Chapter 8

How to code control statements

Objectives

Applied

- Code if/else statements and switch statements to control the logic of an application.
- Code while, do-while, and for loops to control the repetitive processing of an application.
- Code nested loops whenever they are required.
- Use the break statement to jump out of a loop.
- Use the continue statement to jump to the start of a loop.
- Code a try/catch statement that handles any exceptions that may occurs when you use the Integer and Double classes to convert String objects to numbers.

Objectives (cont.)

Knowledge

- Name and describe the relational operators.
- Name and describe the logical operators.
- Compare the if/else and switch statements.
- Explain what it means for execution to fall through a label in a switch statement.
- Differentiate between while and for loops.
- Explain what an exception is in Java.
- Describe the Exception hierarchy and name two of its subclasses.

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 numeric literal

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

subtotal >= 500            // > or = to a numeric literal
quantity <= reorderPoint   // < than or + to 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

Short-circuit AND

```
subtotal > 250 && subtotal < 500
```

Short-circuit OR

```
quantity <= 4 || quantity >= 12
```

2 short-circuit operators (&& and ||)

```
(subtotal > 250 && subtotal < 500) || isValid
```

NOT operator

```
!(counter++ >= years)
```

The syntax of the if/else statement

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

An if statement that only has an if clause

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

With an else clause

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

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;
}
```

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 int variable named productID

```
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 variable named productCode

```
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

```
Welcome to the Invoice 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 nested if/else statement that determines the discount percent

```
double discountPercent = 0;
if (customerType.equalsIgnoreCase("r")) {
    if (subtotal < 100) {
        discountPercent = 0.0;
    } else if (subtotal >= 100 && subtotal < 250) {
        discountPercent = .1;
    } else if (subtotal >= 250) {
        discountPercent = .2;
    }
} else if (customerType.equalsIgnoreCase("c"))  {
   if (subtotal < 250) {
        discountPercent = .2;
   } else if (subtotal >= 250) {
        discountPercent = .3;
   }
} else {
    discountPercent = .1;
}
```

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 + monthlyPayment) *
                  (1 + monthlyInterestRate);
    i++;
}
```

An infinite while loop

```
while (true) {
    // run this code endlessly
}
```

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 + monthlyPayment) *
                  (1 + monthlyInterestRate);
    i++;
} while (i <= months);
```

The syntax of the for loop

```
for (initializationExpr; booleanExpr; incrementExpr) {
    statements
}
```

A for loop that stores numbers in a string

```
String numbers = "";
for (int i = 0; i < 5; i++)  {
    numbers += i;
    numbers += " ";
}
```

The string displayed on the console

```
0 1 2 3 4
```

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

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

An infinite for loop

```
for ( ; ; ) {
    // continue executing loop until the application ends
}
```

A for loop without braces

```
String numbers = "";
for (int i = 0; i < 5; i++)
    numbers += i + " ";
```

The syntax of the break statement

```
break;
```

A statement that exits the loop

```
while (true) {
    int random = (int) (Math.random() * 10);
    System.out.println(random);
    if (random == 7) {
        System.out.println("value found - exit loop!");
        break;
    }
}
```

The console

```
2
5
7
value found - exit loop!
```

The syntax of the continue statement

```
continue;
```

A statement that jumps to the beginning of a loop

```
for (int i = 0; i < 5; i++) {
    int random = (int) (Math.random() * 10);
    if (random > 7) {
        System.out.println(
            "invalid value - continue loop!");
        continue;
    }
    System.out.println(random);
}
```

The console

```
invalid value - continue loop!
0
1
2
invalid value - continue loop!
```

Some of the classes in the Exception hierarchy

```
Exception
    RuntimeException
        IllegalArgumentException
            NumberFormatException
        ArithmeticException
        NullPointerException
```

The console after a NumberFormatException

```
Enter monthly investment:   $100
Exception in thread "main"
java.lang.NumberFormatException: For input string: "$100"
    at sun.misc.FloatingDecimal.readJavaFormatString(
        FloatingDecimal.java:2043)
    at sun.misc.FloatingDecimal.parseDouble(
        FloatingDecimal.java:110)
    at java.lang.Double.parseDouble(
        Double.java:538)
    at murach.futurevalue.FutureValueApp.main(
        FutureValueApp.java:18)
Java Result: 1
```

Two methods that might throw an exception

Class	Method	Throws
`Integer`	`parseInt(String)`	`NumberFormatException`
`Double`	`parseDouble(String)`	`NumberFormatException`

The syntax for a simple try/catch statement

```
try { statements }
catch (ExceptionClass exceptionName) { statements }
```

Code that catches a NumberFormatException

```
String choice = "y";
while (!choice.equalsIgnoreCase("n")) {
    // get the input from the user
    System.out.print("Enter monthly investment:   ");
    double monthlyInvestment;
    try {
       String line = sc.nextLine();
       monthlyInvestment = Double.parseDouble(line);
    } catch (NumberFormatException e)  {
       System.out.println(
          "Error! Invalid number. Try again.\n");
       continue;      // jump to the top of the loop
    }

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

Console output

```
Enter monthly investment:   $100
Error! Invalid number. Try again.

Enter monthly investment:
```

The console

```
Welcome to 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): y

Enter monthly investment:   100
Enter yearly interest rate: 3
Enter number of years:      four
Error! Invalid integer. Try again.
Enter number of years:      4
Future value:               $5,105.85

Continue? (y/n): n

Bye!
```

The Financial class

```
package murach.calculators;

public class Financial {

     public static double calculateFutureValue(
            double monthlyInvestment,
            double yearlyInterestRate, int years) {

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

        // calculate the future value
        double futureValue = 0;
        for (int i = 1; i <= months; i++) {
            futureValue += monthlyInvestment;
            double monthlyInterestAmount = futureValue *
               monthlyInterestRate;
            futureValue += monthlyInterestAmount;
        }

        return futureValue;
    }
}
```

The Console class

```
package murach.ui;

import java.util.Scanner;

public class Console {

    private static Scanner sc = new Scanner(System.in);

    public static void displayLine() {
        System.out.println();
    }

    public static void displayLine(String s) {
        System.out.println(s);
    }

    public static String getString(String prompt) {
        System.out.print(prompt);
        String s = sc.nextLine();
        return s;
    }
```

The Console class (cont.)

```
public static int getInt(String prompt) {
    int i = 0;
    while (true) {
        System.out.print(prompt);
        try {
            i = Integer.parseInt(sc.nextLine());
            break;
        } catch (NumberFormatException e) {
            System.out.println(
                "Error! Invalid integer. Try again.");
        }
     }
     return i;
}
```

The Console class (cont.)

```
public static double getDouble(String prompt) {
        double d = 0;
        while (true) {
            System.out.print(prompt);
            try {
                d = Double.parseDouble(sc.nextLine());
                break;
            } catch (NumberFormatException e) {
                System.out.println(
                    "Error! Invalid decimal. Try again.");
            }
        }
        return d;
    }
}
```

The Main class

```
package murach.ui;

import java.text.NumberFormat;
import murach.calculators.Financial;

public class Main {

    public static void main(String[] args) {
        // displayLine a welcome message
        Console.displayLine(
            "Welcome to the Future Value Calculator");
        Console.displayLine();

        String choice = "y";
        while (choice.equalsIgnoreCase("y")) {

            // get input from user
            double monthlyInvestment = Console.getDouble(
                "Enter monthly investment:   ");
            double yearlyInterestRate = Console.getDouble(
                "Enter yearly interest rate: ");
            int years = Console.getInt(
                "Enter number of years:      ");
```

The Main class (cont.)

```
            // call the future value method
            double futureValue = Financial.calculateFutureValue(
                monthlyInvestment, yearlyInterestRate, years);

            // format and displayLine the result
            Console.displayLine("Future value:                " +
                NumberFormat.getCurrencyInstance()
                            .format(futureValue));
            Console.displayLine();

            // see if the user wants to continue
            choice = Console.getString("Continue? (y/n): ");
            Console.displayLine();
        }
        Console.displayLine("Bye!");
    }
}
```