

Основные языковые конструкции Java

- В данной лекции рассмотрены основные языковые конструкции Java.
- Для организации циклов в Java предназначены три основных конструкции: `while`, `do`, `for`. Для изменения порядка выполнения операторов применяются `continue` и `break` (с меткой или без). Также существуют два оператора ветвления: `if` и `switch`.
- Основной конструкцией обработки исключительных ситуаций является `try-catch-finally`. Для явной инициализации исключительной ситуации служит ключевое слово `throw`.

Темы лекции 5. Основные языковые конструкции Java

- Нормальное и прерванное выполнение операторов
- Блоки и локальные переменные
- Пустой оператор
- Метки
- Оператор `if`
- Оператор `switch`
- Управление циклами
- Операторы `break` и `continue`
- Именованные блоки
- Оператор `return`
- Ошибки при работе программы. Исключения

Оператор if-else

Оператор `if - else` имеет вид

```
if (условие) {...}  
else {...}
```

Данный оператор полностью аналогичен условному оператору в C++.

```
if (i == 10)  
{ if (j < 20)  
    a = b;  
    if (k > 100)  
        c = d;  
    else  
        a = c; // else относится к if (k > 100)  
} else  
    a = d; // else относится к if (i == 10)
```

Тернарный оператор

```
int largerNum;  
int lowNum = 9;  
int highNum = 27;  
if(lowNum < highNum)           // если первое число  
                                // меньше второго  
    largerNum = highNum;  
else // иначе  
    largerNum = lowNum;
```

При тернарном варианте вычисляется выражение слева от знака вопроса. Если оно возвращает **true**, то берётся выражение слева от двоеточия, если возвращается **false**, то берётся выражение справа от двоеточия.

```
int lowNum = 9;  
int highNum = 27;  
int largerNum = lowNum < highNum ? highNum :  
    lowNum;
```

Цепочка вложенных if-else

```
int month = 3; // март
String season; // время года
if(month == 1 || month == 2 || month == 12)
    season = "Зимушка-зима";
else if (month == 3 || month == 4 || month == 5)
    season = "Весна";
else if (month == 6 || month == 7 || month == 8)
    season = "Лето";
else if (month == 9 || month == 10 || month == 11)
    season = "Осень";
else
    season = "Вы с какой планеты?";

textViewInfo.setText("Мартовские песни коты
    поют, когда на дворе " + season);
```

Оператор switch

Оператор `switch` имеет вид

```
switch (целочисленное выражение или enum) {  
    case метка1: оператор1, оператор2, ..., break;  
    case метка2: оператор1, оператор2, ..., break;  
    .....  
    default: .....  
}
```

Данный оператор полностью аналогичен условному оператору в C++. В Java 7 в `switch` можно использовать класс `String`.

Важные свойства оператора **switch**:

- Оператор **switch** отличается от оператора **if** тем, что может выполнять проверку только равенства, а оператор **if** может вычислять результат булева выражения любого типа.
- Две константы **case** в одном и том же операторе **switch** не могут иметь одинаковые значения
- Оператор **switch** эффективнее набора вложенных операторов **if**

Пример:

```
public String
getTypeOfDayWithSwitchStatement (String
dayOfWeekArg) {
    String typeOfDay;
    switch (dayOfWeekArg) {
        case "Monday":    typeOfDay = "Start of work
                                week"; break;
        case "Tuesday": case "Wednesday": case
            "Thursday": typeOfDay = "Midweek";
            break;
        case "Friday":    typeOfDay = "End of work
                                week"; break;
        case "Saturday": case "Sunday":
            typeOfDay = "Weekend"; break;
        default: typeOfDay = "Invalid day of
                                the week: " + dayOfWeekArg);
    }
    return typeOfDay;
}
```

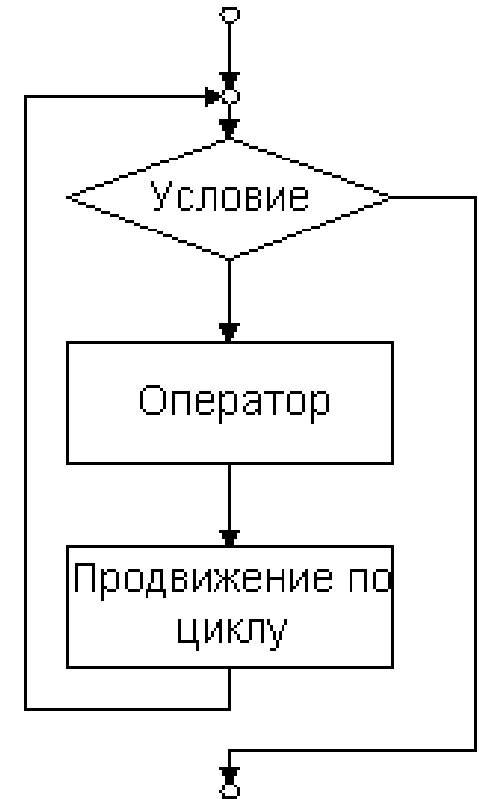
Циклы в Java

1. Цикл for

Цикл for имеет вид

```
for ( инициализация;  
      логическое выражение (условие);  
      шаг (итерация)  
    )  
{ //тело цикла }
```

```
System.out.println("Отсчёт пошёл: ");  
for (int i = 10; i > 10; i--)  
{  
    System.out.println(" " + i);  
}
```



Циклы в Java

2. Цикл `while`.

Цикл `while` имеет вид

```
while (условие) {....};
```

Данный цикл аналогичен циклу `while` в C++.

```
System.out.println("Отсчёт пошёл: ");  
int i = 10;  
while(i > 0)  
{  
    System.out.print(" " + i);  
    i--;  
}
```

Отсчёт пошёл: 10 9 8 7 6 5 4 3 2 1

Циклы в Java

3. Цикл `do - while`

Цикл `do - while` аналогичен циклу `while` в C++ и имеет вид

```
do {...} while (условие);
```

```
System.out.println("Отсчёт пошёл: ");  
int i = 10;  
do  
{  
    System.out.print(" " + i);  
    i--;  
} while(i > 0);
```

Отсчёт пошёл: 10 9 8 7 6 5 4 3 2 1

Вложенные циклы

```
int i, j;  
for (i = 0; i < 10; i++)  
{   for (j = i; j < 10; j++)  
        textInfo.append("*");  
    textInfo.append("\n");  
}
```

**

*

Циклы в Java.

4. Аналог цикла foreach

Цикл `for-each` развитие цикла `for` (идея взята из языка Python). Пример1:

```
int[] nums = { 1, 2, 3, 4, 5 };  
int sum = 0;
```

```
for(int i : nums)  
    sum += i;
```

Этот код эквивалентен коду:

```
int[] nums = { 1, 2, 3, 4, 5 };  
int sum = 0;
```

```
for(int i = 0; i < 5; i++)  
    sum += nums[i];
```

Пример 2:

```
ArrayList a=new ArrayList(10);  
for(int i=0;i<10;i++){  
    Integer n=new Integer(i);  
    a.add(i,n);}  
int sum=0;  
for(Object value:a){  
    sum+=((Integer)value).intValue();}
```

Таким образом, базовая структура цикла `for-each` имеет вид:

```
for (declaration : expression)  
    statement
```

`declaration` – это переменная, которая должна иметь тип, совместимый с каждым элементом списка, массива или коллекции, по которым производится итерация.

`expression` - это выражение. Оно должно вычислять что-либо, по чему можно делать итерацию. Результатом данного выражения может быть массив или тип класса, реализующего интерфейс `Collection`.

Оператор break

Оператор break применяется для выхода из любого блока.

Существует разновидность оператора break с меткой.
Рассмотрим пример:

```
private float[ ][ ] Matrix;
public boolean workOnFlag(float flag) {
    int y, x;
    boolean found = false;
    search: //search это метка
    for (y = 0; y < Matrix.length; y++) {
        for (x = 0; x < Matrix[y].length; x++) {
            if (Matrix[y][x] == flag) {
                found = true;
                break search; //оператор search с меткой.
            }
        }
    }
    if (!found)
        return false;
    else
        return true;}
```

Оператор `continue`

Оператор `continue` осуществляет переход в конец тела цикла и вычисляет значение управляющего логического выражения.

Этот оператор часто используется для пропуска некоторых значений в диапазоне цикла, которые должны игнорироваться.

```
int i=0;
while (i<10) {
.....
    if (i==5) continue;
//в случае если условие i==5 истина, операторы
    ниже выполняться не будут
}
```

Существует оператор `continue` с меткой.

Оператор continue

Рассмотрим пример использования continue

```
public class MyTest {  
    public static void main(String[] args) {  
        label1: for (int i = 0; i < 3; i++)  
        {  
            if (i == 1)  
                continue label1;  
            System.out.println("TEST ");  
        }  
    }  
}
```

На экране получим:

TEST
TEST

Оператор `return`

Оператор `return` завершает выполнение метода и передает управление в точку его вызова. Если метод не возвращает никакого значения, достаточно написать:

```
return;
```

Оператор `goto` в Java отсутствует. Но слово `goto` является зарезервированным.

Ошибки при работе программы.

Исключения

На примере считывания целого числа:

```
BufferedReader r = new BufferedReader(new
InputStreamReader(System.in));
System.out.println("Input number: ");
int n=0; // переменная, в которую считываем
try {
    n = Integer.parseInt(r.readLine());
} catch (NumberFormatException e)
{
    e.printStackTrace();
} catch (IOException e)
{
    e.printStackTrace();
}
```