Урок 5

Ветвления

Условные конструкции.

Ветвления

В программном коде, как и в жизни, множество решений зависит от внешних факторов: «Если случится событие А, я выполню действие Б». По такому принципу строится ветвление во всех языках программирования.

В программировании для ветвления применяются специальные операторы, обеспечивающие выполнение определенной команды (или набора команд) только при условии истинности логического выражения (группы выражений). Ветвление — одна из трех базовых конструкций структурного программирования (наряду с последовательным выполнением команд и циклом).

Для справки: *в дискретной математике (фундаментальной науке, лежащей в основе программирования) условие ветвления — это предикат. Почитать об этом можно в дополнительной литературе.*

Операторы if, if-else

Для реализации ветвления в Java используется оператор **if**:

```
if (условие) {
   последовательность_операторов;
}
```

Условие — это любое выражение, возвращающее булевское значение (**true, false**); то есть вопрос, на который можно ответить только «да» или «нет». Действие выполняется, когда условие истинно (true). Например:

```
if (5 < 10) {
   System.out.println("5 меньше 10");
}
```

В данном примере числовое значение 5 меньше 10, и поэтому условное выражение принимает логическое значение **true**. Следовательно, выполняется метод **println()**.

Рассмотрим пример с противоположным условием:

```
if (10 < 5) {
   System.out.println("Это сообщение никогда не будет выведено");
}
```

Теперь числовое значение 10 не меньше 5 — следовательно, метод **println()** не вызывается и в консоль ничего не выводится.

Операторы сравнения:

Оператор	Значение
<	Меньше
<=	Меньше или равно
>	Больше
>=	Больше или равно
==	Равно
!=	Неравно

Если одного условия недостаточно, применяют ветвление, где в случае истины выполняется одно действие, а иначе — другое:

```
if  ((условие1  &&  условие2)  ||
условие3)  {
  последовательность операторов 1
} else {
  последовательность операторов 2
}
```

Реализуем простой пример:

```
int x = 5;
int y = 42;
if( x > y ) {
   System.out.println(x + y);
}
else {
   System.out.println(x * y);
}
```

Не всегда можно уложить логику ветвления в две ветки, но Java позволяет разделять программу на любое количество вариантов с помощью конструкции **else if**, которая позволяет анализировать дополнительное условие. При этом выполняться будет первое условие, вернувшее **true**.

Представим задачу: даны два произвольных числа, необходимо вывести на экран их соотношение. По сути, будет три варианта: первое число больше второго, второе больше первого или они равны:

```
int x = 5;
int y = 42;
if( x > y )
   System.out.println("$x больше $y");
else if ( x < y )
   System.out.println("$x меньше $y");
```

```
else
   System.out.println("$x равен $y");
```

Оператор Switch

В отличие от **if-else**, оператор **switch** применяется к заранее известному количеству возможных ситуаций:

```
switch(ВыражениеДляСравнения) {
    case Совпадение1:
            команда;
                break;
        case Совпадение2:
            команда;
                break;
        case Совпадение3:
            команда;
                break;
        default:
            оператор;
            break;
}
```

Параметр «**ВыражениеДляСравнения**» может принимать значения простых типов **byte**, **short**, **char**, **int**. С версии Java 7 можно использовать **Enum** и **String**.

Дублирование значений **case** не допускается. Тип каждого значения должен быть совместим с типом выражения.

Если обнаруживается совпадение, исполняется команда или их набор, прописанный за данным оператором. Если совпадений не будет, выполняется команда после ключевого слова **default**. Но оператор **default** не является обязательным. В этом случае при отсутствии совпадений программа не выполняет действий.

Каждая секция **case** обычно заканчивается командой **break**, которая передает управление к концу команды **switch**. При отсутствии у **case** команды **break** выполнение кода продолжится на следующем **case** до того момента, пока не встретится **break** или не закончатся условия. Иногда это используется на практике:

```
int month = 3;

String monthString;

switch (month) {

    case 1:  monthString = "Январь";

             break;

    case 2:  monthString = "Февраль";

             break;

    case 3:  monthString = "Март";

             break;

    case 4:  monthString = "Апрель";

             break;

    case 5:  monthString = "Май";

             break;

    case 6:  monthString = "Июнь";

             break;

    case 7:  monthString = "Июль";

             break;

    case 8:  monthString = "Август";

             break;

    case 9:  monthString = "Сентябрь";

             break;

    case 10: monthString = "Октябрь";

             break;

    case 11: monthString = "Ноябрь";

             break;

    case 12: monthString = "Декабрь";

             break;
```

```
    default: monthString = "Не знаем такого";
             break;
}
mInfoTextView.setText(monthString);
```

```
int month = 2;
int year = 2012;
int numDays = 0;

switch (month) {
case 1:
case 3:
case 5:
case 7:
case 8:
case 10:
case 12:
      numDays = 31;
      break;
case 4:
case 6:
case 9:
case 11:
      numDays = 30;
      break;
case 2:
      if (((year % 4 == 0) && !(year % 100 == 0)) || (year % 400 == 0))
            numDays = 29;
```

```
      else

            numDays = 28;

      break;

default:

      mInfoTextView.setText("Несуществующий месяц");

      break;

}

mInfoTextView.setText("Число дней = " + numDays);
```