:

$interest = balance * (annualInterestRate/1200)$

Write a program that reads the balance and the annual percentage interest rate and displays the interest for the next month.

```
import java.util.Scanner;

public class Exercise02_20 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        // Obtain input
        System.out.print("Enter balance and annual interest rate: ");
        double balance = input.nextDouble();
        double annualInterestRate = input.nextDouble();
```

```
double monthlyInterestRate = annualInterestRate / 1200;

double interest = balance * monthlyInterestRate;

// Display output
System.out.println("The interest is " + (int)(100* interest) / 100.0);
}
}
```

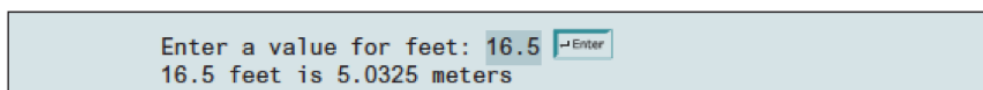
▼ Poglavlje 14

Individualna vežba

ZADACI ZA INDIVIDUALNI RAD 1C, 2C I 3B

Zadaci 1, 2 i 3 se rade samostalno za vreme časova individualnih vežbi

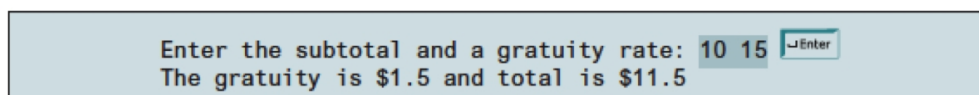
Zadatak 1c: (Pretvorite stope u metre) Napišite program koji čita broj u stopama, pretvara ga u metre i prikazuje rezultat. Jedna stopa je 0,305 metara. Evo primera rada:



```
Enter a value for feet: 16.5 [Enter]
16.5 feet is 5.0325 meters
```

Slika 14.1 Rezultat koji bi trebalo da se dobije za unos od 16.5 stupa

Zadatak 2c: (Finansijska aplikacija: izračunajte napojnice) Napišite program koji čita međuzbir i stopu napojnice, a zatim izračunava napojnicu i zbir. Na primer, ako korisnik unese 10 za međuzbir i 15% za stopu napojnice, program prikazuje 1,5 kao napojnicu i 11,5 kao ukupno. Evo rezultata:

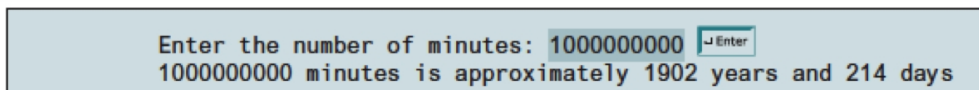


```
Enter the subtotal and a gratuity rate: 10 15 [Enter]
The gratuity is $1.5 and total is $11.5
```

Slika 14.2 Rezultat koji bi trebalo da se dobije za unos od 10 15 (daje 10% od mogućih 15% za napojnicu)

Zadatak 3b: (Pronađite broj godina) Napišite program koji traži od korisnika da unese minute (npr. 1 milijarda) i prikazuje maksimalan broj godina i preostalih dana za minute. Radi jednostavnosti, pretpostavimo da godina ima 365 dana. Evo primera rada:

Rezultat zadatka 3



```
Enter the number of minutes: 1000000000 [Enter]
1000000000 minutes is approximately 1902 years and 214 days
```

Slika 14.3 Očekivani rezultat za unos iste, naznačene vrednosti

ZADACI ZA INDIVIDUALNI RAD 4C I 5B

Zadaci 4c i 5b se rade samostalno za vreme časova individualnih vežbi

Zadatak 4c: (Fizika: ubrzanje) Prosečno ubrzanje se definiše kao promena brzine podeljena vremenom potrebnim da se izvrši promena, kao što je dato sledećom formulom: $a = (v1 - v0)$

/ t

Napišite program koji traži od korisnika da unese početnu brzinu v_0 u metrima/sekundi, krajnju brzinu v_1 u metrima/sekundi i vremenski raspon t u sekundama, a zatim prikazuje prosečno ubrzanje. Evo primera rada

Rezultat zadatka 4

```
Enter v0, v1, and t: 5.5 50.9 4.5
The average acceleration is 10.0889
```

Slika 14.4 Očekivani rezultat za unos iste, naznačene vrednosti

Zadatak 5b: (Geometrija: rastojanje dve tačke) Napišite program koji traži od korisnika da unese dve tačke (k_1, i_1) i (k_2, i_2) i prikazuje njihovu udaljenost. Formula za izračunavanje udaljenosti je:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Slika 14.5 Formula za obračun rastojanja dve tačke

Ovde možete koristiti i Java biblioteku funkciju **Math.pow(a , 0.5)** kod računanja korena od a. Evo primera rada:

ZADACI 6B I 7B

Zadaci 6b i 7c se rade samostalno za vreme časova individualnih vežbi

Zadatak 6b: (Finansijska primena: izračunajte vrednost buduće investicije) Napišite program koji čita iznos investicije, godišnju kamatnu stopu i broj godina i prikazuje buduću vrednost ulaganja koristeći sledeću formulu:

futureInvestmentValue = iznos investicije * (1 + mesečna kamatna stopa)broj godina*12

Na primer, ako unesete iznos 1000, godišnju kamatnu stopu 3,25% i broj godina 1, buduća vrednost ulaganja je 1032,98.

Evo primera rada:

Rezultat zadatka 6

```
Enter investment amount: 1000.56 [Enter]
Enter annual interest rate in percentage: 4.25 [Enter]
Enter number of years: 1 [Enter]
Future value is $1043.92
```

Slika 14.6 Očekivani rezultat za unos iste, naznačene vrednosti

Zadatak 7c: (Cena vožnje) Napišite program koji traži od korisnika da unese razdaljinu koju treba da vozi, efikasnost goriva automobila u miljama po galonu, a cena po galonu zatim prikazuje cenu putovanja. Evo primera rada:

Rezultat zadatka 7

Enter the driving distance: 900.5

Enter miles per gallon: 25.5

Enter price per gallon: 3.55

The cost of driving is \$125.36

Slika 14.7 Očekivani rezultat za unos iste, naznačene vrednosti

✓ Poglavlje 15

Zaključak

ZAKLJUČAK

Rezime lekcije

- 1. Identifikatori su imena za imenovanje elemenata kao što su promenljive, konstante, metode, klase i paketi u programu.
- 2. Identifikator je niz znakova koji se sastoji od slova, cifara, donjih crta (_) i znakova dolara (\$). Identifikator mora da počinje slovom ili donjom crtom. Ne može početi cifrom. Identifikator ne može biti rezervisana reč. Identifikator može biti bilo koje dužine.
- 3. Promenljive se koriste za čuvanje podataka u programu. Deklarisati promenljivu znači reći kompajleru koji tip podataka promenljiva može da sadrži.
- 4. Postoje dve vrste uvoznih izjava: specifičan uvoz i uvoz džoker znakova. Specifični uvoz specificira jednu klasu u naredbi za uvoz. Uvoz džoker znakova uvozi sve klase u paketu.
- 5. U Javi, znak jednakosti (=) se koristi kao operator dodele.
- 6. Promenljivoj deklarisanjoj u metodi mora biti dodeljena vrednost pre nego što se može koristiti.
- 7. Imenovana konstanta (ili jednostavno konstanta) predstavlja trajne podatke koji se nikada ne menjaju.
- 8. Imenovana konstanta se deklarise korišćenjem ključne reči final.
- 9. Java pruža četiri tipa celih brojeva (byte, short, int i long) koji predstavljaju cele brojeve četiri različite veličine.
- 10. Java pruža dva tipa sa pomičnim zarezom (float i double) koji predstavljaju brojeve sa pomičnim zarezom dve različite preciznosti
- 11. Java pruža operatore koji obavljaju numeričke operacije: + (sabiranje), - (oduzimanje), * (množenje), / (deljenje) i % (ostatak).
- 12. Celobrojna aritmetika (/) daje celobrojni rezultat.
- 13. Numerički operatori u Java izrazu se primenjuju na isti način kao u aritmetičkom izrazu.
- 14. Java pruža proširene operatore dodele += (dodeljivanje sabiranja), -= (oduzimanje dodeljivanje), *= (dodeljivanje množenja), /= (dodeljivanje deljenja) i %= (ostatak zadatka).
- 15. Operator povećanja (++) i operator dekrementa (--) povećavaju ili smanjuju promenljivu za 1.
- 16. Kada procenjuje izraz sa vrednostima mešovityh tipova, Java automatski konvertuje operande u odgovarajuće tipove.
- 17. Možete eksplicitno konvertovati vrednost iz jednog tipa u drugi koristeći notaciju (tip) vrednosti.

ZAKLJUČAK (NASTAVAK)

Daje se rezme naučenog u ovoj lekciji

- 18. Prebacivanje promenljive tipa sa malim opsegom u tip sa većim opsegom poznato je kao proširenje tipa.
- 19. Prebacivanje promenljive tipa sa velikim opsegom na tip sa manjim opsegom poznato je kao sužavanje tipa.
- 20. Proširivanje tipa se može izvršiti automatski bez eksplicitnog preliivanja. Sužavanje tipa mora da se izvrši eksplicitno.
- 21. U informatici je ponoć 1. januara 1970. godine poznata kao UNIKS epoha

LITERATURA

Daje se spisak osnovne, i dopunske literature, akao i preporučeni onlajn kursevi

Osnovna literatura:

U pripremi ove onlajn lekcije 2 koja je namenjen studentima Univerziteta Metropolitan ya korišćenje preko sistema za e-učenje Univerziteta Metropolitan, (LAMS), , korišćeno je sledeće poglavlje osnovnog udžbenika koji je preporučen studentima za korišćenje:

1. Y. Daniel Liang, 2019, *Introduction to Java Programming and Data Structures*, **Chapter 2**, Comprehensive Version, 12th edition, Global Edition, Pierson, - preporučeni udžbenik

Dopunska literatura:

1. Deitel, Harvey Deitel, *Java - How To Program*,, 9th Edition, Pearson, ISBN 978-0-13-257566-9, 2012

Preporučeni onlajn Java kursevi:

1. <https://docs.oracle.com/javase/tutorial/>
2. <https://www.ntu.edu.sg/home/ehchua/programming/index.html>
3. <http://www.javatpoint.com/java-tutorial>

