Chapter 7

How to work with primitive types and operators

Objectives

Applied

- Write code that works with any of the eight primitive data types.
- Write code that assigns a new value to a variable.
- Write code that declares and initializes a constant.
- Write code that uses arithmetic operators to perform calculations.
- Write code that uses compound assignment operators.
- Write code that changes the order of arithmetic operations whenever that's necessary.
- Write code that performs casting whenever that's necessary.
- Use the Math and BigDecimal classes to work with numbers.

Objectives (cont.)

Knowledge

- Describe the eight primitive types.
- Distinguish between a variable and a constant.
- Given a list of names, identify the ones that follow the naming recommendations for constants presented in this chapter.
- Explain the difference between a binary operator and a unary operator and give an example of each.
- Explain the difference between prefixing and postfixing an increment or decrement operator.
- List the order of precedence for arithmetic operations.
- Explain what casting is and when it’s performed implicitly.
- List two reasons for using the BigDecimal class.

The eight primitive data types

Type	Bytes	Use
byte	1	Very short integers from -128 to 127.
short	2	Short integers from -32,768 to 32,767.
int	4	Integers from -2,147,483,648 to 2,147,483,647.
long	8	Long integers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.

The eight primitive data types (cont.)

Type	Bytes	Use
float	4	Single-precision, floating-point numbers from -3.4E38 to 3.4E38 with up to 7 significant digits.
double	8	Double-precision, floating-point numbers from -1.7E308 to 1.7E308 with up to 16 significant digits.
char	2	A single Unicode character that’s stored in two bytes.
Boolean	1	A *true* or *false* value.

Technical notes

- To express the value of a floating-point number, you can use scientific notation:

  2.382E+5   // 2.382 * 10^{5}, or 238,200.
  3.25E-8   // 3.25 times 10^{-8}, or .0000000325

- Because of the way floating-point numbers are stored internally, they can’t represent the exact value of the decimal places in some numbers. This can cause a rounding problem.
- By default, Java uses Intel 80-bit extended precision floating-point when it is available from the CPU.

The assignment operator

Operator	Name
=	Assignment

How to declare a variable and assign a value to it in two statements

Syntax

```
type variableName;
variableName = value;
```

Example

```
int counter;                      // declaration statement
counter = 1;                      // assignment statement
```

How to declare a variable and assign a value to it in one statement

Syntax

```
type variableName = value;
```

Examples

```
int counter = 1;               // int variable
double price = 14.95;          // double variable
float interestRate = 8.125F;   // F floating-point value
long numberOfBytes = 20000L;   // L long integer
int population1 = 1734323;     // int variable
int population2 = 1_734_323;   // improve readability J7+
double distance = 3.65e+9;     // scientific notation
char letter1 = 'A';            // 2-digit Unicode
char letter2 = 65;             // integer Unicode
boolean valid = false;         // false is a keyword
int x = 0, y = 0;              // 2 variables/1 statement
```

Naming conventions

- Start variable names with a lowercase letter and capitalize the first letter in all words after the first word.
- Try to use meaningful names that are easy to remember.

How to declare and initialize a constant

Syntax

```
final type CONSTANT_NAME = value;
```

Examples

```
final int DAYS_IN_NOVEMBER = 30;
final float SALES_TAX = .075F;
final double LIGHT_YEAR_MILES = 5.879e+12
```

Naming conventions

- Capitalize all of the letters in constants and separate words with underscores.
- Try to use meaningful names that are easy to remember.

The arithmetic binary operators

Operator	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus

Two integer values

```
int x = 14;
int y = 8;
```

Addition and subtraction

```
int result1 = x + y;        // result1 = 22
int result2 = x - y;        // result2 = 6
```

Multiplication

```
int result3 = x * y;        // result3 = 112
```

Integer division

```
int result4 = x / y;        // result4 = 1
int result5 = x % y;        // result5 = 6
```

Two double values

```
double a = 8.5;
double b = 3.4;
```

Decimal division

```
double result6 = a / b;    // result6 = 2.5
```

The arithmetic unary operators

Operator	Name
++	Increment
--	Decrement
+	Positive sign
-	Negative sign

The increment operator

```
int i = 1;
i++;                    // after execution, i = 2
```

The decrement operator

```
int i = 10;
i--;                    // after execution, i = 9
```

How to postfix an increment operator

```
int x = 14;
int result = x++;  // after execution, x = 15, result = 14
```

How to prefix an increment operator

```
int x = 14;
int result = ++x;  // after execution, x = 15, result = 15
```

How to reverse the value of a number

```
int x = 14;
int result = -x;   // result = -14
```

An arithmetic operation on a character

```
char letter1 = 'C';        // letter1 = 'C'  Unicode integer is 67
char letter2 = ++letter1; // letter2 = 'D'  Unicode integer is 68
```

The compound assignment operators

Operator	Name
`+=`	Addition
`-=`	Subtraction
`*=`	Multiplication
`/=`	Division
`%=`	Modulus

Without compound assignment operators

```
count = count + 1;         // count is increased by 1
count = count - 1;         // count is decreased by 1
total = total + 100.0;     // total is increased by 100.0
total = total - 100.0;     // total is decreased by 100.0
price = price * .8;        // price is multiplied by .8
sum = sum + nextNumber;
        // sum is increased by the value of nextNumber
```

With compound assignment operators

```
count += 1;                // count is increased by 1
count -= 1;                // count is decreased by 1
total += 100.0;            // total is increased by 100.0
total -= 100.0;            // total is decreased by 100.0
price *= .8;               // price is multipled by .8
sum += nextNumber;
        // sum is increased by the value of nextNumber
```

The order of precedence for arithmetic operations

1. Increment and decrement
2. Positive and negative
3. Multiplication, division, and remainder
4. Addition and subtraction

Code that calculates a discounted price

```
double discountPercent = .2;              // 20% discount
double price = 100;                       // $100 price
```

Using the default order of precedence

```
price = price * 1 - discountPercent;    // price = $99.8
```

Using parentheses to specify the order of precedence

```
price = price * (1 – discountPercent);  // price = $80
```

Code that calculates the current value of a monthly investment

```
double currentValue = 5000;          // investment account
double monthlyInvestment = 100;      // added each month
double yearlyInterestRate = .12;     // yrly interest rate
```

Using parentheses to specify order of precedence

```
currentValue = (currentValue + monthlyInvestment) *
               (1 + (yearlyInterestRate/12));
```

Using separate statements to specify order of precedence

```
currentValue += monthlyInvestment;    // add investment
double monthlyInterestRate = yearlyInterestRate / 12;
double monthlyInterest =
    currentValue * monthlyInterestRate;
currentValue += monthlyInterest;      // add interest
```

How implicit casting works

Data types

byte➔short➔int➔long➔float➔double

Examples

```
double grade = 93;             // int to double

double d = 95.0;
int i = 86, j = 91;
double average = (d+i+j)/3;    // i and j to double values
                               // average = 90.666666...
```

How to code an explicit cast

Syntax

```
(type) expression
```

Examples

```
int grade = (int) 93.75;    // double to int (grade = 93)

double d = 95.0;
int i = 86, j = 91;
int average = ((int)d+i+j)/3; // d to int (average = 90)
int remainder = ((int)d+i+j)%3; // d to int (= 2)

double result = (double) i / (double) j; //decimal places
```

How to cast between char and int types

```
char letterChar = 65;  // int to char (letterChar = 'A')
char letterChar2 = (char) 65;    // this works too
int letterInt = 'A';  // char to int (letterInt = 65)
int letterInt2 = (int) 'A';      // this works too
```

How the compound assignment operator can cause an explicit cast

```
int i = 4;
double d = 4.5;
i += d;           // i = 8 (4.5 is cast to the int type)
```

The Math class

```
java.lang.Math
```

Common static methods of the Math class

```
round(number)
pow(number, power)
sqrt(number)
max(a, b)
min(a, b)
random()
```

The round method

```
long result = Math.round(1.667);       // 2
int result = Math.round(1.49F);        // 1
```

The pow method

```
double result = Math.pow(2, 2);        // 4.0 (2*2)
double result = Math.pow(2, 3);        // 8.0 (2*2*2)
double result = Math.pow(5, 2);        // 25.0 (5 squared)
int result = (int) Math.pow(5, 2);     // 25
```

The sqrt method

```
double result = Math.sqrt(20.25);      // 4.5
```

The max and min methods

```
int x = 67;
int y = 23;
int max = Math.max(x, y);             // max is 67
int min = Math.min(x, y);             // min is 23
```

The random method

```
double x = Math.random() * 100;   // >= 0.0 and < 100.0
long result = (long) x;           // from double to long
```

The BigDecimal class

```
java.math.BigDecimal
```

Constructors of the BigDecimal class

```
BigDecimal(int)
BigDecimal(double)
BigDecimal(long)
BigDecimal(String)
```

Methods of the BigDecimal class

```
add(value)
subtract(value)
multiply(value)
divide(value, scale, roundingMode)
setScale(scale, roundingMode)
doubleValue()
toString()
```

The RoundingMode enumeration

```
java.math.RoundingMode
```

Three RoundingMode enumeration values

```
HALF_UP
HALF_DOWN
HALF_EVEN
```

How to round a double value to 2 decimal places

```
discountAmount = new BigDecimal(discountAmount)
    .setScale(2, RoundingMode.HALF_UP).doubleValue();
```

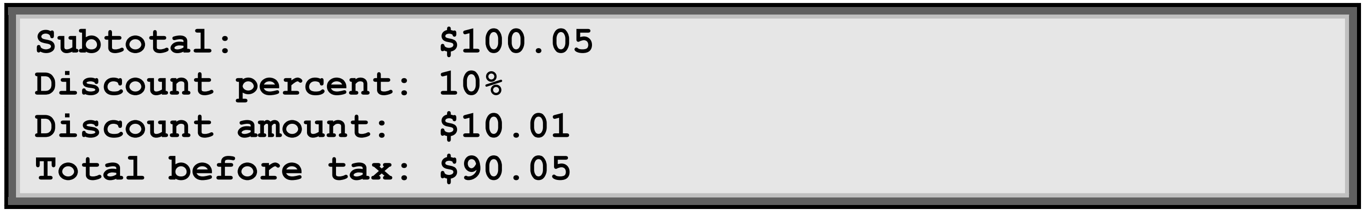
Code that stores monetary values

```
double subtotal = 100.05;                          // 100.050
double discountPercent = .1;
double discountAmount = subtotal * discountPercent; //  10.005
double totalBeforeTax = subtotal - discountAmount;  //  90.045
```

Code that causes a rounding error

```
NumberFormat currency = NumberFormat.getCurrencyInstance();
NumberFormat percent = NumberFormat.getPercentInstance();
String formattedMessage =
    "Subtotal:         " + currency.format(subtotal) + "\n"
  + "Discount percent: " + percent.format(discountPercent) + "\n"
  + "Discount amount:  " + currency.format(discountAmount) + "\n"
  + "Total before tax: " + currency.format(totalBeforeTax) + "\n";
System.out.println(formattedMessage);
```

The console

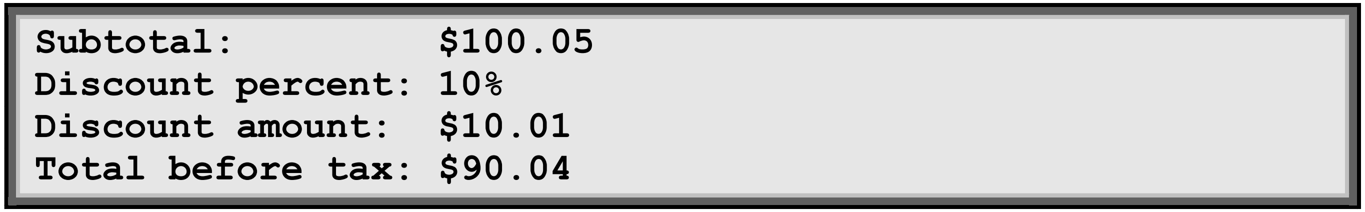


```
Subtotal:         $100.05
Discount percent: 10%
Discount amount:  $10.01
Total before tax: $90.05
```

Code that fixes the rounding error

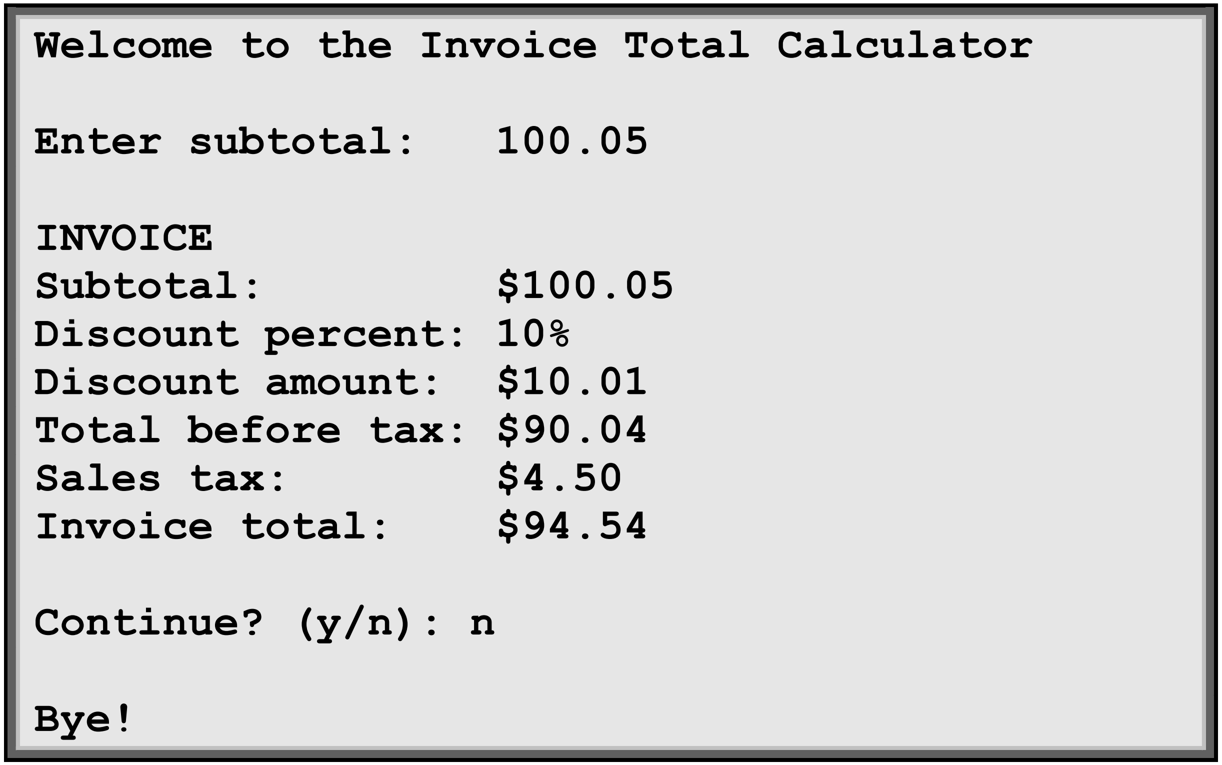
```
BigDecimal subtotal = new BigDecimal("100.05");       // 100.05
BigDecimal discountPercent = new BigDecimal(".1");
BigDecimal discountAmount =
    subtotal.multiply(discountPercent);              // 10.005
discountAmount = discountAmount.setScale(2,
    RoundingMode.HALF_UP);                            // 10.01
BigDecimal totalBeforeTax =
    subtotal.subtract(discountAmount);               // 90.04
```

The console



```
Subtotal:         $100.05
Discount percent: 10%
Discount amount:  $10.01
Total before tax: $90.04
```

The console



```
Welcome to the Invoice Total Calculator

Enter subtotal:   100.05

INVOICE
Subtotal:         $100.05
Discount percent: 10%
Discount amount:  $10.01
Total before tax: $90.04
Sales tax:        $4.50
Invoice total:    $94.54

Continue? (y/n): n

Bye!
```

The code

```
package murach.invoice;

import java.util.Scanner;
import java.text.NumberFormat;
import java.math.BigDecimal;
import java.math.RoundingMode;

public class InvoiceApp {

    public static void main(String[] args) {
        // display a welcome message
        System.out.println(
           "Welcome to the Invoice Total Calculator");
        System.out.println();

        // create a Scanner object named sc
        Scanner sc = new Scanner(System.in);
```

The code (cont.)

```
// perform invoice calculations
// until choice isn't equal to "y" or "Y"
String choice = "y";
while (!choice.equalsIgnoreCase("n")) {
    // get the input from the user
    System.out.print("Enter subtotal:   ");
    String subtotalLine = sc.nextLine();
    double subtotal = new BigDecimal(subtotalLine)
            .setScale(2, RoundingMode.HALF_UP)
            .doubleValue();

    // get discount percent based on subtotal
    double discountPercent;
    if (subtotal >= 200) {
        discountPercent = .2;
    } else if (subtotal >= 100) {
        discountPercent = .1;
    } else {
        discountPercent = 0;
    }
```

The code (cont.)

```
// calculate discount amount
double discountAmount = subtotal * discountPercent;
discountAmount = new BigDecimal(discountAmount)
            .setScale(2, RoundingMode.HALF_UP)
            .doubleValue();

// calculate total before tax
double totalBeforeTax = subtotal - discountAmount;

// calculate sales tax
final double SALES_TAX_PCT = .05;
double salesTax = SALES_TAX_PCT * totalBeforeTax;
salesTax = new BigDecimal(salesTax)
            .setScale(2, RoundingMode.HALF_UP)
            .doubleValue();
```

The code (cont.)

```
// calculate total
double total = totalBeforeTax + salesTax;

// get the currency and percent formatter objects
NumberFormat currency =
    NumberFormat.getCurrencyInstance();
NumberFormat percent =
    NumberFormat.getPercentInstance();
```

The code (cont.)

```
// display the data
String message = "\nINVOICE\n"
  + "Subtotal:         " +
    currency.format(subtotal) + "\n"
  + "Discount percent: " +
    percent.format(discountPercent) + "\n"
  + "Discount amount:  " +
    currency.format(discountAmount) + "\n"
  + "Total before tax: " +
     currency.format(totalBeforeTax) + "\n"
  + "Sales tax:        " +
     currency.format(salesTax) + "\n"
  + "Invoice total:    " +
     currency.format(total) + "\n";

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

The code (cont.)

```
            // see if the user wants to continue
            System.out.print("Continue? (y/n): ");
            choice = sc.nextLine();
            System.out.println();
        }
        System.out.println("Bye!");
    }
}
```