Chapter 3 Selections

Motivations

If you assigned a negative value for <u>radius</u> in Listing 2.2, ComputeAreaWithConsoleInput.java, the program would print an invalid result. If the radius is negative, you don't want the program to compute the area. How can you deal with this situation?

Uploaded By: Jibreel Bornat

Objectives

- To declare **boolean** variables and write Boolean expressions using relational operators (§3.2).
- To implement selection control using one-way **if** statements (§3.3).
- To implement selection control using two-way **if-else** statements (§3.4).
- To implement selection control using nested **if** and multi-way **if** statements (§3.5).
- To avoid common errors and pitfalls in **if** statements (§3.6).
- To generate random numbers using the **Math.random**() method (§3.7).
- To program using selection statements for a variety of examples (**SubtractionQuiz**, **BMI**, **ComputeTax**) (§§3.7–3.9).
- To combine conditions using logical operators (**&&**, ‖, and **!**) (§3.10).
- To program using selection statements with combined conditions (**LeapYear**, **Lottery**) (§§3.11–3.12).
- To implement selection control using **switch** statements (§3.13).
- To write expressions using the conditional expression (§3.14).
- To examine the rules governing operator precedence and associativity (§3.15).
- To apply common techniques to debug errors (§3.16).

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);
```

Uploaded By: Jibreel Bornat 4

Relational Operators

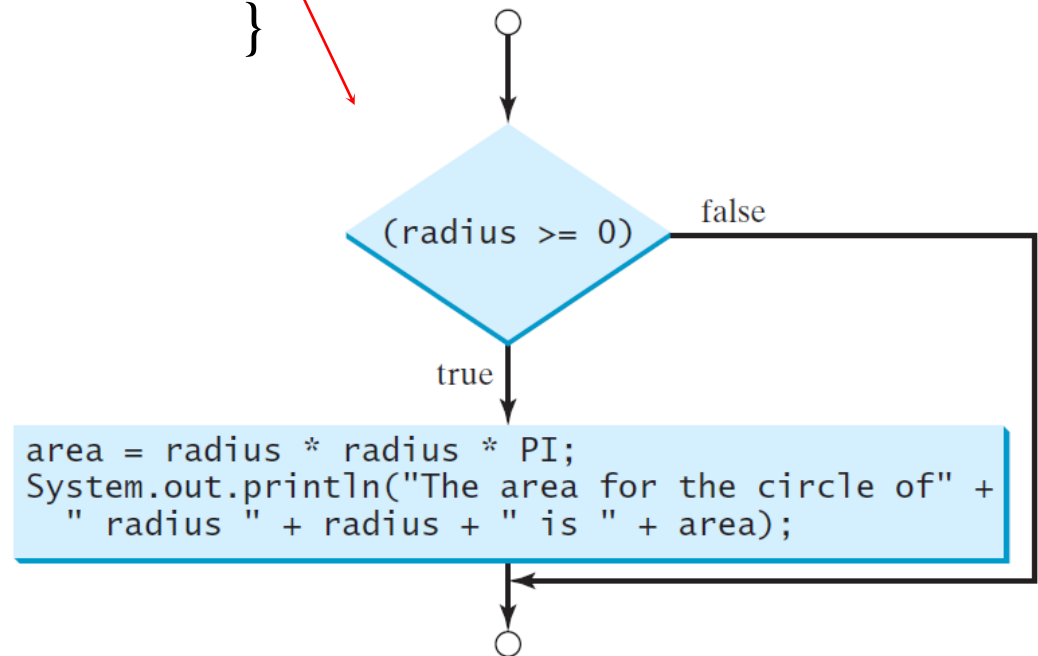| Java Operator	Mathematics Symbol	Name	Example (radius is 5)	Result
<	<	less than	radius < 0	false
<=	≤	less than or equal to	radius <= 0	false
>	>	greater than	radius > 0	true
>=	≥	greater than or equal to	radius >= 0	true
==	=	equal to	radius == 0	false
!=	≠	not equal to	radius != 0	true

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

One-way `if` Statements

if (boolean-expression) {
  statement(s);
}
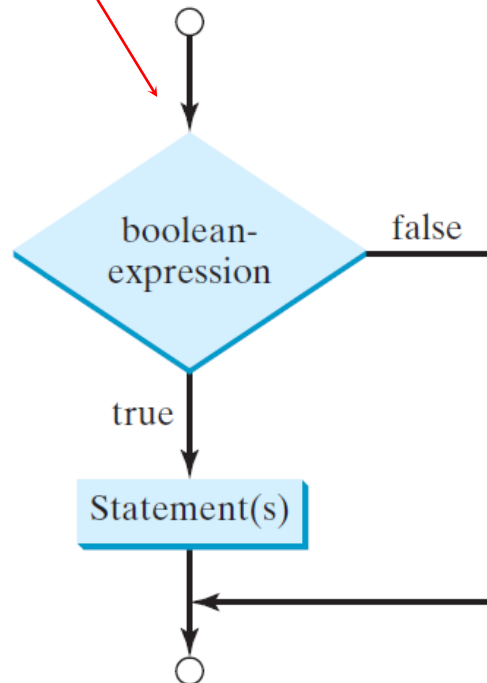
if (radius >= 0) {

  area = radius * radius * PI;

  System.out.println("The area"

    + " for the circle of radius "

    + radius + " is " + area);

}


```
boolean-
expression          false

true

Statement(s)
```

```
(radius >= 0)          false

true

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

Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.
Uploaded By: Jibreel Bornat

Note

```
if i > 0 {
   System.out.println("i is positive");
}
```

(a) Wrong

```
if (i > 0) {
   System.out.println("i is positive");
}
```

(b) Correct

```
if (i > 0) {
   System.out.println("i is positive");
}
```

(a)

Equivalent

```
if (i > 0)
   System.out.println("i is positive");
```

(b)

7

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.
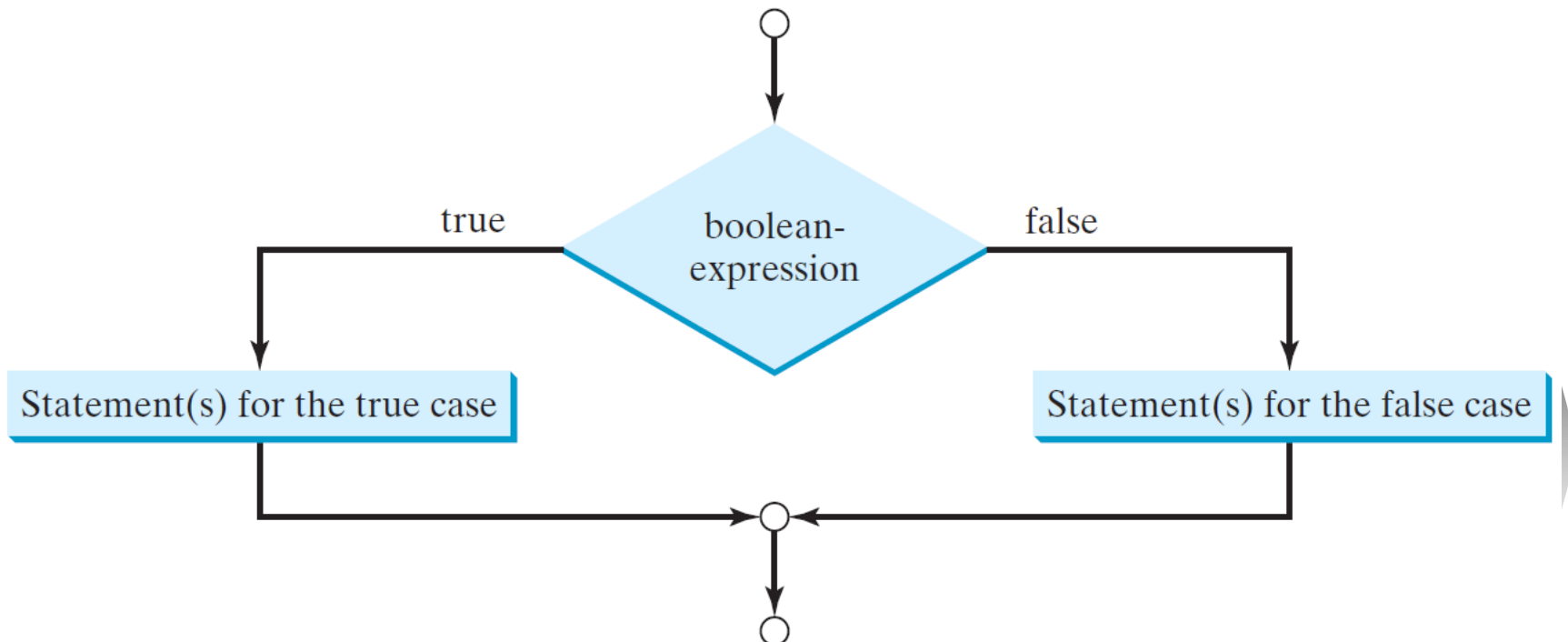
SimpleIfDemo   Run

The Two-way `if` Statement

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

Uploaded By: Jibreel Bornat

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");
}
```

Uploaded By: Jibreel Bornat 10

Multiple Alternative if Statements

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

(a)

Equivalent

This is better

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

(b)

Multi-Way if-else Statements

Uploaded By: Jibreel Bornat 12

Trace if-else statement

Suppose score is 70.0

The condition is false

```java
if (score >= 90.0)
  System.out.print("A");
else if (score >= 80.0)
  System.out.print("B");
else if (score >= 70.0)
  System.out.print("C");
else if (score >= 60.0)
  System.out.print("D");
else
  System.out.print("F");
```
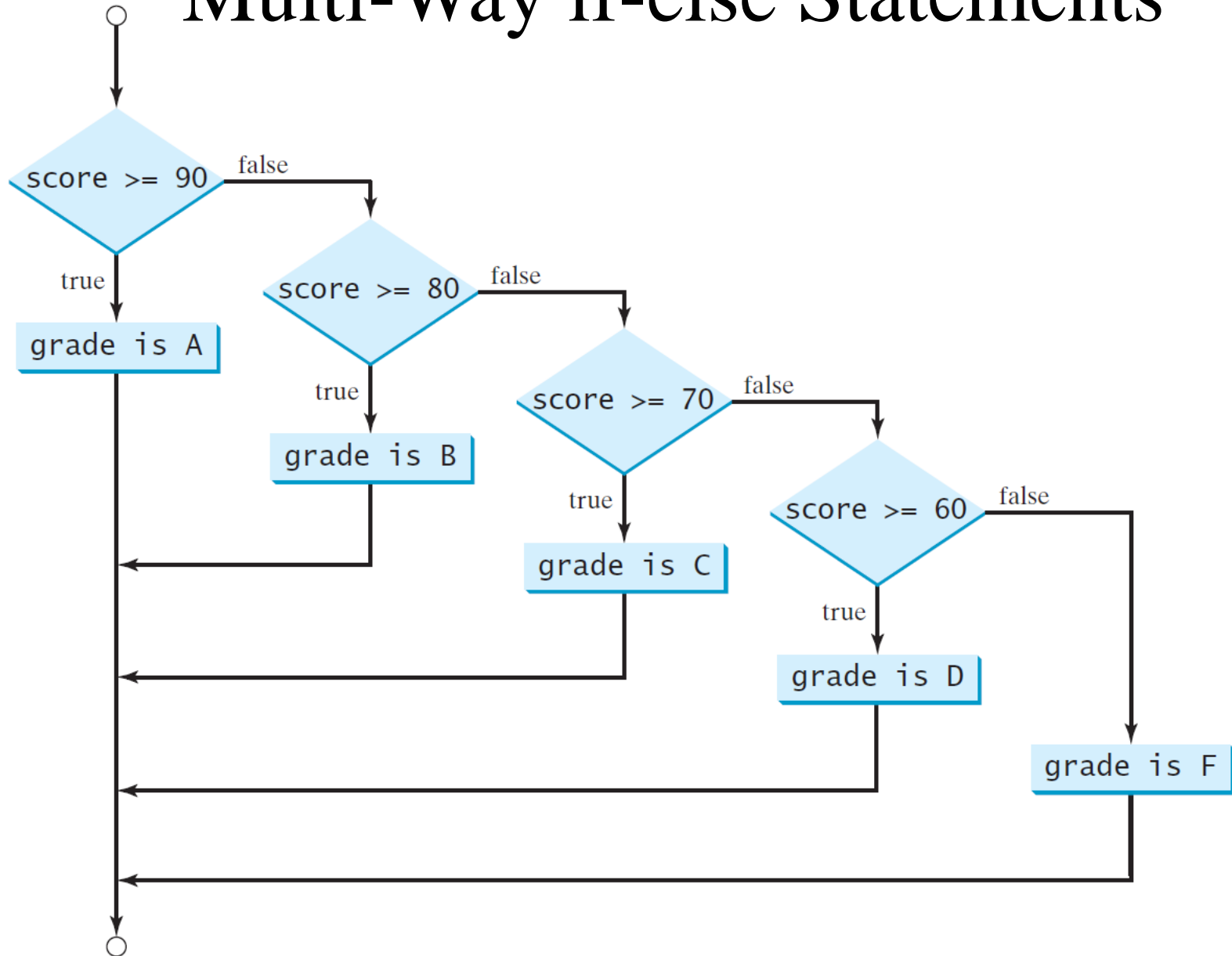
Uploaded By: Jibreel Bornat

Trace if-else statement

> Suppose score is 70.0

> The condition is false

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

Trace if-else statement

animation

> Suppose score is 70.0

> The condition is true

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

Trace if-else statement

Suppose score is 70.0

grade is C

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

Uploaded By: Jibreel Bornat

Trace if-else statement

Suppose score is 70.0

Exit the if statement

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

Uploaded By: Jibreel Bornat

Note

The <u>else</u> clause matches the most recent <u>if</u> clause in the same block.

```
int i = 1, j = 2, k = 3;

if (i > j)
   if (i > k)
     System.out.println("A");
else
     System.out.println("B");
```

(a)

Equivalent

This is better with correct indentation

```
int i = 1, j = 2, k = 3;

if (i > j)
   if (i > k)
     System.out.println("A");
else
     System.out.println("B");
```

(b)

Uploaded By: Jibreel Bornat

Note, cont.

Nothing is printed from the preceding statement. To force the <u>else</u> clause to match the first <u>if</u> clause, you must add a pair of braces:

```
int i = 1;
int j = 2;
int k = 3;
if (i > j) {
  if (i > k)
    System.out.println("A");
}
else
  System.out.println("B");
```

This statement prints B.

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Common Errors

Adding a semicolon at the end of an <u>if</u> clause is a common mistake.

if (radius >= 0); ⟵ Wrong

{

  area = radius*radius*PI;

  System.out.println(

    "The area for the circle of radius " +

    radius + " is " + area);

}

This mistake is hard to find, because it is not a compilation error or a runtime error, it is a logic error.

This error often occurs when you use the next-line block style.

Uploaded By: Jibreel Bornat

TIP

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

(a)

Equivalent

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

(b)

Uploaded By: Jibreel Bornat
21

CAUTION

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

(a)

Equivalent

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

(b)

Uploaded By: Jibreel Bornat 22

Logical Operators

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

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

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.

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Truth Table for Operator &&

p_1	p_2	p_1 && p_2	Example (assume age = 24, weight = 140)
false	false	false	(age <= 18) && (weight < 140) is false, because both conditions are both false.
false	true	false	
true	false	false	(age > 18) && (weight > 140) is false, because (weight > 140) is false.
true	true	true	(age > 18) && (weight >= 140) is true, because both (age > 18) and (weight >= 140) are true.

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

25

Truth Table for Operator ||

p₁	p₂	p₁ || p₂	Example (assume age = 24, weihgt = 140)
false	false	false	
false	true	true	(age > 34) || (weight <= 140) is true, because (age > 34) is false, but (weight <= 140) is true.
true	false	true	(age > 14) || (weight >= 150) is false, because (age > 14) is true.
true	true	true	

Truth Table for Operator ^

p_1	p_2	p_1 ^ p_2	Example (assume age = 24, weight = 140)
false	false	false	(age > 34) ^ (weight > 140) is true, because (age > 34) is false and (weight > 140) is false.
false	true	true	(age > 34) ^ (weight >= 140) is true, because (age > 34) is false but (weight >= 140) is true.
true	false	true	(age > 14) ^ (weight > 140) is true, because (age > 14) is true and (weight > 140) is false.
true	true	false	

Uploaded By: Jibreel Bornat

27

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

28

switch Statements

```
switch (status) {
  case 0:  compute taxes for single filers;
        break;
  case 1:  compute taxes for married file jointly;
        break;
  case 2:  compute taxes for married file separately;
        break;
  case 3:  compute taxes for head of household;
        break;
  default: System.out.println("Errors: invalid status");
        System.exit(1);
}
```

`switch` Statement Flow Chart

Uploaded By: Jibreel Bornat

30
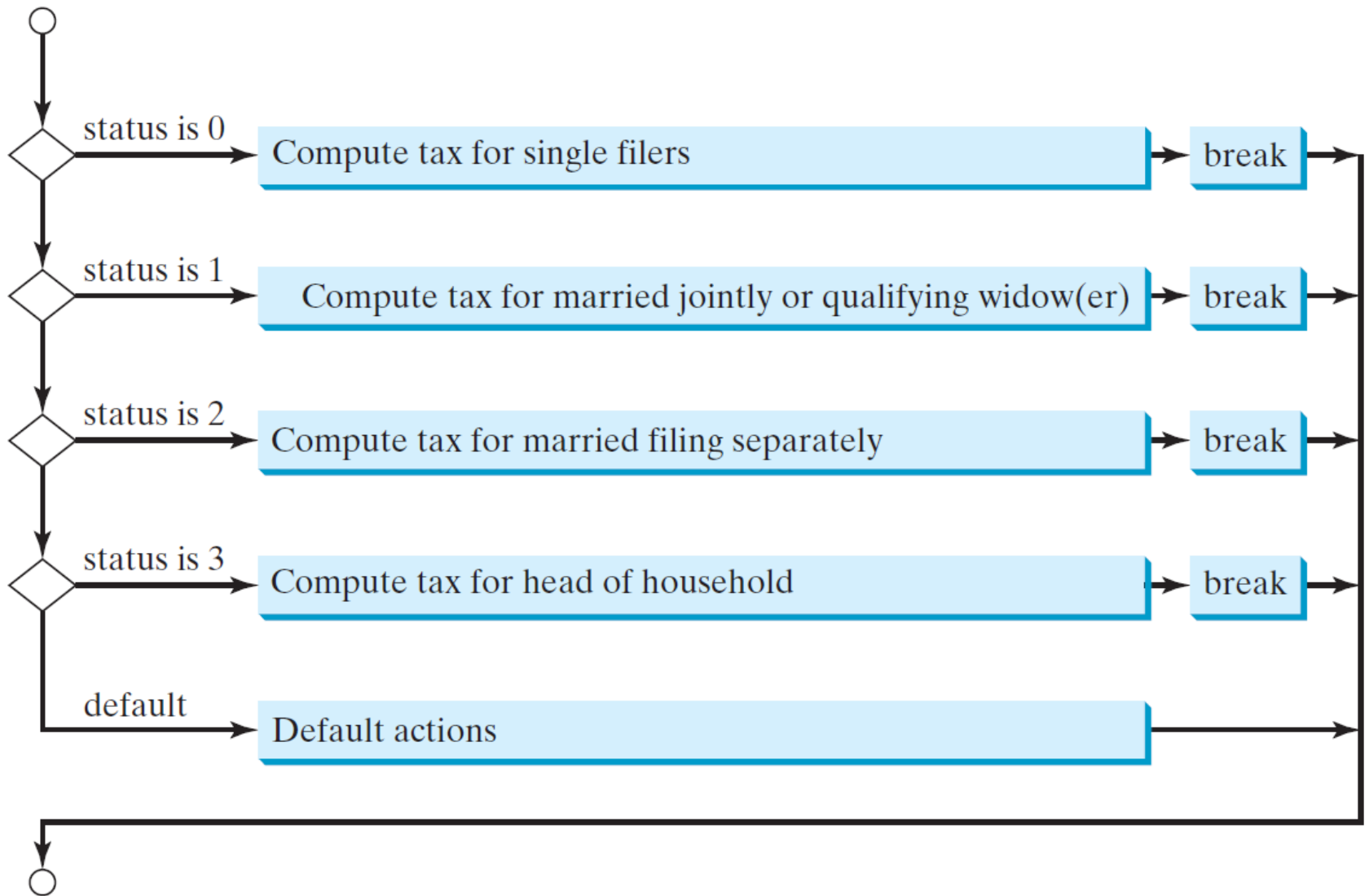
`switch` Statement Rules

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

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

```
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;
}
```

Uploaded By: Jibreel Bornat

31

`switch` Statement Rules

The keyword <u>break</u> is optional,

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

The <u>default</u> 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.

Uploaded By: Jibreel Bornat

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");
}
```

Uploaded By: Jibreel Bornat 33

Trace switch statement

Match case 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");
}
```

Uploaded By: Jibreel Bornat

Trace switch statement

Fall through case 3

```
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");
}
```

Uploaded By: Jibreel Bornat 35

Trace switch statement

Fall through case 4

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

 Uploaded By: Jibreel Bornat   36

Trace switch statement

Fall through case 5

```
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");
}
```

Uploaded By: Jibreel Bornat

Trace switch statement

Encounter break

```
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");
}
```

Uploaded By: Jibreel Bornat

38

Trace switch statement

Exit the statement

```
swit    day) {
  ca    :
  ca  2:
  c   3:
  c   e 4:
  c   e 5: System.out.println("Weekday"); break;
  ase 0:
  case 6: System.out.println("Weekend");
```

Uploaded By: Jibreel Bornat

Conditional Operators

if (x > 0)
  y = 1
else
  y = -1;

is equivalent to

y = (x > 0) ? 1 : -1;
(boolean-expression) ? expression1 : expression2

Uploaded By: Jibreel Bornat 40

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");
```

Uploaded By: Jibreel Bornat

Conditional Operator, cont.

```
boolean-expression ? exp1 : exp2
```

Uploaded By: Jibreel Bornat

Operator Precedence

☞ **+, - (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)**

Uploaded By: Jibreel Bornat   43

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.

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

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))

 Uploaded By: Jibreel Bornat
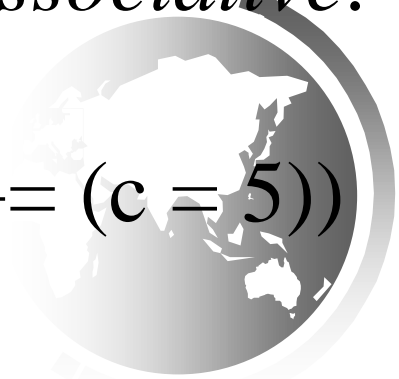
Example

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

```
3 + 4 * 4 > 5 * (4 + 3) - 1
```
(1) inside parentheses first

```
3 + 4 * 4 > 5 * 7 - 1
```
(2) multiplication

```
3 + 16 > 5 * 7 - 1
```
(3) multiplication

```
3 + 16 > 35 - 1
```
(4) addition

```
19 > 35 - 1
```
(5) subtraction

```
19 > 34
```
(6) greater than

```
false
```

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat