6

УПРАВЛЯЮЩИЕ КОМАНДЫ

В предыдущих главах вы узнали много нового. К настоящему моменту вы должны знать базовую структуру программ Java, а также уметь писать простые программы Java с использованием переменных. Кроме того, вы также научились пользоваться различными встроенными методами Java для взаимодействия с пользователями.

В этой главе будет сделан следующий шаг — вы узнаете, как управлять последовательностью выполнения команд вашей программы. По умолчанию команды выполняются по порядку, от начала программы к концу, в порядке их следования. Тем не менее такой порядок можно изменить с помощью *управляющих команд*.

К этой категории относятся команды принятия решений (`if`, `switch`), команды циклов (`for`, `while`, `do-while`) и команды перехода (`break`, `continue`). Все эти команды будут рассмотрены в следующих разделах.

Но сначала рассмотрим операторы сравнения.

6.1. ОПЕРАТОРЫ СРАВНЕНИЯ

Многие управляющие команды используют некоторую разновидность сравнения. Программа выбирает тот или иной путь выполнения в зависимости от результата сравнения.

Из всех операторов сравнения чаще всего используется оператор проверки равенства. Если вы хотите узнать, равны ли две переменные, используйте оператор `==` (два знака `=`). Например, включая в программу выражение `x == y`, вы приказываете программе проверить, равно ли значение `x` значению `y`. Если они равны, значит, условие выполнено и команда дает результат `true`. В противном случае будет получен результат `false`.

Кроме проверки двух значений на равенство существуют и другие операторы сравнения, которые могут использоваться в управляющих командах.

Не равно (`!=`)

Возвращает `true`, если левая часть не равна правой.

```
5 != 2    true
6 != 6    false
```

Больше (`>`)

Возвращает `true`, если левая часть больше правой.

```
5 > 2    true
3 > 6    false
```

Меньше (`<`)

Возвращает `true`, если левая часть меньше правой.

```
1 < 7    true
9 < 6    false
```

Больше или равно (`>=`)

Возвращает `true`, если левая часть больше или равна правой.

```
5 >= 2    true
5 >= 5    true
3 >= 6    false
```

Меньше или равно (<=)

Возвращает `true`, если левая часть меньше или равна правой.

```
5 <= 7    true
7 <= 7    true
9 <= 6    false
```

Также существуют два логических оператора (&&, ||), которые пригодятся для объединения нескольких условий.

Оператор AND (&&)

Возвращает `true`, если выполняются все условия.

```
5==5 && 2>1 && 3!=7  true
5==5 && 2<1 && 3!=7  false, так как второе условие
                     (2<1) дает false
```

Оператор OR (||)

Возвращает `true`, если выполняется хотя бы одно условие.

```
5==5 || 2<1 || 3==7  true, так как первое условие
                    (5==5) дает true
5==6 || 2<1 || 3==7  false, так как все условия дают
                     false
```

6.2. КОМАНДЫ ПРИНЯТИЯ РЕШЕНИЙ

После знакомства с операторами сравнения посмотрим, как пользоваться этими операторами для управления последовательностью выполнения программы. Начнем с команды `if`.

6.2.1. КОМАНДА IF

Команда `if` — одна из наиболее часто используемых команд управления последовательностью выполнения. Она позволяет программе проверить некоторое условие и выполнить соответствующее действие в зависимости от результата проверки.

Команда `if` имеет следующую структуру (номера строк добавлены для удобства):

```
 1 if (выполняется условие 1)
 2 {
 3        действие A
 4 }
 5 else if (выполняется условие 2)
 6 {
 7        действие B
 8 }
 9 else if (выполняется условие 3)
10 {
11        действие C
12 }
13 else
14 {
15        действие D
16 }
```

В строке 1 проверяется первое условие. Если оно выполняется, то выполняются все команды, заключенные в фигурные скобки (строки 2–4). Остальная часть команды `if` (строки 5–16) пропускается.

Если первое условие не выполнено, то блоки `else if`, следующие за условием, могут использоваться для проверки дополнительных условий (строки 5–12). Блоков `else if` может быть несколько. Наконец, блок `else` (стро-

ки 13–116) может использоваться для выполнения кода в том случае, если не выполнено ни одно из предшествующих условий. Блоки `else if` и `else` обязательными не являются. Если нет дополнительных условий, которые требуется проверить, включать их не обязательно.

Чтобы лучше понять, как работает эта команда `if`, рассмотрим пример. Запустите NetBeans и создайте проект с именем `IfDemo`. Замените сгенерированный код следующим:

```
package ifdemo;
import java.util.Scanner;

public class IfDemo{

public static void main(String[] arg)
{

    Scanner input = new Scanner(System.in);
    System.out.print("\nPlease enter your age: ");
    int userAge = input.nextInt();
    if (userAge < 0 || userAge > 100)
    {
        System.out.println("Invalid Age");
        System.out.println("Age must be between 0 and
                           100");
    }
    else if (userAge < 18)
        System.out.println("Sorry you are underage");
    else if (userAge < 21)
        System.out.println("You need parental consent");
    else
    {
        System.out.println("Congratulations!");
        System.out.println("You may sign up for the
                           event!");
    }
}
}
```

Программа запрашивает у пользователя его возраст и сохраняет результат в переменной `userAge`. Команда

```
if (userAge < 0 || userAge > 100)
```

проверяет, не будет ли значение `userAge` меньше нуля или больше 100. Если хотя бы одно из этих условий истинно, то программа выполняет все команды в следующей паре фигурных скобок. В данном примере будет выведена строка «Invalid Age», за которой следует сообщение «Age must be between 0 and 100».

С другой стороны, если оба условия ложны, то программа проверяет следующее условие — `else if (userAge < 18)`. Если `userAge` меньше 18 (но больше либо равно 0, потому что первое условие не выполняется), программа выведет сообщение «Sorry you are underage».

Возможно, вы заметили, что команда:

```
System.out.println("Sorry you are underage");
```

не заключена в фигурные скобки. Дело в том, что фигурные скобки не обязательны, если выполняется *только одна* команда.

Если пользователь не ввел значение, меньшее 18, но введенное значение больше либо равно 18, но меньше 21, будет выполнена следующая команда `else if`. В этом случае выводится сообщение «You need parental consent».

Наконец, если введенное значение больше или равно 21, но меньше или равно 100, то программа выполнит код в блоке `else`. В этом случае будет выведена строка «Congratulations», за которой следует сообщение «You may sign up for the event!».

Запустите программу пять раз и введите при каждом запуске значение `-1`, `8`, `20`, `23` и `121` соответственно. Вы получите следующий результат:

```
Please enter your age: -1
Invalid Age
Age must be between 0 and 100

Please enter your age: 8
Sorry you are underage

Please enter your age: 20
You need parental consent

Please enter your age: 23
Congratulations!
You may sign up for the event!

Please enter your age: 121
Invalid Age
Age must be between 0 and 100
```

6.2.2. ТЕРНАРНЫЙ ОПЕРАТОР

Тернарный оператор (`?`) представляет собой упрощенную форму команды `if`, которая очень удобна для присваивания значения переменной в зависимости от результата условия. Синтаксис выглядит так:

условие `?` *значение для true* `:` *значение для false*;

Например, команда

```
3>2 ? 10 : 5;
```

вернет значение 10, так как 3 больше 2 (т. е. если условие `3 > 2` истинно). Затем это значение можно присвоить переменной.

Если же выполнить команду

```
int myNum = 3>2 ? 10 : 5;
```

myNum будет присвоено значение 10.

6.2.3. КОМАНДА SWITCH

Команда switch похожа на команду if, не считая того, что она не работает с диапазонами значений. Команда switch требует, чтобы каждый случай базировался на одном значении.

Программа выполняет один из нескольких блоков кода в зависимости от значения переменной, используемой для выбора.

Команда switch имеет следующий синтаксис:

```
switch (переменная для выбора)
{
case первыйСлучай:
    действие A;
    break;

case второйСлучай:
    действие B;
    break;

default:
    действие C;
    break;
}
```

Количество случаев в команде switch не ограничено. Блок default не является обязательным; он выполняется, если ни один случай не подходит. Для выбора может использоваться переменная типа byte, short, char или int. Начиная с Java 7 для выбора также может использоваться переменная String.