Потоки вывода-ввода:

Объект, из которого можно считать данные, называется потоком ввода, а объект, в который можно записывать данные, - потоком вывода. Например, если надо считать содержание файла, то применяется поток ввода, а если надо записать в файл - то поток вывода.

В основе всех классов, управляющих потоками байтов, находятся два абстрактных класса: InputStream (представляющий потоки ввода) и OutputStream (представляющий потоки вывода)

Но поскольку работать с байтами не очень удобно, то для работы с потоками символов были добавлены абстрактные классы Reader (для чтения потоков символов) и Writer (для записи потоков символов).

1. Язык Java. Особенности языка.
2. Средства разработки. JDK и JRE.
3. Примитивные типы данных в Java.
4. Работа с переменными. Декларация. Инициализация. Присваивание.
5. Инструкции ветвления и циклов.
6. Операторы и выражения в Java. Особенности вычисления, приоритеты операций.
7. Математические функции в составе стандартной библиотеки Java. Классjava.lang.Math.
8. Форматированный вывод числовых данных.

1) Java — строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). ... На 2019 год Java — один из самых популярных языков программирования.

* простота, объектная ориентированность и понятность;
* надёжность и безопасность;
* переносимость и независимость от платформы;
* высокая производительность;
* интрепретируемость, поточность и динамичность.
* За 22 года жизни язык оброс десятками IDE и фреймворков, сотнями сообществ и форумов, тысячами библиотек и плагинов. Всё это благоприятно сказывается на пороге вхождения в профессию, востребованности и качеству производимых с помощью java продуктов.

2) IntelliJ IDEA, NetBeans, Eclipse

* Java Development Kit (JDK) – Программное обеспечение для тех, кто желает писать программы на Java(представляет инструменты для работы с прогой)
* Java Runtime Environment (JRE) – Программное обеспечение для потребителей, желающих выполнять программы на Java
* Standart Edition (SE) – обычное издания для применения в настольных системах и простых сервисных приложений
* Enterprise Edition – для сложных серверных приложений
* Micro Edition – платформа для мобильных телефонов и других компактных устройств
* Update (u) – для выпусков исправлений ошибок к основной версии

3)Примитивные типы данных хранят адрес на 1 значение.

* Целочисленные типы данных – int (4байта), short(2 байта), long(8 байтов), byte(1 байт)
* Числовые типы данных с плавающей точкой – float(4 байта), double(8 байтов)
* char - 16-разрядное беззнаковое целое, представляющее собой символ UTF-16 (буквы и цифры)
* boolean - булев тип, может иметь значения true или false

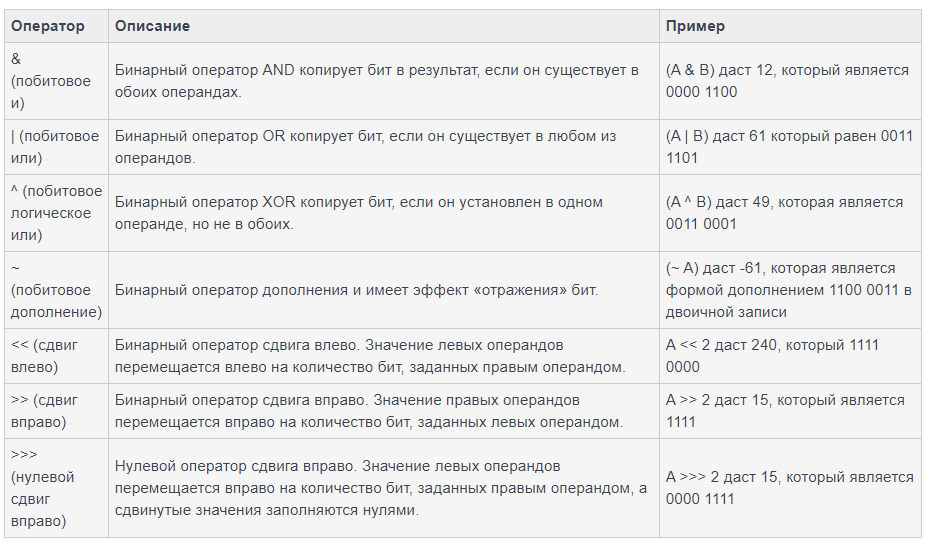
4)

* Декларация – объявление переменной (int k, int [] d = new int [], int [] d = {3, 1, 2})
* Инициализация переменной – присвоение ей определенного значения (обязательно)

Итак, мы запускаем нашу программу. Как мы знаем, Java программу выполняет JVM, то есть виртуальная Java машина. JVM должна знать то, откуда должно начаться выполнение программы. Для этого мы объявляем main метод, который называется «точкой входа». Для выполнения в JVM создаётся основной поток (Thread). При создании потока ему выделяется свой стэк в памяти. Этот стэк состоит из фрэймов. При выполнении каждого нового метода в потоке под него будет выделен новый фрэйм и добавлен на вершину стэка (как новая книжка в стопке книг). Этот фрэйм будет содержит ссылки на объекты и примитивные типы. Да да, наш int будет храниться в стэке, т.к. int это примитивный тип. Прежде чем выделить фрэйм JVM должна понимать, что туда сохранять. Именно по этой причине мы получим ошибку «variable might not have been initialized», ведь если она не инициализирована, то JVM не сможет нам подготовить стэк. Поэтому при компиляции программы умный компилятор поможет нам не допустить ошибку и не сломать всё.

* Память разделяется на следующие разделы: Heap(куча), Stack и Non-Heap.
* Присваивание значения переменной

5) if else while do while (и все) switch case, for, break, continue

6) 

Арифметические операторы

Операторы сравнения

Побитовые операторы

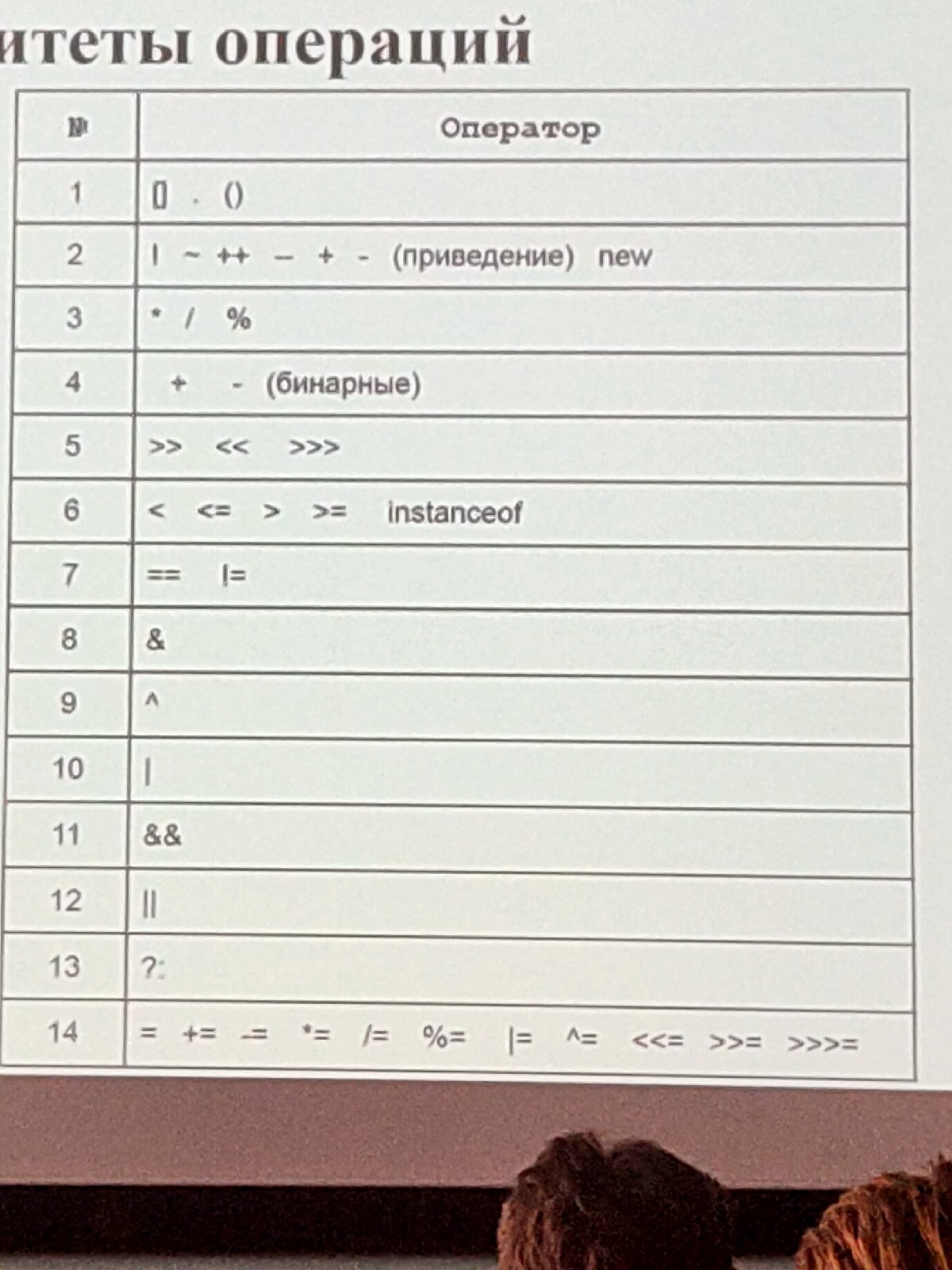
Логические операторы

Операторы присваивания

Тернарный оператор или условный оператор ?:

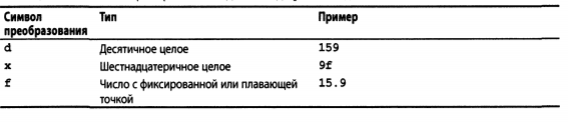
Оператор instanceof — проверяет, является ли объект определенного типа (типа класса или типа интерфейса) и используется только для переменных ссылочного объекта. Оператор instanceof записывается в виде:

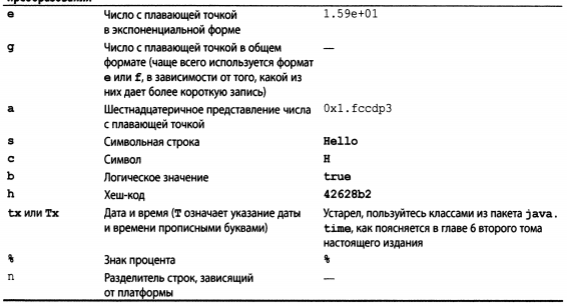
(Переменная ссылочного объекта) instanceof (класс/тип интерфейса)

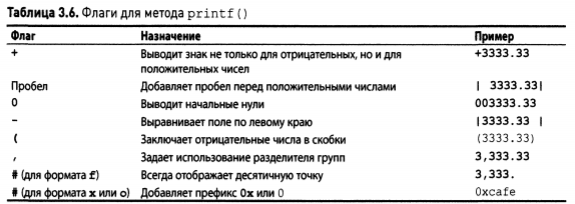
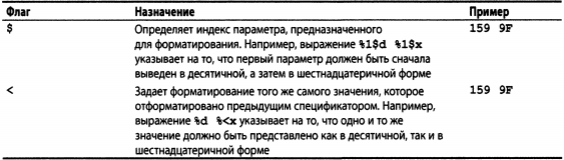


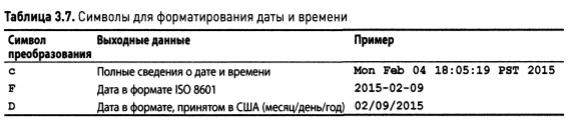
8)

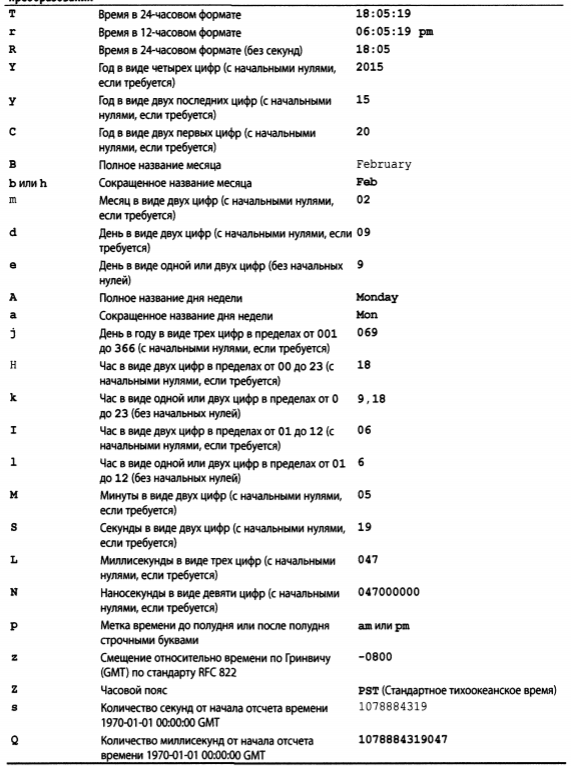
Таблица для printf

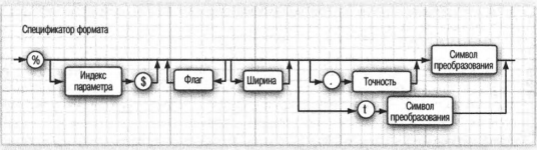


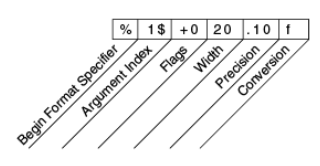


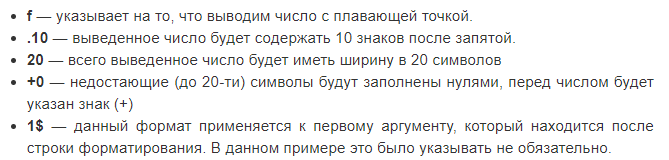


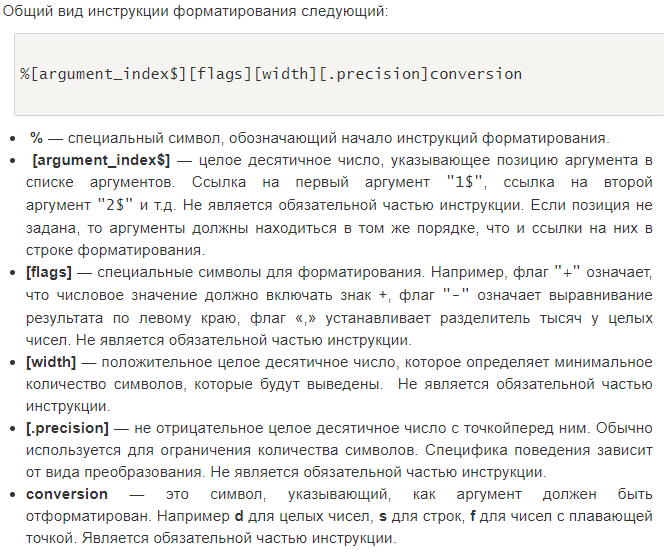












В случае, если нет необходимости выводить отформатированную строку, а нужно просто ее сохранить для дальнейшего использования (например, для записи в лог или базу данных) следует использовать метод format из класса String. Принципы форматирования в этом случае абсолютно такие же, как у описанного выше printf, но этот метод вместо вывода строки сохраняет ее как отформатированную строку.

String s = String.format("Курс валют: %-4s%-8.4f %-4s%-8.4f","USD", 58.4643, "EUR", 63.3695);

7) Класс Math содержит методы, связанные с геометрией и тригонометрией и прочей математики. Методы реализованы как static, поэтому можно сразу вызывать через Math.methodName() без создания экземпляра класса.

В классе определены две константы типа double: E и PI.

Популярные методы для тригонометрических функций принимают параметр типа double, выражающий угол в радианах.

sin(double d)

cos(double d)

tan(double d)

asin(double d)

acos(double d)

atan(double d)

atan2(double y, double x)

Существуют также гиперболические функции: sinh(), cosh(), tanh().

Экспоненциальные функции: cbrt(), exp(), expm1(), log(), log10(), log1p(), pow(), scalb(), sqrt().

Из них хорошо знакомы возведение в степень - pow(2.0, 3.0) вернёт 8.0.

Также популярен метод для извлечения квадратного корня - sqrt(4.0). Если аргумент меньше нуля, то возвращается NaN. Похожий метод cbrt() извлекает кубический корень. Если аргумент отрицательный, то и возвращаемое значение будет отрицательным: -27.0-> -3.0.

Функции округления:

abs() - возвращает абсолютное значение аргумента

ceil() - возвращает наименьшее целое число, которое больше аргумента

floor() - возвращает наибольшее целое число, которое меньше или равно аргументу

max() - возвращает большее из двух чисел

min() - возвращает меньшее из двух чисел

nextAfter() - возвращает следующее значение после аргумента в заданном направлении

nextUp() - возвращает следующее значение в положительном направлении

rint() - возвращает ближайшее целое к аргументу

round() - возвращает аргумент, округлённый вверх до ближайшего числа

ulp() - возвращает дистанцию между значением и ближайшим большим значением

Другие методы

copySign() - возвращает аргумент с тем же знаком, что у второго аргумента

getExponent() - возвращает экспоненту

IEEEremainder() - возвращает остаток от деления

hypot() - возвращает длину гипотенузы

random() - возвращает случайное число между 0 и 1 (единица в диапазон не входит)

signum() - возвращает знак значения

toDegrees() - преобразует радианы в градусы

toRadians() - преобразует градусы в радианы

Вычислим наибольшее и наименьшее числа из двух заданных значений.

double value1 = 9.14;

double value2 = 9.15;

System.out.println("Maximum of value1 and value2 is " + Math.max(value1, value2));

System.out.println("Minimum of value1 and value2 is " + Math.min(value1, value2));

Большие числа

Если вам не хватает точности основных типов для представления целых и вещественных чисел, то можно использовать классы BigInteger и BigDecimal из пакета java.math, которые предназначены для выполнения действий с числами, состоящими из произвольного количества цифр.

Для преобразования обычного числа в число с произвольной точностью (называемое большим числом) вызывается статический метод valueOf():

BigInteger bigIntegerA = BigInteger.valueOf(42);

При работе с большими числами нельзя использовать привычные математические операции с помощью + или \* и т.п. Вместо них следует использовать специальные методы add() (сложение), multiply() (умножение), divide() (деление) и т.д.

BigInteger bigIntegerA = BigInteger.valueOf(42);

BigInteger bigItBigIntegerB = bigIntegerA.add(bigIntegerA);

BigInteger bigIntegerC = bigIntegerA.multiply(bigIntegerA);

В обычной практике эти числа используются редко.

Архивирование через консоль

