CS 501 – Introduction to JAVA Programing

Lecture 2 – Elementary Programming, Selections

STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

Elementary Programming

To learn

1. how to solve practical problems programmatically.
2. to learn Java primitive data types and related subjects, such as variables, constants, data types, operators, expressions, and input and output.

Example – Area of Circle


```
1   public class ComputeArea {
2       /** Main method */
3       public static void main(String[] args) {
4           double radius;
5           double area;
6
7           // Assign a radius
8           radius = 20;
9
10          // Compute area
11          area = radius * radius * 3.14159;
12
13          // Display results
14          System.out.println("The area for the circle of radius " + radius + " is " + area);
15      }
16  }
17
```

Example – Area of Circle

```
1   public class ComputeArea {
2       /** Main method */
3       public static void main(String[] args) {
4           double radius;
5           double area;
6
7           // Assign a radius
8           radius = 20;
9
10          // Compute area
11          area = radius * radius * 3.14159;
12
13          // Display results
14          System.out.println("The area for the circle of radius " + radius + " is " + area);
15      }
16  }
17
```

Allocate memory for radius and area.

- No values

Example – Area of Circle

```
1   public class ComputeArea {
2       /** Main method */
3       public static void main(String[] args) {
4           double radius;
5           double area;
6
7           // Assign a radius
8           radius = 20;
9
10          // Compute area
11          area = radius * radius * 3.14159;
12
13          // Display results
14          System.out.println("The area for the circle of radius " + radius + " is " + area);
15      }
16  }
17
```

Assign 20 to radius:

- Area: No values

Example – Area of Circle

```
1   public class ComputeArea {
2       /** Main method */
3       public static void main(String[] args) {
4           double radius;
5           double area;
6
7           // Assign a radius
8           radius = 20;
9
10          // Compute area
11          area = radius * radius * 3.14159;
12
13          // Display results
14          System.out.println("The area for the circle of radius " + radius + " is " + area);
15      }
16  }
17
```

Computes the area using assigned radius

Example – Reading Input from the Console

```
1   import java.util.Scanner;
2
3   public class ComputeAreaWithConsoleInput {
        Run | Debug
4       public static void main(String[] args) {
5           try (Scanner input = new Scanner(System.in)) {
6               System.out.print(s: "Enter a number for radius: ");
7               double radius = input.nextDouble();
8               System.out.println("The radius of the circle is " + radius);
9               double area = radius * radius * 3.14159;
10              System.out.println("The area for the circle of radius " +
11                      radius + " is " + area);
12          }
13
14      }
15  }
16
```

Explicit Import.
Alternatively, it can be replaced with java.util.*; .

Example – Reading Input from the Console

```
1   import java.util.Scanner;
2
3   public class ComputeAreaWithConsoleInput {
        Run | Debug
4       public static void main(String[] args) {
5           try (Scanner input = new Scanner(System.in)) {
6               System.out.print(s: "Enter a number for radius: ");
7               double radius = input.nextDouble();
8               System.out.println("The radius of the circle is " + radius);
9               double area = radius * radius * 3.14159;
10              System.out.println("The area for the circle of radius " +
11                      radius + " is " + area);
12          }
13
14      }
15  }
16
```

Create a Scanner object

Example – Reading Input from the Console

```
1   import java.util.Scanner;
2
3   public class ComputeAreaWithConsoleInput {
        Run | Debug
4       public static void main(String[] args) {
5           try (Scanner input = new Scanner(System.in)) {
6               System.out.print(s: "Enter a number for radius: ");
7               double radius = input.nextDouble();
8               System.out.println("The radius of the circle is " + radius);
9               double area = radius * radius * 3.14159;
10              System.out.println("The area for the circle of radius " +
11                      radius + " is " + area);
12          }
13
14      }
15  }
16
```

Use the method nextDouble() to obtain a double value.

Identifiers

- Identifiers are the names that identify the elements such as classes, methods, and variables in the program.
- An identifier is a sequence of characters that consist of letters, digits, underscores (_), and dollar signs ($).
- An identifier must start with a letter, an underscore (_), or a dollar sign ($). It cannot start with a digit.
- An identifier cannot be a reserved word. (See Appendix A, “Java Keywords,” for a list of reserved words).
- An identifier cannot be true, false, or null.
- An identifier can be of any length.
- Examples of illegal identifiers:
  - 2A, d+4

Variables

- Variables are used to represent values that may be changed in the program, e.g., radius, area
- Variable declaration tells the compiler to allocate appropriate memory space for the variable based on its data type.
- The syntax for variable declaration is in the order of datatype and the name. For example,

```
int x;           // Declare x to be an
                 // integer variable;
x = 1;           // Assign 1 to x;

double radius;   // Declare radius to
                 // be a double variable;
radius = 1.0;    // Assign 1.0 to radius;

char a;          // Declare a to be a
                 // character variable;
a = 'A';         // Assign 'A' to a;
```

```
int x = 1;
Int x = 5 + 2; // Assign the value for the
               // expression to x

double radius = 1.4;

char a = 'A';
```

Named Constants

- A named constant is an identifier that represents a permanent value.
- The named constant declaration can be done as

```
final datatype CONSTANTNAME = VALUE;

final double PI = 3.14159;
final int SIZE = 3;
```

```
3      public static void main(String[] args) {
4          final double PI = 3.14159;
5          double radius;
6          double area;
7
8          // Assign a radius
9          radius = 20;
10
11         // Compute area
12         area = radius * radius * PI;
13
```

Naming Conventions

- Choose meaningful and descriptive names.

- Variables and method names:
  - Use lowercase. If the name consists of several words, concatenate all in one, use lowercase for the first word, and capitalize the first letter of each subsequent word in the name. For example, the variables radius and area, and the method computeArea.

- Class names:
  - Capitalize the first letter of each word in the name. For example, the class name ComputeArea.

- Constants:
  - Capitalize all letters in constants, and use underscores to connect words. For example, the constant PI and MAX_VALUE

Numerical Data Types and Operations

Name	Range	Storage Size
`byte`	-2^7 to $2^7 - 1$ (-128 to 127)	8-bit signed
`short`	-2^{15} to $2^{15} - 1$ (-32768 to 32767)	16-bit signed
`int`	-2^{31} to $2^{31} - 1$ (-2147483648 to 2147483647)	32-bit signed
`long`	-2^{63} to $2^{63} - 1$ (i.e., -9223372036854775808 to 9223372036854775807)	64-bit signed
`float`	Negative range: -3.4028235E+38 to -1.4E-45 Positive range: 1.4E-45 to 3.4028235E+38	32-bit IEEE 754
`double`	Negative range: -1.7976931348623157E+308 to -4.9E-324 Positive range: 4.9E-324 to 1.7976931348623157E+308	64-bit IEEE 754

Numerical Data Types and Operations

```
Scanner input = new Scanner(System.in);
int value = input.nextInt();
```

Method	Description
`nextByte()`	reads an integer of the `byte` type.
`nextShort()`	reads an integer of the `short` type.
`nextInt()`	reads an integer of the `int` type.
`nextLong()`	reads an integer of the `long` type.
`nextFloat()`	reads a number of the `float` type.
`nextDouble()`	reads a number of the `double` type.

Numerical Data Types and Operations

Name	Meaning	Example	Result
+	Addition	34 + 1	35
-	Subtraction	34.0 - 0.1	33.9
*	Multiplication	300 * 30	9000
/	Division	1.0 / 2.0	0.5
%	Remainder	20 % 3	2

If both variables are integer, 1 / 2 = 0.

Example: % Operator

```
1   import java.util.*;
2
3   public class DisplayTime {
        Run | Debug
4       public static void main(String[] args){
5           Scanner input = new Scanner(System.in);
6
7           System.out.print(s: "Enter an integer for seconds: ");
8           int seconds = input.nextInt();
9
10          int minutes = seconds / 60;
11          int remainingSeconds = seconds % 60;
12          System.out.println(seconds + " seconds is " + minutes +
13                                  " minutes and " + remainingSeconds + "seconds");
14      }
15
16  }
```

Create a scanner

Example: % Operator

```
1   import java.util.*;
2
3   public class DisplayTime {
        Run | Debug
4       public static void main(String[] args){
5           Scanner input = new Scanner(System.in);
6
7           System.out.print(s: "Enter an integer for seconds: ");
8           int seconds = input.nextInt();
9
10          int minutes = seconds / 60;
11          int remainingSeconds = seconds % 60;
12          System.out.println(seconds + " seconds is " + minutes +
13                                  " minutes and " + remainingSeconds + "seconds");
14      }
15
16  }
```

Read an integer seconds
If input is 500, seconds=500

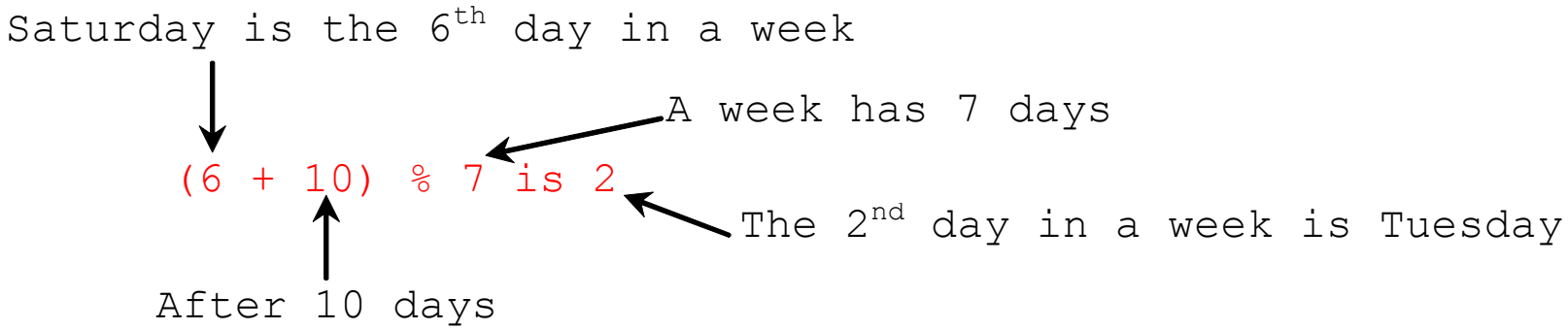
Example: % Operator

```
1   import java.util.*;
2
3   public class DisplayTime {
        Run | Debug
4       public static void main(String[] args){
5           Scanner input = new Scanner(System.in);
6
7           System.out.print(s: "Enter an integer for seconds: ");
8           int seconds = input.nextInt();
9
10          int minutes = seconds / 60;
11          int remainingSeconds = seconds % 60;
12          System.out.println(seconds + " seconds is " + minutes +
13                                  " minutes and " + remainingSeconds + "seconds");
14      }
15
16  }
```

500 / 60 = 8
500 % 60 = 20

Example: % Operator

Suppose today is Saturday and you and your friends are going to meet in 10 days. What day is in 10 days?



Exponent Operation

```
8           // Assign a radius
9           double radius = 20;
10
11          // Compute area
12          //area = radius * radius * PI;
13          area = Math.pow(radius, b: 2) * PI;
14          // Display results
15          System.out.println("The area for the circle of radius " + radius + " is " + area);
16      }
```

Math.pow(a,b) were a is the base and b is the exponent.

Literals

```
8           // Assign a radius
9           double radius = 20;
10
11          // Compute area
12          //area = radius * radius * PI;
13          area = Math.pow(radius, b: 2) * PI;
14          // Display results
15          System.out.println("The area for the circle of radius " + radius + " is " + area);
16      }
```

A literal is a constant value that appears directly in the program.

Literals - Integer

- An integer literal can be assigned to an integer variable as long as **it can fit into the variable**. A compilation error would occur if the literal were too large for the variable to hold.
  - The statement byte b = 1000 would cause a compilation error,
  - because 1000 cannot be stored in a variable of the byte type.
- An integer literal is assumed to be of the **int** type, whose value is between -2^{31} (-2147483648) to $2^{31}-1$ (2147483647).
  - To denote an integer literal of the long type, append it with the letter **L** or **l**. L is preferred because l (lowercase L) can easily be confused with 1 (the digit one).

Literals – Floating-Point

- Floating-point literals are written with a decimal point.
- By default, a floating-point literal is treated as a double type value.
  - 5.0 is considered a **double** value, not a float value.
- Can make a number a float by appending the letter **f** or **F**, and make a number a double by appending the letter **d** or **D**.
  - use 100.2f or 100.2F for a float number
  - 100.2d or 100.2D for a double number.

```
System.out.println("1.0 / 3.0 is " + 1.0 / 3.0);
```

displays `1.0 / 3.0 is 0.3333333333333333`

16 digits

```
System.out.println("1.0F / 3.0F is " + 1.0F / 3.0F);
```

displays `1.0F / 3.0F is 0.33333334`

7 digits

Scientific Notation

- Floating-point literals can also be specified in scientific notation.
- **E** (or **e**) represents an exponent and it can be either in lowercase or uppercase.
  - 1.23456e+2, same as 1.23456e2, is equivalent to 123.456.
  - 1.23456e-2 is equivalent to 0.0123456.

Augmented Assignment Operators

Operator	*Name*	*Example*	*Equivalent*
`+=`	Addition assignment	`i += 8`	`i = i + 8`
`-=`	Subtraction assignment	`i -= 8`	`i = i - 8`
`*=`	Multiplication assignment	`i *= 8`	`i = i * 8`
`/=`	Division assignment	`i /= 8`	`i = i / 8`
`%=`	Remainder assignment	`i %= 8`	`i = i % 8`

Increment and Decrement Operators

Operator	*Name*	*Description*	*Example (assume i = 1)*
`++var`	preincrement	Increment `var` by `1`, and use the new `var` value in the statement	`int j = ++i;` `// j is 2, i is 2`
`var++`	postincrement	Increment `var` by `1`, but use the original `var` value in the statement	`int j = i++;` `// j is 1, i is 2`
`--var`	predecrement	Decrement `var` by `1`, and use the new `var` value in the statement	`int j = --i;` `// j is 0, i is 0`
`var--`	postdecrement	Decrement `var` by `1`, and use the original `var` value in the statement	`int j = i--;` `// j is 1, i is 0`

```
int i = 10;
int newNum = 10 * i++;
```

Same effect as

```
int newNum = 10 * i;
i = i + 1;
```

```
int i = 10;
int newNum = 10 * (++i);
```

Same effect as

```
i = i + 1;
int newNum = 10 * i;
```

Numeric Type Conversion

When performing a binary operation involving two operands of different types, Java automatically converts the operand based on the following rules:

1. If one of the operands is **double**, the other is converted into **double**.
2. Otherwise, if one of the operands is **float**, the other is converted into **float**.
3. Otherwise, if one of the operands is **long**, the other is converted into **long**.
4. Otherwise, **both** operands are converted into **int**.

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.

Implicit casting

```
double d = 3;
```
(type widening)

Explicit casting

```
int i = (int)3.0;
```
(type narrowing)

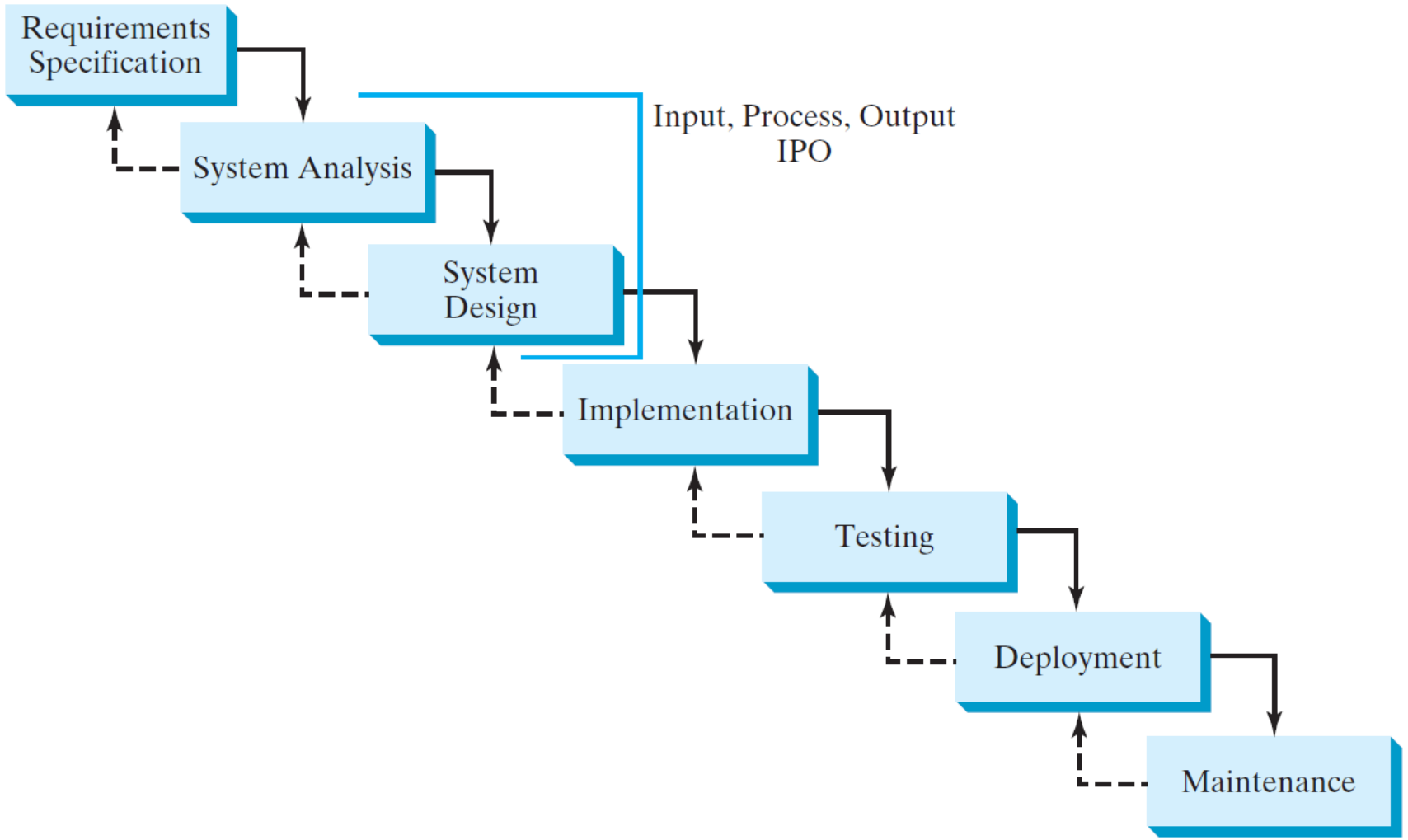
```
int i = (int)3.9;
```
(Fraction part is truncated)

What is wrong? int x = 5 / 2.0;

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

sum += 4.5 is equivalent to **sum = (int)(sum + 4.5)**.

Software Development Process



Selections

```
1   import java.util.Scanner;
2
3   public class ComputeAreaWithConsoleInput {
        Run | Debug
4       public static void main(String[] args) {
5           try (Scanner input = new Scanner(System.in)) {
6               System.out.print(s: "Enter a number for radius: ");
7               double radius = input.nextDouble();
8               System.out.println("The radius of the circle is " + radius);
9               double area = radius * radius * 3.14159;
10              System.out.println("The area for the circle of radius " +
11                      radius + " is " + area);
12          }
13
14      }
15  }
16
```

Selections


```
 8            if (radius >= 0) {
 9                System.out.println("The radius of the circle is " + radius);
10                double area = radius * radius * 3.14159;
11                System.out.println("The area for the circle of radius " +
12                        radius + " is " + area);
13            }
14            else {
15                System.out.println(x: "Incorrect radius");
16            }
```

boolean Data Type, Values, and Expressions

- 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.

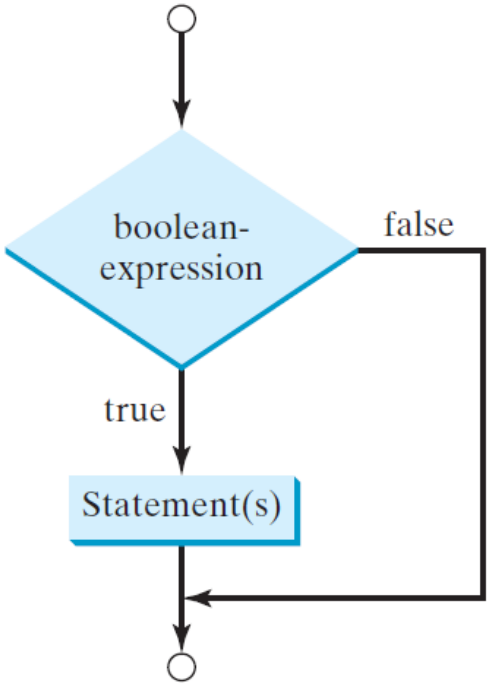
```
boolean b = (1 > 2);
```

boolean Data Type, Values, and Expressions

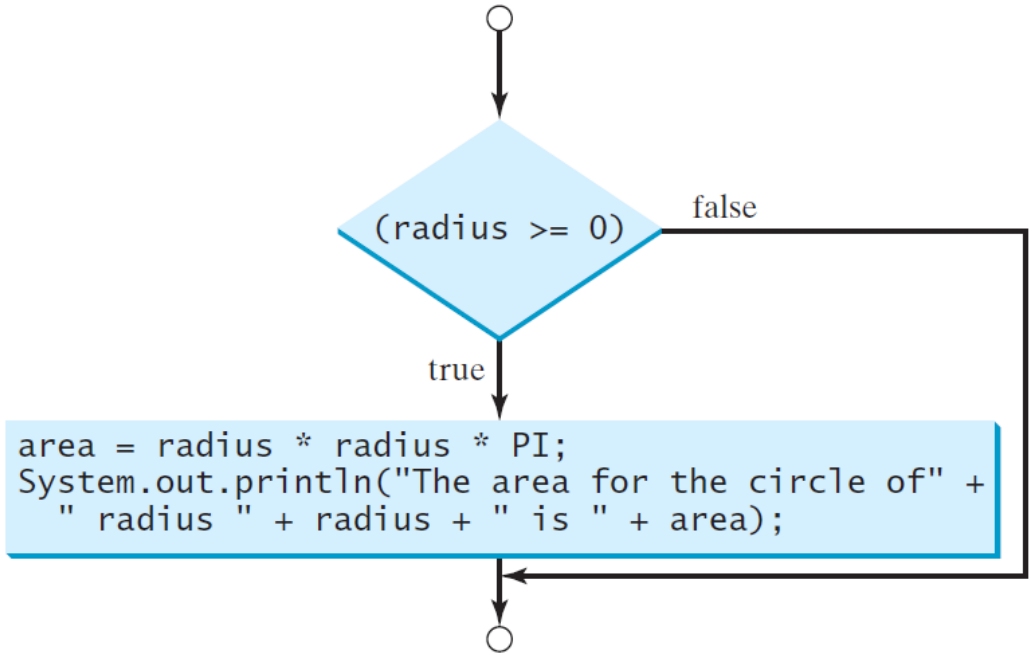
Java Operator	Mathematics Symbol	Name	Example (radius is 5)	Result
`<`	<	less than	`radius < 0`	`false`
`<=`	≤	less than or equal to	`radius <= 0`	`false`
`>`	>	greater than	`radius > 0`	`true`
`>=`	≥	greater than or equal to	`radius >= 0`	`true`
`==`	=	equal to	`radius == 0`	`false`
`!=`	≠	not equal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

if Statements

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




```
8           if (radius >= 0) {
9               System.out.println("The radius of the circle is " + radius);
10              double area = radius * radius * 3.14159;
11              System.out.println("The area for the circle of radius " +
12                      radius + " is " + area);
13          }
```



if Statements

```
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

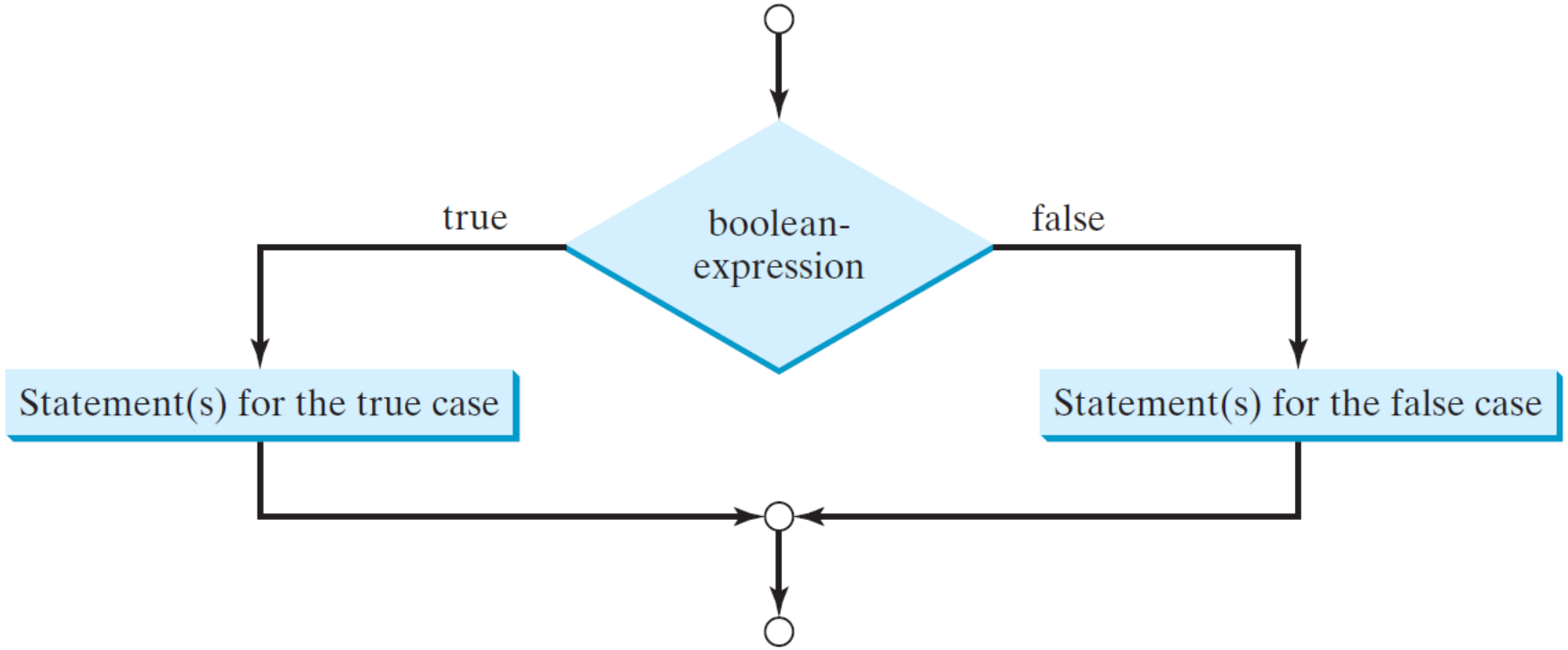
```
if (i > 0)
   System.out.println("i is positive");
```

(b)

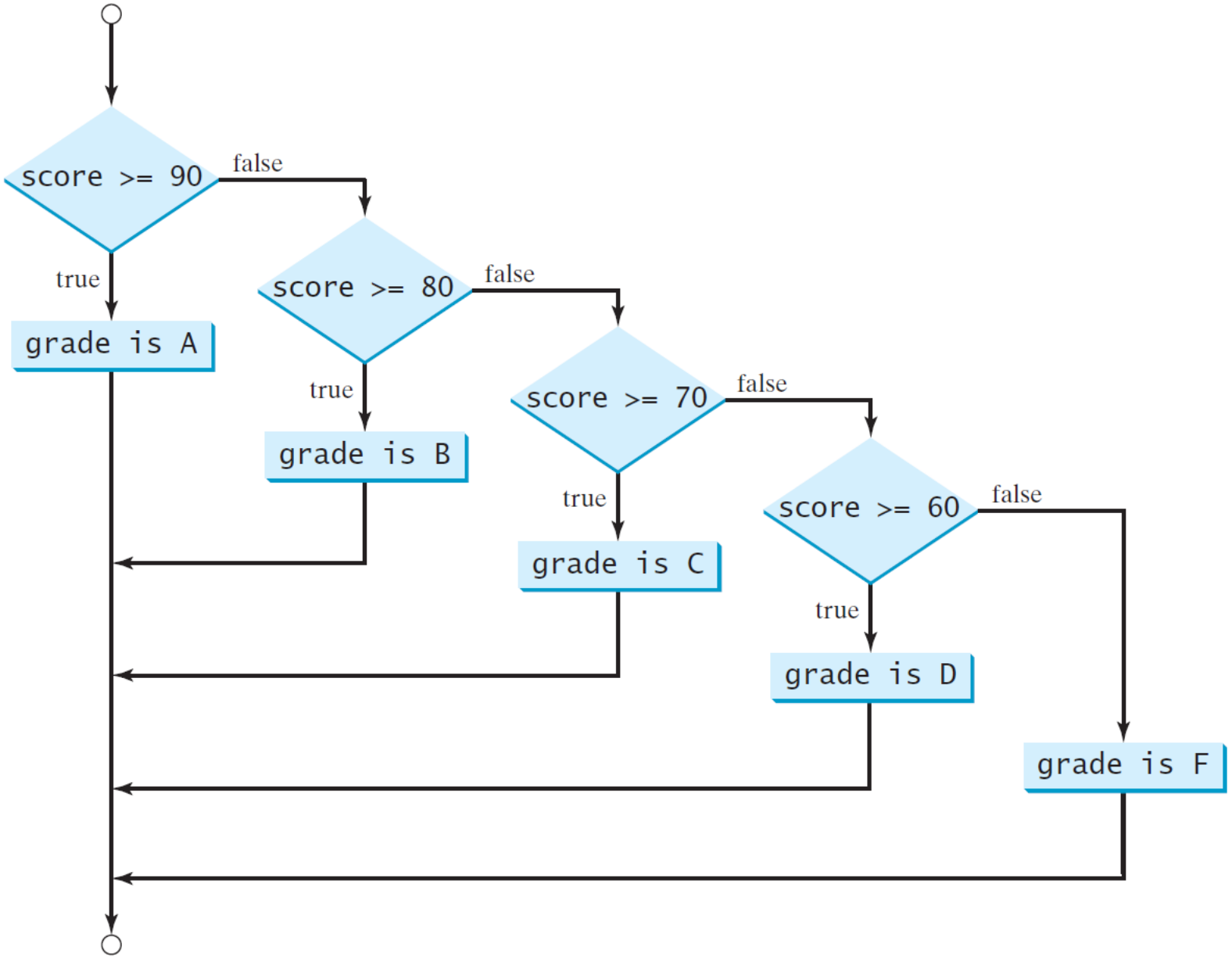
if Statements

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

```
 8            if (radius >= 0) {
 9                System.out.println("The radius of the circle is " + radius);
10                double area = radius * radius * 3.14159;
11                System.out.println("The area for the circle of radius " +
12                        radius + " is " + area);
13            }
14            else {
15                System.out.println(x: "Incorrect radius");
16            }
```



if Statements



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)

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

if Statements

```
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)

To force the **else** clause to match the first **if** clause, you must add a pair of braces.

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

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

(a)

Logical Operators

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

Logical Operators

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.

Logical Operators

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.

Logical Operators

p_1	p_2	$p_1 \mid\mid p_2$	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	

Logical Operators

p_1	p_2	$p_1 \wedge p_2$	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	

Logical Operators - Example

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:

```
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)));
```

switch Statements

```
1   import java.util.Scanner;
2
3   public class WeekDay {
        Run | Debug
4       public static void main(String[] args) {
5           Scanner input = new Scanner(System.in);
6           System.out.print(s: "Enter the day integer: ");
7           int day = input.nextInt() % 7;
8           if (day == 1)
9               System.out.println(x: "Monday");
10          else if (day == 2)
11              System.out.println(x: "Tuesday");
12          else if (day == 3)
13              System.out.println(x: "Wednesday");
14          else if (day == 4)
15              System.out.println(x: "Thursday");
16          else if (day == 5)
17              System.out.println(x: "Friday");
18          else if (day == 6)
19              System.out.println(x: "Saturday");
20          else
21              System.out.println(x: "Sunday");
22      }
23  }
24
```

switch Statements

- 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 Statements

- 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.

switch Statements

```java
import java.util.Scanner;

public class WeekDay {
    Run | Debug
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print(s: "Enter the day integer: ");
        int day = input.nextInt() % 7;
        if (day == 1)
            System.out.println(x: "Monday");
        else if (day == 2)
            System.out.println(x: "Tuesday");
        else if (day == 3)
            System.out.println(x: "Wednesday");
        else if (day == 4)
            System.out.println(x: "Thursday");
        else if (day == 5)
            System.out.println(x: "Friday");
        else if (day == 6)
            System.out.println(x: "Saturday");
        else
            System.out.println(x: "Sunday");
    }
}
```

```java
import java.util.Scanner;

public class WeekDaySwitch {
    Run | Debug
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print(s: "Enter the day integer: ");
        int day = input.nextInt() % 7;
        switch (day) {
            case 1: System.out.println(x: "Monday"); break;
            case 2: System.out.println(x: "Tuesday"); break;
            case 3: System.out.println(x: "Wednesday"); break;
            case 4: System.out.println(x: "Thursday"); break;
            case 5: System.out.println(x: "Friday"); break;
            case 0: System.out.println(x: "Sunday"); break;
            case 6: System.out.println(x: "Saturday");
        }
    }
}
```

Conditional Operators

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

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

(boolean-expression) ? expression1 : expression2

Ternary operator
Binary operator
Unary operator

Conditional Operators

```
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

1. var++, var--
2. +, - (Unary plus and minus), ++var,--var
3. (type) Casting
4. ! (Not)
5. *, /, % (Multiplication, division, and remainder)
6. +, - (Binary addition and subtraction)
7. <, <=, >, >= (Relational operators)
8. ==, !=; (Equality)
9. ^ (Exclusive OR)
10. && (Conditional AND) Short-circuit AND
11. || (Conditional OR) Short-circuit OR
12. =, +=, -=, *=, /=, %= (Assignment operator)

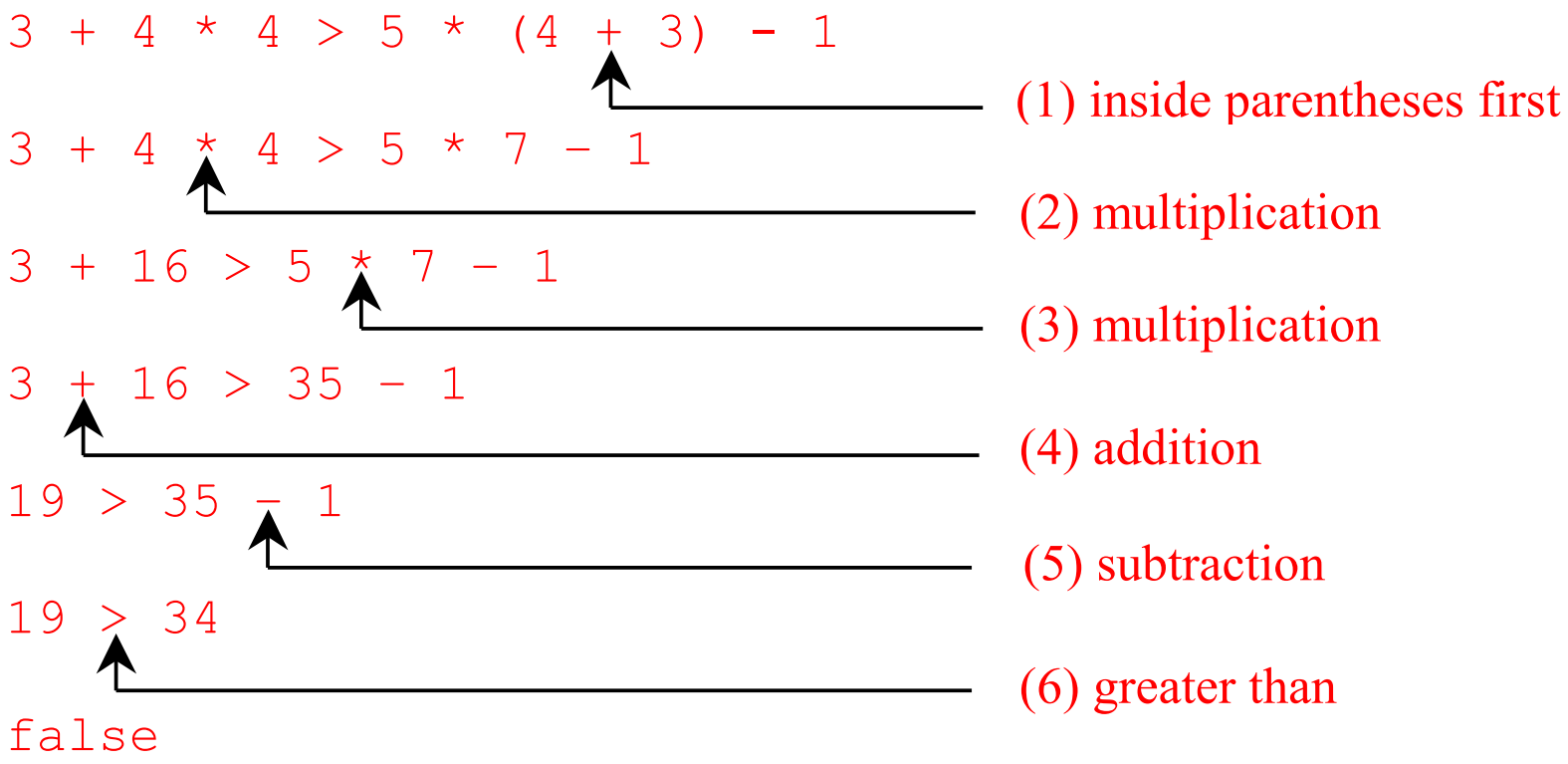
Operator Precedence and Associativity

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

- a – b + c – d is equivalent to ((a – b) + c) – d
- a = b += c = 5 is equivalent to a = (b += (c = 5))

Operator Precedence and Associativity - Example

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



Conclusion

- From the circle area computing program, we learned
  - How to handle variables
  - How to solve problems using operators and methods
  - How to control the condition if necessary (e.g., if statement)
- From the day determining program, we learned
  - How to write the algorithm simpler using switch statement
- We discussed the use of logical operators to make the conditional statement simpler (in a single line statement).
- We discussed the precedence of operators in the problems.