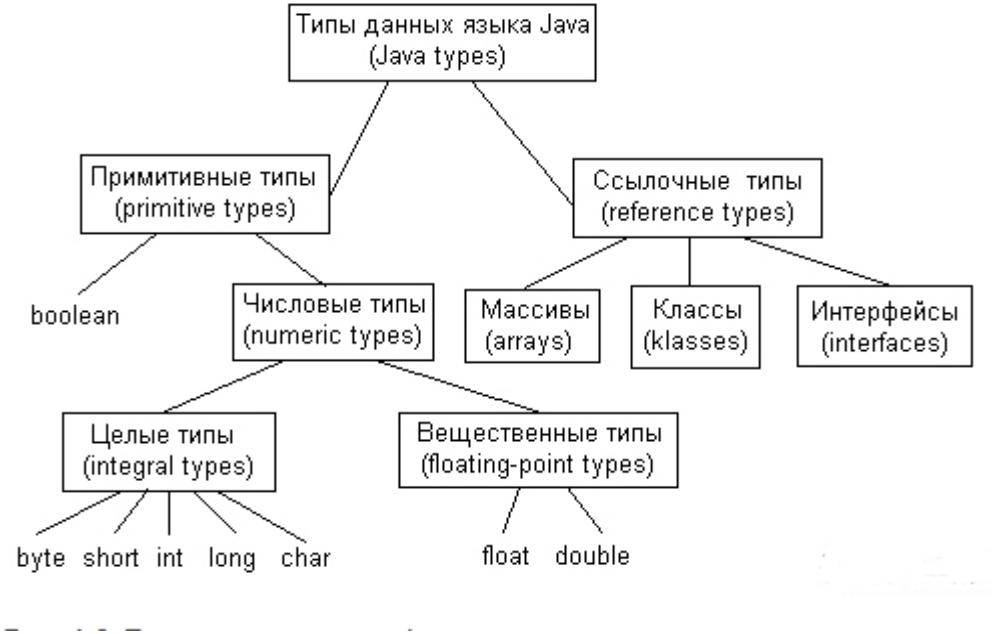
ASTON

Типы данных

Типы данных



Все типы исходных данных, встроенные в язык Java, делятся на две группы: примитивные типы (primitive types) и ссылочные типы (reference types).

- Ссылочные типы делятся на массивы (arrays), массы (classes) и интерфейсы (interfaces).

- Примитивных типов всего восемь. Их можно разделить на логический (иногда говорят булев) тип boolean числовые (numeric).
  Целых типов пять: byte, short, int, long, char.
  Вещественных типов два: float и double.

Примитивы

Тип	Размер (бит)	Минимальное значение	Максимальное значение
`byte`	8	-128	127
`short`	16	-32768	32767
`int`	32	-2147483648	2147483647
`long`	64	-922372036854775808	922372036854775807
`char`	16	0	65536

Тип	Разрядность (бит)	Диапазон	Точность
`float`	32	`3,4e-38 < |x| < 3,4e38`	7-8 цифр
`double`	64	`1,7e-308 < |x| < 1,7e308`	17 цифр

Byte

Наименьший по размеру целочисленный тип - byte. Это 8-битовый тип с диапазоном допустимых значений от -128 до 127. Переменные типа byte часто используются при работе с потоком данных из сети или файла, а также при работе с необработанными двоичными данными или в массивах для экономии памяти.

Объявить переменную типа byte можно следующим образом:

```
byte c, a, t; // объявили сразу три переменные
```

В арифметических выражениях с переменными типа byte вычисления выполняются как с типом int, т.е. с помощью 32-битовой арифметики, а полученный результат будет 32-битовым.

Строку с числом перевести в данный тип можно через метод parseByte(String):

Short

Тип short - 16-битовый тип в диапазоне от -32768 до 32767. Используется очень редко.

short m;

В арифметических выражениях с переменными типа short вычисления выполняются как с типом int, т.е. с помощью 32-битовой арифметики, а полученный результат будет 32-битовым. Например, такой код не пройдёт.

```
// накорми кота

short fishNumber = 3; // три рыбки

short beefNumber = 2; // два кусочка говядины

short breakfast = 0;

breakfast = fishNumber + beefNumber; // завтрак чемпиона
```

Java будет ругаться на последнюю строчку, так как итоговый результат не может быть short. Как вариант, вам нужно преобразовать результат снова в 16-битовое число.

```
breakfast = (short) (fishNumber + beefNumber); // завтрак чемпиона
```

Явно перевести строку с числом в тип short можно через метод parseShort(String):

int/long

Тип int служит для представления 32-битных целых чисел со знаком.

Диапазон допустимых для этого типа значений — от -2147483648 до 2147483647.

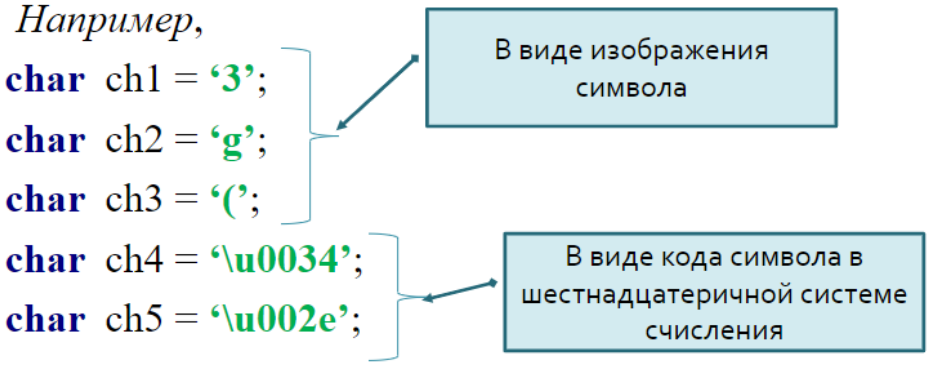
Чаще всего этот тип данных используется для хранения обычных целых чисел со значениями, достигающими двух миллиардов. Этот тип прекрасно подходит для использования при обработке массивов и для счетчиков. В ближайшие годы этот тип будет прекрасно соответствовать машинным словам не только 32-битовых процессоров, но и 64-битовых с поддержкой быстрой конвейеризации для выполнения 32-битного кода в режиме совместимости. Всякий раз, когда в одном выражении фигурируют переменные типов byte, short, int и целые литералы, тип всего выражения перед завершением вычислений приводится к int.

Тип long - это 64-битный тип со знаком, используемый в тех случаях, когда используется очень большое значение, которое не способен хранить тип int. Например, чтобы вычислить расстояние, которое прошёл солнечный луч от солнца до зеркала, превратившись в солнечного зайчика, за которым безуспешно охотится котёнок, вам понадобится именно этот тип.

Можно использовать символы l или L для обозначения числа типа long. Рекомендую использовать заглавную букву, чтобы избежать возможной путаницы.

Вопрос как Long работает в 32 битных системах?

char


Например,
char ch1 = **‘3’**;
char ch2 = **‘g’**;
char ch3 = **‘(’**;
char ch4 = **‘\u0034’**;
char ch5 = **‘\u002e’**;
В виде изображения символа
В виде кода символа в шестнадцатеричной системе счисления

float/double

При присвоении переменной типа float дробного литерала с плавающей точкой, например, 3.1, 4.5 и т.д., Java автоматически рассматривает этот литерал как значение типа double. И чтобы указать, что данное значение должно рассматриваться как float, нам надо использовать суффикс f:
И хотя в данном случае обе переменных имеют практически одно значения, но эти значения будут по-разному рассматриваться и будут занимать разное место в памяти.

```
1 float fl = 30.6f;
2 double db = 30.6;
```

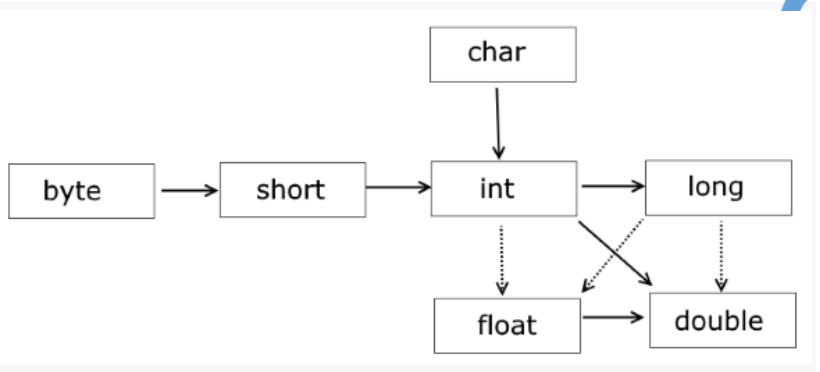
Числа с плавающей точкой нельзя использовать в финансовых расчетах, где ошибки округления недопустимы. Например, в результате выполнения оператора System.out. println(2.0 - 1.1) будет выведено значение 0.8999999999999999, а не 0.9, как было бы логично предположить. Подобные ошибки связаны с внутренним двоичным представлением чисел. Как в десятичной системе счисления нельзя точно представить результат деления 1/3, так и в двоичной системе невозможно точно представить результат деления 1/10. Если же требуется исключить ошибки округления, то следует воспользоваться классом BigDecimal .

Операции

Категория	Оператор	Ассоциативность
Постфикс	() [] . (точка)	Слева направо
Унарный	++ - - ! ~	Справа налево
Мультипликативный	* / %	Слева направо
Аддитивный	+ -	Слева направо
Сдвиг	>> >>> <<	Слева направо
Реляционный	> >= < <=	Слева направо
Равенство	== !=	Слева направо
Побитовое «И» («AND»)	&	Слева направо
Побитовое исключающее «ИЛИ» («XOR»)	^	Слева направо
Побитовое «ИЛИ» («OR»)	|	Слева направо
Логическое «И» («AND»)	&&	Слева направо
Логическое «ИЛИ» («OR»)	||	Слева направо
Условный	?:	Справа налево
Присваивание	= += -= *= /= %= >>= <<= &= ^= |=	Справа налево
Запятая	,	Слева направо

С Java 7 в литералах можно использовать символ подчеркивания для повышения читабельности больших чисел. Например

```
long creditCardNumber = 1234_5678_9012_3456L;
long hexBytes = 0xFF_EC_DE_5E;
byte nybbles = 0b0010_0101;
float pi = 3.14_15F;
```



Преобразование и приведение типов при выполнении операций

byte -> short -> char -> int -> long -> float -> double,

Если преобразование является сужающим (narrowing conversion), т. е. выполняется преобразование

byte <- short <- char <- int <- long <- float <- double,

то такое преобразование может привести к потере точности числа или к его искажению. Поэтому при сужающих преобразованиях при компиляции программы выводится диагностическое сообщение о несовместимости типов и файлы классов не создаются.

- *При приведении float или double к целочисленным типам, дробная часть не округляется, а просто отбрасывается.*
- *Тип boolean не приводится ни к одному из типов.*
- *Тип char приводится к числовым типам, как код символа в системе UNICODE.*
- *Если число больше своего контейнера, результат будет непредсказуемым.*

Проблема с 32 битной архитектурой

Тип double и long имеют размерность 64 бита, а это значит что в 32 битной архитектуре процессора, операции с этими значениями будут происходить в два действия. Сначала произойдет считывание/запись старших 32 битов, после - считывание младших 32 битов.
Проблема возникает как раз тогда, когда множество потоков пытается считывать одни и те же данные так, что в один момент может произойти некорректная манипуляция с данными. Например, один поток выполнил операцию запись в старшие 32 бита, а вторую операцию не закончил, и в это время второй поток начал чтение этой переменной. Второй поток и выдаст ошибку чтения, так как данные находятся в промежуточном состоянии, то есть измененные наполовину.

Решений есть несколько

- добавить модификатор volatile к этим переменным, которой говорит JVM работать с переменными, как с атомарными (так же этот модификатор говорит процессору не использовать кэш переменных для консистетности разделяемых ресурсов между потоками)
- сделать переменные закрытого типа и добавить методы доступа к нему с модификатором synchronize
- перейти с 32-битной архитектуры на 64-битную.

Литература

- https://metanit.com/java/tutorial/2.2.php
- https://vertex-academy.com/tutorials/ru/prividenie-tipov-v-java/
- https://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.2