Flow of Control

Chapter 3

Flow of Control

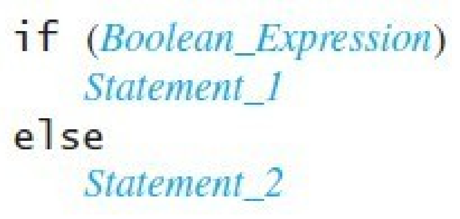
- Flow of control is the order in which a program performs actions.
  - Up to this point, the order has been sequential.
- A branching statement chooses between two or more possible actions.
- A loop statement repeats an action until a stopping condition occurs.

The `if-else` Statement

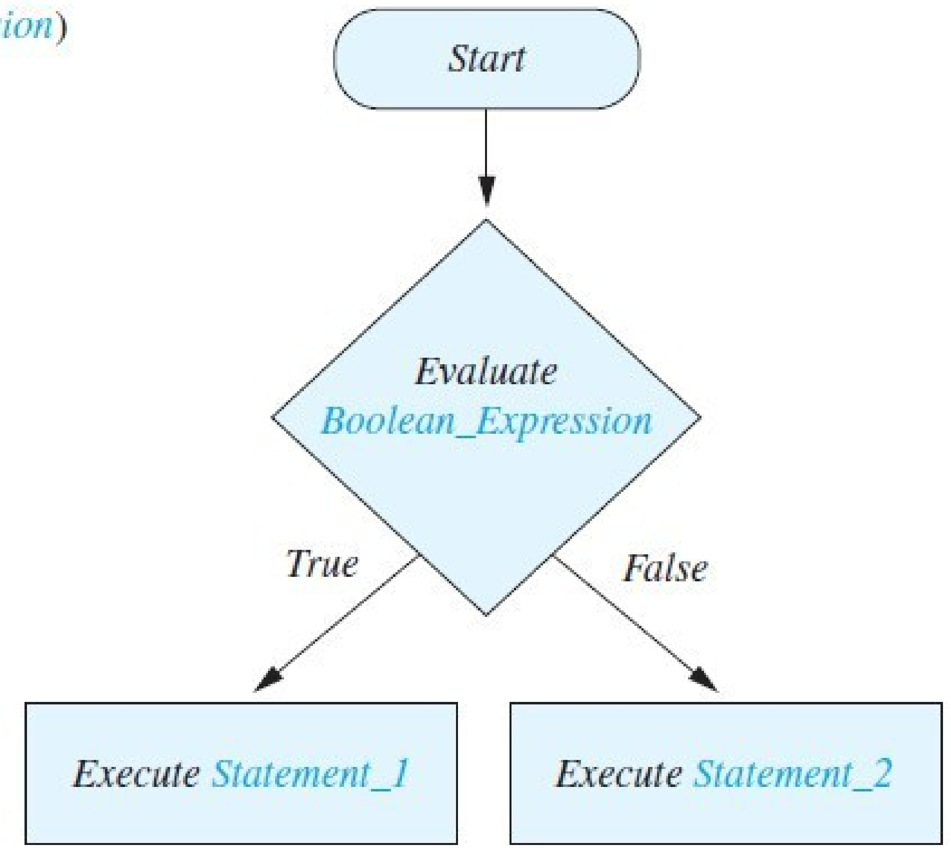
- A branching statement that chooses between two possible actions.
- Syntax

```
if (Boolean_Expression)
  Statement_1
else
  Statement_2
```

Semantics of the `if-else` Statement



```
if (Boolean_Expression)
    Statement_1
else
    Statement_2
```



The `if-else` Statement

- Example

```
if (balance >= 0)
    balance = balance + (INTEREST_RATE * balance) / 12;
else
    balance = balance - OVERDRAWN_PENALTY;
```

Compound Statements

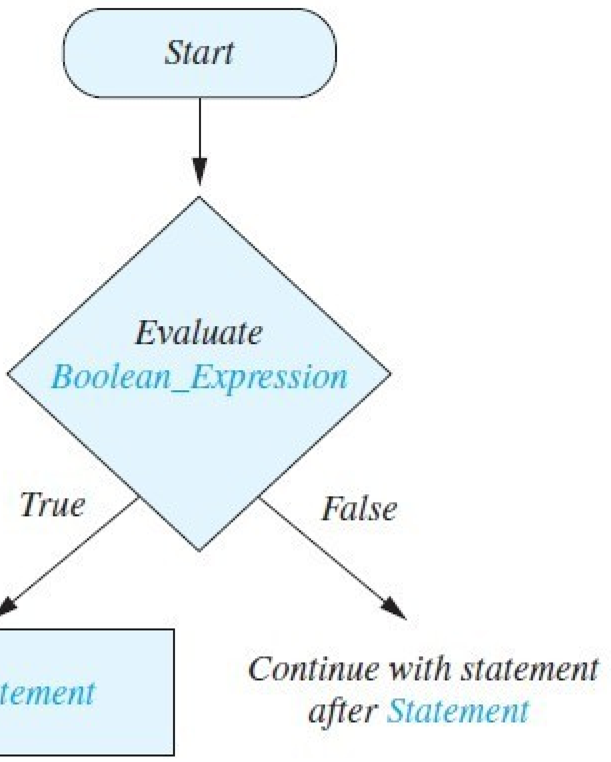
- To include multiple statements in a branch, enclose the statements in braces.

```
if (count < 3)
   {
      total = 0;
      count = 0;
   }
```

Omitting the **`else`** Part

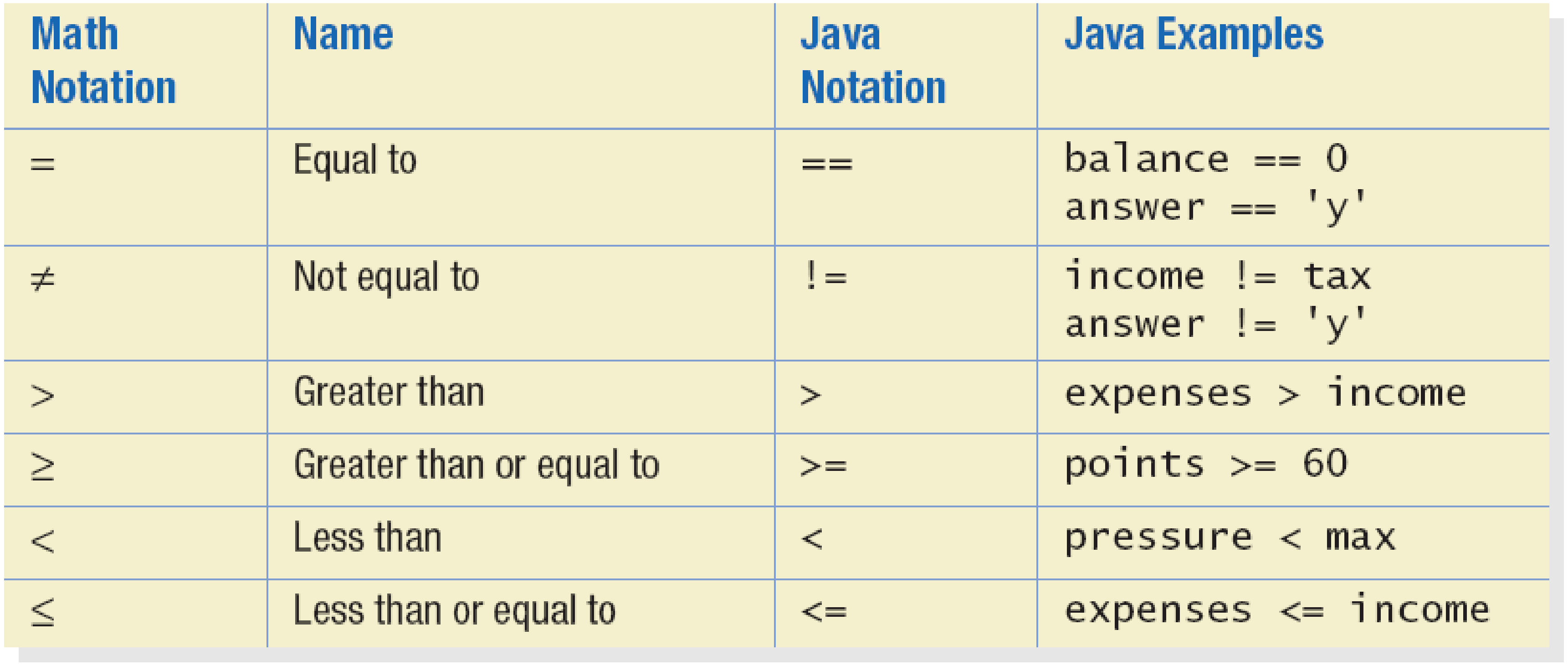
- The Semantics of an **`if`** Statement without an **`else`**

```
if (Boolean_Expression)
    Statement
```


Start
Evaluate Boolean_Expression
True
False
Execute Statement
Continue with statement after Statement

Java Comparison Operators

- Figure 3.4 Java Comparison Operators



Math Notation	Name	Java Notation	Java Examples

=	Equal to	`==`	`balance == 0`
			`answer == 'y'`

≠	Not equal to	`!=`	`income != tax`
			`answer != 'y'`

>	Greater than	`>`	`expenses > income`
≥	Greater than or equal to	`>=`	`points >= 60`
<	Less than	`<`	`pressure < max`
≤	Less than or equal to	`<=`	`expenses <= income`

Compound Boolean Expressions

- Boolean expressions can be combined using the "and" **(&&)** operator.
- Example

```
if ((score > 0) && (score <= 100))
...
```

- Not allowed

```
if (0 < score <= 100)
...
```

Compound Boolean Expressions

- Syntax

```
(Sub_Expression_1) && (Sub_Expression_2)
```

- Parentheses often are used to enhance readability.
- The larger expression is true only when both of the smaller expressions are true.

Compound Boolean Expressions

- Boolean expressions can be combined using the "or" (**||**) operator.
- Example

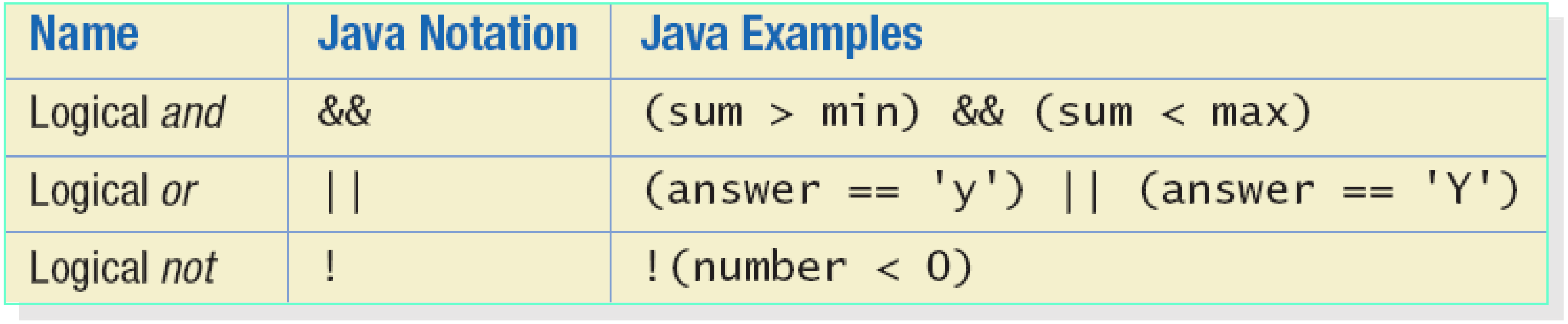
```
if ((quantity > 5) || (cost < 10))
...
```

- Syntax

```
(Sub_Expression_1) || (Sub_Expression_2)
```

Java Logical Operators

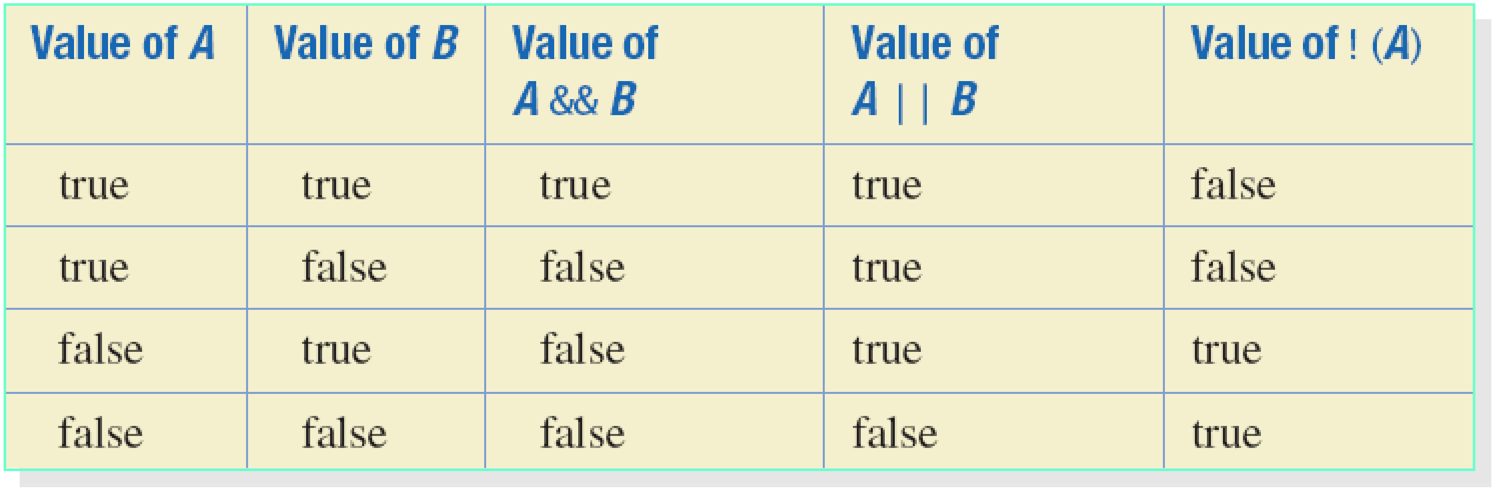
- Figure 3.6



Name	Java Notation	Java Examples
Logical *and*	`&&`	`(sum > min) && (sum < max)`
Logical *or*	`||`	`(answer == 'y') || (answer == 'Y')`
Logical *not*	`!`	`!(number < 0)`

Boolean Operators

- FIGURE 3.7 The Effect of the Boolean Operators **&&** (and), **||** (or), and **!** (not) on Boolean values



Value of *A*	Value of *B*	Value of *A* && *B*	Value of *A* || *B*	Value of ! (*A*)
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

Compound Statements

- When a list of statements is enclosed in braces (`{}`), they form a single compound statement.
- Syntax

```
{
    Statement_1;
    Statement_2;
  …
}
```

Compound Statements

- A compound statement can be used wherever a statement can be used.
- Example

```
if (total > 10)
{
   sum = sum + total;
   total = 0;
}
```

Multibranch `if-else` Statements

- Syntax

```
if (Boolean_Expression_1)
   Statement_1
else if (Boolean_Expression_2)
   Statement_2
else if (Boolean_Expression_3)
   Statement_3
else if …
else
   Default_Statement
```

The `exit` Method

- Sometimes a situation arises that makes continuing the program pointless.
- A program can be terminated normally by

```
System.exit(0).
```

Input Validation

- You should check your input to ensure that it is within a valid or reasonable range. For example, consider a program that converts feet to inches. You might write the following:

```
int feet = key.nextInt();
int inches = feet * 12;
```

- What if:
  - The user types a negative number for feet?
  - The user enters an unreasonable value like 100? Or a number larger than can be stored in an int? (2,147,483,647)

Input Validation

- Address these problems by ensuring that the entered values are reasonable:

```
int feet = key.nextInt();
if ((feet >= 0) && (feet < 10))
{
  int inches = feet * 12;
  ...
}
```