Chapter 4

How to code control statements

Objectives

Applied

1. Code if/else statements and switch statements to control the logic of an application.
2. Code while, do-while, and for loops to control the repetitive processing that an application requires.
3. Code nested for loops whenever they are required.
4. Use the break and continue statements to jump out of a loop or to jump to the start of a loop.
5. Code a static method that performs a given operation, and code a statement that calls that method.
6. 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.

Objectives (cont.)

Knowledge

1. Explain how a reference type like String is different from a primitive data type.
2. Explain what a logical operator is and why you would use one.
3. Compare the if/else and switch statements.
4. Explain what it means for execution to fall through a label in a switch statement.
5. Describe the differences between while, do-while, and for loops.
6. Explain what an infinite loop is.
7. Describe the difference between the break statement and the continue statement.

Relational operators

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

Boolean expressions

```
discountPercent == 2.3  // equal to a numeric literal
letter == 'y'           // equal to a char literal
isValid == true         // equal to a true value

subtotal != 0           // not equal to a numeric literal

years > 0               // greater than a numeric literal
i < months              // less than a variable

subtotal >= 500           // greater than or equal to a
                          // numeric literal
quantity <= reorderPoint  // less than or equal to a
                          // variable

isValid                 // isValid is equal to true
!isValid                // isValid is equal to false
```

Logical operators

Operator	Name
&&	And
||	Or
!	Not

Boolean expressions with logical operators

```
subtotal > 250 && subtotal < 500    // short-circuit AND
quantity <= 4 || quantity >= 12     // short-circuit OR

(subtotal > 250 && subtotal < 500) || isValid
                                   // 2 logical operators

!(counter++ >= years)              // NOT
```

The syntax of the if/else statement

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

An if statement with only an if clause

```
double discountPercent = .05;
if (subtotal >= 100) {
    discountPercent = .1;
}
```

An if statement with an else clause

```
double discountPercent;
if (subtotal >= 100) {
    discountPercent = .1;
} else {
    discountPercent = .05;
}
```

An if statement with multiple else if clauses

```
double discountPercent;
if (subtotal >= 100 && subtotal < 200) {
    discountPercent = .1;
} else if (subtotal >= 200 && subtotal < 300) {
    discountPercent = .2;
} else if (subtotal >= 300) {
    discountPercent = .3;
} else {
    discountPercent = .05;
}
```

An if statement with clauses that contain multiple statements

```
double discountPercent;
String shippingMethod = "";
if (subtotal >= 100) {
    discountPercent = .1;
    shippingMethod = "UPS";
} else {
    discountPercent = .05;
    shippingMethod = "USPS";
}
```

An if statement without braces

```
double discountPercent;
if (subtotal >= 100)
    discountPercent = .1;
else
    discountPercent = .05;
```

Another way to code an if statement without braces

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

Nested if statements

```
double discountPercent;
if (customerType.equals("r")) {
    if (subtotal >= 100) {          // begin nested if
        discountPercent = .2;
    } else {
        discountPercent =.1;
    }                               // end nested if
} else {
    discountPercent = .4;
}
```

The syntax of the switch statement

```
switch (switchExpression)  {
    case label1:
        statements
        break;
    [case label2:
        statements
        break;] ...
    [default:
        statements
        break;]
}
```

A switch statement that uses an integer

```
switch (productID) {
    case 1:
        productDescription = "Hammer";
        break;
    case 2:
        productDescription = "Box of Nails";
        break;
    default:
        productDescription = "Product not found";
        break;
}
```

A switch statement that uses a string

```
switch (productCode) {
    case "hm01":
        productDescription = "Hammer";
        break;
    case "bn03":
        productDescription = "Box of Nails";
        break;
    default:
        productDescription = "Product not found";
        break;
}
```

A switch statement that falls through case labels

```
switch (dayOfWeek)  {
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
        day = "weekday";
        break;
    case 6:
    case 7:
        day = "weekend";
        break;
}
```

The console for the enhanced Invoice application

```
The Invoice Total Calculator

Enter customer type (r/c): r
Enter subtotal:   100

INVOICE
Subtotal:         $100.00
Discount percent: 10%
Discount amount:  $10.00
Total before tax: $90.00
Sales tax:        $4.50
Invoice total:    $94.50

Continue? (y/n): n

Bye!
```

The switch statement and nested if/else statements

```
double discountPercent = 0;
switch(customerType) {
    case "r":
    case "R":
        if (subtotal < 100) {
            discountPercent = 0.0;
        } else if (subtotal >= 100 && subtotal < 250) {
            discountPercent = .1;
        } else if (subtotal >= 250) {
            discountPercent = .2;
        }
        break;
    case "c":
    case "C":
        if (subtotal < 250) {
            discountPercent = .2;
        } else if (subtotal >= 250) {
            discountPercent = .3;
        }
        break;
```

The switch statement and nested if/else statements (cont.)

```
        default:
            discountPercent = .1;
            break;
    }
```

The syntax of the while loop

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

A while loop that calculates a future value

```
int i = 1;
int months = 36;
while (i <= months) {
    futureValue = (futureValue + monthlyInvestment) *
                  (1 + monthlyInterestRate);
    i++;
}
```

The syntax of the do-while loop

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

A do-while loop that calculates a future value

```
int i = 1;
int months = 36;
do {
    futureValue = (futureValue + monthlyInvestment) *
                  (1 + monthlyInterestRate);
    i++;
} while (i <= months);
```

The syntax of the for loop

```
for (initializationExpression; booleanExpression;
        incrementExpression) {
    statements
}
```

A for loop that stores the numbers 0 through 19 in a string

```
String numbers = "";
for (int i = 0; i < 20; i++)  {
    numbers += i + " ";
}
```

The console after the string is printed to it

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
```

A for loop that adds the numbers 8, 6, 4, and 2

```
int sum = 0;
for (int i = 8; i > 0; i -= 2) {
    sum += i;
}
```

A for loop that calculates a future value

```
int months = 36;
for (int i = 1; i <= months; i++) {
    futureValue = (futureValue + monthlyInvestment) *
                  (1 + monthlyInterestRate);
}
```

The console for the Future Value application

```
The Future Value Calculator

Enter monthly investment:   100
Enter yearly interest rate: 3
Enter number of years:      3
Future value:               $3,771.46

Continue? (y/n):
```

The code for the Future Value application

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

public class FutureValueApp {

    public static void main(String[] args) {
        System.out.println(
            "The Future Value Calculator\n");
        Scanner sc = new Scanner(System.in);
        String choice = "y";
        while (choice.equalsIgnoreCase("y")) {
            // get the input from the user
            System.out.print(
                "Enter monthly investment:   ");
            double monthlyInvestment = sc.nextDouble();
            System.out.print(
                "Enter yearly interest rate: ");
            double interestRate = sc.nextDouble();
            System.out.print(
                "Enter number of years:      ");
            int years = sc.nextInt();
```

The code for the Future Value application (cont.)

```
// convert yearly values to monthly values
double monthlyInterestRate =
    interestRate / 12 / 100;
int months = years * 12;

// use a for loop to calculate the future value
double futureValue = 0.0;
for (int i = 1; i <= months; i++) {
    futureValue =
        (futureValue + monthlyInvestment) *
        (1 + monthlyInterestRate);
}
```

The code for the Future Value application (cont.)

```
            // format the result and display it to the user
            NumberFormat currency =
                NumberFormat.getCurrencyInstance();
            System.out.println(
                "Future value:                " +
                currency.format(futureValue));
            System.out.println();

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

The console for an application with nested loops

```
Monthly investment: $100.00

Year        5.0%          5.5%          6.0%          6.5%
1     $1,233.00     $1,236.36     $1,239.72     $1,243.10
2     $2,529.09     $2,542.46     $2,555.91     $2,569.45
3     $3,891.48     $3,922.23     $3,953.28     $3,984.64
4     $5,323.58     $5,379.83     $5,436.83     $5,494.59
5     $6,828.94     $6,919.65     $7,011.89     $7,105.68
6     $8,411.33     $8,546.33     $8,684.09     $8,824.66
```

Nested loops that print a table of future values

```
// get the currency and percent formatters
NumberFormat currency = NumberFormat.getCurrencyInstance();
NumberFormat percent = NumberFormat.getPercentInstance();
percent.setMinimumFractionDigits(1);

// set the monthly payment to 100 and display it
int monthlyInvestment = 100;
System.out.println("Monthly investment: " +
        currency.format(monthlyInvestment) + "\n");

// create the header row and add it to the table
String table  = "";
String headerRow  = "Year      ";
for (double rate = 5.0; rate < 7.0; rate += .5) {
    headerRow += percent.format(rate/100) + "         ";
}
table += headerRow + "\n";
```

Nested loops that print a table of future values (cont.)

```
// loop through the years
for (int year = 1; year < 7; year++) {
    // add year to the start of the row
    String row = year + "    ";

    // loop through each interest rate
    for (double rate = 5.0; rate < 7.0; rate += .5) {
        int months = year * 12;
        double monthlyInterestRate = rate/12/100;

        // calculate the future value
        double futureValue = 0.0;
        for (int i = 1; i <= months; i++) {
            futureValue =
                (futureValue + monthlyInvestment) *
                (1 + monthlyInterestRate);
        }
```

Nested loops that print a table of future values (cont.)

```
            // add the calculation to the row
            row += currency.format(futureValue) + "    ";
        }
        // add the row to the table
        table += row + "\n";
    }
    System.out.println(table);
```

A break statement that exits the loop

```
while (true) {
    System.out.print("Enter a color: ");
    String line = sc.nextLine();
    if (line.equalsIgnoreCase("exit")) {
        break;
    }
    System.out.println("You entered: " + line + "\n");
}
System.out.println("Bye!");
```

The console

```
Enter a color: blue
You entered: blue

Enter a color: exit
Bye!
```

A continue statement that jumps to the beginning of a loop

```
String choice = "y";
while (choice.equalsIgnoreCase("y")) {
    System.out.print("Enter a number: ");
    int number = sc.nextInt();
    if (number <= 0) {
        System.out.println(
            "Number must be greater than 0. Try again.");
        continue;
    }
    System.out.println("You entered: " + number + "\n");

    System.out.print("Continue? (y/n): ");
    choice = sc.next();
    System.out.println();
}
System.out.println("Bye!");
```

The console when the loop is executed

```
Enter a number: -100
Number must be greater than 0. Try again.
Enter a number: 100
You entered: 100

Continue? (y/n):
```

The console for the Guess the Number application

```
Guess the number!
I'm thinking of a number from 1 to 10

Your guess: 11
Invalid guess. Try again.
Your guess: 5
Too low.
Your guess: 7
You guessed it in 2 tries.

Bye!
```

The code for the Guess the Number application

```
import java.util.Scanner;

public class GuessNumberApp {

    public static void main(String[] args) {
        final int LIMIT = 10;

        System.out.println("Guess the number!");
        System.out.println(
            "I'm thinking of a number from 1 to " + LIMIT);
        System.out.println();

        // get a random number between 1 and the limit
        double d = Math.random() * LIMIT;
                                 // d is >= 0.0 and < limit
        int number = (int) d;    // convert double to int
        number++;                // int is >= 1 and <= limit

        Scanner sc = new Scanner(System.in);
        int count = 1;
```

The Guess the Number application (cont.)

```
        while (true) {
            System.out.print("Your guess: ");
            int guess = sc.nextInt();

            if (guess < 1 || guess > LIMIT) {
                System.out.println(
                    "Invalid guess. Try again.");
                continue;
            }

            if (guess < number) {
                System.out.println("Too low.");
            } else if (guess > number) {
                System.out.println("Too high.");
            } else {
                System.out.println("You guessed it in " +
                                   count + " tries.\n");
                break;
            }
            count++;
        }
        System.out.println("Bye!");
    }
}
```