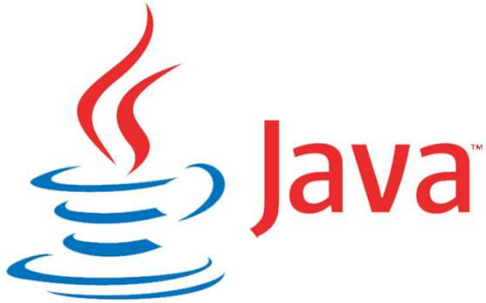
Тема 2.
Операторы управления



Условные конструкции

Одним из фундаментальных элементов многих языков программирования являются условные конструкции.

Данные конструкции позволяют направить работу программы по одному из путей в зависимости от определенных условий.

В языке программирования Java используются следующие условные конструкции:

- ***if … else***
- ***switch … case***

Условные конструкции

Конструкция if … else

Выражение if/else проверяет истинность некоторого условия, и в зависимости от результатов проверки выполняет определенный код:

```
int num1 = 6;
int num2 = 4;

if(num1 > num2){
    System.out.println(“The first number is bigger then the second one”);
}
```

Условные конструкции

Конструкция if … else

- После ключевого слова if ставится условие;
- Если это условие выполняется, то срабатывает код, который помещен в блоке if в {};
- В примере выше в качестве условия выступает операция сравнения двух чисел;
- Так как, в данном случае, первое число больше второго, то выражение num1 > num2 истинно и возвращает значение true;
- Следовательно, управление переходит в блок кода в фигурных скобках и начинает выполнять указанные там инструкции, т.е. метод System.out.println(“The first number is bigger then the second one”);
- Если бы первое число оказалось меньше второго, то блок кода условия if не выполнялся.

Условные конструкции

Конструкция if … else

- Но что делать, если необходимо, чтобы при несоблюдении условия также выполнялись какие-либо действия?
- В этом случае необходимо добавить блок else:

```
int num1 = 6;
int num2 = 4;

if(num1 > num2){
      System.out.println(“The first number is bigger then the second
one”);
} else{
     System.out.println(“The first number is less then the second one”);
}
```

Условные конструкции

Конструкция if … else

Но при сравнении чисел мы можем насчитать три состояния:

1. Первое число больше второго;
2. Первое число меньше второго;
3. Числа равны.

С помощью выражения else … if можно обрабатывать дополнительные условия:

Условные конструкции

Конструкция if … else

```
public static void main(String[] args) {
        int num1 = 6;
        int num2 = 8;

        if(num1 > num2){
           System.out.println("The first number is bigger thet the second one");
        }else if(num1 < num2){
           System.out.println("The first number is less then the second one");
        }else {
           System.out.println("Numbers are equal");
        }
 }
```

Условные конструкции

Конструкция if … else

Также можно соединять сразу несколько условий, используя логические операторы:

```
int num1 = 8;
int num2 = 6;

if(num1 > num2 && num1 > 7){
    System.out.println(“The first number is bigger then the second one
and bigger then 7”);
}
```

Условные конструкции

Тернарный оператор “?:”

В Java можно заменить конструкцию if … else на сокращенную форма – тернарный оператор:

```
public static void main(String[] args){
    int num1 = 8;
    int num2 = 6;

    String result = num1 > num2 ? “num1 > num2” : “num1 < num2”;
    System.out.println(result);
}
```

Условные конструкции

Конструкция switch

Конструкция switch/case аналогична конструкции if/else, так как позволяет обработать сразу несколько условий:

Условные конструкции

Конструкция switch

```
public static void main(String[] args) {
      int num = 8;
      switch (num) {
         case 1:
            System.out.println("Number is 1!");
            break;
         case 8:
            System.out.println("Number is 8!");
            break;
         case 9:
            System.out.println("Number is 9!");
            break;
         default:
            System.out.println("Number is not 1, 8 or 9");
      }
   }
```

Условные конструкции

Конструкция switch

- После ключевого слова switch в скобках идет сравниваемое выражение.
- Значение этого выражения последовательно сравнивается со значениями, помещенными после оператора case.
- Если совпадение будет найдено, то будет выполняться определенный блок case.
- Выражение в switch должно иметь тип char, byte, short, int, enum (начиная с Java 6) или String(начиная с Java 7)

Циклы

Еще одним видом управляющих конструкций являются циклы.

Циклы позволяют в зависимости от определенных условий выполнять определенное действие множество раз.

В языке Java есть следующие виды циклов:

- for
- while
- do … while

Циклы

Цикл for

Цикл for имеет следующее формальное определение:

```
for([инициализация счетчика]; [условие]; [изменение счетчика]){
    //тело цикла
}
```

Стандартный цикл for выглядит следующим образом:

```
for(int i = 1; i < 9; i++){
    System.out.println(i * i);
}
```

Циклы

Цикл for

- Первая часть объявления цикла int i = 1 создает и инициализирует счетчик i;
- Счетчик необязательно должен представлять тип int – это может быть и любой другой числовой тип, например, float;
- Перед выполнением цикла значение счетчика будет равно 1;
- В данном случае это то же самое, что и объявление переменной;
- Вторая часть – условие, при котором будет выполняться цикл;
- В данном примере цикл будет выполняться, пока i не достигнет 9;
- Третья часть – приращение счетчика на 1;
- Опять же, не обязательно увеличивать счетчик на 1, можно и уменьшать (i--);
- В итоге, блок цикла сработает 8 раз, пока значение i не станет равным 9;
- И каждый раз это значение будет увеличиваться на 1.

Циклы

Цикл for

Необязательно указывать все условия при объявлении цикла, такой вариант тоже корректен:

```
public static void main(String[] args) {
      int i = 1;
      for(;;){
         System.out.println(i * i);
      }
   }
```

Определение цикла осталось тем же, только теперь блоки в определении пустые for(;;) – нет инициализированной переменной-счетчика, нет условия, поэтому цикл будет работать вечно – **бесконечный цикл**.

Циклы

Цикл for

Либо можно опустить ряд блоков:

```
public static void main(String[] args) {
      int i = 1;
      for(;i < 9;){
         System.out.println(i * i);
         i++;
      }
   }
```

Этот пример эквивалентен первому примеру: здесь также есть счетчик, только создан он вне цикла. Есть и условие выполнения цикла, а также приращение счетчика уже в самом теле цикла for.

Циклы

Цикл for

Специальная версия цикла for предназначена для перебора элементов в наборах элементов, например, в массивах и коллекциях.

Она аналогична действия цикла foreach, который имеется в других языках программирования.

Ее формальное объявление выглядит следующим образом:

```
for(data_type  variable_name : container){
    //actions
}
```

```
int[] array = new int[]{1, 2, 3, 4, 5};
for(int a : array){
    System.out.println(a);
}
```

Циклы

Цикл for

То же самое можно сделать и с помощью обычной версии for:

```
int[] array = new int[]{1, 2, 3, 4, 5};
for(int i = 0; i < array.length; i++){
    System.out.println(array[i]);
}
```

В то же время эта версия цикла for более гибкая по сравнению с foreach.

В частности, в этой версии можно изменять элементы массива:

```
int[] array = new int[]{1, 2, 3, 4, 5};
for(int i = 0; i < array.length; i++){
    array[i] = array[i] * 2;
    System.out.println(array[i]);
}
```

Циклы

Оператор break

Иногда требуется выйти из цикла, не дожидаясь его завершения. В этом случае на помощь приходит оператор break:

```
int[] nums = new int[]{1, 2, 3, 4, 12, 9};
for(int i = 0; i < nums.length; i++){
    if(nums[i] > 10)
        break;
    System.out.println(nums[i]);
}
```

В данном случае два последних элемента не будут видны в консоли, так как когда nums[i] окажется больше 10 (то есть равно 12), сработает оператор break, и цикл завершится.

Циклы

Оператор continue

Теперь сделаем так, чтобы цикл не завершался, если число больше 10, а просто переходил к следующему элементу. Для этого используем оператор continue:

```
int[] nums = new int[]{1, 2, 3, 4, 12, 9};
for(int i = 0; i < nums.length; i++){
    if(nums[i] > 10)
        continue;
    System.out.println(nums[i]);
}
```

В данном случае, когда выполнение цикла дойдет до числа 12, которое не удовлетворяет условию проверки, программа просто пропустит это число и перейдет к следующему элементу массива.

Практика

Задача: Сделать цикл (for) от 1го до 10ти, каждая итерация которого будет выведена словами на экран. Например:

“один”

“два”

…

“десять”

Циклы

Цикл while

Цикл while сразу проверяет истинность некоторого условия, и если условие истинно, код цикла выполняется.

Например:

```
int j = 6;
while(j > 0){
    System.out.println(j);
    j--;
}
```

Практика

Задача: «С помощью цикла while посчитать сумму четных элементов от 10 до 55».

Циклы

Цикл do

Цикл do сначала выполняет код цикла, а потом проверяет условие в инструкции while. И пока это условие истинно, цикл повторяется.

Например:

```
int j = 7;
do{
    System.out.println(j);
    j--;
}while(j  > 0);
```

В данном случае код цикла сработает 7 раз, пока j не окажется равным нулю.

Циклы

Цикл do

Важно отметить, что цикл do гарантирует хотя бы однократное выполнение действий, даже если условие в инструкции while не будет истинно.

Например:

```
int j = -1;
do{
    System.out.println(j);
    j--;
}while(j  > 0);
```

Хотя переменная j изначально меньше 0, цикл все равно выполнится один раз.