**Глава 2. Управляющие инструкции Java**

К управляющим инструкциям относят условные инструкции и инструкции цикла. В Java таких инструкций несколько, и каждая из них имеет свои особенности.

**Условная инструкция if()**

Если не считать тернарного оператора, в Java существует две условных конструкции, которые позволяют выполнять разные операции в зависимости от некоторого условия. В первую очередь рассмотрим условную инструкцию if().

package com.jolken;  
  
public class IfElse {  
 public static void main(String[] args) {  
 int a = 42;  
 //условие, вместо цифр могут быть переменные  
 if (a > 0) { //Код выполняемый, если условие верно  
 System.*out*.println("a > 0");  
 }  
 else { //Код выполняемый, если условие неверно  
 System.*out*.println("a < 0");  
 }  
 }  
  
}

Условная инструкция if() выполняется в следующей последовательности. Сначала проверяется условие, указанное в круглых скобках после ключевого слова if. Если условие верное (значение соответствующего выражения равно true), выполняется блок инструкций, указанный сразу после инструкции if

В противном случае, то есть если значение выражения в круглых скобках после ключевого слова if равно false, выполняется блок инструкций, указанных после ключевого слова else

Ветвь else условной инструкции не является обязательной

На практике нередко используются вложенные инструкции if(). С точки зрения

синтаксиса языка Java такая ситуация проста: в ветви else условной инструкции

указывается другая условная инструкции и т. д. Синтаксическая конструкция

имеет вид:

package com.jolken;  
  
public class IfElse {  
 public static void main(String[] args) {  
  
 int a = 42;  
 if (a > 0) { System.*out*.println("a > 0"); }  
 else if (a > 2) { System.*out*.println("a > 2"); }  
 else if (a > 6) { System.*out*.println("a > 6"); }  
 else if (a > 12) { System.*out*.println("a > 12"); }  
 else if (a > 999) { System.*out*.println("a > 999"); }  
 }  
  
}

**Условная инструкция switch()**

Обычно к услугам условной инструкции switch() прибегают в случае, когда при проверке условия альтернатив больше, чем две. Эту инструкцию еще называют инструкцией выбора. Синтаксис вызова инструкции switch() следующий:

package com.jolken;  
  
public class Switch {  
 public static void main(String[] args) {  
 int a = 42;  
 switch (a) {  
 case 1:  
 System.*out*.println("a=1");  
 break;  
 case 12:  
 System.*out*.println("a=12");  
 break;  
 case 42:  
 System.*out*.println("a=42");  
 break;  
 default:  
 System.*out*.println("Значение a неизвестно");  
 break;  
 }  
 }  
}

**Инструкция цикла for()**

Для выполнения однотипных многократно повторяющихся действий используют инструкции цикла. В Java существует несколько инструкций цикла. Рассмотрим инструкцию цикла for().

package com.jolken;  
  
public class For {  
 public static void main(String[] args) {  
 // переменная, видимая только в теле цикла  
 // ↓↓↓↓↓ условие, когда оно будет неверным, то цикл закончится  
 // ↓↓↓↓↓ ↓↓↓↓↓ действие в конце цикла  
 // ↓↓↓↓↓ ↓↓↓↓↓ ↓↓↓↓↓  
 for (int i = 0; i < 16; i++) {  
 System.*out*.println(i);  
 }  
 //Выведет в столбик числа от 0 до 15  
 }  
  
}

В круглых скобках после ключевого слова for указываются три группы, или блока, выражений. Блоки разделяются точкой с запятой. Первый блок обычно называется блоком инициализации. Как правило, в этом блоке размещается команда (или команды), которая перед выполнением цикла присваивает индексным переменным начальные значения. Второй блок — условие, при выполнении которого продолжается работа инструкции цикла. Третий блок содержит команды, которыми изменяются значения индексных переменных. Первый и третий блоки могут состоять из нескольких команд. Команды одного блока разделяются запятыми.

Далее, после ключевого слова for и круглых скобок в фигурных скобках идет блок команд, которые выполняются в рамках каждого цикла и фактически формируют тело инструкции цикла. Если тело инструкции цикла состоит из одной команды, фигурные скобки можно не ставить. Выполняется инструкция цикла по следующему алгоритму.

Начинается выполнение инструкции цикла с блока инициализации: последовательно выполняются все команды этого блока. Далее проверяется условие во втором блоке. Если оно истинно (значение true), выполняются команды тела инструкции цикла (команды в фигурных скобках). Далее выполняются команды третьего блока в круглых скобках и проверяется условие во втором блоке. Если условие истинно, выполняются команды основного тела инструкции цикла, команды блока изменения индексных переменных (третий блок), затем проверяется условие и т. д. Вся эта процедура продолжается до тех пор, пока при проверке условия его значение не становится равным false.

**Инструкция цикла while()**

Для организации циклов кроме инструкции for() часто используется инструкция while(). Далее приведен синтаксис вызова инструкции while():

package com.jolken;  
  
public class While {  
 public static void main(String[] args) {  
 int a = 0;  
 while (a < 10) {  
 a++;  
 }  
 }  
}

После ключевого слова while в круглых скобках указывается условие. В начал выполнения инструкции проверяется это условие. Если условие истинно, выполняются команды цикла — они заключаются в фигурные скобки. После этого снова проверяется условие и т. д.

От инструкции for() инструкция while() принципиально отличается тем, что инициализация индексной переменной, если такая имеется, выполняется до вызова инструкции, а команда изменения этой переменной размещается в теле цикла. Поэтому инструкция while() более гибкая в плане возможных вариантов ее использования. Все, что запрограммировано с помощью инструкции for(), может быть запрограммировано и с помощью инструкции while().

**Инструкция do-while()**

Инструкция do-while() является модификацией инструкции while(). Синтаксис ее вызова такой:

package com.jolken;  
  
public class DoWhile {  
 public static void main(String[] args) {  
 int a = 0;  
 do {  
 a++;  
 } while (a < 10);  
 }  
}

Выполнение инструкции начинается с блока команд цикла, размещенных в фигурных скобках после ключевого слова do. Затем проверяется условие, указанное в круглых скобках после ключевого слова while. Если условие истинно, выполняются команды цикла и снова проверяется условие и т. д. Таким образом, хотя бы один раз команды цикла будут выполнены — в этом отличие инструкции do-while() от инструкции while().

**Метки и инструкции break() и continue()**

С инструкцией break() мы уже сталкивались. В общем случае она завершает работу инструкции цикла или инструкции выбора. Если после инструкции break() указать метку, то управление передается команде, размещенной за помеченной этой меткой инструкцией цикла, выбора или блоком команд (напомним, блок команд заключается в фигурные скобки).

Инструкция continue() используется для завершения текущего цикла в инструкции цикла и переходу к выполнению следующего цикла. Если после инструкции continue() указать метку, то выполняется переход для выполнения итерации помеченного меткой внешнего цикла. Таким образом, инструкция continue() с меткой применяется только тогда, когда имеются вложенные циклы.

package com.jolken;  
  
public class BreakContinue {  
 public static void main(String[] args) {  
 for (int i = 0; i < 98; i++) {  
 if (i < 9) {  
 continue;  
 }  
 if (i == 25) {  
 break;  
 }  
 System.*out*.println(i);  
 }  
 //Программа выведет числа от 9 до 24  
 }  
}

просмотреть примеры программ на страничке 63-79

выполнить задачи 1.11, 1.25-1.61, 4.6-4.33