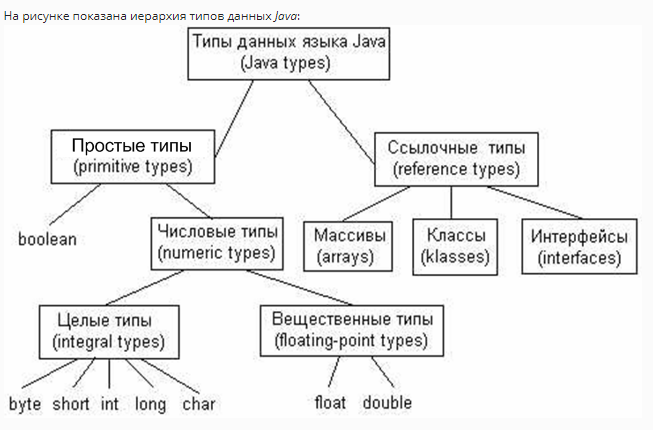
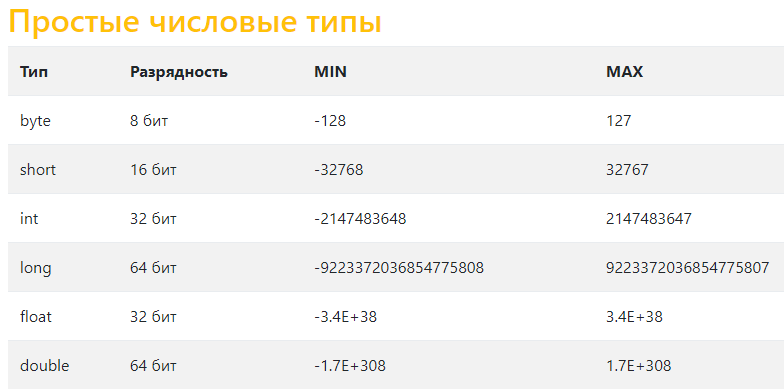
Переменные – это не что иное как зарезервированные места памяти для хранения значений. Это означает, что при создании переменной Вы резервируете некоторое пространство в памяти.

Основываясь на типе данных, который присвоен переменной, операционная система выделяет память и решает, что может быть сохранено в зарезервированную памяти. Поэтому, назначая различные типы данных для переменных, в Java можно хранить целые числа, десятичные дроби или символов в этих переменных.

Существует два типа данных в Java:

* простые или примитивные типы данных;
* ссылочные типы данных (ссылка/объект).





**Тип byte**

* Тип данных byte является 8-разрядным знаковым целым числом.
* Минимальная величина равна -128 (-27).
* Максимальное значение равно 127 (включительно) (27-1).
* По умолчанию – 0.
* byte предназначен для экономии места в больших массивах, главным образом вместо целых чисел, поскольку byte в четыре раза меньше, чем int.
* Пример:
* byte a = 100;

byte b = -50;

Слово «байт» (byte) возникло в компании IBM примерно в 1956 году. Оно произошло от слова bite («кусок»), но его было решено писать через букву **y**, чтобы не путать со словом «bit» («бит»). В течение некоторого времени слово «байт» обозначало просто число битов в конкретном потоке данных. Однако в середине 1960-х, в связи с разработкой семейства компьютеров System/360 в компании IBM, это слово стало обозначать группу из восьми бит.

Любопытно, что **bite** имеет также значение "укус" (сущ.) или "укусить" (глагол). Таким образом это наш родной "Кусь!"



### Тип short

* Тип данных short является 16-разрядным знаковым целым числом.
* Минимальное значение равно -32768 (-215).
* Максимальная величина равна 32 767 (включительно) (215-1).
* Тип short в Java может также использоваться для экономии памяти как byte. Размер short в 2 раза меньше, чем int.
* По умолчанию — 0.
* Пример:
* short s = 10000;

short r = -20000;

### Тип int

* В языке Java тип данных int является 32-разрядным знаковым целым числом.
* Минимальный размер int — 2 147 483 648 (-231).
* Максимальная величина равна 2,147,483,647 (включительно)(231-1).
* Тип int обычно используется для целых значений. Если нет озабоченности по поводу памяти.
* По умолчанию равно 0.
* Пример:
* int a = 100000;

int b =-200000;

### Тип long

* Тип данных long является 64-разрядным знаковым целым числом.
* Минимальное значение равно — 9,223,372,036,854,775,808 (-263).
* Максимальная величина — 9,223,372,036,854,775,807 (включительно). (263-1).
* В Java Применяется когда требуется более широкий диапазон, чем int.
* По умолчанию — 0L.
* Пример:
* long a = 100000L;

long b =-200000L;

### Тип float

* Тип данных float является c одинарной точностью 32-битный IEEE 754 с плавающей точкой.
* Тип float используется главным образом для сохранения памяти в больших массивах чисел с плавающей точкой.
* По умолчанию — 0.0f.
* Тип float никогда не должен применяется для точного значения, например, валюты.
* Пример:

float f1 = 234.5f;

### Тип double

* Тип данных double является c двойной точностью 64-битный IEEE 754 с плавающей точкой.
* Обычно используется для десятичных значений.
* Тип double никогда не должен применяется для точного значения, например, валюты.
* По умолчанию — 0.0d.
* Пример:

double d1 = 123.4;

### Тип boolean

### 

* Тип данных boolean представляет собой один бит информации.
* Существует только два возможных значения: true и false.
* Предназначен для простых признаков, которые позволяют отслеживать условия true или false.
* По умолчанию — false.
* Пример:

boolean one = true;

### Тип char

* Тип данных char является одним 16-разрядным символом Юникода.
* Минимальная величина равна «\u0000» (или 0).
* Максимальная величина равна «\uffff» (или 65535 включительно).
* В Java char нужен для хранения любого символа.
* Пример:

char letterA ='A';

Для char в Java предусмотрена Unicode кодировка. Диапазон значений составляет от 0 до 65536. Размер составляет 2 байта. При этом, в отличие от предыдущих типов переменных, отрицательных значений у char нет. Важно понимать, что в этой переменной не хранится сам символ. Здесь содержится его кодировка из Unicode. Соответственно, мы можем проводить целочисленные операции с символами.

С объявлением переменной типа char все также просто, как и с предыдущими:

char a;

char b;

В первом случае мы объявили символьную переменную а, а во втором – b.

**Ссылочные типы данных**

Ссылочные типы данных хранят ссылку на объект или же тип данных **null**, то есть нулевую (пустую) ссылку.

**Массивы**

Для объявления типа массива используются квадратные скобки. В приведенной ниже строке объявляется переменная month\_days, тип которой — “массив целых чисел типа int”.

***int month\_days [];***

Для того, чтобы зарезервировать память под массив, используется специальный оператор new. В приведенной ниже строке кода с помощью оператора new массиву month\_days выделяется память для хранения двенадцати целых чисел.

***month\_days = new int [12];***

Итак, теперь month\_days — это ссылка на двенадцать целых чисел.

**Многомерные массивы**

На самом деле, настоящих многомерных массивов в Java не существует. Зато имеются массивы массивов, которые ведут себя подобно многомерным массивам, за исключением нескольких незначительных отличий. Приведенный ниже код создает традиционную матрицу из шестнадцати элементов типа double, каждый из которых инициализируется нулем. Внутренняя реализация этой матрицы — массив массивов double.

***double matrix [][] = new double [4][4];***

## Класс String

Отдельно хотелось бы упомянуть класс [String](https://javarush.ru/groups/posts/2347-klass-string-v-java). Это базовый класс, предназначен для хранения и работы с данными, которые хранятся в виде строки. Пример:

String text = **new** String("This TV is very loud");

**Классы**

Java — это объектно-ориентированный язык программирования. Все программы состоят из объектов, которые как-то связываются между собой. Класс — это, по сути, шаблон для объекта. Он определяет, как объект будет выглядеть и какими функциями обладать. **Каждый объект является объектом какого-то класса**.

**Интерфейсы**

**Интерфейс** — это дитя Абстракции и Полиморфизма. Интерфейс очень напоминает абстрактный класс, у которого все методы абстрактные. Он объявляется так же, как и класс, только используется ключевое слово interface.

Операции с простыми типами данных:

- сложение

- вычитание

-деление

- умножение