Курс «Программирование на Java» - Типы данных. Переменные. Операторы

IT-Academy

Рассматриваемые вопросы

- Типы данных
- Переменные
- Нумерические литералы
- Виды операторов
- Приоритет операторов

IT-Academy

Типы данных

Типы данных в языке Java делятся на:

- Примитивные
- Ссылочные

В свою очередь примитивных типы можно разделить на:

- Целочисленные
- Дробные
- Символьные
- Логические

Все остальные типы - ссылочные.

IT-Academy

Примитивные типы данных

Тип	Размер	Мин	Макс	По умолчанию
byte	1 байт	-128	127	0
short	2 байта	-215	215 -1	0
int	4 байта	-231	231 -1	0
long	8 байт	-263	263 -1	0
float	4 байта	1.4*10-45	1.4*1038	0
double	8 байт	4.9*10-324	1.8*10308	0
char	2 байта	‘\u0000’	‘\uffff’	‘\u0000’
boolean	1 бит			FALSE

IT-Academy

Целые числа

Целочисленные типы данных используются для хранения целых чисел без дробной части. В Java есть 4 целочисленных типа данных: byte, short, int и long.

- byte: занимает 1 байт и может хранить целые числа от -128 до 127.
- short: занимает 2 байта и может хранить целые числа от -32,768 до 32,767.
- int: занимает 4 байта и может хранить целые числа от -2,147,483,648 до 2,147,483,647.
- long: занимает 8 байт и может хранить целые числа от -9,223,372,036,854,775,808 до 9,223,372,036,854,775,807.

Пример использования:

```
int a = 10;
long b = 123456789L;
```

Вещественные числа

Типы данных с плавающей точкой
Типы данных с плавающей точкой используются для хранения чисел с дробной частью. В Java есть 2 типа данных с плавающей точкой: float и double.

- float: занимает 4 байта и может хранить дробные числа с точностью до 7 знаков после запятой.
- double: занимает 8 байт и может хранить дробные числа с точностью до 15 знаков после запятой.

Пример использования:

```
float a = 3.14f;
double b = 1.23456789;
```

Логический тип данных

Логический тип данных boolean может принимать только два значения: true (истина) или false (ложь).

Пример использования:

```
boolean a = true;
boolean b = false;
```

Символьный тип данных

Символьный тип данных char используется для хранения одного символа Unicode. Он занимает 2 байта и может хранить любой символ из Unicode-таблицы.

Пример использования:

```
char a = 'A';
char b = '\u0041'; // тот же символ 'A', записанный в шестнадцатеричном формате Unicode
```

Ссылочаные типы данных

Ссылочные типы данных используются для хранения ссылок на объекты в памяти. Они включают классы, интерфейсы, массивы и т.д. При создании объекта ссылочного типа данных выделяется память для самого объекта и для его полей и методов. В переменной ссылочного типа данных хранится адрес объекта в памяти.

Пример использования:

```
String str = "Hello, World!";
int[] arr = {1, 2, 3, 4, 5};
```

Типы данных

В Java есть множество различных типов данных, каждый из которых имеет свои особенности и применение. При написании программы необходимо правильно выбирать тип данных для каждой переменной в зависимости от ее значения и ожидаемого диапазона значений.

```
// Объявление переменной типа int
int age;

// Инициализация значения
age = 18;

// Объявление и инициализация
int apples = 31;

// Объявление нескольких переменных одного типа
double a, b, c;

// Объявление и инициализация нескольких переменных
byte d = 10, e = 10;

// !!! Переменная должна быть обязательно объявлена
f = 44;
```

Переменные

Переменная – это именованная ячейка памяти, содержимое которой может изменяться.

При объявлении переменной сначала указывается тип переменной, а затем идентификатор задаваемой переменной.

Имя переменной не должно:
- начинаться на цифру
- содержать знаков пунктуации и пробелов
- быть одним из ключевых слов языка Java

Необходимо объявить все переменные, прежде чем их использовать

IT-Academy

Переменные

Переменная - это именованное место в памяти, которое содержит определенное значение.

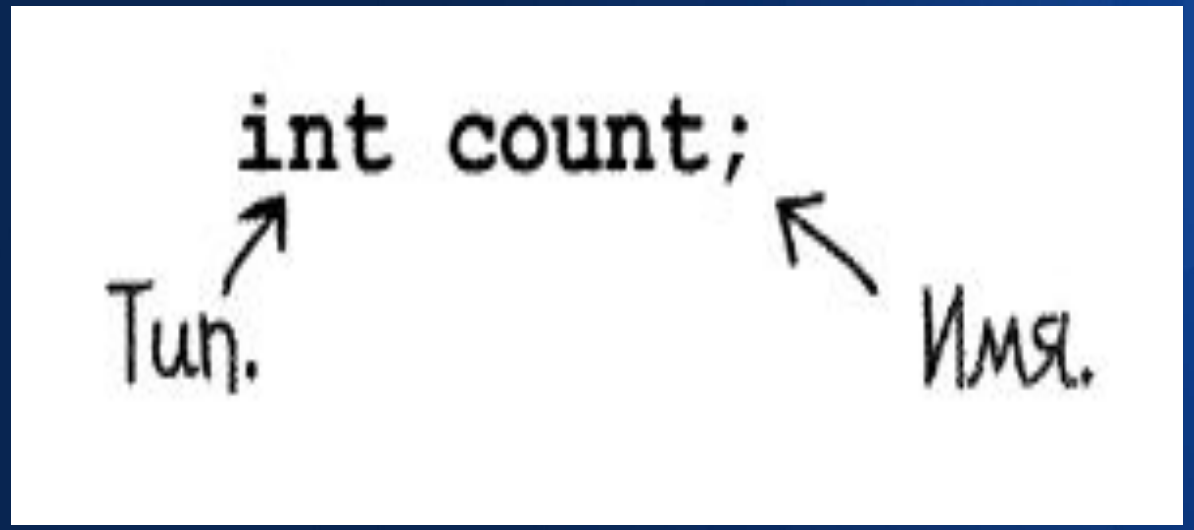
Думайте о переменных в Java как о чашках. Кофейных, чайных, больших пивных кружках, стаканах для попкорна, чашечках с симпатичными изогнутыми ручками, бокалах с металлической обводкой, которые ни при каких обстоятельствах нельзя ставить в микроволновую печь.

Переменная — это просто посуда. Контейнер. Она предназначена для *хранения*.

IT-Academy

Объявление переменной

Переменная дожна иметь ТИП.
Переменная дожна иметь ИМЯ.



IT-Academy

Именование переменных

- Имя должно начинаться с буквы, знака подчеркивания (_) или знака доллара ($). Нельзя начинать имя с цифры.
- После первого символа можно использовать числа без ограничений. Главное, не начинать имя с цифры.
- Вы вправе выбрать любое имя, следуя предыдущим двум правилам, но только если оно не совпадает с одним из зарезервированных ключевых слов (и других названий) в языке Java, которые распознаются компилятором.

Вспоминаем и обсуждаем

1. Есть два типа языков программирования. К какому относится Java?
2. Что такое Java JDK и IntelliJ IDEA ? Могу ли я обойтись без этого?
3. Что такое консольное приложение? Какие элементы входят в структуру программы?
4. Как создать первую программу на Java, выводящую на экран сообщение "Hello World!"? Какие комментарии можно использовать в коде? Как запустить программу?
5. Что такое алгоритм? Какие способы описания алгоритмов существуют?
6. Какой оператор используется для вывода информации на экран в Java? Какие аргументы он принимает?
7. Какие типы данных поддерживает язык Java? Какие основные отличия между ними?
8. Что такое переменная? Какие правила именования переменных существуют в Java? Какой тип данных следует выбрать для хранения числа с плавающей точкой?

Преобразование типов в Java

Что такое преобразование типов в Java и зачем оно нужно?

Преобразование типов в Java - это изменение типа значения переменной из одного типа в другой.

IT-Academy

Виды преобразования типов в Java

1) Явное (происходит при использовании оператора приведения типа)

2) Неявное (происходит автоматически)

Вспомним еще раз примитивные типы данных:

Тип	Диапазон	Значение по умолчанию	Размер	Примеры литералов
boolean	true или false	false	1 бит	true, false
byte	-128… 127	0	8 бит	1, -90, -128
char	Символ юникода или от 0 до 65 536	\u0000	16 бит	'a', '\u0031', '\201', '\n', 4
short	-32,768… 32,767	0	16 бит	1, 3, 720, 22,000
int	-2 147 483 648… 2 147 483 647	0	32 бит	-2, -1, 0, 1, 9
long	-9 223 372 036 854 775 808 до 9 223 372 036 854 775 807	0	64 бит	-4000L, -900L, 10L, 700L
float	3.40282347 x 1038, 1.40239846 x 10-45	0.0	32 бит	1.67e200f, -1.57e-207f, .9f, 10.4F
double	1.7976931348623157 x 10308, 4.9406564584124654 x 10-324	0.0	64 бит	1.e700d, -123457e, 37e1d

IT-Academy

Неявное преобразование типов

Из меньшего в большее!!!

Пример:

```
int a = 5;
double b = a; // неявное преобразование int в double
char c = 'a';
int d = c; // неявное преобразование char в int
short e = 10;
long f = e; // неявное преобразование short в long
```

Неявное преобразование типов

Из большего в меньшее!!!

Пример:

```
double g = 3.14;
int h = (int) g; // явное преобразование double в int

long i = 10000000000L;
int j = (int) i; // явное преобразование long в int
```

Важно отметить...

Для приведения типов перед присваиваемым выражением надо поставить нужный нам тип в круглых скобках (часто остальное выражение тоже приходится брать в скобки из-за приоритетов). При этом часть данных может теряться: например, при преобразовании из дробного числа в целое всегда отбрасывается дробная часть.

Немного практики...

Напишите программу, которая объявляет и инициализтрует целое и дробное, затем производит следующие операции:

1. Преобразование целого числа в дробное (с сохранением точности) и вывод результата на экран.
2. Преобразование дробного числа в целое (с отбрасыванием дробной части) и вывод результата на экран.

Константы

Константы - это значения, которые не могут быть изменены в процессе выполнения программы.

Константы упрощают код и уменьшают количество ошибок, связанных с изменением значений в процессе выполнения программы.

Константы

Константы — переменные, значения которых нельзя изменять. Выглядит создание неизменяемой переменной так же, как и создание обычной, только перед типом переменной нужно написать слово **final**:

```
final тип ИМЯ = значение;
```

Если присвоить final-переменной другое значение, ваша программа просто не скомпилируется.

Глобальные константы

Если вы решите объявить в своей программе глобальные константы, для этого нужно создать *статические переменные класса*, а также сделать их **public** и **final**.

```
class Solution {
  public static final String SOURCE_ROOT = "c:\\projects\\my\\";
  public static final int DISPLAY_WIDTH = 1024;
  public static final int DISPLAY_HEIGHT = 768;
}
```

Литералы в Java

Литералы - это значения, которые напрямую включаются в код программы. Например:

```
int age = 25; // целочисленный литерал
double price = 10.99; // литерал с плавающей точкой
char letter = 'a'; // символьный литерал
String message = "Hello, world!"; // строковый литерал
boolean flag = true; // литерал булевого типа
```

Операторы

Все операторы Java можно разделить на следующие группы

- Арифметические операторы
- Операторы сравнения
- Побитовые операторы
- Логические операторы
- Операторы присваивания
- Прочие операторы

Арифметические операторы

```
int a = 5, b = 40, c = 57, d = 70;
                                    // 45
System.out.println("a + b = " + (a + b));
                                    // -35
System.out.println("a - b = " + (a - b));
                                    // 200
System.out.println("a * b = " + (a * b));
                                    // 8
System.out.println("b / a = " + (b / a));
                                    // 0
System.out.println("b % a = " + (b % a));
                                    // 2
System.out.println("c % a = " + (c % a));
                                    // 5
  System.out.println("a++   = " + (a++));
                                    // 40
  System.out.println("b--   = " + (b--));
                                    // 70
  System.out.println("d++   = " + (d++));
                                    // 72
  System.out.println("++d   = " + (++d));
```

Используются в математических выражениях:

+ - * / % ++ --

IT-Academy

Арифметические операции

В Java доступны следующие арифметические операции:

- Сложение (+)
- Вычитание (-)
- Умножение (*)
- Деление (/)
- Остаток от деления (%)

Операция сложения (+)

- Операция сложения (+) выполняется для чисел и строк. Если хотя бы один операнд является строкой, то происходит конкатенация (слияние) строк.

Пример:

```
String str1 = "Hello";
String str2 = "World";
String result = str1 + " " + str2; // result = "Hello World"
```

Операция деления (/)

Операция деления (/) для целочисленных типов данных (int, long, short, byte) производит целочисленное деление, т.е. результат будет округлен до ближайшего целого числа.

Пример:

```
int a = 5;
int b = 2;
int c = a / b; // c = 2
```

Операция остаток от деления (%)

- Операция остаток от деления (%) возвращает остаток от деления первого операнда на второй.

Пример:

```
int a = 5;
int b = 2;
int c = a % b; // c = 1
```

Унарный минус (-)/плюс (+)

- Унарный минус (-) и плюс (+) используются для изменения знака числа.

Пример:

```
int a = 5;
int b = -a; // b = -5
```

Операция ++ и --

Операция ++ и -- в языке программирования Java используется для увеличения или уменьшения значения переменной на 1.

Операция ++ (инкремент) увеличивает значение переменной на 1. Например:

```
int x = 5;
x++; // x станет равным 6
```

Операция -- (декремент) уменьшает значение переменной на 1. Например:

```
int y = 10;
y--; // y станет равным 9
```

Еще немного практики...

У вас есть три переменные:

```
int a = 4;

int b = 2;

int c = 7;
```

Используя арифметические операции и правила порядка выполнения операций получите переменную со значение 18.
!!! Нужно использовать все возможные арифметические операции.

IT-Academy

```
                    int a = 60;    // 0b11_1100
                       int b = 13;    // 0b1101
    // 0b1111_1111_1111_1111_1111_1111_1010_1000
                                    int c = -88;
                                /* 12 = 0b1100 */
        System.out.println("a & b = " + (a & b));
                               /* 61 = 0b11_1101 */
        System.out.println("a | b = " + (a | b));
                               /* 49 = 0b11_0001 */
        System.out.println("a ^ b = " + (a ^ b));
  -61 = 0b1111_1111_1111_1111_1111_1111_1100_0011
              System.out.println("~a = " + (~a));
                              /* 240 = 0b1111_0000 */
       System.out.println("a << 2 = " + (a << 2));
                                /* 15 = 0b1111 */
       System.out.println("a >> 2  = " + (a >> 2));
                                /* 15 = 0b1111 */
     System.out.println("a >>> 2 = " + (a >>> 2));

// -22 = 0b1111_1111_1111_1111_1111_1111_1110_1010
       System.out.println("c >> 2 = " + (c >> 2));
                                 // 1073741802 =
        0b0011_1111_1111_1111_1111_1111_1110_1010
     System.out.println("c >>> 2 = " + (c >>> 2));
```

Побитовые операторы

& | ^ ~ << >> >>>

Побитовые операторы могут быть применены только для целочисленных типов:
int, long, short, char и byte

В Java побитовый оператор работает над битами и выполняет операцию бит за битом

IT-Academy

```
boolean a = true;
boolean b = false;

// false
System.out.println("a && b = " + (a && b));
// true
System.out.println("a || b = " + (a || b) );
// true
System.out.println("!(a && b) = " + !(a && b));

int x = 15;
//false
System.out.println(
x > 15 && x < 50 || x % 2 == 0);
```

Логические операторы

Используется для объединения условий

&& || !

IT-Academy

```
int a = 5, b = 10, c = 0;


c = a + b; // 15
c += a; // 20
c -= a; // 15
c *= a; // 75


a = 10; c = 15; c /= a; // 1
a = 10; c = 15; c %= a; // 5


c <<= 2 ; // 20
c >>= 2 ; // 5
c >>= 2 ; // 1
c &= a ; // 0
c ^= a ; // 10
c |= a ; // 10
```

Операторы присваивания

Используется для объединения условий

= += -= *= /= %=

<<= >>= &= ^= |=

IT-Academy

```
int a , b;
a = 10;
b = (a == 1) ? 22 : 33;
System.out.println( "Значение b: " +  b ); // 33
b = (a == 10) ? 55 : 811;
System.out.println( "Значение b: " + b ); // 55


String name = "George";
System.out.println(name instanceof Integer); //
false
System.out.println(name instanceof String); //
true


class Vehicle {/* ... */}
class Car extends Vehicle {/* ... */}
Vehicle a = new Car();
System.out.println(a instanceof Car); // true
System.out.println(a instanceof Vehicle);// true
System.out.println(a instanceof Object); // true
```

Прочие операторы

Есть несколько других операторов, поддерживаемых языком Java:

- Тернарный оператор или условный оператор (?:)
- Оператор instanceof

IT-Academy

Приоритет операторов

Приоритет операторов определяет группирование терминов в выражении. Это влияет как вычисляется выражение.

При одинаковом приоритете операторов будут выполняться слева направо.

1. () [] .
2. ++ -- ! ~
3. * / %
4. + -
5. >> >>> <<
6. > >= < <=
7. == !=
8. &
9. ^
10. |
11. &&
12. ||
13. ?:

. = += -= *= /= %= >>= <<= &= ^= |=

• ,

IT-Academy