ОПЕРАТОРЫ ПРИНЯТИЯ РЕШЕНИЙ (IF, SWITCH)
ЦИКЛЫ (WHILE, DO WHILE, FOR)
ОПЕРАТОРЫ ВЕТВЛЕНИЯ (BREAK, CONTINUE)

Операторы принятия решений используются, когда в зависимости от условия необходимо выполнить разные действия.

Оператор if

```
if (выражение) {
    инструкции, которые выполнятся, если выражение true,
    в противном случае данный блок игнорируется
}
```

```
if (answer == 13) { // если значение  переменной answer равно 13
    // в консоль будет выведено: Ответ верный
    System.out.println("Ответ верный");
}
```

```
// { } можно опустить, если инструкция, соответствующая if одна
if (done) System.out.println("Завершено");
```

Необязательный блок else

```
if (выражение) {
    инструкции, которые выполнятся, если выражение true
} else {
    инструкции, которые выполнятся, если выражение false
}
```

```
if (answer == 13) {  /* если значение переменной answer равно 13
    в консоль будет выведено: Ответ правильный */
    System.out.println("Ответ правильный");
} else { /* если значение переменной answer не равно 13
    в консоль будет выведено: Ошибка в ответе */
    System.out.println("Ошибка в ответе");
}
```

```
// фигурные скобки можно опустить
if (done) System.out.println("Завершено");
else System.out.println("Действие не было завершено");
```

Несколько условий else if

Используются, если необходимо добавить новые варианты условий.
Каждое новое выражение будет проверяться только, если предыдущие ложны

```
 1 if (выражение 1) {
 2     инструкции, которые выполнятся, если выражение 1 true
 3 } else  if (выражение 2){
 4     инструкции, которые выполнятся,
 5     если выражение 1 false и выражение 2 true
 6 } else  if (выражение 3){
 7     инструкции, которые выполнятся,
 8     если выражение 1 false, выражение 2 false и выражение 3 true
 9 } else {
10     инструкции, которые выполнятся, если все выражения false
11 }
```

Несколько условий else if

```
if (answer == 13) { /* если значение переменной answer равно 13
    в консоль будет выведено: Ответ правильный */
    System.out.println("Ответ правильный");
} else if (answer < 13) { /* если значение переменной answer меньше 1
    в консоль будет выведено: Попробуйте число больше */
    System.out.println("Попробуйте число больше");
} else {  /* если все условия будут ложными
    в консоль будет выведено: Попробуйте число меньше */
    System.out.println("Попробуйте число меньше");
}
```

```
// фигурные скобки можно опустить
if (done) System.out.println("Ответ правильный");
else if (answer < 13) System.out.println("Попробуйте число больше");
else System.out.println("Попробуйте число меньше");
```

Конструкция switch

```
switch(выражение/переменная) {
    case значение1:
         инструкции
         [break]
    case значение2:
         инструкции
         [break]
    case значение3:
    case значение4:
        инструкции
        [break]
    default:
        инструкции, выполнятся, если ни один case не совпал
        [break]
}
```

Switch работает со следующими типами данных:

- byte / Byte
- short / Short
- int / Integer
- char / Character
- enum
- String

Выражение/переменная проверяется на равенство со значениями в case.
Если соответствие установлено инструкции начинают выполняться от соответствующего case и далее до ближайшего break или до конца switch.

Конструкция switch. Пример

```
String item = “какое-то значение”;
switch (item) {
    case "Oranges": // если item == "Oranges",
        System.out.println("Oranges - $0.59 a pound.");
    case "Apples": // если item == "Apples",
        System.out.println("Apples - $0.32 a pound.");
        break;
    case "Mangoes": // если item == "Mangoes"
    case "Papayas": // если item == "Papayas"
        System.out.println("Mangoes and papayas are $2.79 a pound.");
        break;
    default:
        System.out.println("Sorry, we are out of " + item + ".");
}
```

Циклы

Позволяют выполнять однотипное действие несколько раз.
Каждое повторение цикла называется итерацией.

- Цикл с предусловием while
- Цикл с постусловием do while
- Цикл for

Цикл с предусловием while

Конструкция while выполняет инструкции в фигурных скобках до тех пор, пока логическое выражение имеет истинное значение (true). Этот параметр является условием выполнения цикла. Истинность логического выражения (условия) проверяется перед каждым выполнением тела цикла.

```
1 while(логическое выражение) { /* проверка выражение на true */
2     тело цикла выполняется, если выражение истинно
3 } /* если выражение ложно, программа выходит из цикла
4      и продолжает выполнение инструкций после цикла */
```

Цикл с предусловием while

```
1 // бесконечный цикл
2 while(true) {
3     инструкции
4 }
5 // цикл не совершит ни одной итерации
6 while(false) {
7     инструкции
8 }
```

Цикл с предусловием while. Пример.

```
1 int count = 3;
2 while (count > 0) {
3     System.out.println(count);
4     count--;
5 }
```

Первая проверка условия: count = 3, значит условие истинно и тело цикла выполнится.
Итерация 1 (первое выполнение тела цикла): в консоль будет выведено: 3, уменьшение значения count на 1, значит count будет равен 2
Вторая проверка условия: count = 2 (после уменьшения в теле цикла), значит условие истинно и тело цикла выполнится
Итерация 2 (второе выполнение тела цикла): в консоль будет выведено: 2, уменьшение значения count на 1, значит count будет равен 1
Третья проверка условия: count = 1(после уменьшения в теле цикла), значит условие истинно и тело цикла выполнится
Итерация 3 (третье выполнение тела цикла): в консоль будет выведено: 1, уменьшение значения count на 1, значит count будет равен 0
Четвертая проверка условия: count = 0 (после уменьшения в теле цикла), значит условие ложно и тело цикла не будет выполнено, программа выйдет из цикла.

Цикл с постусловием do while

Сначала выполнится тело цикла, затем проверяется логическое выражение, если оно истинно, тело цикла будет выполнятся еще раз. Так будет происходить, пока логическое выражение истинно, когда логическое выражение станет ложным, программы выйдет из цикла. Если логическое выражение изначально ложно, тело цикла выполняется один раз, т.к. условие проверяется после выполнения тела цикла.

```
1 do {
2     тело цикла выполнится первый раз в любом случае,
3     далее будет выполняться, если логическое выражение истинно
4 } while (логическое выражение);
```

Цикл с постусловием do while. Пример.

```
1 int count = 2;
2 do { // тело цикла:
3     System.out.println(count);  // вывод в консоль:  значение count
4     count--; // уменьшение значения count на 1
5 } while (count > 0);
```

Итерация 1 (первое повторение тела цикла):
в консоль будет выведено: 2
уменьшение значения count на 1, значит count будет равен 1
Первая проверка условия: count = 1, значит условие истинно и тело цикла выполнится.
Итерация 2 (второе повторение тела цикла):
в консоль будет выведено: 1
уменьшение значения count на 1, значит count будет равен 0
Вторая проверка условия: count = 0, значит условие ложно и тело цикла выполняться не будет, программа выйдет из цикла

Цикл for

[] - квадратные скобки в описании используются для обозначения необязательных параметров

```
1 for ( [ начало-инициализация ]; [ условие ]; [ шаг ] ) {
2     тело цикла  выполняется, если условие истинно
3 }
```

Начало-инициализация выполняется один раз, при входе в цикл. Обычно здесь инициализируются один или несколько счётчиков.

Условие проверяется перед каждой итерацией, если оно истинно, тело цикла выполняется, если ложно, программа выходит из цикла.

Шаг выполняется после выполнения тела цикла, но перед проверкой условия.

Цикл for. Пример.

```
1 int count = 3;
2 for (int i = 0; i < count ; i++) {
3     System.out.println(count);
4     count--;
5 }
```

Начало цикла: int i = 0 - инициализация счетчика
Первая проверка условия: i < count, значение i равно 0, а значение count равно 3, получаем 0 < 3, значит условие истинно и тело цикла выполнится.
Итерация 1 (первое выполнение тела цикла): в консоль будет выведено: 3, уменьшение значения count на 1, значит count будет равен 2
Первое обновление счетчика (шаг): i++ - значение i станет равно 1
Вторая проверка условия: i < count, значение i равно 1(после первого обновления счетчика), а count равно 2 (после уменьшения в теле цикла), получаем 1 < 2, значит условие истинно и тело цикла выполнится.
Итерация 2 (второе выполнение тела цикла): в консоль будет выведено: 2 уменьшение значения count на 1, значит count будет равен 1.
Второе обновление счетчика (шаг): i++; значение i станет равно 2
Третья проверка условия: i < count, значение i равно 2(после второго обновления счетчика), а значение count равно 1 (после уменьшения в теле цикла), получаем 2 < 1, значит условие ложно и тело не выполнится, программа выйдет из цикла

Break и Continue

- break позволяет выйти из цикла в любой момент и продолжить выполнение кода после цикла.
- continue прекращает выполнение текущей итерации цикла.
- метки позволяют указать из какого цикла необходимо выйти (в случае break) или какой включающий цикл следует продолжить (в случае continue).

Break и Continue. Пример

```
1 outer: while (true) {
2     for (int i = 0; i < 100; i++) {
3         if (i % 2 == 0) continue; // относится к циклу for
4         System.out.println(i);
5         if (i == 51) break outer; // относится к циклу с меткой outer
6     }
7 }
```