НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Объектно-ориентированное программирование на языке Java

Yevhen Berkunskyi, NUoS
eugeny.berkunsky@gmail.com
http://www.berkut.mk.ua

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Основные языковые конструкции JAVA

- Условные конструкции: **if**, **if-else** и **switch**.
- Циклические конструкции : **while**, **do-while**, **for**, и ***усовершенствованный*** **for**.
- Конструкции передачи управления : **break**, **continue**, **return**, **try-catch-finally**, **throw**, and **assert**.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Условные конструкции

- неполная форма **if** оператора
- **if-else** оператор
- **switch** оператор

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Неполная форма if оператора

- Синтаксис неполной формы if оператора:

if (< условное выражение >) <оператор>

Примеры:

```
// emergency это логическая переменная
if (emergency) operate();

if (temperature > critical)
    soundAlarm();
```

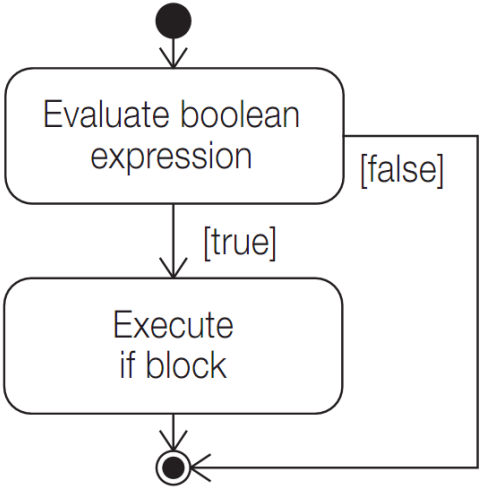
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

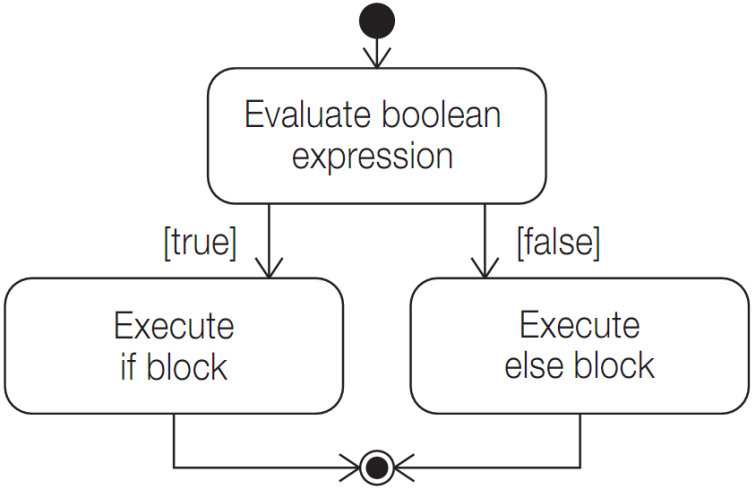
Неполная форма if оператора

Обратите внимание, что <оператор> может содержать несколько команд, в таком случае их необходимо заключить в фигурные скобки.

```
if (catIsAway()) {      // Block
  getFishingRod();
  goFishing();
}
```

Диаграмма активности (деятельности) для оператора if





Неполная форма if оператора

Обратите внимание, что блок if может быть любым оператором. В частности, это может быть пустой оператор (;) или пустой блок ({}). Общей ошибкой программирования является непреднамеренное использование пустого оператора.

```
if (emergency);
// Empty if block operate();
```

if-else оператор

Оператор if-else используется для выбора между двумя действиями, в зависимости от значения условия. Синтаксис :

```
if (<логическое выражение>)
    <оператор1>
else
    <оператор2>
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

if-else оператор

```
if (emergency)
  operate();
else
  joinQueue();
if (temperature > critical)
  soundAlarm();
else
  businessAsUsual();
```

```
if (catIsAway()) {
  getFishingRod();
  goFishing();
} else
  playWithCat();
```

вложенный if оператор

```
if (temperature >= upperLimit) {          //(1)
  if (danger)                    // (2) Simple if.
    soundAlarm();
  if (critical)                  // (3)
    evacuate();
  else // Goes with if at (3).
    turnHeaterOff();
} else // Goes with if at (1).
    turnHeaterOn();
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Использование блок-нотации {}

```
// (A): Block notation
if (temperature > upperLimit) { // (1)
  if (danger) soundAlarm();     // (2)
} else                          // Goes with if at (1).
  turnHeaterOn();
// (B): Without block notation
if (temperature > upperLimit)   // (1)
  if (danger) soundAlarm();     // (2)
else turnHeaterOn();            // Goes with if at (2).
// (C):
if (temperature > upperLimit)   // (1)
  if (danger)                   // (2)
    soundAlarm();
  else                          // Goes with if at (2).
    turnHeaterOn();
```

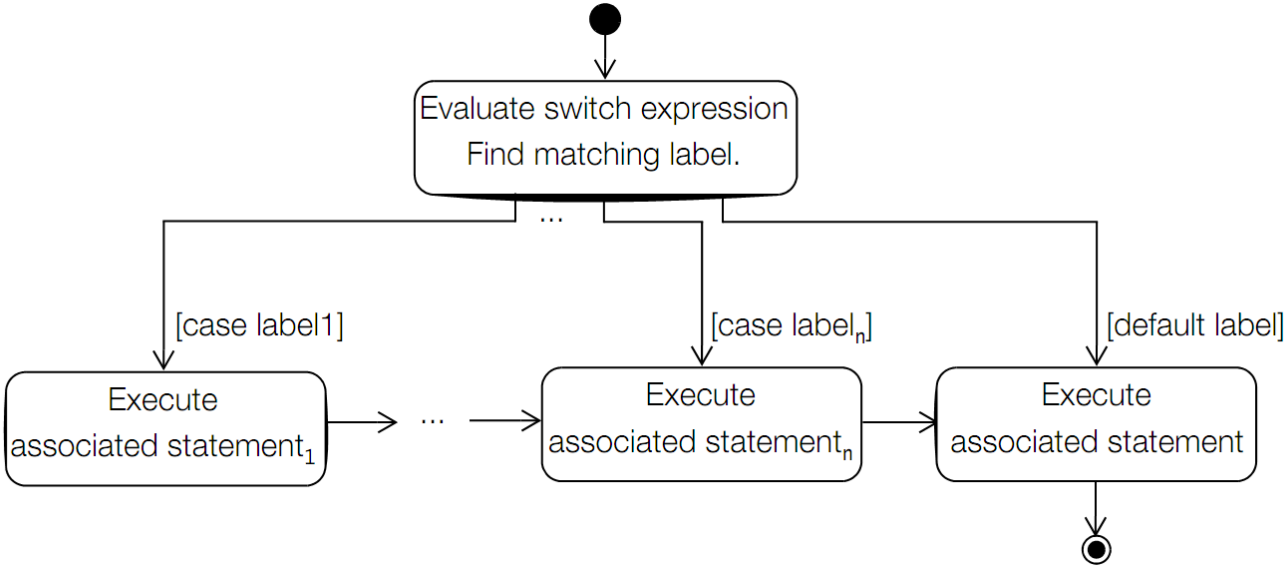
Использование блок-нотации {}

```
if (temperature >= upperLimit) {                    // (1)
  soundAlarm();
  turnHeaterOff();
} else if (temperature < lowerLimit) {              // (2)
  soundAlarm();
  turnHeaterOn();
} else if (temperature == (upperLimit+lowerLimit)/2) {
                                                    // (3)
    doingFine();
} else                                              // (4)
  noCauseToWorry();
```

switch оператор

```
switch (<выражение>) {
   case label1: <оператор1>
   case label2: <оператор2>
   ...
   case labeln: <операторn>
   default: <оператор>
} // end switch
```

Диаграмма активности (деятельности) для оператора switch



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Принцип работы switch оператора

```
public class Advice {
  public final static int LITTLE_ADVICE  = 0;
  public final static int MORE_ADVICE    = 1;
  public final static int LOTS_OF_ADVICE = 2;
  public static void main(String[] args) {
    dispenseAdvice(LOTS_OF_ADVICE);
  }
……// in the next slide
}
```

```
public static void dispenseAdvice(int howMuchAdvice){
    switch(howMuchAdvice) {                    // (1)
      case LOTS_OF_ADVICE:
        System.out.println("See no evil.");   // (2)
      case MORE_ADVICE:
        System.out.println("Speak no evil.");// (3)
      case LITTLE_ADVICE:
        System.out.println("Hear no evil."); // (4)
        break;                                 // (5)
      default:
        System.out.println("No advice.");     // (6)
    }
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Использование break в switch

```
public static String digitToString(char dig) {
    String str = "";
    switch(dig) {
      case '1': str = "one";   break;
      case '2': str = "two";   break;
      case '3': str = "three"; break;
      case '4': str = "four";  break;
      case '5': str = "five";  break;
      case '6': str = "six";   break;
      case '7': str = "seven"; break;
      case '8': str = "eight"; break;
      case '9': str = "nine";  break;
      case '0': str = "zero";  break;
      default:
      System.out.println(dig + " is not a digit!");
    }
    return str;
  }
```

Циклические операторы

Java содержит четыре языковые конструкции для построения цикла:

- **while** оператор
- **do-while** оператор
- ***basic* for** оператор
- ***усовершенствованный* for** оператор

while оператор

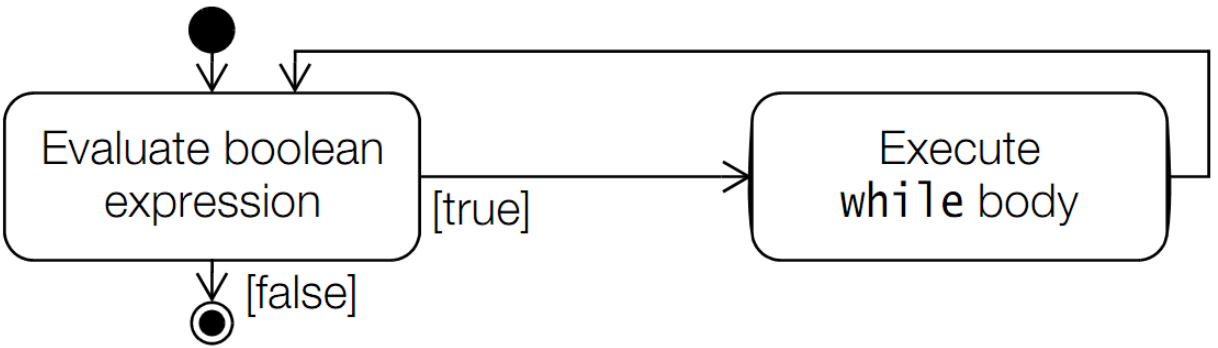
Синтаксис:

```
while (< условие цикла >)
    <тело цикла>
```

Условие <условие цикла> выполняется перед выполнением <тела цикла>. Оператор while выполняет <тело цикла>, пока условие <условие цикла> истинно.

Когда условие <условие цикла> становится ложным, цикл завершается, и управление передается на оператор сразу после цикла.

Диаграмма активности (деятельности) для оператора `while`



Оператор while обычно используется, когда количество итераций неизвестно.

```
while (noSignOfLife())
   keepLooking();
```

while оператор

Поскольку <тело цикла> может быть любым допустимым оператором, непреднамеренное завершение каждой строки пустым оператором (;) может привести к непредвиденным результатам.

Всегда используйте оператор блока, {...}, поскольку это поможет избежать таких проблем.

```
//пустое тело цикла!
while (noSignOfLife());
  keepLooking();
// нет оператора в теле цикла.
```

do-while оператор

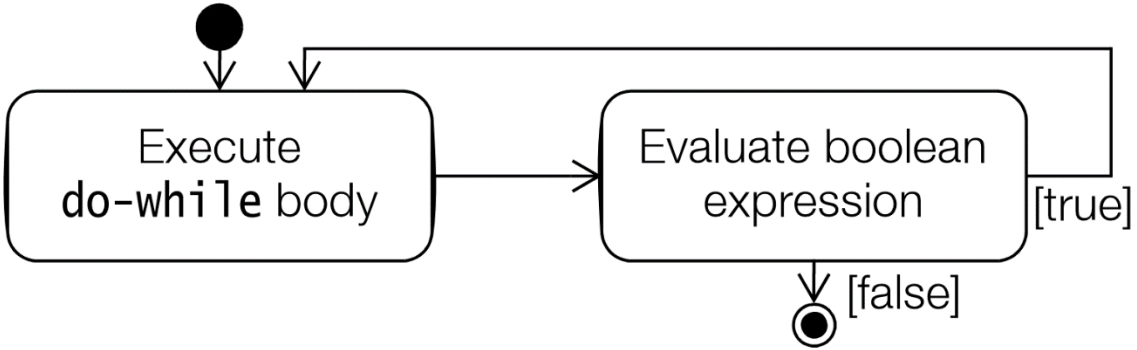
Синтаксис:

```
do
<тело цикла>
while (<условие цикла>);
```

Условие <условие цикла> проверяется после выполнения <тела цикла>. Оператор do-while выполняет <тело цикла> до тех пор, пока условие <условие цикла> не станет ложным.

Когда условие <условие цикла> становится ложным, цикл завершается, и выполнение продолжается с оператором, стоящим сразу после цикла.

Диаграмма активности (деятельности) для оператора do-while



while и do-while

<Тело цикла> в цикле do-while всегда является блоком оператора. Сравним циклы while и do-while.

В приведенных ниже примерах мыши, возможно, никогда не смогут играть, если кошка не ушла, как в цикле в (1). Мышам приходится играть хотя бы один раз (рискуя потерять жизнь) в цикле (2).

```
while (cat.isAway()) {        // (1)
  mice.play();
}

do {                          // (2)
  mice.play();
} while (cat.isAway());
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

The for(;;) оператор

Цикл for (;;) является наиболее общим из всех циклов. Он в основном используется когда количество итераций известно заранее.

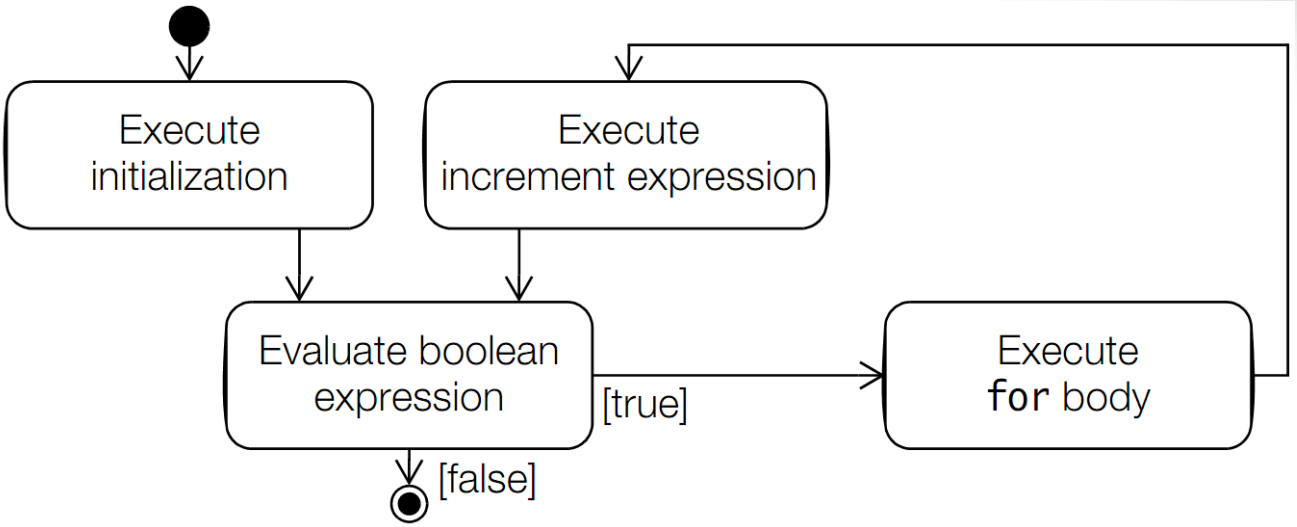
Синтаксис :

```
for (<инициализатор>;
     <условие цикла>;
     <инкрементное выражение>)
   <тело цикла>
```

Семантика for(;;)

```
< инициализатор >
while (< условие цикла >) {
   < тело цикла >
   < инкрементное выражение >
}
```

Диаграмма активности (деятельности) для оператора for



Примеры с оператором for

```
int sum = 0;
int[] array = {12, 23, 5, 7, 19};
for (int index = 0; index < array.length; index++) // (1)
  sum += array[index];

for (int i = 0, j = 1, k = 2; ... ; ...) ...;      // (2)

for (int i = 0, String str = "@"; ... ; ...) ...;  // (3)
//error.

int i, j, k; // Variable declaration
for (i = 0, j = 1, k = 2; ... ; ...) ...;          // (4)
//Only initialization
```

Примеры с оператором for

Нельзя объединять объявления переменных с операторами в разделе <initialization>, как в случае (5) в следующем примере. Факторинг объявления переменной, как в (6), позволяет применять список выражений, разделенных запятыми.

```
// (5) Нерабочий код :
for (int i = 0, System.out.println("not legal!");
         flag; i++) { //Error!
  // loop body
}

// (6) Рабочий код:
int i;        // declaration factored out.
for (i = 0, System.out.println("legal!");
     flag; i++) {  // OK.
  // loop body
}
```

Примеры с оператором for

Приращение <increment> также может быть списком выражений, разделенным запятыми. Следующий код указывает цикл for (;;), который содержит список из трех переменных в разделе <initialization>, разделенный запятыми, и список из двух выражений, разделенных запятыми, в разделе <increment expression>:

Примеры с оператором for

```
// Legal usage but not recommended.
int[][] sqMatrix = { {3, 4, 6}, {5, 7, 4}, {5, 8, 9} };
for (int i = 0,
     j = sqMatrix[0].length - 1,
     asymDiagonal = 0;                    // initialization
     i < sqMatrix.length;                 // loop condition
     i++, j--) // increment expression
  asymDiagonal += sqMatrix[i][j];         // loop body
```

Оператор `for(;;)`

Все разделы заголовка for (;;) являются необязательными. Любой или все из них могут быть пустыми, но две точки с запятой являются обязательными. В частности, отсутствие условия <loop condition> означает, что условие цикла является истинным.

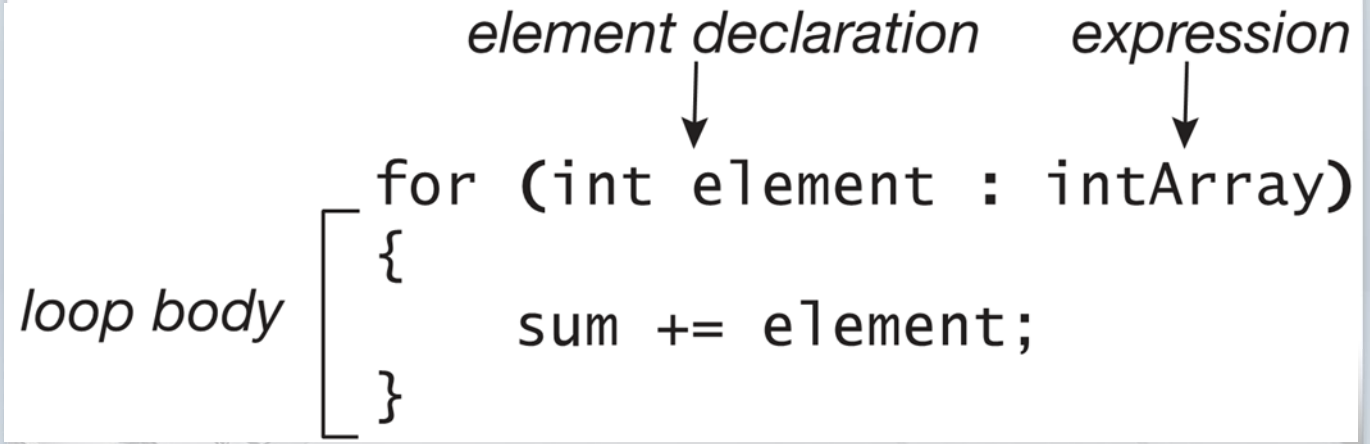
«Краб», (;;) обычно используется для построения бесконечного цикла, где окончание, по-видимому, достигается посредством кода в теле цикла (см. Следующий раздел о операторах передачи):

```
for (;;) Java.programming();
// Infinite loop
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Оператор for(:)

Улучшенный цикл более удобный, когда нам нужно перебирать массив или коллекцию, особенно когда необходимо выполнить некоторую операцию для каждого элемента массива или коллекции.



Оператор for(:)

Element является локальной переменной для блока цикла и недоступна после завершения цикла.

Кроме того, изменение значения текущей переменной не изменяет никакого значения в массиве.

Тело цикла, которое может быть простым оператором или составным, выполняется для каждого элемента массива, и нет опасности возникновения каких-либо ошибок выхода за границы.

Конструкции передачи управления

Java предоставляет шесть языковых конструкций для передачи управления в программе :

- break
- continue
- return
- try-catch-finally
- throw
- assert

метки

У оператора может быть метка.

<label>: <statement>

Метка - это идентификатор, и он всегда предшествует оператору.

Имена меток существуют в их собственном пространстве имен, так что они не конфликтуют с именами пакетов, классов, интерфейсов, методов, полей и локальных переменных.

метки

Оператор может иметь несколько меток:

LabelA: LabelB:

System.out.println (? "Mutliple labels. Используйте разумно.");

Перед объявлением переменной не может быть метки:

L0: int i = 0; // Ошибка времени компиляции.

Помеченный оператор выполняется так, как если бы он был без метки, если только это не оператор break или continue.

оператор break

Оператор break имеет две формы:
с меткой и без метки.

```
break;          // the unlabeled form
break <label>; // the labeled form
```

break без метки

Оператор break без метки используется в циклических конструкциях

(for(;;), for(:), while, do-while)

а также в операторе switch и управляющих конструкциях.

Принцип работы : остальная часть тела конструкции пропускается, а выполнение передается оператору, следующему за конструкцией.

break с меткой

Оператор break с меткой может использоваться для завершения любого помеченного оператора, содержащего оператор break.

Затем элемент управления переносится в конструкцию, которая следует после помеченного оператора.

В случае помеченного блока остальная часть блока пропускается, и выполнение продолжается с оператором, следующим за блоком:

break с меткой

```
out:
{ // (1) Labeled block
  // ...
  if (j == 10) break out;
  // (2) Terminate block. Control to (3).
  System.out.println(j);
  // Rest of the block not executed if j == 10.
  // ...
}
// (3) Continue here.
```

Оператор `continue`

Как и оператор break, оператор continue также имеет две формы: с меткой и без метки.

```
continue;                 // the unlabeled form
continue <label>;         // the labeled form
```

Оператор `continue`

Оператор continue может использоваться только в for (;;), для for(:), while или do-while, чтобы преждевременно остановить текущую итерацию тела цикла и при необходимости продолжить следующую итерацию.

Оператор `continue`

- В случае циклов while и do-while остальная часть тела цикла пропускается, то есть останавливает текущую итерацию, причем выполнение продолжается с проверки условия цикла <**loop condition**>.
- В случае цикла for (;;) остальная часть тела цикла пропускается, причем выполнение продолжается с выражением <**increment**>.

Оператор `return`

Оператор return используется, чтобы остановить выполнение метода и передать управление обратно вызывающему коду.

Использование двух форм возвращаемого выражения продиктовано тем, что метод может быть void или возвращать результат определенного типа.

Оператор `return`

Форма return	В void методе	В не-void методе
return;	необязательный	Не допускается
return <выражение>;	Не допускается	Обязательно, если метод явно не прекращен

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Questions?