**Содержание урока**

1. [Преобразования примитивных типов](https://stepik.org/lesson/%D0%9F%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D1%82%D0%B8%D0%BF%D0%BE%D0%B2-12760/step/2?course=Java-%D0%91%D0%B0%D0%B7%D0%BE%D0%B2%D1%8B%D0%B9-%D0%BA%D1%83%D1%80%D1%81&unit=3108)
2. [Задача](https://stepik.org/lesson/%D0%9F%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D1%82%D0%B8%D0%BF%D0%BE%D0%B2-12760/step/3?course=Java-%D0%91%D0%B0%D0%B7%D0%BE%D0%B2%D1%8B%D0%B9-%D0%BA%D1%83%D1%80%D1%81&unit=3108)
3. [Классы-обертки и связанные преобразования](https://stepik.org/lesson/%D0%9F%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D1%82%D0%B8%D0%BF%D0%BE%D0%B2-12760/step/4?course=Java-%D0%91%D0%B0%D0%B7%D0%BE%D0%B2%D1%8B%D0%B9-%D0%BA%D1%83%D1%80%D1%81&unit=3108)
4. [Преобразования в строку и обратно](https://stepik.org/lesson/%D0%9F%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D1%82%D0%B8%D0%BF%D0%BE%D0%B2-12760/step/5?course=Java-%D0%91%D0%B0%D0%B7%D0%BE%D0%B2%D1%8B%D0%B9-%D0%BA%D1%83%D1%80%D1%81&unit=3108)
5. [Полезный совет](https://stepik.org/lesson/%D0%9F%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D1%82%D0%B8%D0%BF%D0%BE%D0%B2-12760/step/6?course=Java-%D0%91%D0%B0%D0%B7%D0%BE%D0%B2%D1%8B%D0%B9-%D0%BA%D1%83%D1%80%D1%81&unit=3108)
6. [Вопрос](https://stepik.org/lesson/%D0%9F%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D1%82%D0%B8%D0%BF%D0%BE%D0%B2-12760/step/7?course=Java-%D0%91%D0%B0%D0%B7%D0%BE%D0%B2%D1%8B%D0%B9-%D0%BA%D1%83%D1%80%D1%81&unit=3108)
7. [Вопрос](https://stepik.org/lesson/%D0%9F%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D1%82%D0%B8%D0%BF%D0%BE%D0%B2-12760/step/8?course=Java-%D0%91%D0%B0%D0%B7%D0%BE%D0%B2%D1%8B%D0%B9-%D0%BA%D1%83%D1%80%D1%81&unit=3108)
8. [Задача](https://stepik.org/lesson/%D0%9F%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D1%82%D0%B8%D0%BF%D0%BE%D0%B2-12760/step/9?course=Java-%D0%91%D0%B0%D0%B7%D0%BE%D0%B2%D1%8B%D0%B9-%D0%BA%D1%83%D1%80%D1%81&unit=3108)

Реализуйте метод, который возвращает букву, стоящую в таблице UNICODE после символа "\" (обратный слэш) на расстоянии a.

В качестве примера написано заведомо неправильное выражение. Исправьте его.

public static char charExpression(int a) {

return '?' - 1;

}

Зачем в Java, объектно-ориентированном языке, есть примитивные типы, не являющиеся объектами? Ведь есть полноценные объекты-аналоги java.lang.Integer и т.п.

Примитивные типы нужны для производительности вычислений. Работа с объектами намного медленнее.

Примитивные типы — это просто синонимы для соответсвующих ссылочных типов. Т.е. int — синоним для java.lang.Integer и т.п.

Чтобы всех запутать.

Примитивные типы — пережиток ранних версий Java. Теперь их не могут убрать из-за обратной совместимости.

Укажите преобразования типов, которые компилятор делает автоматически. Другими словами, для каких преобразований не требуется явно писать в программе оператор приведения типа?

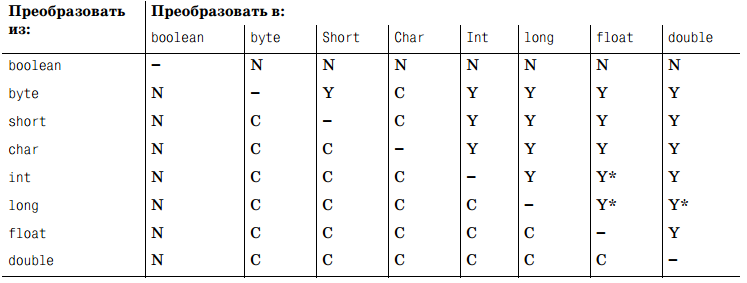


Таблица представленная ВЫШЕ представляет собой сетку, где для каждого примитивного типа указаны типы, в которые их можно преобразовать, и способ преобразования.

Буква **N** в таблице означает невозможность преобразования.

Буква **Y**означает расширяющее преобразование, которое выполняется автоматически.

Буква **С** означает сужающее преобразование, требующее явного приведения.

Наконец, **Y\*** означает автоматическое расширяющее преобразование, в процессе которого значение может потерять некоторые из наименее значимых разрядов. Это может произойти при преобразовании int или long во float или double.

Int -> long =Y

Long -> float = Y\*  
String -> int = N

byte -> char = C

char -> int = Y

int -> boolean = N

char -> Character = Y

float -> long = C