

Chapter 3 Selections

Relational Operators

Java Operator	Mathematics Symbol	Name	Example (radius is 5)	Result
<	<	less than	<code>radius < 0</code>	<code>false</code>
<=	≤	less than or equal to	<code>radius <= 0</code>	<code>false</code>
>	>	greater than	<code>radius > 0</code>	<code>true</code>
>=	≥	greater than or equal to	<code>radius >= 0</code>	<code>true</code>
==	=	equal to	<code>radius == 0</code>	<code>false</code>
!=	≠	not equal to	<code>radius != 0</code>	<code>true</code>

The `boolean` Type and Operators

Often in a program you need to compare two values, such as whether `i` is greater than `j`. Java provides six comparison operators (also known as relational operators) that can be used to compare two values. The result of the comparison is a Boolean value: `true` or `false`.

```
boolean b = (1 > 2) ;
```

Problem: A Simple Math Learning Tool

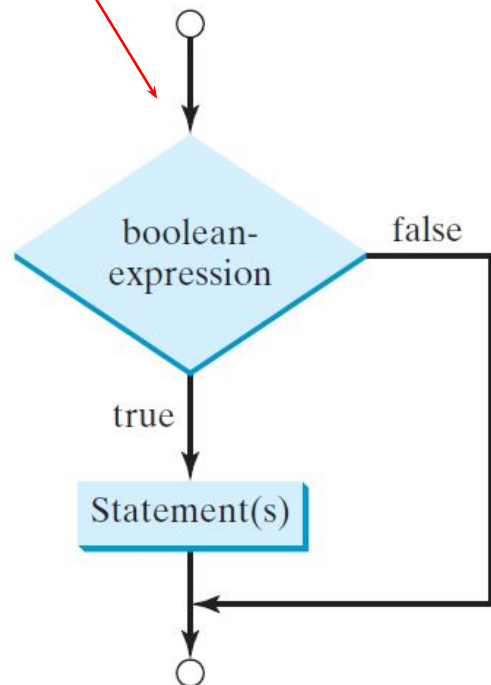
This example creates a program to let a first grader practice additions. The program randomly generates two single-digit integers number1 and number2 and displays a question such as “What is $7 + 9$?” to the student. After the student types the answer, the program displays a message to indicate whether the answer is true or false.

AdditionQuiz

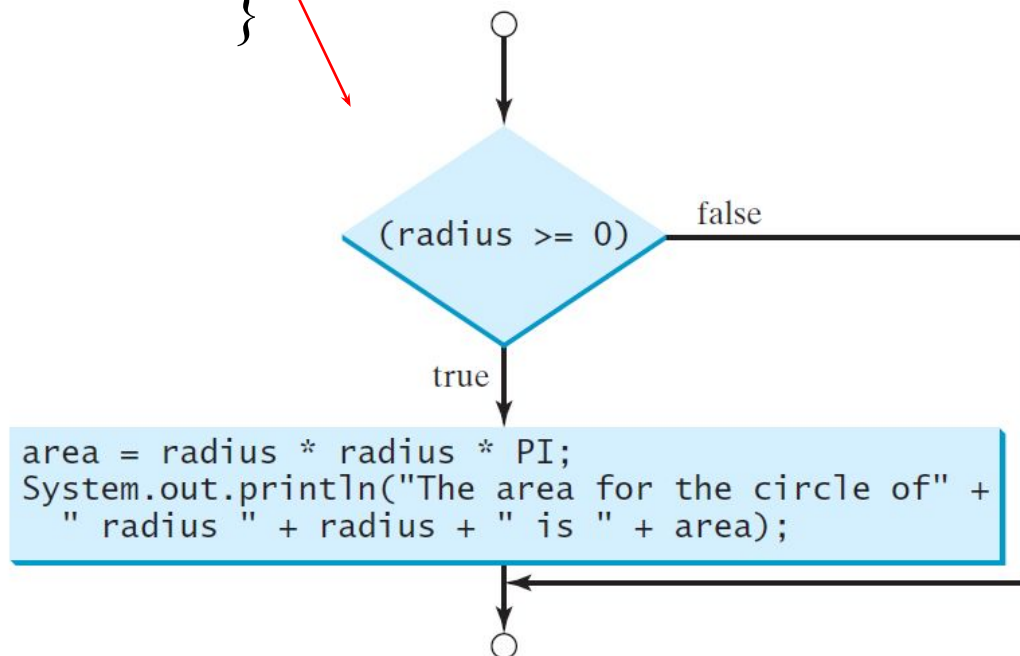
Run

One-way if Statements

```
if (boolean-expression) {
    statement(s);
}
```



```
if (radius >= 0) {
    area = radius * radius * PI;
    System.out.println("The area"
        + " for the circle of radius "
        + radius + " is " + area);
}
```



Simple if Demo

Write a program that prompts the user to enter an integer. If the number is a multiple of 5, print HiFive. If the number is divisible by 2, print HiEven.



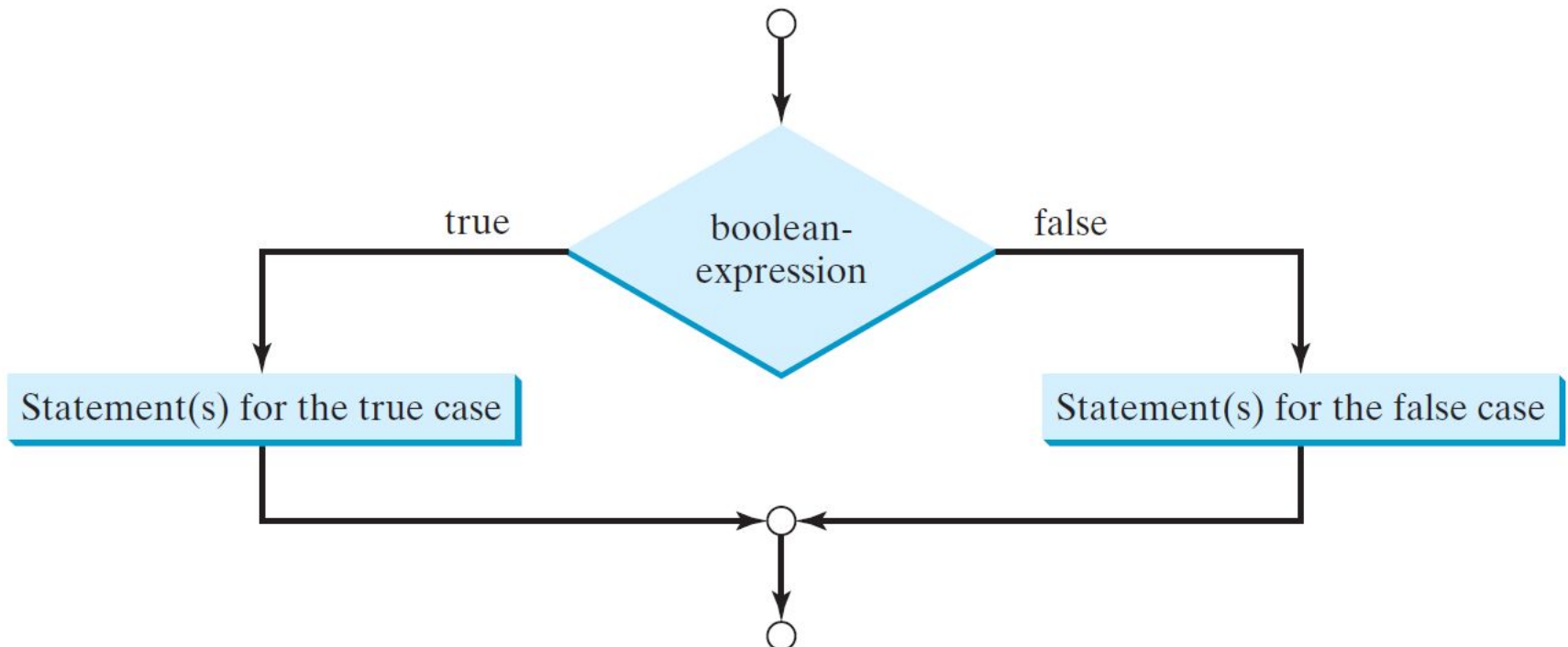
SimpleIfDemo

Run

The Two-way `if` Statement

```

if (boolean-expression) {
    statement(s) -for-the-true-case;
}
else {
    statement(s) -for-the-false-case;
}
    
```



if-else Example

```
if (radius >= 0) {  
    area = radius * radius * 3.14159;  
  
    System.out.println("The area for the "  
        + "circle of radius " + radius +  
        " is " + area);  
}  
else {  
    System.out.println("Negative input");  
}
```


Multiple Alternative if Statements

```

if (score >= 90.0)
    System.out.println("A");
else
    if (score >= 80.0)
        System.out.println("B");
    else
        if (score >= 70.0)
            System.out.println("C");
        else
            if (score >= 60.0)
                System.out.println("D");
            else
                System.out.println("F");
    
```

(a)

Equivalent

```

if (score >= 90.0)
    System.out.println("A");
else if (score >= 80.0)
    System.out.println("B");
else if (score >= 70.0)
    System.out.println("C");
else if (score >= 60.0)
    System.out.println("D");
else
    System.out.println("F");
    
```

This is better

(b)

TIP

```
if (number % 2 == 0)
    even = true;
else
    even = false;
```

(a)

Equivalent

```
boolean even
    = number % 2 == 0;
```

(b)

CAUTION

```
if (even == true)
    System.out.println(
        "It is even.");
```

(a)

Equivalent

```
if (even)
    System.out.println(
        "It is even.");
```

(b)

Problem: An Improved Math Learning Tool

This example creates a program to teach a first grade child how to learn subtractions.

The program randomly generates two single-digit integers number1 and number2 with number1 \geq number2 and displays a question such as “What is $9 - 2$?” to the student. After the student types the answer, the program displays whether the answer is correct.

SubtractionQuiz

Run

Logical Operators

Operator	Name	Description
!	not	logical negation
&&	and	logical conjunction
	or	logical disjunction
^	exclusive or	logical exclusion

Truth Table for Operator !

p	!p	Example (assume age = 24, weight = 140)
true	false	!(age > 18) is false, because (age > 18) is true.
false	true	!(weight == 150) is true, because (weight == 150) is false.

Truth Table for Operator &&

p_1	p_2	$p_1 \ \&\& \ p_2$	Example (assume age = 24, weight = 140)
false	false	false	$(\text{age} \leq 18) \ \&\& \ (\text{weight} < 140)$ is false, because both conditions are both false.
false	true	false	
true	false	false	$(\text{age} > 18) \ \&\& \ (\text{weight} > 140)$ is false, because $(\text{weight} > 140)$ is false.
true	true	true	$(\text{age} > 18) \ \&\& \ (\text{weight} \geq 140)$ is true, because both $(\text{age} > 18)$ and $(\text{weight} \geq 140)$ are true.

Truth Table for Operator ||

p_1	p_2	$p_1 \parallel p_2$	Example (assume age = 24, weight = 140)
false	false	false	
false	true	true	$(\text{age} > 34) \parallel (\text{weight} \leq 140)$ is true, because $(\text{age} > 34)$ is false, but $(\text{weight} \leq 140)$ is true.
true	false	true	$(\text{age} > 14) \parallel (\text{weight} \geq 150)$ is false, because $(\text{age} > 14)$ is true.
true	true	true	

Truth Table for Operator \wedge

p_1	p_2	$p_1 \wedge p_2$	Example (assume age = 24, weight = 140)
false	false	false	$(\text{age} > 34) \wedge (\text{weight} > 140)$ is true, because $(\text{age} > 34)$ is false and $(\text{weight} > 140)$ is false.
false	true	true	$(\text{age} > 34) \wedge (\text{weight} \geq 140)$ is true, because $(\text{age} > 34)$ is false but $(\text{weight} \geq 140)$ is true.
true	false	true	$(\text{age} > 14) \wedge (\text{weight} > 140)$ is true, because $(\text{age} > 14)$ is true and $(\text{weight} > 140)$ is false.
true	true	false	

Examples

```
System.out.println("Is " + number + " divisible by 2 and 3? " +  
((number % 2 == 0) && (number % 3 == 0)));
```

```
System.out.println("Is " + number + " divisible by 2 or 3? " +  
((number % 2 == 0) || (number % 3 == 0)));
```

```
System.out.println("Is " + number +  
" divisible by 2 or 3, but not both? " +  
((number % 2 == 0) ^ (number % 3 == 0)));
```

TestBooleanOperators

Run

Problem: Determining Leap Year?

This program first prompts the user to enter a year as an int value and checks if it is a leap year.

A year is a leap year if it **is divisible by 4** but not by 100, or it is divisible by 400.

```
(year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)
```

LeapYear

Run

switch Statement Rules

The switch-expression must yield a value of char, byte, short, or int type and must always be enclosed in parentheses.

The value1, ..., and valueN must have the same data type as the value of the switch-expression. The resulting statements in the case statement are executed when the value in the case statement matches the value of the switch-expression. Note that value1, ..., and valueN are constant expressions, meaning that they cannot contain variables in the expression, such as $1 + x$.

```

switch (switch-expression) {
    case value1: statement(s)1;
                break;
    case value2: statement(s)2;
                break;
    ...
    case valueN: statement(s)N;
                break;
    default: statement(s)-for-default;
}
    
```


switch Statement Rules

The keyword break is optional, but it should be used at the end of each case in order to terminate the remainder of the switch statement. If the break statement is not present, the next case statement will be executed.

The default case, which is optional, can be used to perform actions when none of the specified cases matches the switch-expression.

```

switch (switch-expression) {
    case value1: statement(s)1;
                break;
    case value2: statement(s)2;
                break;
    ...
    case valueN: statement(s)N;
                break;
    default: statement(s)-for-default;
}
    
```



When the value in a **case** statement matches the value of the **switch-expression**, the statements *starting from this case* are executed until either a **break** statement or the end of the **switch** statement is reached.

Trace switch statement

Suppose day is 2:

```
switch (day) {  
  case 1:  
  case 2:  
  case 3:  
  case 4:  
  case 5: System.out.println("Weekday"); break;  
  case 0:  
  case 6: System.out.println("Weekend");  
}
```

Conditional Operators

```
if (x > 0)
```

```
    y = 1
```

```
else
```

```
    y = -1;
```

is equivalent to

```
y = (x > 0) ? 1 : -1;
```

```
(boolean-expression) ? expression1 : expression2
```

Conditional Operator

```
if (num % 2 == 0)
```

```
    System.out.println(num + "is even");
```

```
else
```

```
    System.out.println(num + "is odd");
```

```
System.out.println(
```

```
    (num % 2 == 0)? num + "is even" :
```

```
    num + "is odd");
```


Operator Precedence

- `var++`, `var--`
- `+`, `-` (Unary plus and minus), `++var`, `--var`
- (type) Casting
- `!` (Not)
- `*`, `/`, `%` (Multiplication, division, and remainder)
- `+`, `-` (Binary addition and subtraction)
- `<`, `<=`, `>`, `>=` (Relational operators)
- `==`, `!=`; (Equality)
- `^` (Exclusive OR)
- `&&` (Conditional AND) Short-circuit AND
- `||` (Conditional OR) Short-circuit OR
- `=`, `+=`, `-=`, `*=`, `/=`, `%=` (Assignment operator)

Operator Precedence and Associativity

The expression in the parentheses is evaluated first. (Parentheses can be nested, in which case the expression in the inner parentheses is executed first.) When evaluating an expression without parentheses, the operators are applied according to the precedence rule and the associativity rule.

If operators with the same precedence are next to each other, their associativity determines the order of evaluation. All binary operators except assignment operators are left-associative.

Operator Associativity

When two operators with the same precedence are evaluated, the *associativity* of the operators determines the order of evaluation. All binary operators except assignment operators are *left-associative*.

$a - b + c - d$ is equivalent to $((a - b) + c) - d$

Assignment operators are *right-associative*. Therefore, the expression

$a = b += c = 5$ is equivalent to $a = (b += (c = 5))$

Example

Applying the operator precedence and associativity rule, the expression $3 + 4 * 4 > 5 * (4 + 3) - 1$ is evaluated as follows:

