

TM105: Introduction to Computer Programming

Meeting 3 (Chapter 3)

Selections

Motivations

- If you assigned a negative value for radius in the program that we wrote last meeting (to compute the circle area), 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?



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Objectives

- To learn about algorithms a procedure for solving a problem.
- To introduce the **Math** class and compound assignment operators.
- To declare **boolean** variables and write Boolean expressions using relational operators.
- To implement selection control using one-way **if** statements.
- To implement selection control using two-way **if-else** statements.
- To implement selection control using nested **if** and multi-way **if** statements .
- To avoid common errors and pitfalls in **if** statements.
- To combine conditions using logical operators (**&&**, ||, and **!**).
- To implement selection control using **switch** statements.



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Algorithms

- Any computing problem can be solved by executing a series of actions in a specific order.
- An algorithm is a procedure for solving a problem in terms of
  - the **actions** to execute and
  - the **order** in which these actions execute
- Algorithms could be represented using
  - Pseudocode (text representation)
  - UML activity diagram (graphical representation)



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Pseudocode

- Pseudocode is an informal language (similar to everyday English) that helps you develop algorithms without having to worry about the strict details of Java language syntax.
- It helps you "think out" a program before writing it in Java.
- Pseudocode can be converted to a corresponding Java program.



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

UML activity diagram

- The UML (Unified Modeling Language) is a graphical language that allows people who build software systems to represent their object-oriented designs in an industry standard notation.
- Activity diagrams are part of the UML. It models the workflow (the activity) of a portion of a software system.

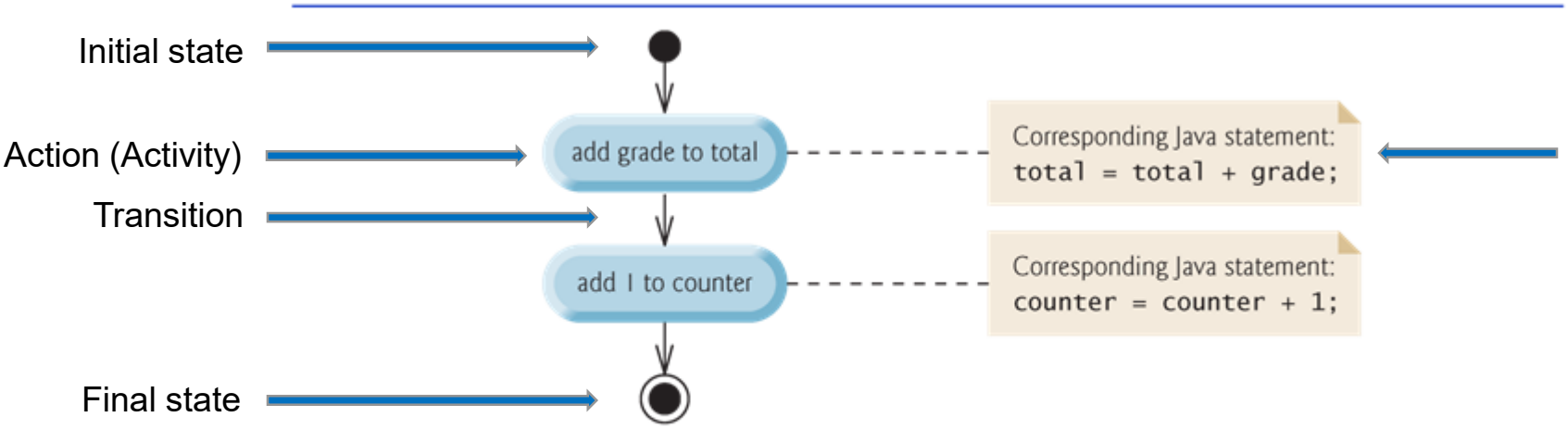


Fig. 4.1 | Sequence structure activity diagram.

Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Exercise

Write an application that inputs from the user the radius of a circle as a real number and prints the circle's diameter, circumference and area rounded to 2 decimal places.

Use the following formulas (r is the radius):

- $diameter = 2r$
- $circumference = 2\pi r$
- $area = \pi r^2$



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

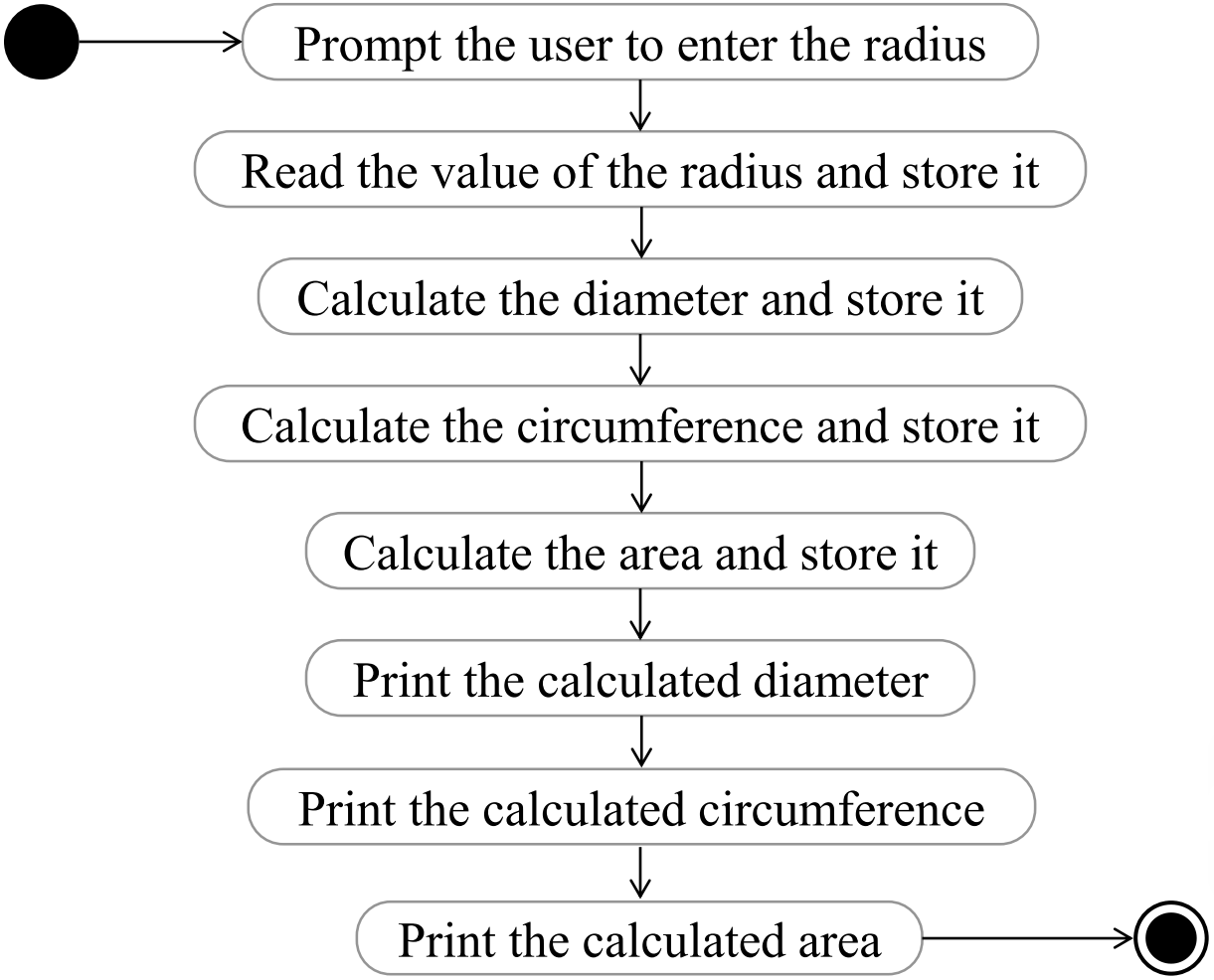
Exercise: The Algorithm as a Pseudocode

- Prompt the user to enter the value of the radius of the circle
- Read the value of the radius and store it
- Calculate the diameter of the circle and store it
- Calculate the circumference of the circle and store it
- Calculate the area of the circle and store it
- Print the calculated value of the diameter
- Print the calculated value of the circumference
- Print the calculated value of the area



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Exercise: The Algorithm as a UML Activity Diagram



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

```java
// To calculate diameter, circumference and area for a circle
import java.util.Scanner;
public class Circle {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        double radius, diameter, circumference, area;
        System.out.print("Enter radius: "); // prompt
        radius = keyboard.nextDouble(); // read radius
        diameter = 2 * radius;
        circumference = 2 * Math.PI * radius;
        // area = Math.PI * radius * radius;
        area = Math.PI * Math.pow(radius, 2);
        System.out.printf("Diameter is %.2f\n", diameter);
        System.out.printf("Circumference is %.2f\n", circumference);
        System.out.printf("Area is %.2f\n", area);
    } // end main method
} // end class Circle
```

```
-----output-----
Enter radius: 6.2
Diameter is 12.40
Circumference is 38.96
Area is 120.76
```

Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Class `Math`

- Class `Math` is part of `java.lang` package, which is implicitly imported by the compiler.
- Class `Math` declares commonly used mathematical constants
  - Math.PI (3.141592653589793)
  - Math.E (2.718281828459045)
- Class `Math` provides a collection of methods that enable you to perform common mathematical calculations.



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Method	Description	Example
`abs(x)`	absolute value of x	`abs(23.7)` is 23.7 `abs(0.0)` is 0.0 `abs(-23.7)` is 23.7
`ceil(x)`	rounds x to the smallest integer not less than x	`ceil(9.2)` is 10.0 `ceil(-9.8)` is -9.0
`cos(x)`	trigonometric cosine of x (x in radians)	`cos(0.0)` is 1.0
`exp(x)`	exponential method e^x	`exp(1.0)` is 2.71828 `exp(2.0)` is 7.38906
`floor(x)`	rounds x to the largest integer not greater than x	`floor(9.2)` is 9.0 `floor(-9.8)` is -10.0
`log(x)`	natural logarithm of x (base e)	`log(Math.E)` is 1.0 `log(Math.E * Math.E)` is 2.0
`max(x, y)`	larger value of x and y	`max(2.3, 12.7)` is 12.7 `max(-2.3, -12.7)` is -2.3
`min(x, y)`	smaller value of x and y	`min(2.3, 12.7)` is 2.3 `min(-2.3, -12.7)` is -12.7
`pow(x, y)`	x raised to the power y (i.e., x^y)	`pow(2.0, 7.0)` is 128.0 `pow(9.0, 0.5)` is 3.0
`sin(x)`	trigonometric sine of x (x in radians)	`sin(0.0)` is 0.0
`sqrt(x)`	square root of x	`sqrt(900.0)` is 30.0
`tan(x)`	trigonometric tangent of x (x in radians)	`tan(0.0)` is 0.0



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Class `Math`

- In addition, there is another method in class `Math` called `round(x)` which rounds x the to the nearest integer to x.
- For example:
  - `Math.round(3.5)` is 4 ← 4 of type long
  - `Math.round(3.45)` is 3
  - `Math.round(-3.2)` is -3 `// Be careful with negative values`

- Let us write a single java statement equivalent to the following equation:

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

```
// declaration and initialization of a, b, c, x;
x = (-b + Math.sqrt(Math.pow(b, 2) - 4 * a * c)) / (2 * a);
```

Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Compound Assignment Operators

Example:

```
c = c + 3;
```

can be written with the addition compound assignment operator, +=, as:

Assignment operator	Sample expression	Explanation	Assigns
Assume: `int c = 3, d = 5, e = 4, f = 6, g = 12;`			
+=	`c += 7`	`c = c + 7`	10 to c
-=	`d -= 4`	`d = d - 4`	1 to d
*=	`e *= 5`	`e = e * 5`	20 to e
/=	`f /= 3`	`f = f / 3`	2 to f
%=	`g %= 9`	`g = g % 9`	3 to g

Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Increment and Decrement Operators

- Unary increment operator, `++`, adds one to its operand.

Operator	Operator name	Sample expression	Explanation
`++`	prefix increment	`++a`	Increment `a` by 1, then use the new value of `a` in the expression in which `a` resides.
`++`	postfix increment	`a++`	Use the current value of `a` in the expression in which `a` resides, then increment `a` by 1.
`--`	prefix decrement	`--b`	Decrement `b` by 1, then use the new value of `b` in the expression in which `b` resides.
`--`	postfix decrement	`b--`	Use the current value of `b` in the expression in which `b` resides, then decrement `b` by 1.

Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

```java
// Prefix increment and postfix increment operators.
public class Increment {
   public static void main(String[] args) {
      int c;
      // demonstrate postfix increment operator
      c = 5; // assign 5 to c
      System.out.println(c);   // prints 5
      System.out.println(c++); // prints 5 then postincrements
      System.out.println(c);   // prints 6
      System.out.println(); // skip a line
      // demonstrate prefix increment operator
      c = 5; // assign 5 to c
      System.out.println(c);   // prints 5
      System.out.println(++c); // preincrements then prints 6
      System.out.println(c);   // prints 6
   } // end main method
} // end class Increment
```

```
-----output-----
5
5
6

5
6
6
```



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Increment and Decrement Operators

- In Java, an augmented expression of the form:
  `x1 op= x2`
  is implemented as:
  `x1 = (T)(x1 op x2)`, where `T` is the type for `x1`.

- Therefore, the following code is correct:

```
int sum = 0;
sum += 4.5; // sum becomes 4 after this statement
```

- `sum += 4.5` is equivalent to:
  `sum = (int)(sum + 4.5)`.



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

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 (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);
```



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Relational Operators

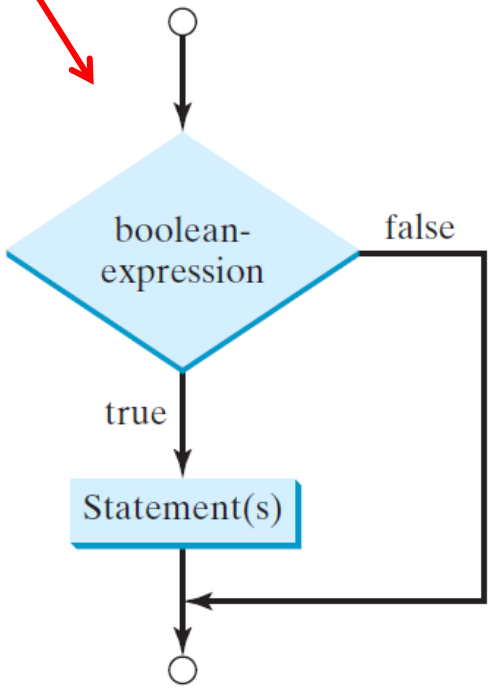
Java Operator	Mathematics Symbol	Name	Example (radius is 5)	Result
`<`	<	less than	`radius < 0`	`false`
`<=`	≤	less than or equal to	`radius <= 0`	`false`
`>`	>	greater than	`radius > 0`	`true`
`>=`	≥	greater than or equal to	`radius >= 0`	`true`
`==`	=	equal to	`radius == 0`	`false`
`!=`	≠	not equal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



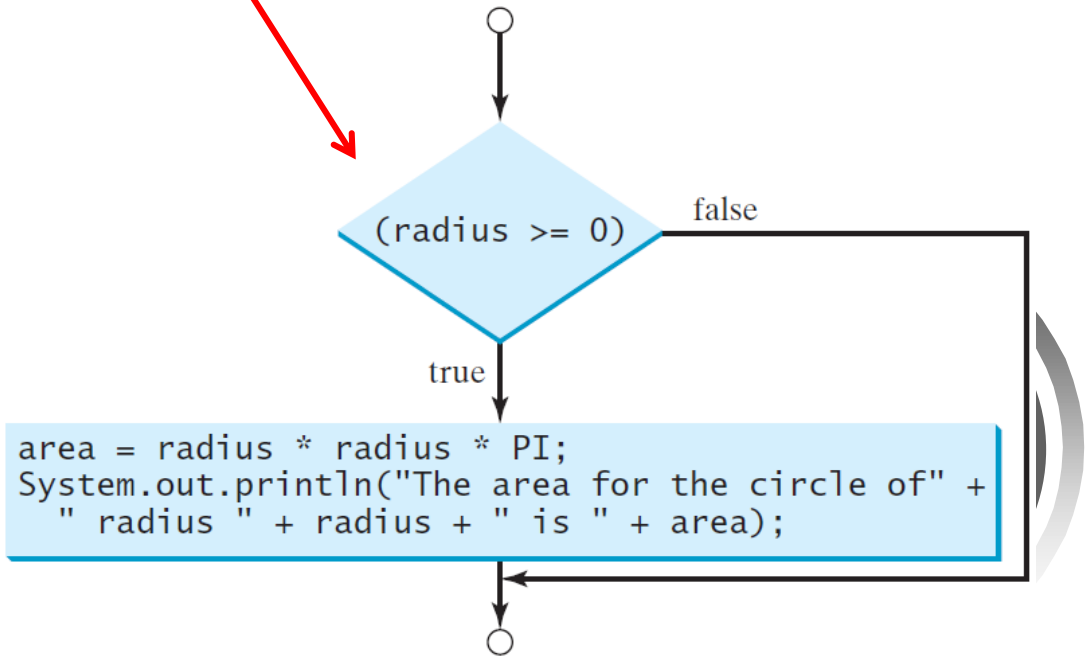
Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

One-way `if` Statements

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




```
if (radius >= 0) {
  area = radius * radius * Math.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.

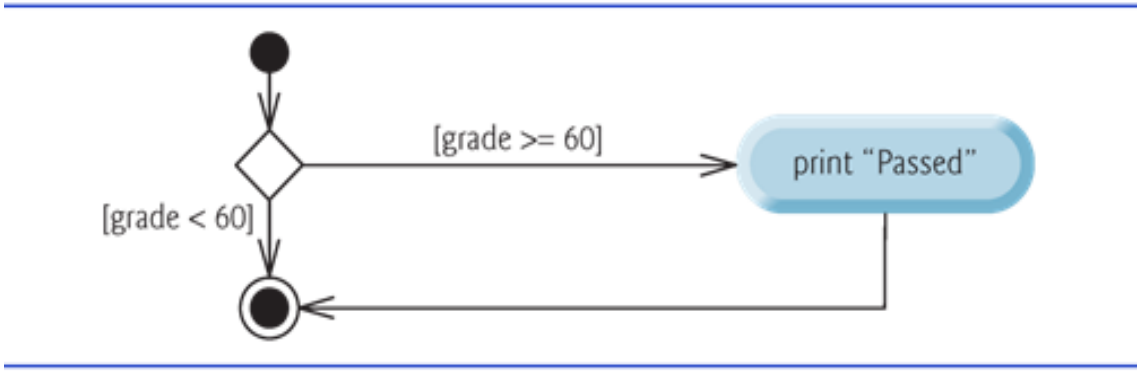
One-way `if` Statements

- Pseudocode:

  If student's grade is greater than or equal to 60
  Print "Passed"

- In Java:

```
if (studentGrade >= 60)
   System.out.println("Passed");
```





Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

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)



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.

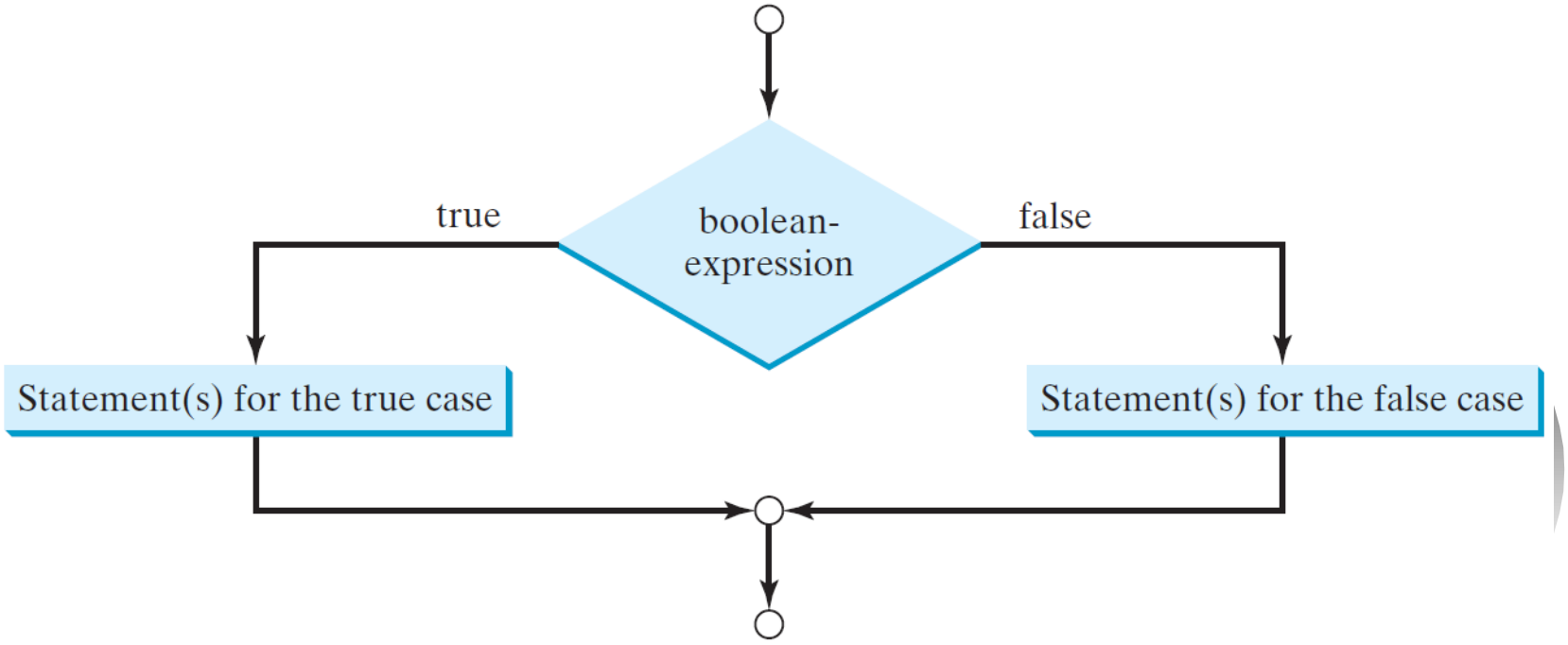
SimpleIfDemo Run



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

`if-else` Double-Selection Statement

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



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

`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("Invalid input");
}
```



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

`if-else` Example

- Pseudocode:

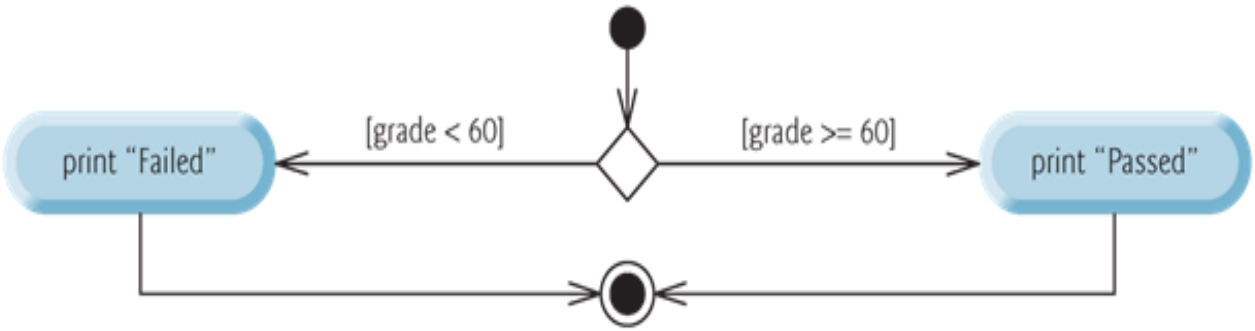
  If student's grade is greater than or equal to 60
  Print "Passed"
  Else
  Print "Failed"

- In Java:

```
if (grade >= 60)

System.out.println("Passed");
else

System.
```



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Multiple Alternative `if` Statements

```
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

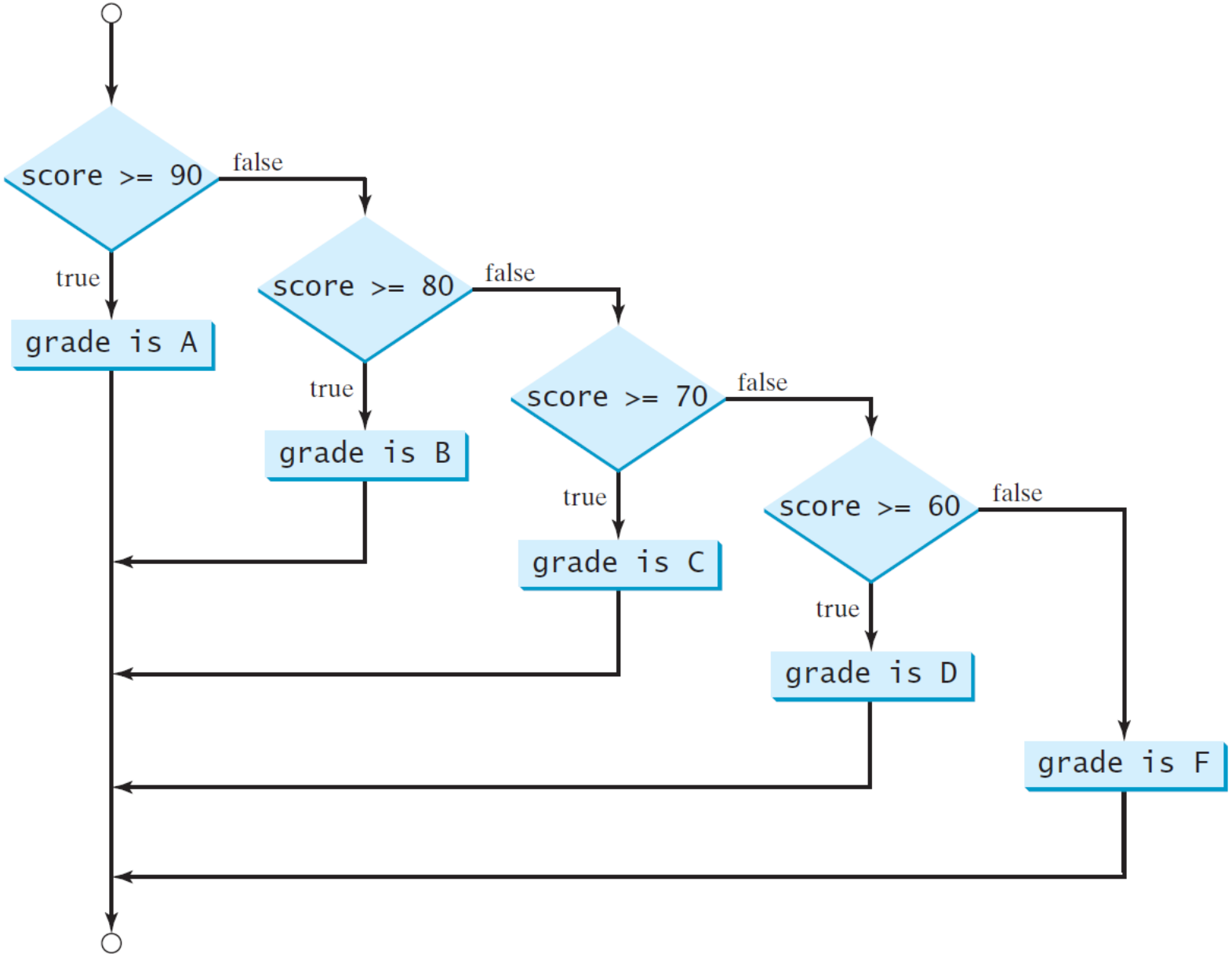
This is better

```
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");
```

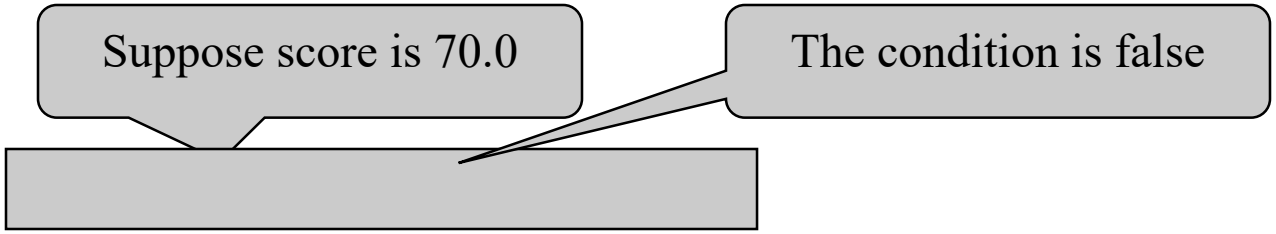
(b)

Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Multi-Way `if-else` Statements



Trace `if-else` statement




```
  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");
```



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Trace `if-else` statement

Suppose score is 70.0

The condition is false

```
if (score >= 90.0)
  System.out.print("A");
  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");
```



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Trace `if-else` statement

Suppose score is 70.0

The condition is true

```
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");
```



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

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)

else if (score >= 60.0)
  System.out.print("D");
else
  System.out.print("F");
```



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");
```



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Note

The `else` clause matches the most recent `if` 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)

Nothing will be printed!



Note

To force the `else` clause to match the first `if` 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");
```

B will be printed



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

TIP

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

(a)

Equivalent

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

(b)

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

(c)

Equivalent

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

(d)

Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Conditional operator (?:)

- Conditional operator (`?:`)—shorthand `if...else`.
- Ternary operator (takes three operands)

`boolean` expression `?` value if `true` `:` value if `false`

Example:

```
String result = studentGrade >= 60 ?
  "Passed" : "Failed";
  System.out.println(result);
```

Or simply:

```
System.out.println(studentGrade >= 60 ?
"Passed" : "Failed");
```



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Conditional operator (?:)

Exercise 1:

What is the value of s after executing the following statement?

```
String s = (6 > 6) ? "P" : "W";
```

W

Exercise 2:

Use the ternary operator `?:` instead of the following `if` statement:

```
if (a >= 0)
   b = a;
else
   b = -a;
```

```
b = (a >= 0) ? a : -a;
```



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Testing two `String`s for equality

- Method `equals` tests any two `String`s for equality
  - The method returns `true` if the contents of the strings are equal, and `false` otherwise. It uses a lexicographical comparison.
  - Example:

```
String word1 = "Yes", word2 = "No";
if (word1.equals("Yes"))
```

returns true

- To check if the contents of 2 strings are different, the negation operator `!` should be used:
  - `if (! word1.equals(word2))`

returns true

- The equality operator `==` is used for checking if 2 `String` variables refer to the same `String` object or not.



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Types of errors

- **Compilation (Syntax) errors** are caught by the compiler.
  - Example: when one brace in a block is left out
- A **logic error** has its effect at execution time.
  - Example: when both braces in a block are left out.
  - A **fatal logic error** causes a program to fail and terminate immediately.
  - A **nonfatal logic error** allows a program to continue executing but causes it to produce incorrect results.



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Types of errors

- Examples for compilation errors:
  - `int 5a;`
  - `double b; c;`
  - `int d = 6.7; int e = "3"; String s = 8;`
  - `int f; int g = f;`
  - `int i = 1; System.out.println("i = ", i);`
  - `int j = 2; System.out.println("j = %d", j);`
- Examples for fatal logic errors:
  - `int h = 10 / 0;`
  - `int k = 3; System.out.printf("k = %f\n", k);`
  - `int m = 4; System.out.printf("m = %d\n" + m);`
- Examples for nonfatal logic errors:
  - `int n = 5; System.out.println("n = %d" + n);`
  - `int p = 6; System.out.printf("p = \n", p);`



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

`If-else` Double-Selection

- It is possible to have an empty statement by placing a semicolon (`;`) where a statement would normally be.
- Example:

```
if ( grade >= 60 )
   System.out.println("Passed");
else
   ;
```



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Common errors

- **Example 1:**

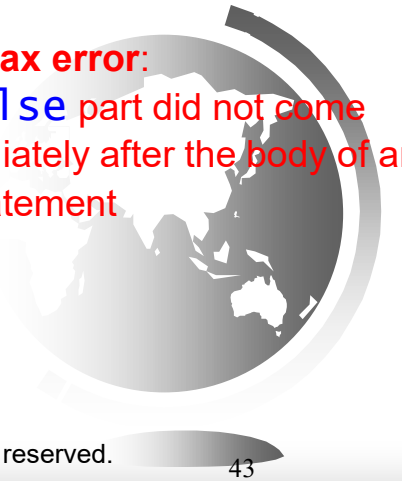
```
if ( grade >= 60 );
   System.out.println("Passed");
```

A **nonfatal logic error**: `Passed` will be printed in all cases

- **Example 2:**

```
if ( grade >= 60 );
   System.out.println("Passed");
else
   System.out.println("Failed");
```

A **syntax error**: The `else` part did not come immediately after the body of an `if` statement



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Logical Operators

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



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Logical Operators

- Here is an example:

```
if (score < 0 || score > 100)
   System.out.println("Invalid score");
else
   if (score < 60)
      System.out.println("Failed");
   else
      System.out.println("Passed");
```

If you enter any score less then 0 or greater than 100, you will get the following output:

```
Invalid score
```



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

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.

Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

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.

Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Truth Table for Operator ||

p_1	p_2	p_1 || p_2	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	

Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

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	

Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

`switch` Multiple-Selection Statement

- `switch` **multiple-selection statement** performs different actions based on the possible values of a **constant integral expression** of type `byte`, `short`, `int` or `char`.
- The `switch` statement consists of a block that contains a sequence of `case` **labels** and an optional `default` **case.**
- The program evaluates the **controlling expression** in the parentheses following keyword `switch`.
- The program compares the controlling expression's value with each `case` label.
- If a match occurs, the program executes that `case`'s statements.
- The `break` **statement** causes program control to proceed with the first statement after the `switch`.



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

```java
/* SwitchCase class uses switch statement to print the name
 * of the day according to a day number entered by the user */
import java.util.Scanner;
public class SwitchCase {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      System.out.print("Enter a day number (between 1 and 7): ");
      int day = input.nextInt();
      switch (day) {
         case 1: System.out.println("Saturday"); break;
         case 2: System.out.println("Sunday"); break;
         case 3: System.out.println("Monday"); break;
         case 4: System.out.println("Tuesday"); break;
         case 5: System.out.println("Wednesday"); break;
         case 6: System.out.println("Thursday"); break;
         case 7: System.out.println("Friday"); break;
         default: System.out.println("Invalid day number!");
      } // end switch
   } // end main method
} // end class SwitchCase
```



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

`switch` Multiple-Selection Statement

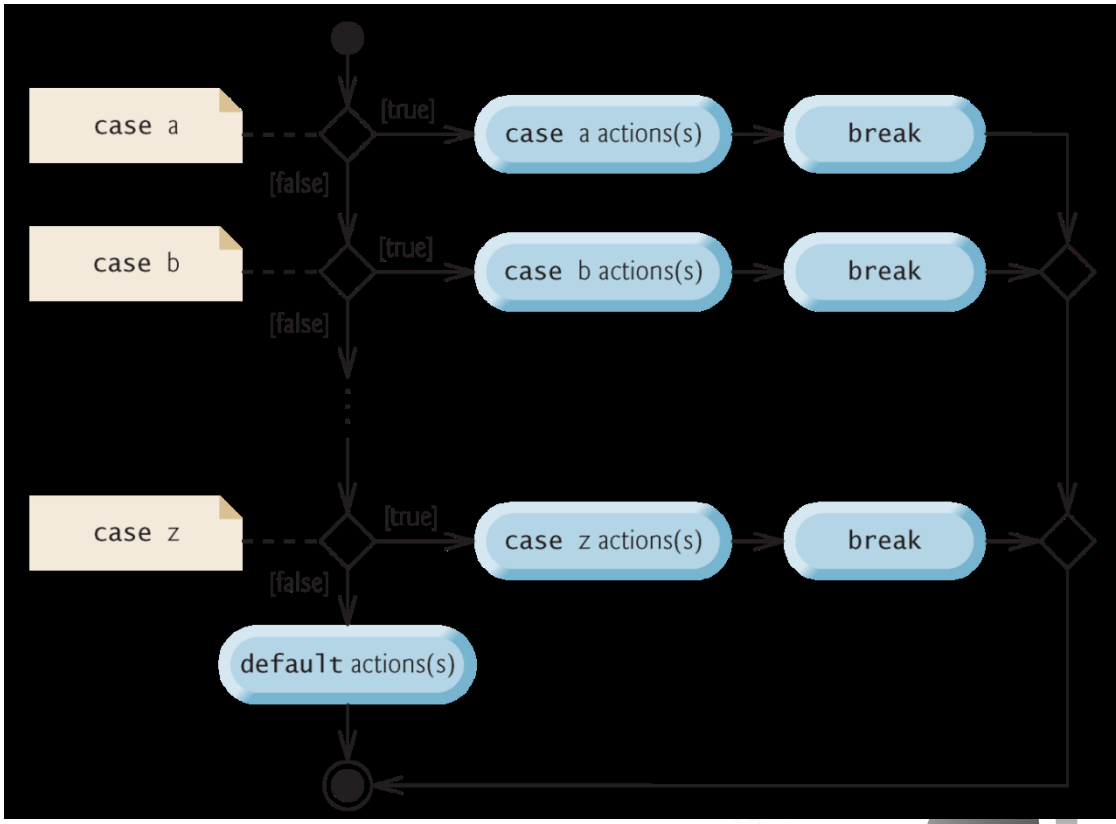
- `switch` does not provide a mechanism for testing ranges of values.
- Each `case` can have multiple statements.
- `switch` does not require braces around multiple statements in a `case`.
- Without `break`, the statements for a matching case and subsequent cases execute until a `break` or the end of the `switch` is encountered. This is called "falling through."
- If no match occurs between the controlling expression's value and a `case` label, the `default` case executes.
- If no match occurs and there is no `default` case, program control simply continues with the first statement after the `switch`.



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

`switch` Multiple-Selection Statement

- The figure shows the UML activity diagram for the general `switch` statement.
- Most `switch` statements use a `break` in each `case` to terminate the `switch` statement after processing the `case`.
- The `break` statement is not required for the `switch`'s last `case` (or the optional `default` case, when it appears last), because execution continues with the next statement after the `switch`.



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

`switch` Multiple-Selection Statement

Exercise 1: Find the output of the following code:

```
int i = 8;
switch (i%2) {
   case 0: System.out.printf("%d is ", i);
           System.out.printf("even\n");
           break;
   case 1: System.out.printf("%d is ", i);
           System.out.printf("odd\n");
}
System.out.println("See you!");
```

8 is even

See you!



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

`switch` Multiple-Selection Statement

Exercise 2:

Find the output of the following code:

```
int m = 7;
switch(m / 3) {
   case 1: System.out.println("ONE");
   case 2: System.out.println("TWO");
   case 3: System.out.println("THREE");
}
```

TWO

THREE



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

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)



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Operator Precedence and Associativity

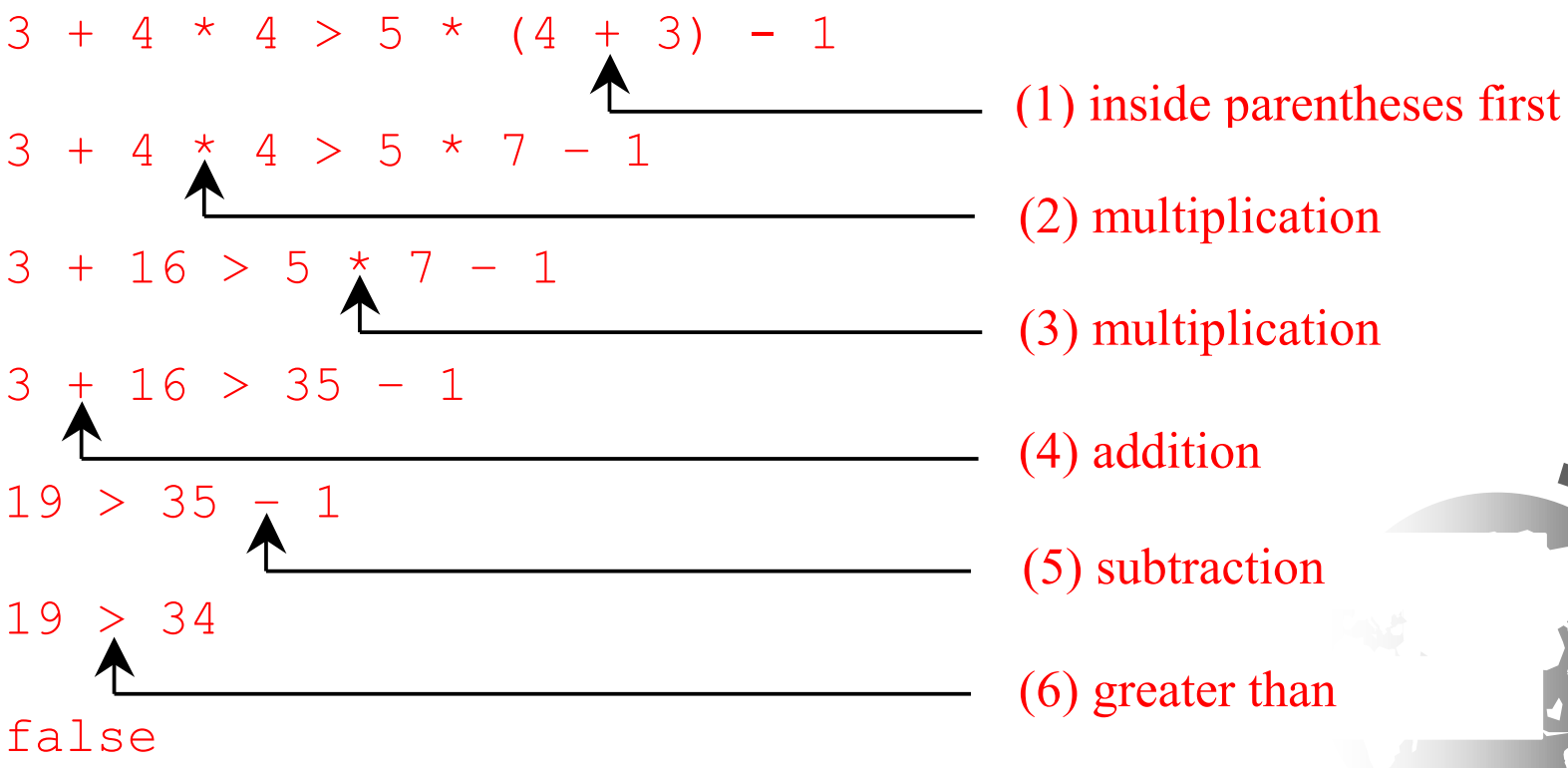
- The expression in the parentheses is evaluated first.
  - Parentheses can be nested, in which case the expression in the inner parentheses is executed first.
- When evaluating an expression without parentheses, the operators are applied according to the precedence rule and the associativity rule.
-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



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Example

Apply the operator precedence and associativity rule to find the value of: **`3 + 4 * 4 > 5 * (4 + 3) - 1`**



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

Thanks!



Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.