Chapter 2

How to write your first Java applications

Objectives

Applied

1. Given the specifications for an application that requires only the language elements presented in this chapter, write, test, and debug the application.
2. Given the Java code for an application that uses any of the language elements presented in this chapter, explain what each statement in the application does.
3. Given the name of a package and a class, look it up in the documentation for the Java API.

Objectives (cont.)

Knowledge

1. Name two types of comments that are provided by Java and explain how to code them.
2. Given a list of names, identify the ones that are valid for Java classes, and variables.
3. Given a list of names, identify the ones that follow the naming recommendations for classes presented in this chapter.
4. Given a list of names, identify the ones that follow the naming recommendations for variables presented in this chapter.
5. Describe the difference between a main method and other methods.
6. Name three things you can assign to a numeric variable.
7. Distinguish between the int and double data types.

Objectives (cont.)

8. Explain what happens when an arithmetic expression uses both int and double values.
9. Name three things you can assign to a String variable.
10. Explain what an escape sequence is and when you would use one.
11. Explain what *importing a class* means and when you typically do that.
12. Explain what a static method is and how it differs from other methods.
13. Explain what the System.out object can be used for.
14. Explain what a Scanner object can be used for.
15. Explain what a Boolean expression is and when you might use one.
16. Explain how an if/else statement works and what it allows you to do.
17. Explain what it means for a variable to have block scope.

Objectives (cont.)

18. Explain how a while loop works and what it allows you to do.
19. Describe the difference between testing an application and debugging an application.
20. Describe the difference between a compile-time error, a runtime error, and a logical error.

A sample application

```
import java.util.Scanner;

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();  // print a blank line

        // get the input from the user
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter subtotal:   ");
        double subtotal = sc.nextDouble();

        // calculate the discount amount and total
        double discountPercent = .2;
        double discountAmount = subtotal * discountPercent;
        double invoiceTotal = subtotal - discountAmount;
```

A sample application (cont.)

```
        // format and display the result
        String message = "Discount percent: "
           + discountPercent + "\n"
           + "Discount amount:  " + discountAmount + "\n"
           + "Invoice total:    " + invoiceTotal + "\n";
        System.out.println(message);
    }
}
```

A block comment

```
/*
 * Author:  J. Murach
 * Purpose: This program uses the console to get a subtotal
 * from the user, and it calculates the discount amount and
 * total and displays them.
 */
```

Valid identifiers

```
InvoiceApp          $orderTotal      i
Invoice             _orderTotal      x
InvoiceApp2         input_string     TITLE
subtotal            _get_total       MONTHS_PER_YEAR
discountPercent     $_64_Valid
```

The rules for naming an identifier

- Start each identifier with a letter, underscore, or dollar sign. Use letters, dollar signs, underscores, or digits for subsequent characters.
- Use up to 255 characters.
- Don’t use Java keywords.

Keywords

boolean	if	interface	class	true
char	else	package	volatile	false
byte	final	switch	while	throws
float	private	case	return	native
void	protected	break	throw	implements
short	public	default	try	import
double	static	for	catch	synchronized
int	new	continue	finally	const
long	this	do	transient	goto
abstract	super	extends	instanceof	null

The syntax for declaring a class

```
public|private class ClassName {
   statements
}
```

The syntax for declaring a main method

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

A public class that contains a main() method

```
public class InvoiceApp {  // declare and begin the class
   public static void main(String[] args){
      System.out.println(
          "Welcome to the Invoice Total Calculator");
   }
}                          // end the class
```

The same class with different brace placement

```
public class InvoiceApp              // declare the class
{                                    // begin the class
    public static void main(String[] args)
    {
        System.out.println(
            "Welcome to the Invoice Total Calculator");
    }
}                                    // end the class
```

The rules for naming a class

- Start the name with a capital letter.
- Use letters and digits only.
- Follow the other rules for naming an identifier.

Recommendations for naming a class

- Start every word within a class name with an initial cap.
- Each class name should be a noun or a noun that's preceded by one or more adjectives.

Two of the eight primitive data types

```
int
double
```

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

Syntax

```
type variableName = value;
```

Examples

```
int counter = 1;                // declare and initialize
                                // an int variable
double unitPrice = 14.95;       // declare and initialize
                                // a double variable
```

An example that uses assignment statements

```
int quantity = 0;                // declare and initialize
                                 // an int variable
int maxQuantity = 100;           // declare and initialize
                                 // another int variable

// two assignment statements
quantity = 10;                   // quantity is now 10
quantity = maxQuantity;          // quantity is now 100
```

Naming recommendations for variables

- Start variable names with a lowercase letter and capitalize the first letter in all words after the first word.
- Each variable name should be a noun or a noun preceded by one or more adjectives.
- Try to use meaningful names that are easy to remember.

The basic operators for arithmetic expressions

Operator	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division

Statements that use simple arithmetic expressions

```
// integer arithmetic
int x = 14;
int y = 8;
int result1 = x + y;          // result1 = 22
int result2 = x - y;          // result2 = 6
int result3 = x * y;          // result3 = 112
int result4 = x / y;          // result4 = 1

// double arithmetic
double a = 8.5;
double b = 3.4;
double result5 = a + b;       // result5 = 11.9
double result6 = a - b;       // result6 = 5.1
double result7= a * b;        // result7 = 28.9
double result8 = a / b;       // result8 = 2.5
```

Statements that increment a counter variable

```
int invoiceCount = 0;
invoiceCount = invoiceCount + 1;      // invoiceCount = 1
invoiceCount = invoiceCount + 1;      // invoiceCount = 2
```

Statements that add amounts to a total

```
double invoiceAmount1 = 150.25;
double invoiceAmount2 = 100.75;
double invoiceTotal = 0.0;
invoiceTotal = invoiceTotal + invoiceAmount1;
                                  // invoiceTotal = 150.25
invoiceTotal = invoiceTotal + invoiceAmount2;
                                  // invoiceTotal = 251.00
```

Statements that mix int and double variables

```
double result9 = invoiceTotal / invoiceCount;
                                 // result9  = 125.50
int result10 = (int) invoiceTotal / invoiceCount;
                                 // result10 = 125
double result11 = (double) invoiceCount / 4;
                                 // result11 = 0.5
```

The syntax for declaring and initializing a string variable

```
String variableName = value;
```

Statements that declare and initialize a string

```
String message1 = "Invalid data entry.";
String message2 = "";
String message3 = null;
```

How to join strings

```
String firstName = "Bob";           // firstName is Bob
String lastName = "Smith";          // lastName is Smith
String name = firstName + " " + lastName;
                                    // name is Bob Smith
```

How to join a string and a number

```
double price = 14.95;
String priceString = "Price: " + price;
```

How to append one string to another with the + operator

```
firstName = "Bob";      // firstName is Bob
lastName = "Smith";     // lastName is Smith
name = firstName + " ";
                        // name is Bob followed by a space
name = name + lastName;
                        // name is Bob Smith
```

How to append one string to another with the += operator

```
firstName = "Bob";      // firstName is Bob
lastName = "Smith";     // lastName is Smith
name = firstName + " ";
                        // name is Bob followed by a space
name += lastName;       // name is Bob Smith
```

Common escape sequences

```
\n
\t
\r
\"
\\
```

A string with a new line

String

```
"Code: JSP\nPrice: $49.50"
```

Result

```
Code: JSP
Price: $49.50
```

A string with tabs and returns

String

```
"Joe\tSmith\rKate\tLewis"
```

Result

```
Joe       Smith
Kate      Lewis
```

A string with quotation marks

String

```
"Type \"x\" to exit"
```

Result

```
Type "x" to exit
```

A string with a backslash

String

```
"C:\\java\\files"
```

Result

```
C:\java\files
```

Common packages

```
java.lang
java.util
java.text
java.time
java.io
```

How to import a single class from a package

Syntax

```
import packagename.ClassName;
```

Examples

```
import java.util.Scanner;
import java.util.Date;
import java.text.NumberFormat;
```

How to import all classes in a package

Syntax

```
import packagename.*;
```

Examples

```
import java.util.*;
import java.text.*;
```

How to use the Scanner class to create an object

With an import statement

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

Without an import statement

```
java.util.Scanner sc = new java.util.Scanner(System.in);
```

How to create an object from a class

Syntax

```
ClassName objectName = new ClassName(arguments);
```

Examples

```
Scanner sc = new Scanner(System.in);
                        // creates a Scanner object named sc
Date now = new Date();
                        // creates a Date object named now
```

How to call a method from an object

Syntax

```
objectName.methodName(arguments)
```

Examples

```
double subtotal = sc.nextDouble();
                     // get a double entry from the console
String currentDate = now.toString();
                     // convert the date to a string
```

How to call a static method from a class

Syntax

```
ClassName.methodName(arguments)
```

Examples

```
String priceString = Double.toString(price);
                                // convert a double to a string
double total = Double.parseDouble(inputStr);
                                // convert a string to a double
```

The documentation for the Scanner class

Scanner (Java Platform SE ...

https://docs.oracle.com/javase/8/docs/api/

java.time.format
java.time.temporal
java.time.zone
java.util
java.util.concurrent
java.util.concurrent.atomi
java.util.concurrent.locks
java.util.function
java.util.jar

PropertyPermission
PropertyResourceBundle
Random
ResourceBundle
ResourceBundle.Control
Scanner
ServiceLoader
SimpleTimeZone
Spliterators
Spliterators.AbstractDoub
Spliterators.AbstractIntSp
Spliterators.AbstractLong
Spliterators.AbstractSplite
SplittableRandom
Stack
StringJoiner
StringTokenizer
Timer
TimerTask
TimeZone
TreeMap
TreeSet
UUID

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

Java™ Platform Standard Ed. 8

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

compact1, compact2, compact3

java.util

Class Scanner

java.lang.Object
    java.util.Scanner

All Implemented Interfaces:

Closeable, AutoCloseable, Iterator<String>

```
public final class Scanner
extends Object
implements Iterator<String>, Closeable
```

A simple text scanner which can parse primitive types and strings using regular expressions.

A Scanner breaks its input into tokens using a delimiter pattern, which by default matches whitespace. The resulting tokens may then be converted into values of different types using the various next methods.

For example, this code allows a user to read a number from System.in:

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
```

Two methods of the System.out object

```
println(data)
print(data)
```

Code that uses the println() method

```
System.out.println(
    "Welcome to the Invoice Total Calculator");
System.out.println("Total: " + total);
System.out.println(message);
System.out.println();          // print a blank line
```

Code that use the print() method

```
System.out.print("Total: ");
System.out.print(total);
System.out.print("\n");
```

An application that prints data to the console

```
public class InvoiceApp {

    public static void main(String[] args) {

        // set and calculate the numeric values
        double subtotal = 100;
                                // set subtotal to 100
        double discountPercent = .2;
                                // set discountPercent to 20%
        double discountAmount =
            subtotal * discountPercent;
        double invoiceTotal = subtotal - discountAmount;

        // print the data to the console
        System.out.println(
            "Welcome to the Invoice Total Calculator");
        System.out.println();
        System.out.println(
            "Subtotal:         " + subtotal);
```

An application that prints data to the console (cont.)

```
        System.out.println(
            "Discount percent: " + discountPercent);
        System.out.println(
            "Discount amount:  " + discountAmount);
        System.out.println(
            "Total:            " + invoiceTotal);
        System.out.println();
    }
}
```

The console

```
Welcome to the Invoice Total Calculator

Subtotal:          100.0
Discount percent: 0.2
Discount amount:  20.0
Total:             80.0
```

The Scanner class

```
java.util.Scanner
```

How to create a Scanner object

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

Common methods of a Scanner object

```
next()
nextInt()
nextDouble()
nextLine()
```

How to use the methods of a Scanner object

```
String name = sc.next();
int count = sc.nextInt();
double subtotal = sc.nextDouble();
String cityName = sc.nextLine();
```

Note

- The Scanner class was introduced in version 1.5 of the JDK.

Code that gets three values from the user

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

// read a string
System.out.print("Enter product code: ");
String productCode = sc.next();

// read a double value
System.out.print("Enter price: ");
double price = sc.nextDouble();

// read an int value
System.out.print("Enter quantity: ");
int quantity = sc.nextInt();

// perform a calculation and display the result
double total = price * quantity;
System.out.println();
System.out.println(quantity + " " + productCode +
                   " @ " +  price + " = " + total);
System.out.println();
```

The console after the program finishes

```
Enter product code: cshp
Enter price: 49.50
Enter quantity: 2

2 cshp @ 49.5 = 99.0
```

Code that reads three values from one line

```
// read three int values
System.out.print("Enter three integer values: ");
int i1 = sc.nextInt();
int i2 = sc.nextInt();
int i3 = sc.nextInt();

// calculate the average and display the result
int total = i1 + i2 + i3;
int avg = total / 3;
System.out.println("Average: " + avg);
System.out.println();
```

The console after the program finishes

```
Enter three integer values: 99 88 92
Average: 93
```

Relational operators

Operator	Name
`==`	Equality
`!=`	Inequality
`>`	Greater Than
`<`	Less Than
`>=`	Greater Than Or Equal
`<=`	Less Than Or Equal

Examples of Boolean expressions

```
count == 5                  // equal to a numeric literal
testScore != 0              // not equal to a numeric literal
years > 0                   // greater than a numeric literal
i < months                  // less than a numeric variable
subtotal >= 9.99
            // greater than or equal to a numeric literal
quantity <= reorderPoint
            // less than or equal to a numeric variable
```

Two methods of the String class

```
equals(String)
equalsIgnoreCase(String)
```

Examples

```
userEntry.equals("Y") // equal to a string literal
userEntry.equalsIgnoreCase("Y")
                      // equal to a string literal
!lastName.equals("Jones")
                      // not equal to a string literal
code.equalsIgnoreCase(productCode)
                      // equal to another string variable
```

The syntax of the if/else statement

```
if (booleanExpression) { statements }
[else if (booleanExpression) { statements }] ...
[else { statements }]
```

If statements without else if or else clauses

With a single statement

```
double discountPercent = .1;
if (subtotal >= 100)
    discountPercent = .2;
```

With a block of statements

```
double discountPercent = .1;
if (subtotal >= 100) {
    discountPercent = .2;
    status = "Bulk rate";
}
```

An if statement with an else clause

```
double discountPercent = .1;
if (subtotal >= 100) {
    discountPercent = .2;
} else {
    discountPercent = .1;
}
```

An if statement with else if and else clauses

```
double discountPercent = .1;
if (customerType.equals("T")) {
    discountPercent = .4;
} else if (customerType.equals("C")) {
    discountPercent = .2;
} else if (subtotal >= 100) {
    discountPercent = .2;
} else {
    discountPercent = .1;
}
```

The syntax of the while loop

```
while (booleanExpression) {
   statements
}
```

A loop that continues while choice is “y” or “Y”

```
Scanner sc = new Scanner(System.in);
String choice = "y";
while (choice.equalsIgnoreCase("y")) {
    // get the invoice subtotal from the user
    System.out.print("Enter subtotal:   ");
    double subtotal = sc.nextDouble();

    // print the user input to the console
    System.out.println("You entered: " + subtotal);

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

A loop that calculates the sum of the numbers 1 through 4

```
int i = 1;
int sum = 0;
while (i < 5) {
    sum = sum + i;
    i = i + 1;
}
```

The console for the Invoice application

```
Welcome to the Invoice Total Calculator

Enter subtotal:   150
Discount percent: 0.1
Discount amount:  15.0
Invoice total:    135.0

Continue? (y/n):
```

The code for the Invoice application

```
import java.util.Scanner;

public class InvoiceApp  {

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

        Scanner sc = new Scanner(System.in);

        // perform invoice calculations until choice
        // isn't equal to "y" or "Y"
        String choice = "y";
        while (choice.equalsIgnoreCase("y")) {
            // get the invoice subtotal from the user
            System.out.print("Enter subtotal:   ");
            double subtotal = sc.nextDouble();
```

The code for the Invoice application (cont.)

```
// calculate the discount amount and total
double discountPercent = 0.0;
if (subtotal >= 200) {
    discountPercent = .2;
} else if (subtotal >= 100) {
    discountPercent = .1;
} else {
    discountPercent = 0.0;
}
double discountAmount =
    subtotal * discountPercent;
double total = subtotal - discountAmount;

// display the results
String message = "Discount percent: "
               + discountPercent + "\n"
               + "Discount amount:  "
               + discountAmount + "\n"
               + "Invoice total:    "
               + total + "\n";
System.out.println(message);
```

The code for the Invoice application (cont.)

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

The console for the Test Score application

```
Enter test scores that range from 0 to 100.
To end the program, enter 999.

Enter score: 90
Enter score: 80
Enter score: 75
Enter score: 999

Score count:   3
Score total:   245
Average score: 81.66666666666667
```

The code for the Test Score application

```
import java.util.Scanner;

public class TestScoreApp {

    public static void main(String[] args) {

        System.out.println(
            "Enter test scores that range from 0 to 100.");
        System.out.println(
            "To end the program, enter 999.");
        System.out.println();  // print a blank line

        // initialize variables and create a Scanner object
        int scoreTotal = 0;
        int scoreCount = 0;
        int testScore = 0;
        Scanner sc = new Scanner(System.in);
```

The code for the Test Score application (cont.)

```
// get a series of test scores from the user
while (testScore <= 100) {
    // get the input from the user
    System.out.print("Enter score: ");
    testScore = sc.nextInt();

    // accumulate score count and score total
    if (testScore <= 100) {
        scoreCount = scoreCount + 1;
        scoreTotal = scoreTotal + testScore;
    }
}
```

The code for the Test Score application (cont.)

```
        // display the score count, score total,
        // and average score
        double averageScore =
            (double) scoreTotal / scoreCount;
        String message = "\n"
                + "Score count:   " + scoreCount + "\n"
                + "Score total:   " + scoreTotal + "\n"
                + "Average score: " + averageScore + "\n";
        System.out.println(message);
    }
}
```

A runtime error that occurred while testing the Invoice application

```
Output - ch02_Invoice (run)
Welcome to the Invoice Total Calculator

Enter subtotal:   $100
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Scanner.java:864)
        at java.util.Scanner.next(Scanner.java:1485)
        at java.util.Scanner.nextDouble(Scanner.java:2413)
        at InvoiceApp.main(InvoiceApp.java:18)
C:\murach\java\netbeans\book_apps\ch02_Invoice\nbproject\build-impl.xml:1026: The following error
C:\murach\java\netbeans\book_apps\ch02_Invoice\nbproject\build-impl.xml:791: Java returned: 1
BUILD FAILED (total time: 6 seconds)
```

Incorrect output produced by the Test Score application

Output - ch02_TestScore (run)

```
compile:
run:
Enter test scores that range from 0 to 100.
To end the program, enter 999.


Score count:   0
Score total:   0
Average score: NaN

BUILD SUCCESSFUL (total time: 0 seconds)
```

Debugging tips

- For a runtime error, go to the line in the source code that was running when the program crashed. In most IDEs, you can do that by clicking on the link to the line of source code. That should give you a strong indication of what caused the error.
- For a logical error, first figure out how the source code produced that output. Then, fix the code and test the application again.