Понятие переменных

Создание переменных. Понятие типизации.

[Типизация в Java](#_1nofwc83v3b)

[Операции с переменными](#_x798yvywg5dp)

# Понятие переменных

В школьных уроках математики вы уже сталкивались с переменными. Это были атрибуты в формулах, которые могли заменяться на числовые значения, влияя при этом на поведение функций. В программировании переменная обретает более глубокий смысл.

С точки зрения Java переменная делится на две составляющие: область памяти и имя переменной. Область памяти — это адрес, который можно использовать для доступа к данным переменной (ее значению). JVM легко ориентируется среди адресов переменных, но человеку трудно понять машинную запись адреса. Поэтому в Java переменным можно задавать имена — например, car или table. Синоним будет сопоставляться с областью памяти, в которой хранится значение переменной. Более подробно работу с памятью мы рассмотрим в этом курсе позже. По сути, переменная является контейнером для хранения (как банка на кухне).

# Типизация в Java

Язык Java имеет строгую типизацию. Разберемся, что это означает.

Продолжая аналогию с контейнерами и банками, отметим, что каждый такой предмет имеет свою форму и объем. Так же и переменные характеризуются размером и типом. В Java две группы типов данных:

* Примитивные;
* Ссылочные (объектные).

Познакомимся с примитивными типами, чтобы перейти к изучению более сложных контейнеров.

Java заботится о том, чтобы вы не могли разместить в переменной одного типа значение другого типа (например, положить строку в числовую переменную). Чтобы меры безопасности срабатывали еще при написании кода, нужно указывать тип переменной при ее создании. Вне зависимости от типа переменные объявляются по общим правилам: переменная должна иметь тип и имя.

|  |
| --- |
| int a = 42; |

Имена переменных должны начинаться с буквы, $ или \_, а далее может идти любая последовательность символов. Идентификаторы чувствительны к регистру, и ими не могут быть ключевые слова Java (перечислены ниже):

**abstract, assert, boolean, break, byte, case, catch, char, class, const, continue, default, do, double, else, enum, extends, final, finally, float, for, goto, if, implements, import, instanceof, int, interface, long, native, new, package, private, protected, public, return, short, static, strictfp, super, switch, synchronized, this, throw, throws, transient, try, void, volatile, while**.

В Java есть 8 примитивных типов данных (о них часто спрашивают на собеседованиях):4 типа целых чисел со знаком:

* + — 8-битное число;
  + short — 16-битное число;
  + int — 32-битное число;
  + long — 64-битное число.
* 2 типа чисел с плавающей точкой:
  + float — 32-битное число с плавающей точкой;
  + double — 64-битное число с плавающей точкой.
* Еще 2 типа:
  + boolean — логический тип (true или false);
  + char — 16-битный тип данных, предназначенный для хранения символов в кодировке Unicode.

|  |
| --- |
| byte b = 10;  short s = 2404;  int i = 123456;  long l = 1500L; // Для объявления long в конце ставится буква L  float f = 120.0f; // Для объявления float в конце ставится буква f  double d = 15.72775;  boolean bool = true;  char c = 'A'; |

# Операции с переменными

У переменных есть размер, в который можно не уместиться. Например, когда результат не известен заранее, а рассчитывается в рамках логики программы. По правилам Java нельзя положить большую вещь в маленькую баночку. Произойдет переполнение, и часть данных потеряется. На практике невозможно, например, положить значение **int** в переменную **short**.

|  |
| --- |
| int a = 42;  short s = a;  // Здесь будет ошибка |

В принципе, число 42 вполне совместимо с типом **short**. Но это известно нам, а не компилятору, который интересует только соответствие типов переменных, а не их значений. Он пытается предотвращать даже потенциальные угрозы переполнения.

Чтобы переменная не могла менять значение при выполнении программы, можно определить ее как константу с помощью ключевого слова **final** (написать его перед указанием типа данных переменной):

|  |
| --- |
| final int a = 20; |

Разумеется, инициализация переменных не обязательно производится вместе с ее объявлением. Но нельзя начинать работать с переменной до ее инициализации!

Рассмотрим пример функционала, рассчитывающего объем цилиндра:

|  |
| --- |
| Public static void main(String args[]) {  float volume; // Объявление переменной  float radius = 2.0f, height = 10.0f;  // volume инициализируется динамически во время выполнения программы  volume = 3.1416f \* radius \* radius \* height;  System.out.println("Объем цилиндра равен " + volume); } |

С примитивными типами можно выполнять простейшие арифметические операции:

|  |  |
| --- | --- |
| **Операция** | **Описание** |
| + | Сложение |
| - | Вычитание |
| \* | Умножение |
| / | Деление |
| % | Деление по модулю |
| ++ | Инкремент (приращение на 1) |
| += | Сложение с присваиванием |
| -= | Вычитание с присваиванием |
| \*= | Умножение с присваиванием |
| /= | Деление с присваиванием |
| %= | Деление по модулю с присваиванием |
| -- | Декремент (отрицательное приращение на 1) |

# 