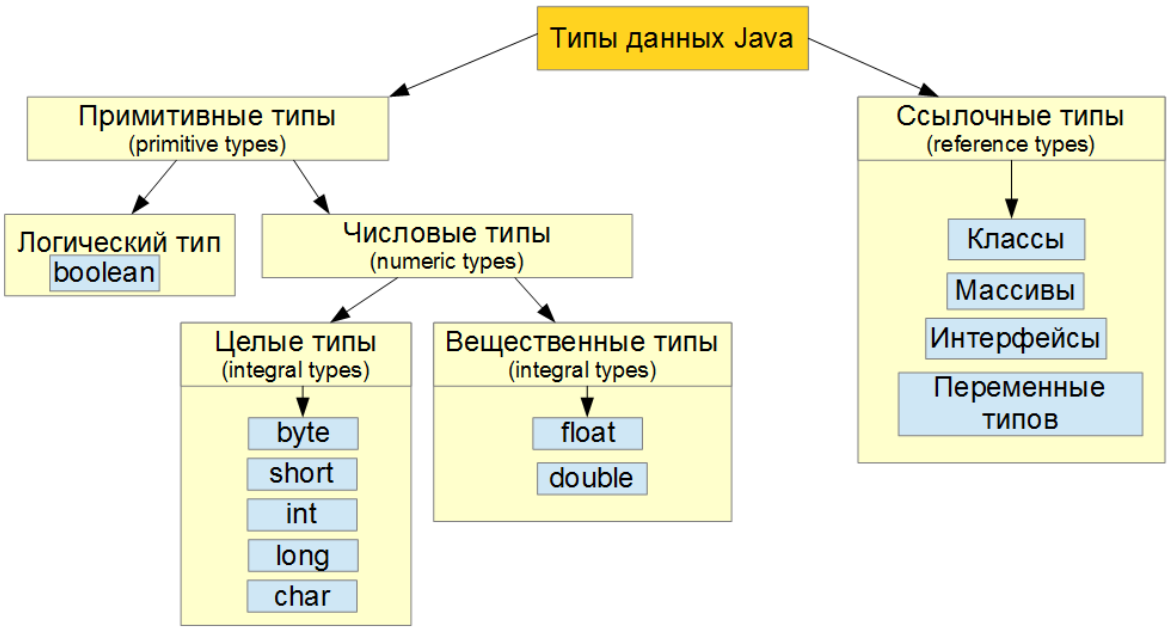
Вопросы по теме Типы данных:

1. Перечислите типы данных в Java?
2. В чем разница между ссылочными и примитивными типами данных?
3. В чем разница между стеком (stack) и кучей (heap)?
4. Типы ссылок в Java?
5. Что такое приведение типов?
6. Что такое автоупаковка / автораспаковка?
7. Примитивные типы данных: значения по умолчанию и диапазон значений?
8. В чем отличие i++ и ++i?
9. Чем отличается использование && и & для типа boolean?
10. Что такое литералы?

Типы данных:



В Java данные делятся на примитивные типы и ссылочные типы.

Примитивный тип данных – это самый простой, неделимый тип данных в Java. Он делится на логический, числовой и вещественный (с плавающей точкой) типы.

Тип данных	Мин. значение	Макс. значение	Объем памяти	Знач. по умолчанию	
byte	-2^7	$2^7 - 1$	1 байт	0	числовой
short	-2^{15}	$2^{15} - 1$	2 байта	0	
char	символ Unicode		2 байта	\u0000	
int	-2^{31}	$2^{31} - 1$	4 байта	0	
long	-2^{63}	$2^{63} - 1$	8 байт	0L	
float	3.4e-038	3.4e+038	4 байта	0.0f	вещественный
double	1.7e-308	1.7e+308	8 байт	0.0d	
boolean	true / false		1 байт	false	логический

1 байт = 8 бит, 1 бит может принимать только 2 значения – 0 или 1. Поэтому из одного бита мы можем получить 2^8 = 256 различных значений. Но у нас старший бит (крайний левый) отведен под знак + / - поэтому мы можем закодировать под значение только 7 бит. Итого 1 байт в Java принимает значения от -2^7 = -128 до $2^7 - 1$ = 127 (-1 – это мы отнимаем значение 0, оно положительное).

Ссылочный тип данных – это сложный, составной тип данных в Java. Он состоит из примитивных типов и образует различные структуры – классы. Т.к. объект ссылочного тип данных занимает много места в памяти, то переменные этого типа хранят в себе не сам объект, а ссылку на его расположение в памяти (4-х байтовый адрес).

Разница между стеком (stack) и кучей (heap).

Стек – область оперативной памяти, работающая по принципу LIFO (Last In - First Out – последний пришел - первый вышел, как колода карт, можем брать только сверху). В стеке хранятся значения локальных примитивных переменных, создаваемых в методах, аргументы метода, а также ссылки на объекты в куче, на которые ссылается метод. Переменные в стеке существуют до тех пор, пока выполняется метод в котором они были созданы.

Доступ к этой области памяти осуществляется быстрее, чем к куче. Также стек является потокобезопасным, т.к. для каждого потока создается свой отдельный стек.

Размер стека — это фиксированная величина, и превышение лимита выделенной на стеке памяти приведёт к переполнению стека (будет брошено исключение StackOverFlowError).

Куча – область оперативной памяти, работающая по принципу динамического выделения памяти. В куче хранятся объекты (ссылки на эти объекты хранятся в стеке), а также примитивные типы данных, которые являются полями объекта. Память в куче освобождается с помощью сборщика мусора. Если память заполнится, то будет брошено исключение OutOfMemoryError.

Доступ к этой области памяти осуществляется медленнее, чем к стеку. Также куча является не потокобезопасной. Доступ к данных, хранящимся в куче, может быть получен из любого места программы.

Типы ссылок в Java.

В Java используются разные типы ссылок на объект: сильные, слабые, мягкие и фантомные ссылки. С помощью этих ссылок определяется, нужно ли удалять объект.

Сильная ссылка (StrongReference) – самый распространенный вид ссылок. Объект в куче не будет удален сборщиком мусора, пока на него указывает сильная ссылка или если он явно доступен через цепочку сильных ссылок.

Мягкая ссылка (SoftReference) – если на объект ссылаются только мягкие ссылки, то он не будет удален сборщиком мусора, но если программе не хватает памяти, то такой объект может быть удален.

Слабая ссылка (WeakReference) – если на объект ссылаются только слабые ссылки, то он будет удален при сборке мусора (а может и не удалится, сборщик мусора непредсказуем).

Фантомная ссылка (PhantomReference) – если на объект ссылаются только фантомные ссылки, то будет выполнена попытка его удаления. Сам объект при этом не будет удален из памяти до тех пор, пока на него существует фантомная ссылка или данная фантомная ссылка не будет очищена с помощью вызова метода clear().

Приведение типов.

Приведение типа – это преобразование одного типа данных в другой тип данных.

Возможно два вида приведения типа:

Сужение типа – преобразование данных большего размера в тип меньшего размера. При таком виде приведения типа часть информации может потеряться (например, при преобразовании long в int и при этом значение long > макс / мин значения int). Поэтому компилятор требует указывать оператор преобразования типа.

```
long bigNumber = 3_000_000_000L;

int number = (int) bigNumber; // вывод -1_294_967_296
```

Расширение типа – преобразование данных меньшего размера в тип большего размера. С целочисленными типами никаких проблем при таком приведении не возникнет. А вот при преобразовании целых чисел в дробные могут теряться младшие части числа

Также понятие приведение типа применимо и к ссылочным типам данных. В этом случаем приведение типа допустимо только в рамках одной иерархии классов (наследовании).

Сужение ссылочного типа – преобразование объекта от типа предка к типу потомка. При сужении типа лучше проверять можно ли сохранить объект определенного типа в переменную определенного типа с помощью оператора instanceof.

Расширение ссылочного типа – преобразование объекта от типа потомка к типу предка.

```
class Parent {}

class Child extends Parent {}

    Parent parent = new Child(); // расширение типа

    if (parent instanceof Child) {

        Child child = (Child) parent; // сужение типа;

    }
```

Автоупаковка / автораспаковка.

Автоупаковка – процесс скрытого преобразования примитивных типов данных в соответствующие ссылочные классы-оболочки. Компилятор использует метод valueOf().

Автораспаковка – процесс скрытого преобразования ссылочных объектов в соответствующие примитивные типы данных. Компилятор использует метод intValue(), doubleValue() и т.д.

примитив ↔ класс-обертка: byte ↔ Byte, short ↔ Short, char ↔ Character, int ↔ Integer, long ↔ Long, float ↔ Float, double ↔ Double, boolean ↔ Boolean

Недостаток процесса автоупаковки / автораспоковки – при неправильном использовании может замедлить работу программы. Например:

Integer number = 1;

int sum = 0;

sum = sum + number;

Сначала идет автораспаковка переменной number, т.к. нельзя слаживать ссылочный тип данных с примитивным, затем идет автоупаковка. Если это поместить в цикл с большим количеством итераций, то лишние операции по преобразованию типов замедлят выполнение программы.

Отличие i++ и ++i.

Префиксная форма ++i – сначала происходит инкремент, затем вычисление.

Постфиксная форма i++ – сначала происходит вычисление, затем инкремент.

int i = 1; int j = 1;

int a = 10 + ++i; // 12

int b = 10 + j++; // 11

Использование && и & для типа boolean.

&& – ленивое вычисление, если результат выражения уже ясен, то дальнейшее вычисление не будет происходить.

& - полное вычисление выражения.

int i = 1; int j = 1;

if (true & false & ++i == 0) { } // i == 2, операция ++i выполнилась

if (true && false && ++j == 0) { } // j == 1, операция ++j не выполнилась

Литерал.

Литерал – явно заданное значение переменной в коде программы.

int i = 1; \\ 1 - целочисленный литерал

String s = "Hi"; \\ "Hi" – строковый литерал

Бесконечность и NaN.

В Java тип double имеет специальные значения:

– **NEGATIVE_INFINITY** – «минус бесконечность»

– **POSITIVE_INFINITY** – «плюс бесконечность»

– **NaN** – «не число» (Not-a-Number) – особое состояние числа с плавающей запятой.