第3章
演算子と分岐文

Java 运算符[演算子]

- 运算符用于对变量和值执行运算。
- 在下面的示例中，我们使用 + 运算符将两个值相加：

```
int sum1 = 100 + 50;         // 150 (100 + 50)
int sum2 = sum1 + 250;       // 400 (150 + 250)
int sum3 = sum2 + sum2;      // 800 (400 + 400)
```

- Java运算符分为以下几组：
  - 算术运算符[算術演算子]
  - 赋值运算符[代入演算子]
  - 比较运算符[比較演算子]
  - 逻辑运算符[論理演算子]
  - 按位运算符[ビット演算子]

尝试代码：
Operators.java

算术运算符[算術演算子]

- 算术运算符用于执行常见的数学运算。

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

思考：++x和x++这两种写法有什么区别？

赋值运算符[代入演算子]

- 赋值运算符用于为变量赋值。

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

比较运算符[比較演算子]

- 比较运算符用于比较两个值：

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

逻辑运算符[論理演算子]

- 逻辑运算符用于确定变量或值之间的逻辑：

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

布尔表达式

- 布尔表达式是返回一个布尔值[ブーリアン型]true或false的Java表达式。
- 比如：
  - x > y
  - 19 > 9
  - 10 == 10
  - (1 == 2) && (2 == 2)

- 练习：以下代码输出什么？

```
public class MyClass {
    public static void main(String[] args) {
        int x = 1;
        int y = 2;
        System.out.println((x > y) || x > (y + 1) && (x < 0));
    }
}
```

if语句

- 先看句法：

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

- condition通常是一个布尔表达式，如果condition结果为true，则运行花括号{}里面的程序，否则跳过此花括号。

- 练习：以下代码会输出什么？

```
int x = 20;
int y = 18;
if (x > y) {
    System.out.println("I'm Bruce Wayne.");
}
System.out.println("I'm batman!");
```

if...else...语句

- 先看句法：

```
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```

- 如果condition结果为true，则只运行第一个花括号里的程序，结果为false，则只运行第二个花括号里的程序。
- 翻译成自然语言就是：如果…就…,否则就…
- 练习：以下代码会输出什么？

```
int time = 20;
if (time < 18) {
  System.out.println("Good day.");
} else {
  System.out.println("Good evening.");
}
```

if...else...语句

- 先看句法：

```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2 is true
} else {
  // block of code to be executed if the condition1 is false and condition2 is false
}
```

- 如果condition1是false，则检查condition2，这里的else if可以写任意次，如果某一个condition是true则运行它对应的花括号，其他花括号不运行，如果所有condition都是false，则运行最后一个else的花括号。

```
int time = 22;
if (time < 10) {
  System.out.println("Good morning.");
} else if (time < 20) {
  System.out.println("Good day.");
} else {
  System.out.println("Good evening.");
}
// Outputs "Good evening."
```

简写if…else（三元运算符[条件 (三項) 演算子]）

- 先看句法：

```
variable = (condition) ? expressionTrue :  expressionFalse;
```

- 多用于有条件的给变量赋值

```
int time = 20;
if (time < 18) {
  System.out.println("Good day.");
} else {
  System.out.println("Good evening.");
}
```

```
int time = 20;
String result = (time < 18) ? "Good day." : "Good evening.";
System.out.println(result);
```

switch语句

- 先看句法：

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

再看例子：

```
int day = 4;
switch (day) {
  case 1:
    System.out.println("Monday");
    break;
  case 2:
    System.out.println("Tuesday");
    break;
  case 3:
    System.out.println("Wednesday");
    break;
  case 4:
    System.out.println("Thursday");
    break;
  case 5:
    System.out.println("Friday");
    break;
  case 6:
    System.out.println("Saturday");
    break;
  case 7:
    System.out.println("Sunday");
    break;
}
// Outputs "Thursday" (day 4)
```

- 到达break关键字时，它将跳出switch块。（可无）
- 如果没有匹配的情况下，default关键字指定一些代码来运行。（可无）