Murach's Java Programming (6th Edition)

Chapter 2

How to write your first applications

Objectives (part 1)

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.

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.

Objectives (part 2)

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 the 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.
8. Explain how you can use type inference when initializing a variable.
9. Explain what happens when an arithmetic expression uses both int and double values.
10. Name three things you can assign to a String variable.
11. Explain what an escape sequence is and when you would use one.
12. Explain what *importing a class* means and when you typically do that.

Objectives (part 3)

13. Explain what a static method is and how it differs from other methods.
14. Explain what the System.out object can be used for.
15. Explain what a Scanner object can be used for.
16. Explain what a Boolean expression is and when you might use one.
17. Explain how an if/else statement works and what it allows you to do.
18. Explain what it means for a variable to have block scope.
19. Explain how a while loop works and what it allows you to do.
20. Describe the difference between testing an application and debugging an application.
21. Describe the difference between a compile-time error, a runtime error, and a logical error.

The syntax for declaring a class that contains a main() method

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

A class named TestApp with a main() method

```
public class TestApp {
    public static void main(String[] args) {
        System.out.println("Hi!");
    }
}
```

The same class with different brace placement

```
public class TestApp
{
    public static void main(String[] args)
    {
        System.out.println("Hi!");
    }
}
```

The console after running the TestApp class

```
Hi!
```

Rules and recommendations for naming a class

- Start every word within the name with a capital letter.
- Use letters and digits only.
- The class name must match the name of the .java file for the class.

A Java application with statements and comments

```
/*
 * Author:  Joel Murach
 * Purpose: This application uses the console to get a
 * subtotal from the user. Then, it calculates and displays
 * the discount amount and total.
 */
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:   ");
        String input = sc.nextLine();
        double subtotal = Double.parseDouble(input);
```

A Java application with statements and comments (continued)

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

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

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
```

How to use type inference when initializing a variable (Java 10 and later)

Syntax

```
var variableName = value;
```

Examples

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

How to assign a new value to a variable

```
int quantity = 0;               // declare and initialize
                                // an int variable
quantity = 10;                  // assign a new value –
                                // quantity is now 10
```

Rules for naming variables

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

Java 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
assert	enum	exports	module	requires
strictfp	var			

Recommendations for naming variables

- Start names with a lowercase letter.
- Use camel case notation.
- Use meaningful names that are easy to remember.

Names that follow these recommendations

```
counter
quantity
lineItem
invoiceTotal
firstName
lastName
productList
```

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;
var message4 = "Let the compiler infer the data type";
```

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 priceStr = "Price: " + price;
                    // priceStr is Price: 14.95
```

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
```

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

```
"Joe Smith\nKate Lewis"
```

Result

```
Joe Smith
Kate Lewis
```

A string with tabs and returns

String

```
"Joe\tSmith\nKate\tLewis"
```

Result

```
Joe       Smith
Kate      Lewis
```

A string with quotation marks

String

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

Result

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

A string with 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

```
new ClassName(arguments)
```

Examples

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

How to call a method from an object

Syntax

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

Examples

```
String inputStr = sc.nextLine();
                                // call from Scanner object
String currentDate = now.toString();
                                // call from Date object
```

How to call a static method from a class

Syntax

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

Examples

```
String priceStr = Double.toString(9.99);
                                  // call from Double class
double total = Double.parseDouble("27.97");
                                  // call from Double class
```

The println() and print() 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
```

Codes that uses 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 (continued)

```
        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);
```

A method that’s available from a Scanner object

```
nextLine()
```

Static methods of the Integer and Double classes

- `parseInt(String)`
- `parseDouble(String)`

How to use a Scanner object to get data from a user

How to get a string

```
String city = sc.nextLine();  // returns a String object
```

How to get an integer

```
String input = sc.nextLine();
int count = Integer.parseInt(input);
```

How to get a double

```
String input = sc.nextLine();
double subtotal = Double.parseDouble(input);
```

Relational operators

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

Boolean expressions that compare numbers

```
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 a String object

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

Boolean expressions that compare strings

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

An if statement with else if and else clauses

```
double discountPercent;
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")) {
    System.out.print("Continue? (y/n): ");
    choice = sc.nextLine();
}
```

The console after the loop runs

```
Continue? (y/n): y
Continue? (y/n): y
Continue? (y/n): n
```

A loop that displays the numbers 1 through 4

```
int i = 1;
while (i < 5) {
    System.out.print(i + " ");
    i = i + 1;
}
System.out.println("Bye!");
```

The console after the code runs

```
1 2 3 4 Bye!
```

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;
}
System.out.println(sum);   // displays 10
```

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 (part 1)

```
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);

        String choice = "y";
        while (choice.equalsIgnoreCase("y")) {
            // get the invoice subtotal from the user
            System.out.print("Enter subtotal:   ");
            String input = sc.nextLine();
            double subtotal = Double.parseDouble(input);
```

The code for the Invoice application (part 2)

```
 // 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 (part 3)

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

The console for the Test Score application

```
Enter test scores that range from 0 to 100.
To exit, 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 (part 1)

```
import java.util.Scanner;

public class TestScoreApp {

    public static void main(String[] args) {
        // display operational messages
        System.out.println(
            "Enter test scores that range from 0 to 100.");
        System.out.println("To exit, 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 (part 2)

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

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

The code for the Test Score application (part 3)

```
        // 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

```
Welcome to the Invoice Total Calculator

Enter subtotal:   $100
Exception in thread "main" java.lang.NumberFormatException:
For input string: "$100"
    at java.base/jdk.internal.math.FloatingDecimal.
        readJavaFormatString(FloatingDecimal.java:2054)
    at java.base/jdk.internal.math.FloatingDecimal.
        parseDouble(FloatingDecimal.java:110)
    at java.base/java.lang.Double.parseDouble(Double.java:556)
    at murach.InvoiceApp.main(InvoiceApp.java:19)
```

Incorrect output produced by the Test Score application

```
Enter test scores that range from 0 to 100.
To exit, enter 999.

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

Score count:   0
Score total:   170
Average score: Infinity
```

Debugging tips

- For a runtime error, go to the line in the source code that's identified in the error message. In most IDEs, you can click on the link in the error message to go to the line of source code. This should give you an idea of what might be causing the error.
- For a logic error, first determine how the source code produced that output. This should give you an idea of what might be causing the error.
- Once you determine the cause of an error, you can usually fix the error.

Search results for the Scanner class