Chapter 2
Java basics

ELEMENTARY PROGRAMMING

1. Writing a Simple Program

- Problem: compute the area of a circle.
- How do we write a program for solving this problem?
  - designing algorithms
    1. Read in the circle's radius.
    2. Compute the area using the following formula:

       area = pi * radius * radius
    3. Display the result.
  - translating algorithms into code

1. Writing a Simple Program

every Java program begins with a class definition

```
public class ComputeArea {
 // Details to be given later

}
```

every Java program must have a main method

```
public class ComputeArea {
 public static void main(String[] args) {

        // Step 1: Read in radius
        // Step 2: Compute area
        // Step 3: Display the area

} }
```

1. Writing a Simple Program

```
public class ComputeArea {
 public static void main(String[] args) {
     double radius; double area;
         // Step 1: Read in radius
         // Step 2: Compute area
         // Step 3: Display the area
}
}
```

```
public class ComputeArea {
 public static void main(String[] args) {
     double radius; double area;
     radius = 20; // radius is now 20
     // Step 2: Compute area
     area = radius * radius * 3.14159;
     // Step 3: Display the area
     System.out.println("The area for the circle of radius " + radius + " is " + area);

}
}
```

1. Reading Input from the Console

- Use **Scanner** class to create an object to read input from System.in

```
Scanner input = new Scanner(System.in);
```

- Use scanner object to invoke its method to perform a task.

```
double radius = input.nextDouble();
```

```
import java.util.Scanner;
// Scanner is in the java.util package

public class ComputeAreaWithConsoleInput {
  public static void main(String[] args) {
    // Create a Scanner object
    Scanner input = new Scanner(System.in);

    // Prompt the user to enter a radius
    System.out.print("Enter a number for radius: ");
    double radius = input.nextDouble();

    // Compute area
    double area = radius * radius * 3.14159;

    // Display result
    System.out.println("The area for the circle of radius " +
      radius + " is " + area);
  }
}
```

2. Identifiers

- Identifiers are the names that identify the elements such as classes, methods, and variables in a program.
- All identifiers must obey the following rules:
  - consisting of letters, digits, underscores (_), and dollar signs ($).
  - start with a letter, an underscore (_), or a dollar sign ($), cannot start with a digit.
  - cannot be a reserved word.
  - cannot be true, false, or null.
  - can be of any length.

```
import java.util.Scanner;
// Scanner is in the java.util package

public class ComputeAreaWithConsoleInput {
  public static void main(String[] args) {
    // Create a Scanner object
    Scanner input = new Scanner(System.in);

    // Prompt the user to enter a radius
    System.out.print("Enter a number for radius: ");
    double radius = input.nextDouble();

    // Compute area
    double area = radius * radius * 3.14159;

    // Display result
    System.out.println("The area for the circle of radius " +
      radius + " is " + area);
  }
}
```

3. Variables

- Variables are used to represent values that may be changed in the program.
- A variable must be assigned a value before it can be used.
- The syntax for declaring a variable is
  datatype variableName;
  Example:

```
int count; // Declare count to be an integer variable
double radius; // Declare radius to be a double variable
```

- If variables are of the same type, they can be declared together, as follows:
  datatype variable1, variable2, ..., variablen;
  Example:

```
int i, j, k;
```

- Declare a variable and initialize it in one step.
  Example:

```
int count = 1;
```

4. Assignment Statements and Assignment Expressions

- An assignment statement designates a value for a variable.
- An assignment statement can be used as an expression in Java.
- The syntax for assignment statements is as follows:
  variable = expression;

```
int y = 1;
double radius = 1.0;
int x = 5 * (3 / 2);
x = y + 1;
double area = radius * radius * 3.14159;
```

5. Named Constants

- A named constant is an identifier that represents a permanent value.
- A constant must be declared and initialized in the same statement.
- The syntax for declaring a constant:
  final datatype CONSTANTNAME = value;

5. Named Constants

```
import java.util.Scanner; // Scanner is in the java.util package
public class ComputeAreaWithConstant {
 public static void main(String[] args) {
   final double PI = 3.14159; // Declare a constant
   // Create a Scanner object
   Scanner input = new Scanner(System.in);
   // Prompt the user to enter a radius
   System.out.print("Enter a number for radius: ");
   double radius = input.nextDouble();
   // Compute area
   double area = radius * radius * PI;
   // Display result
   System.out.println("The area for the circle of radius " +
     radius + " is " + area);
 }}
```

6. Naming Conventions

- Name: descriptive, straightforward meanings
- The conventions for naming variables, methods, and classes.
  - Use lowercase for variables and methods.
    If a name consists of several words, use camelCase rule,
    Example: area, studentName
  - Capitalize the first letter of each word in a class name
    Example: ComputeArea
  - Capitalize every letter in a constant, and use underscores between words
    Example: PI, MAX_VALUE.

7. Numeric Data Types and Operations

Numeric Types:

- Every data type has a range of values.
- The compiler allocates memory space for each variable or constant according to its data type.
- eight primitive data types for numeric values, characters, and Boolean values

7. Numeric Data Types and Operations

Numeric Types

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 $2^{63} - 1$	64-bit signed
	(i.e., -9223372036854775808 to 9223372036854775807)	
`float`	Negative range: $-3.4028235E + 38$ to $-1.4E - 45$	32-bit IEEE 754
	Positive range: $1.4E - 45$ to $3.4028235E + 38$	
`double`	Negative range: $-1.7976931348623157E + 308$ to $-4.9E - 324$	64-bit IEEE 754
	Positive range: $4.9E - 324$ to $1.7976931348623157E + 308$	

7. Numeric Data Types and Operations

Numeric Types

Reading Numbers from the Keyboard

TABLE 2.2 Methods for `Scanner` Objects

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.

7. Numeric Data Types and Operations

Numeric Operators

TABLE 2.3 Numeric Operators

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

7. Numeric Data Types and Operations

```
import java.util.Scanner;

public class DisplayTime {
 public static void main(String[] args) {
   Scanner input = new Scanner(System.in);
   // Prompt the user for input
   System.out.print("Enter an integer for seconds: ");
   int seconds = input.nextInt();
   int minutes = seconds / 60; // Find minutes in seconds
   int remainingSeconds = seconds % 60; // Seconds remaining
   System.out.println(seconds + " seconds is " + minutes +
     " minutes and " + remainingSeconds + " seconds");
 }
}
```

7. Numeric Data Types and Operations

Exponent Operations

Math.pow(a, b) computes a^b

```
System.out.println(Math.pow(2, 3)); // Displays 8.0
System.out.println(Math.pow(4, 0.5)); // Displays 2.0
System.out.println(Math.pow(2.5, 2)); // Displays 6.25
System.out.println(Math.pow(2.5, -2)); // Displays 0.16
```

8. Numeric Literals

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

Example:

34 and 0.305 are literals

```
int numberOfYears = 34;
double weight = 0.305;
```

127
0B1111: 15
2147483648L or 2147483648l: long

100.2f or 100.2F: float
100.2 or 100.2d or 100.2D: double

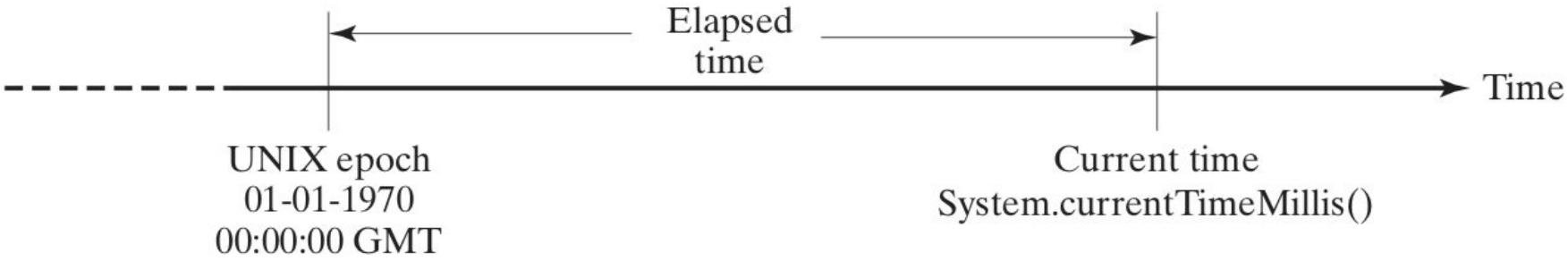
Scientific Notation

1.23456 * (10^2) is written as 1.23456E2 or 1.23456E+2
1.23456 * 10^-2 as 1.23456E-2

9. Case Study: Displaying the Current Time

The currentTimeMillis method in the System class returns the current time in milliseconds elapsed since midnight, January 1, 1970 GMT

Code: ShowCurrentTime.java



10. Augmented Assignment Operators

The operators +, -, *, /, and % can be combined with the assignment operator to form augmented operators.

count = count + 1; count += 1;

TABLE 2.4 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`

11. Increment and Decrement Operators

The increment operator (++) and decrement operator (– –) are for incrementing and decrementing a variable by 1.

```
int i = 3, j = 3;
i++; // i becomes 4
j--; // j becomes 2

++i; // i becomes 4
--j; // j becomes 2
```

11. Increment and Decrement Operators

TABLE 2.5 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`

12. Numeric Type Conversions

Java will automatically widen a type, but you must narrow a type explicitly.

- If an integer and a floating-point number are involved in a binary operation, Java automatically converts the integer to a floating-point value.
  3 * 4.5 is same as 3.0 * 4.5

- Casting: specify the target type in parentheses, followed by the variable's name or the value to be cast
  (int) 1.7

SELECTIONS

1 Introduction

The program can decide which statements to execute based on a condition.

```
if (radius < 0) {
    System.out.println("Incorrect input");
} else {
    area = radius * radius * 3.14159;
    System.out.println("Area is " + area);
}
```

2. boolean Data Type

- The boolean data type declares a variable with the value either true or false.
- Six relational operators

TABLE 3.1 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`

3 if Statements

- specify alternative paths of execution.
- one-way if statement

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

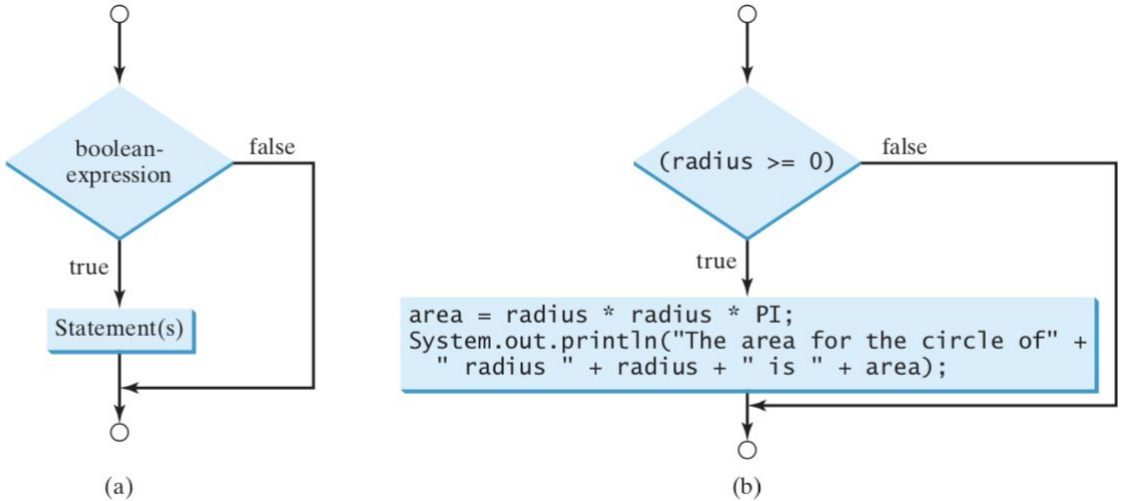


FIGURE 3.1 An `if` statement executes statements if the `boolean-expression` evaluates to `true`.

4 Two-Way if-else Statements

- decide the execution path based on whether the condition is true or false.
- two-way if-else statement

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

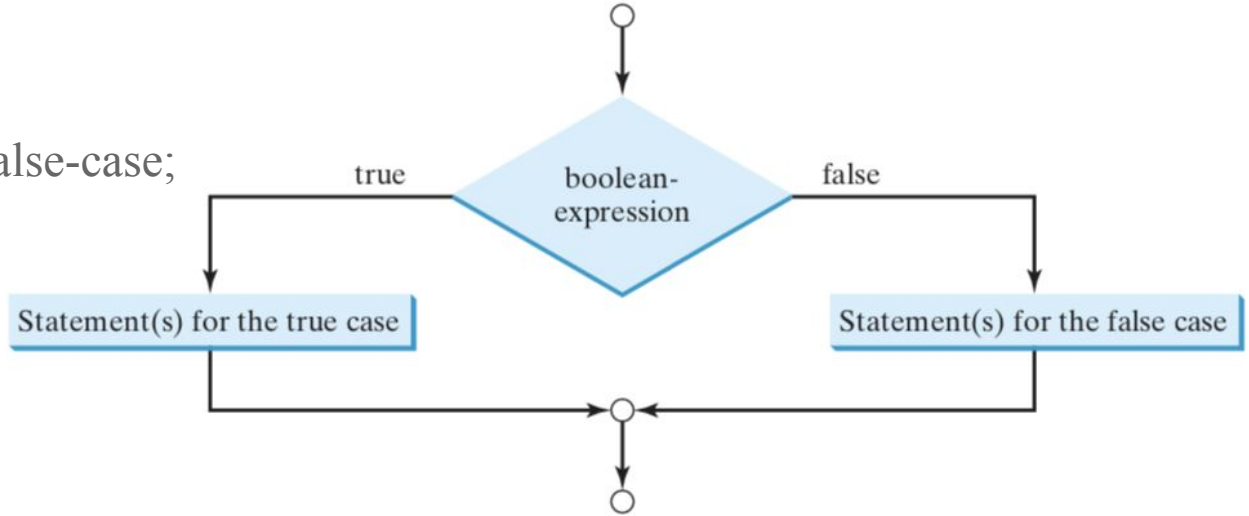
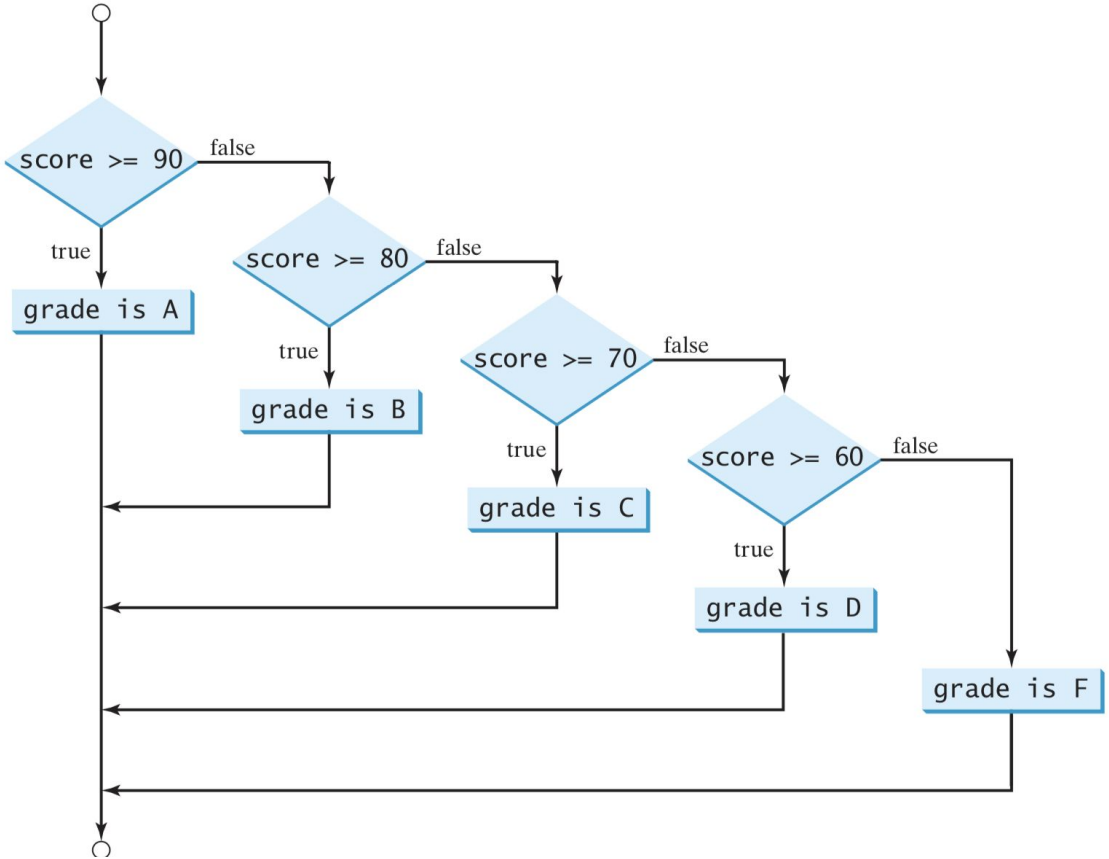


FIGURE 3.2 An `if-else` statement executes statements for the true case if the `Boolean-expression` evaluates to `true`; otherwise, statements for the false case are executed.

5 Nested if and Multi-Way if-else 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");
}
```

6 Generating Random Numbers

Math.random() to obtain a random double value between 0.0 and 1.0, excluding 1.0.

The program can work as follows:

1. Generate two single-digit integers into number1 and number2.
2. If number1 < number2, swap number1 with number2.
3. Prompt the student to answer, "What is number1 – number2?"
4. Check the student's answer and display whether the answer is correct.

Code: SubtractionQuiz

7 Logical Operators

The logical operators ***!, &&,*** ||, ***and*** ^ can be used to create a compound Boolean expression.

TABLE 3.3 Boolean Operators

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

8 switch Statements

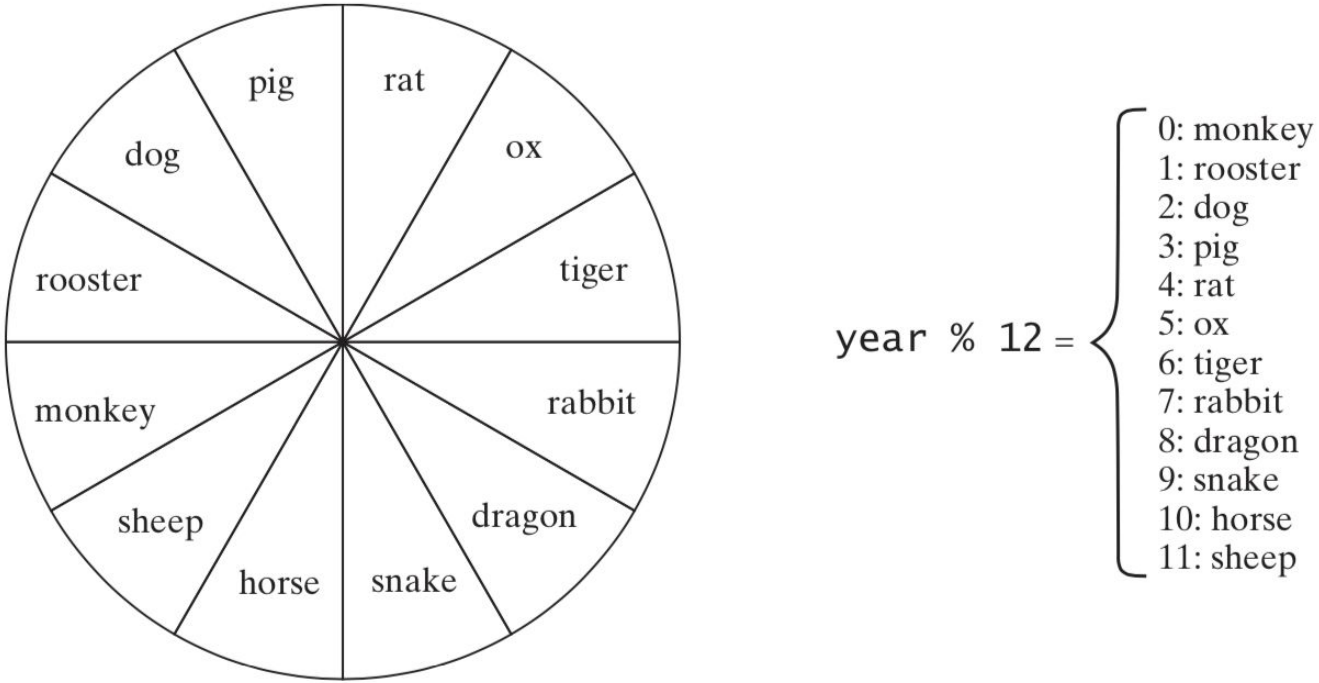


FIGURE 3.6 The Chinese Zodiac is based on a twelve-year cycle.

8 switch Statements

A ***switch*** statement executes statements based on the value of a variable or an expression.

```
switch (year % 12) {
  case 0: System.out.println("monkey"); break;
  case 1: System.out.println("rooster"); break;
  case 2: System.out.println("dog"); break;
  case 3: System.out.println("pig"); break;
  case 4: System.out.println("rat"); break;
  case 5: System.out.println("ox"); break;
  case 6: System.out.println("tiger"); break;
  case 7: System.out.println("rabbit"); break;
  case 8: System.out.println("dragon"); break;
  case 9: System.out.println("snake"); break;
  case 10: System.out.println("horse"); break;
  case 11: System.out.println("sheep"); break;
}
```

9 Conditional Expressions

A conditional expression evaluates an expression based on a condition.

boolean-expression ? expression1 : expression2

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

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

10 Operator Precedence and Associativity

TABLE 3.8 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)
	`<`, `<=`, `>`, `>=` (Relational)
	`==`, `!=` (Equality)
	`^` (Exclusive OR)
	`&&` (AND)
	`||` (OR)
	`=`, `+=`, `-=`, `*=`, `/=`, `%=` (Assignment operator)