1. **Какие есть типы данных в Java?**

Примитивные типы данных.

В Java есть 8 примитивных типов, которые делят на 3 группы:

1. Целые числа - byte, short, char, int, long.
2. Числа с плавающей точкой (вещественные) - float, doubleю
3. Логический - boolean

## Тип String.

## Ссылочные типы данных (или объектные) (это все остальные типы: классы, перечисления и интерфейсы, например, объявленные в стандартной библиотеке Java, а также массивы.)C:\Users\Максим\Desktop\Java_data_types_thumb[3].png

**2) Какие есть типы операций в Java и в чём их отличие?**

## Операция присваивания

## Унарные операции

* унарный минус "-" – меняет знак числа или выражения на противоположный;
* унарный плюс "+" – не выполняет никаких действий над числом или выражением;
* побитовое дополнение "~" (только для целых) – инвертирует все биты поля числа (меняет 0 на 1 и 1 на 0);
* инкремент "++" (только для целых) – увеличивает значение переменной на 1;
* декремент "--" (только для целых) – уменьшает значение переменной на 1.

## Арифметические бинарные операции

* сложение "+";
* вычитание "-";
* умножение "\*";
* деление "/";
* вычисление остатка от деления целых чисел "%" (возвращает остаток от деления первого числа на второе, причем результат будет иметь тот же знак, что и делимое), например, результат операции 5%3 будет равен 2, а результат операции (-7)%(-4) будет равен -3. В Java операция может использоваться и для вещественных переменных (типа float или double).

## Побитовые операции

 установка бита в i-ой позиции поля результата в 1, если оба бита в i-ых позициях операндов равны 1, или в 0 в противном случае – побитовое И ("&");

 установка бита в i-ой позиции поля результата в 1, если хотя бы один бит в i-ых позициях операндов равен 1, или в 0 в противном случае – побитовое ИЛИ ("|");

 установка бита в i-ой позиции поля результата в 1, если биты в i-ых позициях операндов не равны друг другу, или в 0 в противном случае – побитовое исключающее ИЛИ ("^");

 сдвиг влево битов поля первого операнда на количество битов, определяемое вторым операндом (бит знака числа при этом не меняется) – побитовый сдвиг влево с учетом знака "<<";

 сдвиг вправо битов поля первого операнда на количество битов, определяемое вторым операндом (бит знака числа при этом не меняется) – побитовый сдвиг вправо с учетом знака ">>";

 сдвиг вправо битов поля первого операнда на количество битов, определяемое вторым операндом (бит знака числа при этом также сдвигается) – побитовый сдвиг вправо без учета знака ">>>".

## Комбинированные операции

1. Выражение x += b означает x = x + b.
2. Выражение x -= b означает x = x - b.
3. Выражение x \*= b означает x = x \* b.
4. Выражение x /= b означает x = x / b.
5. Выражение x %= b означает x = x % b.
6. Выражение x &= b означает x = x & b.
7. Выражение x |= b означает x = x | b.
8. Выражение x ^= b означает x = x ^ b.
9. Выражение x <<= b означает x = x << b.
10. Выражение x >>= b означает x = x >> b.
11. Выражение x >>>= b означает x = x >>> b.

## Операции сравнения

 ==" (равно), "!=" (не равно),

 ">" (больше), ">=" (больше или равно),

 "<" (меньше) "<=" (меньше или равно)

## Булевские операции

Булевские операции выполняются над булевскими переменными и их результатом также является значение типа boolean. В Java определены следующие булевские операции:

* отрицание "!" – замена false на true, или наоборот;
* операция И "&" – результат равен true, только, если оба операнда равны true, иначе результат – false;
* операция ИЛИ "|" – результат равен true, только, если хотя бы один из операндов равен true, иначе результат – false.
* операция исключающее ИЛИ "^" – результат равен true, только, если операнды не равны друг другу, иначе результат – false.

Операции "&", "|" и "^" можно, также как и соответствующие побитовые операции использовать в составных операциях присваивания: "&=", "|=" и "^="

Кроме того, к булевским операндам применимы операции "==" (равно) и "!=" (не равно).

## Условная операция

Условная операция записывается в форме выражение-1?выражение-2:выражение-3. **3) Какие есть группы операций над примитивными типами данных, какой  
основной смысл каждой группы (в чём заключается идея сбора операций в соответствующие группы)?**

## Все операции делятся на 5 групп:

## арифметические операции (+, —, \*, / и прочие);

## поразрядные логические операции (~, &, | и прочие);

## логические операции (&, |, &&, || и прочие);

## операции отношения (==, !=, >, < и прочие);

## дополнительные операции для особых случаев.

Идея сбора заключается в упорядочивании операций по виду операторов и их функционалу.  
**4) Для каких типов данных в Java нет операций и почему? А как для данных  
типов всё-таки выполнить операции?**

Для ссылочных  типов данных (или объектных) нет операций. Мы не можем сложить 2 объекта и получить результат, или не можем их сравнить их между собой.

Переменная типа класса является ссылочной переменной, она не хранит данные (как переменные простых типов int, char и т.д.), а указывает на место в памяти, где эти данные хранятся (как переменные типа массива).

Единственное мы можем сравнивать хэш-коды объектов, а также на равенство или не равенство ссылок на эти объекты.  
5) **Какие есть операции для типа данных boolean?**

## Логические операции.

## Булевские операции.

## Условная операция.

## Часть операций сравнения(или отношения), а именно переменные типа boolean можно сравнивать только на равенство и неравенство.

## Над типом boolean определено часть побитовых операций (а также операций составного присваивания), которые аналогичны по действию и результату соответствующим логическим операциям.

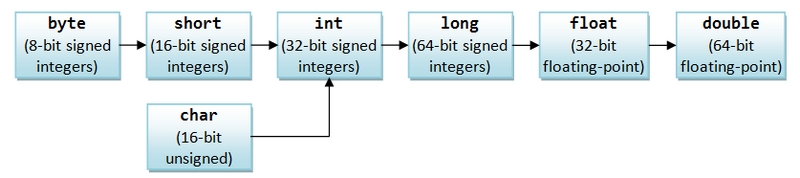
**6) Можно ли значение типа boolean получить из другого типа? А наоборот?**

Значение типа boolean нельзя преобразовать ни к какому типу, а также его  
нельзя получить ни из какого типа **7) Что такое явное и неявное (автоматическое) преобразование типов?**

Когда в одной операции вовлечены данные разных типов, не всегда необходимо использовать операцию преобразования типов. Некоторые виды преобразований выполняются неявно, автоматически.

Неявное преобразование типов выполняется в случае если выполняются условия:

1. Оба типа совместимы
2. Длина целевого типа больше или равна длине исходного типа

[](https://lh3.googleusercontent.com/-bHiKtoojZx8/VoIuedd2HBI/AAAAAAAAOvg/TfGOmcceOh4/s1600-h/JavaBasics_ImplicitTypeCastingPrimit%5B3%5D.png)

Во всех остальных случаях должно использоваться явное преобразование типов.

Явное - это когда разработчик в скобках (либо через вызов методов) преобразует переменную одного типа к другому. При этом, не факт, что это преобразование пройдёт и пройдёт правильно. То есть разработчик берет ответственность на себя за это преобразование.

**8) Какие операции всегда сами преобразовывают тип?**

Некоторые операции присваивания (Assignment operators) такие, как +=, -=, /=, \*=. **9) Какие операции в Java перегружены? В чём заключается смысл перегрузки для соответствующих операций?**

+ для строк

-  также перегружен для операций с плавающей запятой и целыми числами, как и \* и /.

Чтобы внести дополнительный функционал при работе с разными типами данных.

**10)Какие вы знаете спецификаторы типов для вставки соответствующих  
значений в форматную строку?**

|  |  |
| --- | --- |
| Спецификатор формата | Выполняемое форматирование |
| %a | Шестнадцатеричное значение с плавающей точкой |
| %b | Логическое (булево) значение аргумента |
| %c | Символьное представление аргумента |
| %d | Десятичное целое значение аргумента |
| %h | Хэш-код аргумента |
| %e | Экспоненциальное представление аргумента |
| %f | Десятичное значение с плавающей точкой |
| %g | Выбирает более короткое представление из двух: %е или %f |
| %o | Восьмеричное целое значение аргумента |
| %n | Вставка символа новой строки |
| %s | Строковое представление аргумента |
| %t | Время и дата |
| %x | Шестнадцатеричное целое значение аргумента |
| %% | Вставка знака % |