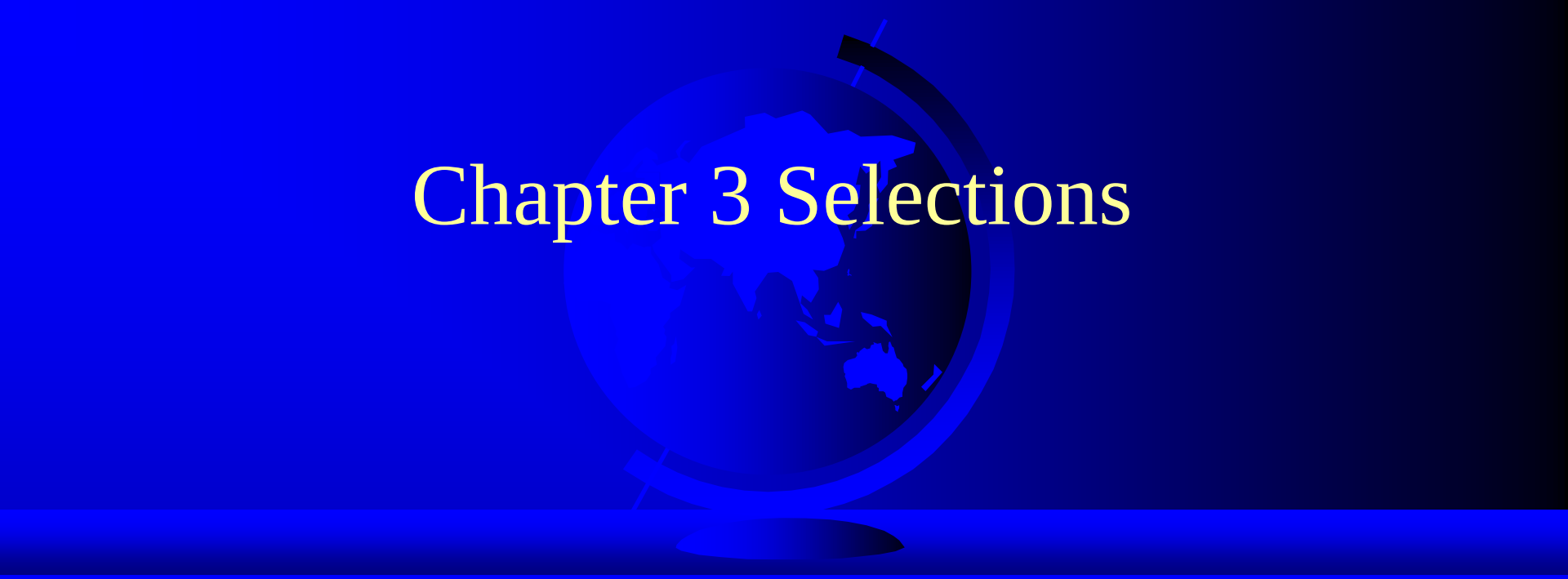
# Chapter 3 Selections



# Objectives

To declare boolean type and write Boolean expressions using comparison operators

- –To program AdditionQuiz using Boolean expressions

To implement selection control using one-way if statements

- –To program the GuessBirthday game using one-way if statements .
- –To implement selection control using two-way if statements).
- –To implement selection control using nested if statements
- –To avoid common errors in if statements
- –To program using selection statements for a variety of examples (BMI, ComputeTax, SubtractionQuiz)
- –To generate random numbers using the Math.random() method

To combine conditions using logical operators (&&, ||, and !)

- –To program using selection statements with combined conditions (LeapYear, Lottery).

To implement selection control using switch statements

- –To write expressions using the conditional operator

To format output using the System.out.printf method and to format strings using the String.format method

- –To examine the rules governing operator precedence and associativity

(GUI) To get user confirmation using confirmation

# The `boolean` Type and Operators

Compare two values: such as whether i is greater than j. Java provides six comparison operators (also known as relational operators)

The result of the comparison is a Boolean value: true or false.

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

# Comparison Operators

| *Operator* | *Name* |
|---|---|
| < | less than |
| <= | less than or equal to |
| > | greater than |
| >= | greater than or equal to |
| == | equal to |
| != | not equal to |

- Comparing characters is the same as comparing their Unicodes.
  - For example, '**a**' is larger than '**A**'

# ASCII Character Set

TABLE B.2 ASCII Character Set in the Hexadecimal Index

| | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *A* | *B* | *C* | *D* | *E* | *F* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | nul | soh | stx | etx | eot | enq | ack | bel | bs | ht | nl | vt | ff | cr | so | si |
| 1 | dle | dcl | dc2 | dc3 | dc4 | nak | syn | etb | can | em | sub | esc | fs | gs | rs | us |
| 2 | sp | ! | " | # | $ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4 | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 5 | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6 | ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 7 | p | q | r | s | t | u | v | w | x | y | z | { | \| | } | ~ | del |

# Problem: AdditionQuiz

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.
  - There are several ways to generate random numbers. For now, generate the first integer : System.currentTimeMillis() % 10
    and the second: System.currentTimeMillis()* 7 % 10.

```
What is 1 + 7? 8 ↵Enter
1 + 7 = 8 is true
```

```
What is 4 + 8? 9 ↵Enter
4 + 8 = 9 is false
```

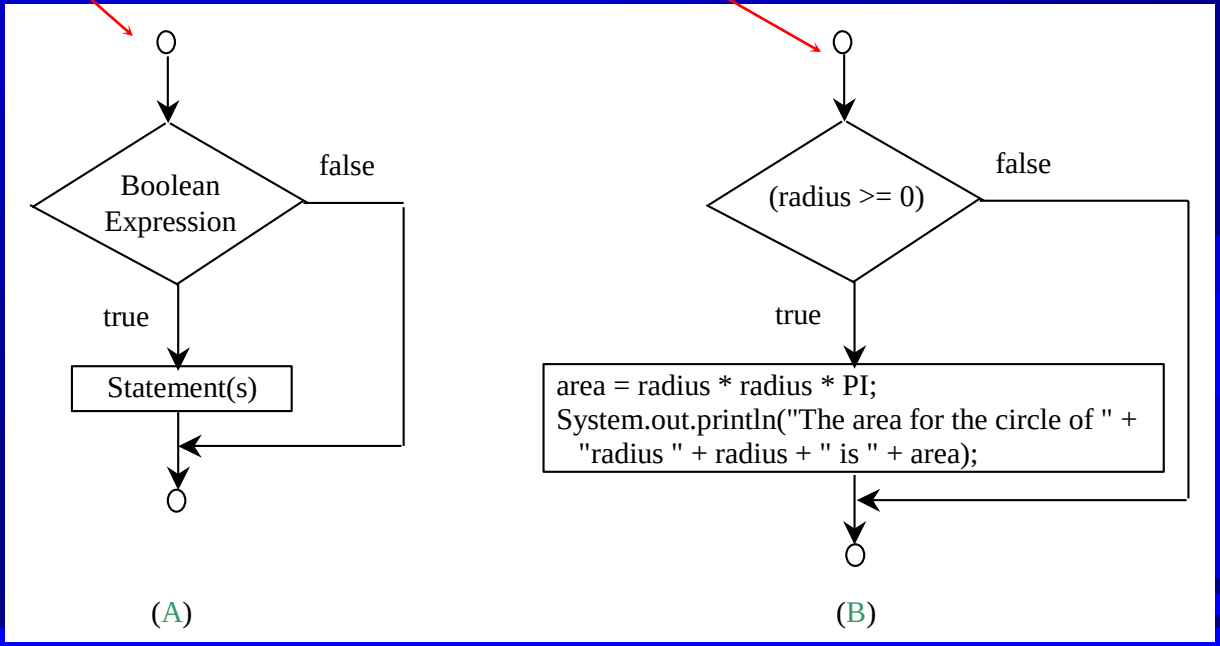
## LISTING 3.1 AdditionQuiz.java

```
 1 import java.util.Scanner;
 2
 3 public class AdditionQuiz {
 4   public static void main(String[] args) {
 5     int number1 = (int)(System.currentTimeMillis() % 10);          generate number1
 6     int number2 = (int)(System.currentTimeMillis() * 7 % 10);      generate number2
 7
 8     // Create a Scanner
 9     Scanner input = new Scanner(System.in);
10
11     System.out.print(                                              show question
12       "What is " + number1 + " + " + number2 + "? ");
13
14     int answer = input.nextInt();
15
16     System.out.println(                                            display result
17       number1 + " + " + number2 + " = " + answer + " is " +
18       (number1 + number2 == answer));
19   }
20 }
```

# 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);
}
```



# 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)

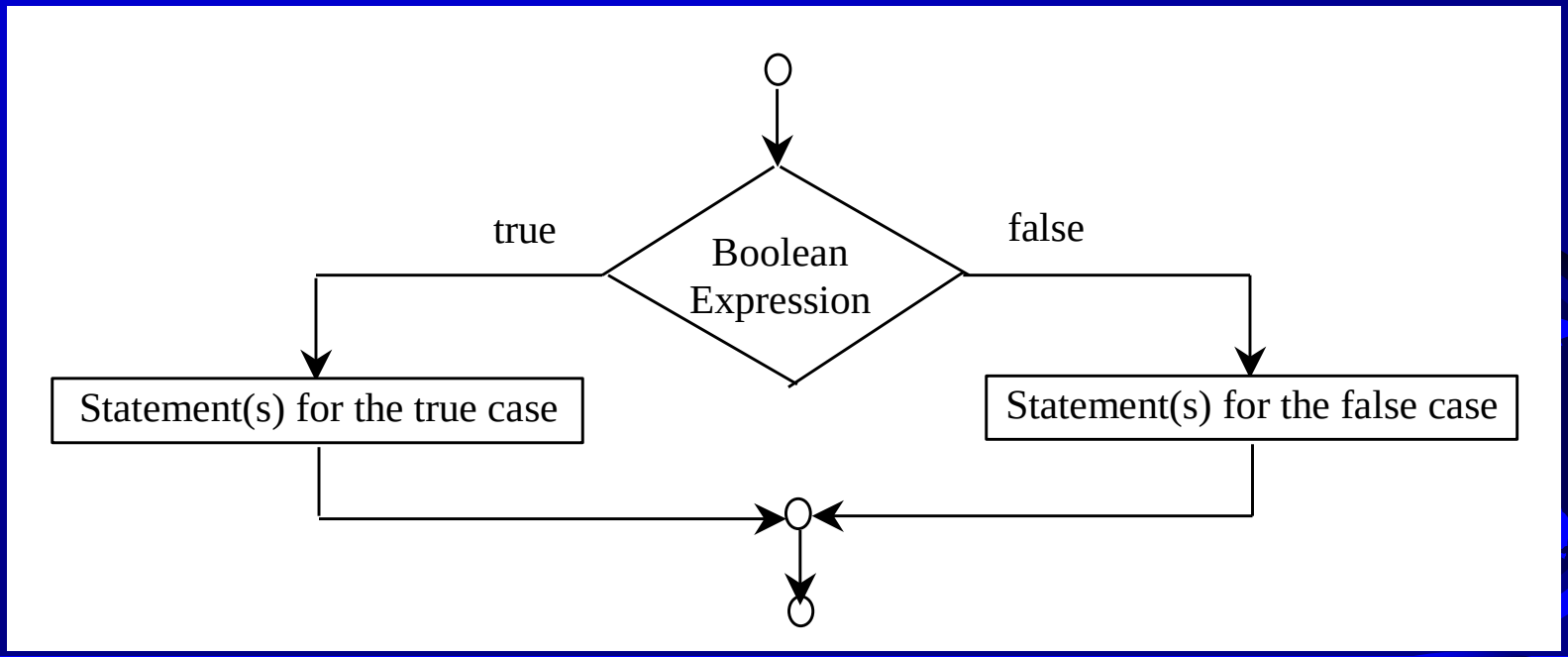
Equivalent

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

(b)

# The Two-way `if` Statement

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



# Multiple Alternative if Statements

```
if (score >= 90.0)
  grade = 'A';
else
  if (score >= 80.0)
    grade = 'B';
  else
    if (score >= 70.0)
      grade = 'C';
    else
      if (score >= 60.0)
        grade = 'D';
      else
        grade = 'F';
```

Equivalent

```
if (score >= 90.0)
  grade = 'A';
else if (score >= 80.0)
  grade = 'B';
else if (score >= 70.0)
  grade = 'C';
else if (score >= 60.0)
  grade = 'D';
else
  grade = 'F';
```

animation

# Trace if-else statement

Suppose score is 70.0

```
if (score >= 90.0)
  grade = 'A';
else if (score >= 80.0)
  grade = 'B';
else if (score >= 70.0)
  grade = 'C';
else if (score >= 60.0)
  grade = 'D';
else
  grade = 'F';
```

Exit the if statement
grade=?

# Note

The else clause matches the most recent if clause in the same block.

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

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

(a)

Equivalent

```
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)

- Nothing is printed.
- To force the else clause to match the first if clause, you must add a pair of braces.

# Note, cont.

```
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 ?

# Common Errors

Adding a semicolon at the end of an if clause

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

Wrong

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.

# 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: SubstractionQuiz

Teach a first grade child how to learn subtractions. The program randomly generates two single-digit integers number1 and number2 and displays a question such as "What is 9 – 2?". After the student types the answer, the program indicate whether the answer is correct.

To generate random numbers:

- use the random() method in the Math class, returns a random double value d such that $0.0 \leq d < 1.0$
- (int) (Math.random()*10) : a random integer between 0 and 9

If number1 < number2, swap number1 with number2.

```
What is 6 - 6? 0 Enter
You are correct!
```

```
What is 9 - 2? 5 Enter
Your answer is wrong
9 - 2 should be 7
```

## LISTING 3.4 SubtractionQuiz.java

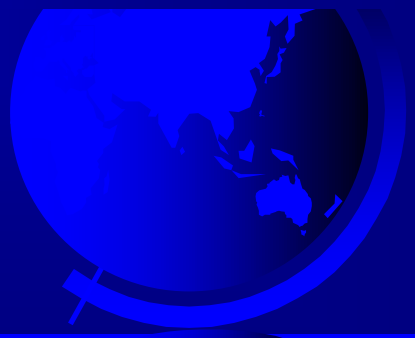
```
1 import java.util.Scanner;
2
3 public class SubtractionQuiz {
4   public static void main(String[] args) {
5     // 1. Generate two random single-digit integers
6     int number1 = (int)(Math.random() * 10);                    random numbers
7     int number2 = (int)(Math.random() * 10);
8
9     // 2. If number1 < number2, swap number1 with number2
10    if (number1 < number2) {
11      int temp = number1;
12      number1 = number2;
13      number2 = temp;
14    }
15
16    // 3. Prompt the student to answer "What is number1 - number2?"
17    System.out.print
18      ("What is " + number1 + " - " + number2 + "? ");
19    Scanner input = new Scanner(System.in);
20    int answer = input.nextInt();                               get answer
21
22    // 4. Grade the answer and display the result
23    if (number1 - number2 == answer)                            check the answer
24      System.out.println("You are correct!");
25    else
26      System.out.println("Your answer is wrong\n" + number1 + " - "
27        + number2 + " should be " + (number1 - number2));
28  }
29 }
```

# Logical Operators

| *Operator* | *Name* |
|---|---|
| ! | not |
| && | and |
| \|\| | or |
| ^ | exclusive or |

# Truth Table for Operator ^

| p1 | p2 | p1 ^ p2 | Example (assume age = 24, gender = 'F') |
|---|---|---|---|
| false | false | false | (age > 34) ^ (gender == 'F') is true, because (age > 34) is false but (gender == 'F') is true. |
| false | true | true | |
| true | false | true | (age > 34) \|\| (gender == 'M') is false, because (age > 34) and (gender == 'M') are both false. |
| true | true | false | |



# Examples

Here is a program that checks whether a number is divisible by 2 and 3, whether a number is divisible by 2 or 3, and whether a number is divisible by 2 or 3 but not both:

```
Enter an integer: 18 ↵Enter
Is 18
  divisible by 2 and 3? true
  divisible by 2 or 3? true
  divisible by 2 or 3, but not both? false
```

## LISTING 3.7 TestBooleanOperators.java

```
 1 import java.util.Scanner;                                          import class
 2
 3 public class TestBooleanOperators {
 4   public static void main(String[] args) {
 5     // Create a Scanner
 6     Scanner input = new Scanner(System.in);
 7
 8     // Receive an input
 9     System.out.print("Enter an integer: ");
10     int number = input.nextInt();                                  input
11
12     System.out.println("Is " + number +
13       "\n\tdivisible by 2 and 3? " +
14       (number % 2 == 0 && number % 3 == 0 )                        and
15       + "\n\tdivisible by 2 or 3? " +
16       (number % 2 == 0 || number % 3 == 0 ) +                      or
17       "\n\tdivisible by 2 or 3, but not both? "
18       + (number % 2 == 0 ^ number % 3 == 0 ));                     exclusive or
19   }
20 }
```

# 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.

- the boolean expression?

# 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.

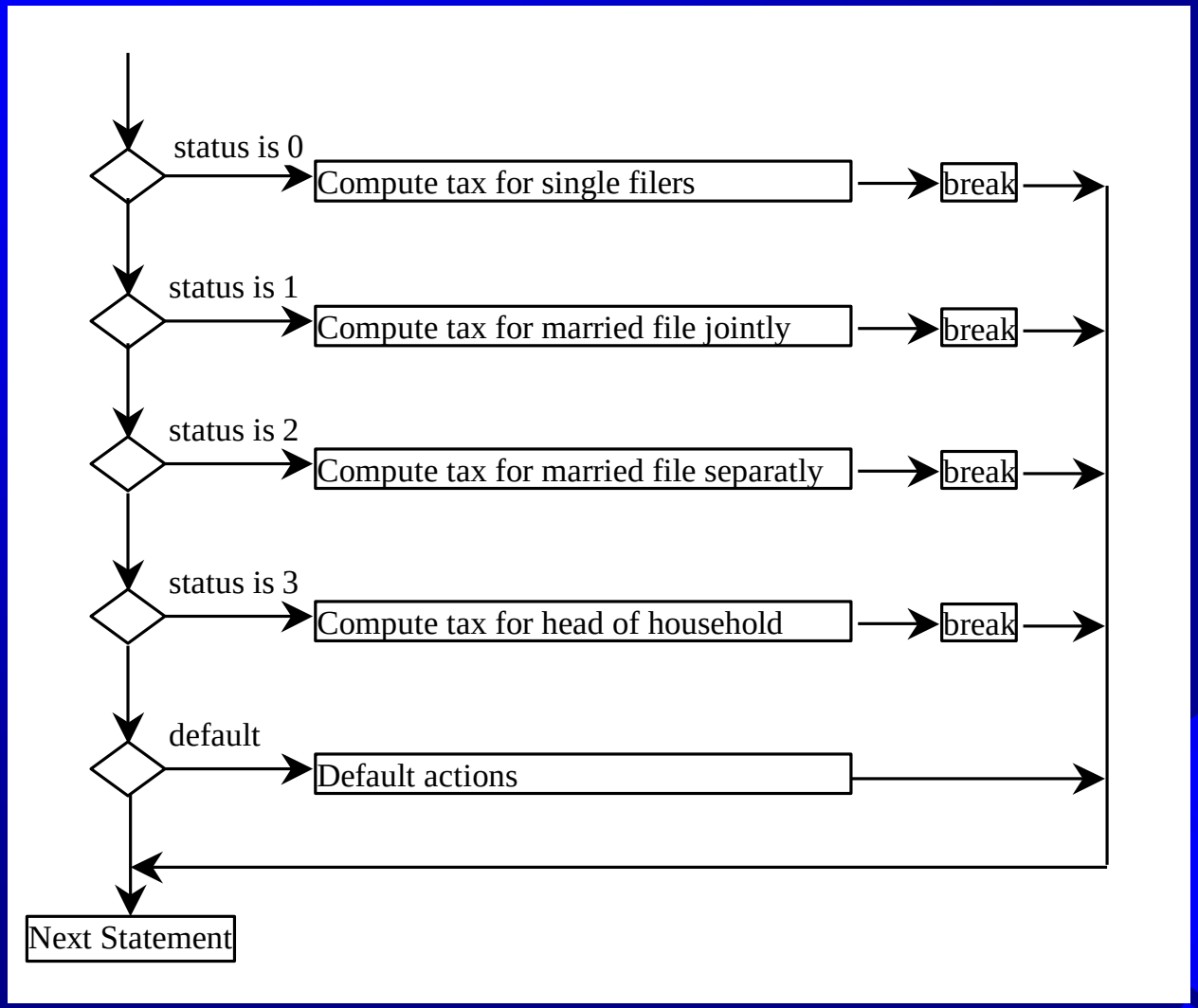
A year is a leap year if it is divisible by 4 but not by 100, or it is divisible by 400.

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

# `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(0);
}
```

# `switch` Statement Flow Chart



# `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;
}
```

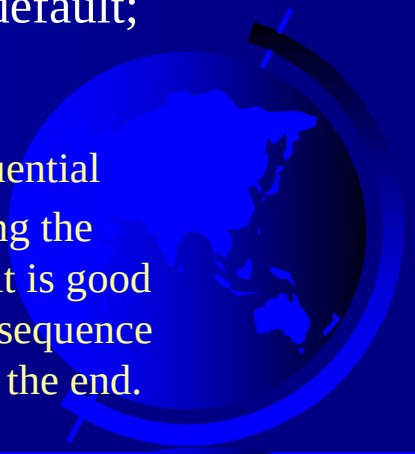
# `switch` Statement Rules

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;
}
```

The case statements are executed in sequential order, but the order of the cases (including the default case) does not matter. However, it is good programming style to follow the logical sequence of the cases and place the default case at the end.



animation

# Trace switch statement

Suppose ch is 'a':

```
switch (ch) {
  case 'a': System.out.println(ch);
  case 'b': System.out.println(ch);
  case 'c': System.out.println(ch);
}
```

animation

# Trace switch statement

ch is 'a':

```
switch (ch) {
  case 'a': System.out.println(ch);
  case 'b': System.out.println(ch);
  case 'c': System.out.println(ch);
}
```

animation

# Trace switch statement

Execute this line

```
switch (ch) {
  case 'a': System.out.println(ch);
  case 'b': System.out.println(ch);
  case 'c': System.out.println(ch);
}
```

animation

# Trace switch statement

Execute this line

```
switch (ch) {
   case 'a': System.out.println(ch);
   case 'b': System.out.println(ch);
   case 'c': System.out.println(ch);
}
```

animation

# Trace switch statement

Execute this line

```
switch (ch) {
  case 'a': System.out.println(ch);
  case 'b': System.out.println(ch);
  case 'c': System.out.println(ch);
}
```

animation

# Trace switch statement

Execute next statement

```
switch (ch)
  case 'a': System.out.println(ch);
  case 'b': System.out.println(ch);
  case 'c': System.out.println(ch);
}
```

Next statement;

animation

# Trace switch statement

Suppose ch is 'a':

```
switch (ch) {
  case 'a': System.out.println(ch);
            break;
  case 'b': System.out.println(ch);
            break;
  case 'c': System.out.println(ch);
}
```

animation

# Trace switch statement

ch is 'a':

```
switch (ch) {
  case 'a': System.out.println(ch);
            break;
  case 'b': System.out.println(ch);
            break;
  case 'c': System.out.println(ch);
}
```

animation

# Trace switch statement

Execute this line

```
switch (ch) {
  case 'a': System.out.println(ch);
            break;
  case 'b': System.out.println(ch);
            break;
  case 'c': System.out.println(ch);
}
```

animation

# Trace switch statement

Execute this line

```
switch (ch) {
  case 'a': System.out.println(ch);
            break;
  case 'b': System.out.println(ch);
            break;
  case 'c': System.out.println(ch);
}
```

animation

# Trace switch statement

Execute next statement

```
switch (ch)
  case 'a': System.out.println(ch);
            break;
  case 'b': System.out.println(ch);
            break;
  case 'c': System.out.println(ch);
}
```

Next statement;

# Conditional Operator

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

is equivalent to

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

(boolean-expression) ? expression1 : expression2

Ternary operator
Binary operator
Unary operator

# 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 and Associativity

**TABLE 3.10** Operator Precedence Chart

| *Precedence* | *Operator* |
|---|---|
| ↓ | `var++` and `var--` (Postfix) |
| | `+`, `-` (Unary plus and minus), `++var` and `--var` (Prefix) |
| | (type) (Casting) |
| | `!` (Not) |
| | `*`, `/`, `%` (Multiplication, division, and remainder) |
| | `+`, `-` (Binary addition and subtraction) |
| | `<`, `<=`, `>`, `>=` (Comparison) |
| | `==`, `!=` (Equality) |
| | `^` (Exclusive OR) |
| | `&&` (AND) |
| | `\|\|` (OR) |
| | `=`, `+=`, `-=`, `*=`, `/=`, `%=` (Assignment operator) |

# Operator Associativity

All binary operators except assignment operators are *left-associative*.

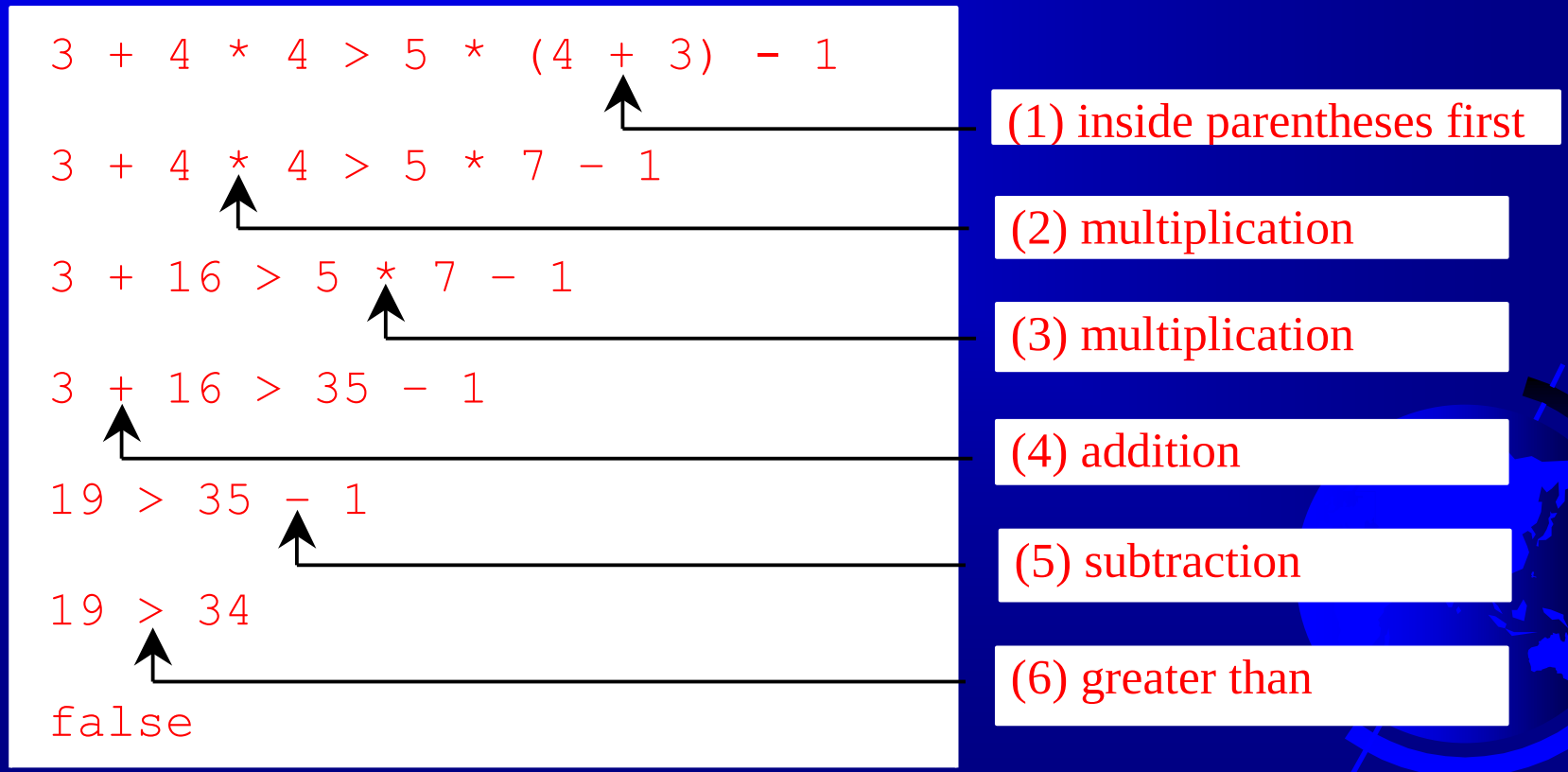
a – b + c – d is equivalent to ((a – b) + c) – d

Assignment operators are *right-associative*.

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

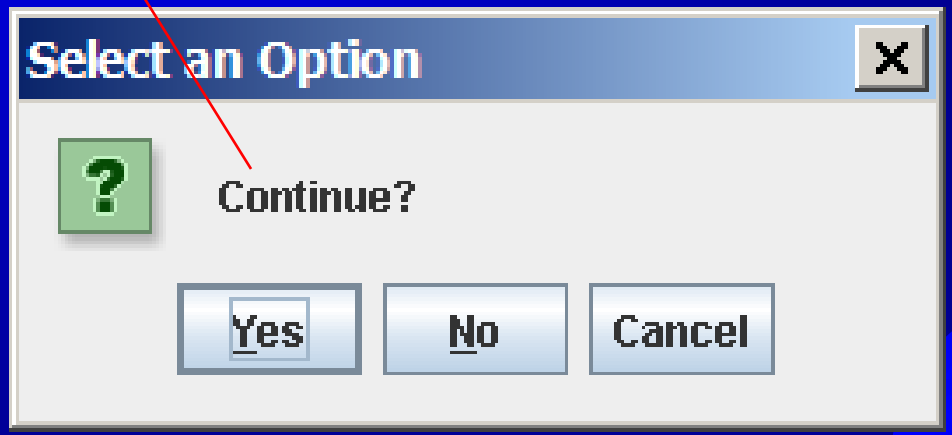
# Example

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



# (GUI) Confirmation Dialogs

```
int option = JOptionPane.showConfirmDialog
    (null, "Continue");
```



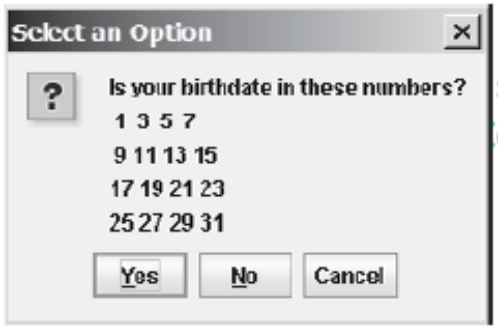

Return value：

JOptionPane.YES_OPTION (0) for the *Yes* button, JOptionPane.NO_OPTION (1) for the *No* button, and JOptionPane.CANCEL_OPTION (2) for the *Cancel* button.

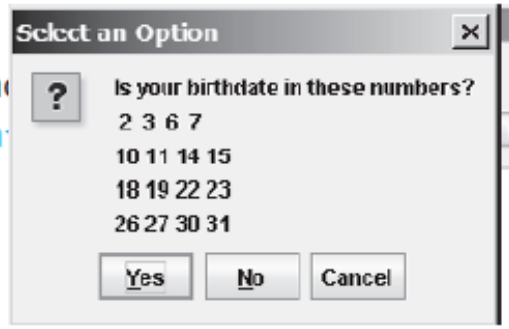
# Example

input using 5 Confirm dialogbox

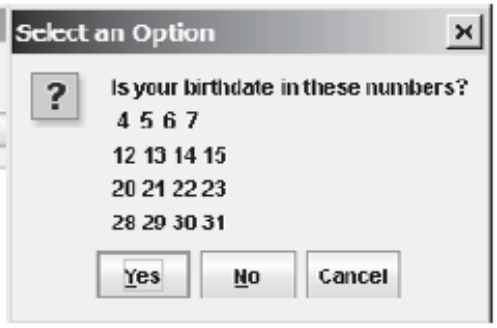
output using 1 Message dialogbox



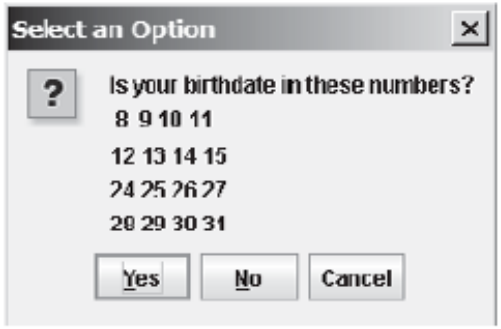

(a)



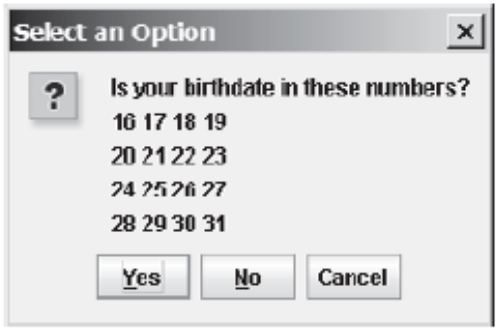

(b)




(c)




(d)




(e)


(f)