Часть 1 (java.lang)

О пакете java.lang

Это библиотека поддержки средств языка.
Пакет `java.lang` импортируется автоматически.

Класс Object

Этот класс является суперклассом для всех классов.

Конструктор класса Object

`Object()`

конструктор класса; параметров не принимает; создает объект своего класса; возвращает ничтоже;

Методы класса Object

`clone()`

этот метод предназначен для клонирования объекта (чтобы объекту был доступен этот метод, он должен реализовать интерфейс `Cloneable`); параметров не принимает; создает клон объекта; возвращает объект класса `Object`;

`equals()`

этот метод предназначен для проверки на равенство своего объекта и указанного в параметрах; принимает параметр `(Object)`; сравнивает свой объект с указанным в параметрах (с точки зрения этого метода объекты равны, если у них одинаковое содержимое, и одинаковый хэш — такое возможно, если это один и тот же объект); возвращает значение типа `boolean` (`true` если да, `false` если нет);

`finalize()`

этот метод используется в системных нуждах; параметров не принимает; вызывается неявно при удалении своего объекта; возвращает ничтоже;

`getClass()`

этот метод предназначен для запроса класса своего объекта; параметров не принимает; исполняет запрос класса своего объекта; возвращает объект класса `Class`; этот метод объявлен как `final`;

`hashCode()`

этот метод служит для возрата хэш- кода своего объекта; параметров не принимает; возвращает хэш- код своего объекта; возвращает значение типа `int`;

`toString()`

этот метод служит для возврата строкового представления своего объекта; параметров не принимает; возвращает строковое представление своего объекта; возвращает объект класса `String`;

`notify()`

этот метод служит для работы с многопоточностью; параметров не принимает; прерывает исполнение потока, ожидающего на своем объекте этого метода; возвращает ничтоже; этот метод объявлен как `final`;

`notifyAll()`

этот метод служит для работы с многопоточностью; параметров не принимает; прерывает исполнение всех потоков, ожидающих на своем объекте этого метода; возвращает ничтоже; этот метод объявлен как `final`;

`wait()`

этот метод служит для работы с многопоточностью; параметров либо не принимает, либо принимает `(long)` — количество миллисекунд, либо принимает `(long, int)` — количество миллисекунд и наносекунд; приводит к ожиданию завершения другого потока до указанного в параметрах времени (если указано); возвращает ничтоже; этот метод объявлен как `final`;

Класс Class

Этот класс содержит описание класса (состояние класса времени выполнения класса/ интерфейса). Его экземпляры создаются загрузчиком, объявить же экземпляры этого класса нельзя (хотя их можно получить вызовом метода `Object.getClass()`).
Подробности — в других источниках.

Класс Void

Это класс, имеющий служебное назначение. Он содержит единственный член `TYPE` – используется как ссылка на объект класса `Class` для класса `Void`. Объекты этого класса не могут быть созданы программистом.

Класс Boolean

Это класс- обертка для элементарного типа данных `boolean`.

Конструктор класса Boolean

`Boolean()`

конструктор класса; принимает параметры — либо принимает `boolean` — значение инициализации, либо принимает `String` — строковое представление значения инициализации, не зависит от регистра; создает объект своего класса; возвращает ничтоже;

Методы класса Boolean

`valueOf()`

этот метод служит для возврата значения; принимает параметр — либо `(boolean)` — логическое значение, либо `(String)` - строковое представление логического значения, не зависит от регистра; возвращает значение своего операнда; возвращает значение типа `boolean`;
этот метод статический;

`equals()`

этот метод служит для проверки на равенство своего объекта, и занного в параметрах; принимает параметр – `Boolean` (указанное значение); проверяет на равенство свой объект и указанное значение; возвращает значение типа `boolean` (`true` – равны, `false` – нет);

`hashCode()`

этот метод служит для возрата хэша своего объекта; параметров не принимает; возвращает хэш своего объекта; возвращает значение типа `int`;

`toString()`

этот метод служит для возврата строкового представления своего объекта; параметров либо не принимает, либо принимает параметр (boolean); возвращает строковое представление либо своего объекта, либо своего операнда (соответственно); возвращает объект класса String;

`booleanValue()`

этот метод служит для возврата значения своего объекта; параметров не принимает; возвращает значение своего объекта в виде значения типа `boolean`; возвращает значение типа `boolean`;

`compare()`

этот метод служит для сравнения объектов этого класса; принимает параметры `(Boolean, Boolean)`; сравнивает указанные в параметрах объекты (с точки зрения этого метода объекты равны, если инкапсулируют одно и то же значение; причем, метод возвращает число ноль — если объекты равны, положительное число — если первый больше, и отрицательное число — если первый меньше; как обычно, значение `true` считается больше, чем `false`); возвращает значение типа `int`; этот метод объявлен как `static`;

`compareTo()`

этот метод служит для сравнения объектов этого класса; принимеает параметр `(Boolean)`; сравнивает свой объект с указанным в параметрах объектом (при этом, механизм сравнения тот же, что у метода `compare()`); возвращает значение типа `int`;

`getBoolean()`

этот метод используется для системных нужд; принимает параметр – `String` (указанное значение); интерпретирует указанное значение как имя системного свойства, и возвращает его значение (полагается, что это свойство имеет логическое значение); возвращает значение типа `boolean`;
этот метод статический;

`logicalAnd()`

логическое И; принимает параметры – `boolean`, и `boolean`; исполняет логическое И над указанными операндами; возвращает значение типа `boolean`;
этот метод статический;

`logicalOR()`

логическое ИЛИ; принимает параметры – `boolean`, и `boolean`; исполняет логическое ИЛИ над указанными операндами; возвращает значение типа `boolean`;
этот метод статический;

`logicalXOR()`

логическое ИСКЛЮЧАЮЩЕЕ ИЛИ; принимает параметры – `boolean`, и `boolean`; исполняет логическое ИСКЛЮЧАЮЩЕЕ ИЛИ над указанными операндами; возвращает значение типа `boolean`;
этот метод статический;

`parseBoolean()`

этот метод служит для возврата значения; принимает параметр - `String` (строковое представление логического значения, не зависит от регистра); возвращает значение своего операнда; возвращает значение типа `boolean`;
этот метод статический;

Класс Character

Это класс- обертка для элементарного типа данных `char`.

Конструктор класса Character

`Character()`

конструктор класса `Character`; принимает параметры — символ (это значение будет заключено в обертку); создает объект своего класса; возвращает ничтоже;

Методы класса Character

`forDigit()`

используется для преобразования значения к цифре; принимает параметры — целочисленное (преобразуемое значение), и `int` (основание системы счисления результата); преобразует указанное число в представление в указанной системе счисления; возвращает результат `char`;
этот метод статический;

`digit()`

используется для преобразования значения к цифре; принимает параметры — `char` (преобразуемое значение), и `int` (основание системы счисления результата); преобразует указанное значение в представление в указанной системе счисления; возвращает целочисленное;
этот метод статический;

`compareTo()`

используется для сравнения указанного символа со значением обертки; принимает параметр — обертку символа (это и есть указанное значение); сравнивает указанное значение со значением своей обертки (возвращает ноль — если равно, отрицательное значение — если значение своей обертки меньше, положительное значение — если больше); возвращает `int`;

`equals()`

используется для сравнения указанного значения со своим инкапсуллируемым; принимает параметр — `Character` (указанное значение); сравнивает свое

инкапсуллируемое значение с указанным; возвращает логическое значение (`true` – равны, `false` – нет);

`hashCode()`

используется для возврата хэша своего объекта; параметров не принимает; возвращает хэш своего объекта; возвращает `int;`

`isDefinied()`

используется для определения есть ли символ в Unicode; принимает параметры — `char` (указанный символ); проверяет, есть ли указанный симол в Unicode (`true` – есть, `false` – нет); возвращает логическое значение;
этот метод статический;

`isDigit()`

используется для определения является ли символ цифрой; принимает параметры — `char` (указанный символ); проверяет является ли указанный символ цифрой (`true` – есть, `false` – нет); возвращает логическое значение;
этот метод статический;

`isIdentifierIgnorable()`

используется для проверки является ли символ игнорируемым; принимает параметры — `char` (указанный символ); проверяет является ли указанный символ игнорируемым — не опустимым в именах, `true` – да, `false` – нет; возвращает логическое значение;
этот метод статический;

`isJavaIdentifierPart()`

используется для проверки является ли символ неигнорируемым; принимает параметры — `char` (указанный символ); проверяет является ли указанный символ неигнорируемым — допустимым в именах, `true` – да, `false` – нет; возвращает логическое значение;
этот метод статический;

`isJavaIdentifierStart()`

используется для проверки является ли символ допустимым как начальный в именах; принимает параметры — `char` (указанный символ); проверяет является

ли указанный символ допустимым как начальный в именах, `true` – да, `false` – нет; возвращает логическое значение;
этот метод статический;

`isISOControl()`

используется для проверки, является ли символ управляющим символом ISO; принимает параметры — символ (указанный символ); проверяет, является ли указаный символ управляющим символом ISO, `true` – да, `false` – нет; возвращает логическое значение;
этот метод статический;

`isLetter()`

используется для проверки является ли символ литерой; принимает параметры — `char` (указанный символ); проверяет является ли указанный символ литерой, `true` – да, `false` – нет; возвращает логическое значение;
этот метод статический;

`isLetterOrDigit()`

используется для проверки является ли символ литерой/ цифрой; принимает параметры — `char` (указанный символ); проверяет является ли указанный символ литерой/ цифрой, `true` – да, `false` – нет; возвращает логическое значение;
этот метод статический;

`isSpaceChar()`

используется для проверки является ли символ пробельным; принимает параметры — `char` (указанный символ); проверяет является ли указанный символ пробельным в Unicode, `true` – да, `false` – нет; возвращает логическое значение;
этот метод статический;

`isUpperCase()`

используется для проверки является ли символ символом в верхнем регистре; принимает параметры — `char` (указанный символ); проверяет является ли указанный символ символом в верхнем регистре, `true` – да, `false` – нет; возвращает логическое значение;
этот метод статический;

`isLowerCase()`

используется для проверки является ли символ символом в нижнем регистре; принимает параметры — `char` (указанный символ); проверяет является ли указанный символ символом в нижнем регистре, `true` – да, `false` – нет; возвращает логическое значение;
этот метод статический;

`isMirrored()`

используется для проверки является ли символ зеркально отображаемым в Unicode; принимает параметры — `char` (указанный символ); проверяет является ли указанный символ зеркально отображаемым в Unicode, `true` – да, `false` – нет; возвращает логическое значение;
этот метод статический;

`isTitleCase()`

используется для проверки является ли символ обозначающим заглавную; принимает параметры — `char` (указанный символ); проверяет является ли указанный символ обозначающим заглавную, `true` – да, `false` – нет; возвращает логическое значение;
этот метод статический;

`isUnicodeIdentifierPart()`

используется для проверки, является ли символ допустимым в именах; принимает параметры — `char` (указанный символ); проверяет является ли указанный символ допустимым в именах в Unicode, исключая первый символ имен, `true` – да, `false` – нет; возвращает логическое значение;
этот метод статический;

`isUnicodeIdentifierStart()`

используется для проверки, является ли символ допустимым в именах (как первый символ); принимает параметры — `char` (указанный символ); проверяет является ли указанный символ допустимым в именах в Unicode (как первый символ), `true` – да, `false` – нет; возвращает логическое значение;
этот метод статический;

`isWhiteSpace()`

используется для проверки является ли символ пробельным; приимает параметры — `char` (указанный символ); проверяет являетсяли указанный символ пробельным, `true` – да, `false` – нет; возвращает логическое значение;
этот метод статический;

`toUpperCase()`

используется для возврата символа в верхнем регистре; принимает параметры — `char` (указанный символ); возвращает символ в верхнем регистре эквивалентный указанному; возвращает `char`;
этот метод статический;

`toLowerCase()`

используется для возврата символа в нижнем регистре; принимает параметры — символ (указанный); возвращает символ в нижнем регистре эквивалентный указанному; возвращает `char`;
этот метод статический;

`toTitleCase()`

используется для возврата символа (заглавная); принимает параметры — `char` (указанный символ); возвращает символ (заглавная) эквивалентный указанному; возвращает `char`;
этот метод статический;

Переменные класса Character

`BYTES`

константа; длина значения в байтах;

`MAX_RADIX`

костанта; максимальный корень системы счисления;

`MIN_RADIX`

константа; минимальный корень системы счисления;

`MAX_VALUE`

константа; максимальное значение;

`MIN_VALUE`

константа; минимальное значение;

`TYPE`

константа; объект класса `Class` для типа `char`;

Класс Number

Это абстрактный класс, являющийся суперклассом для численных типов (за исключением символов). Класс `Number` инкапсуллирует ряд методов, имеющих имя вида `TypeNameValue()`, где `TypeName` это имя конкретного численного типа (за исключением символов); эти методы переопределяются в подклассах, и предназначены для явного приведения к соответствующим счисленным типам.

Классы — обертки целочисленных

Это следующие классы.

Класс Byte

Конструктор класса Byte

`Byte()`

конструктор класса `Byte`; принимает параметр — `byte` или строку (должна содержать запись числа); создает объект своего класса; возвращает ничтоже;

Методы класса Byte

`valueOf()`

используется для возврата инкапсуллируемого значения; параметров не принимает; возвращает свое инкапсуллируемое значение; возвращает `byte`;

перегруженный вариант этого метода принимает параметр — строку (содержит число), или строку и число (основание системы счисления результата), возвращает число получая его из параметра, эта версия метода статическая;

`hashCode()`

используется для возврата хэша своего объекта; параметров не принимает; возвращает хэш своего объекта; возвращает `int`;
этот метод статический;

`toString()`

используется для возврата строкового представления инкапсуллированного значения; параметров не принимает; возвращает строковое представление инкапсуллированного значения; возвращает строку;
перегруженная версия этого метода принимает параметр — `byte`, и преобразует его в строку, эта версия метода статическая;

`Методы byteValue(), shortValue(), intValue(), longValue(), floatValue(), doubleValue()`

используются для возврата инкапсуллируемого значения в соответствующем представлении; параметров не принимают; возвращают инкапсуллируемое значение в соответствующем представлении; возвращают число;

`compare()`

используется для сравнения чисел; принимает параметры — `byte` (указанное число), `byte` (указанное число); сравнивает указанные числа; возвращает int (ноль — если равны, отрицательное — если первое меньше, положительное — если первое больше);
этот метод статический;

`compareTo()`

используется для сравнения своего значения с указанным значением; принимает параметр — `Byte` (указанное значение); сравнивает свое инкапсуллируемое значение с указанным значением; возвращает `int` (ноль — если равны, отрицательное значение — если свое значение меньше, положительное значение — если больше);

`equals()`

используется для сравнения инкапсуллируемого значения с указанным значение; принимает параметр — `Byte`; сравнивает свое инкапсуллируемое значение с указанным значением; возвращает логическое значение (`true` – равны, `false` – не равны);

`decode()`

используется для возврата значения `Byte`; принимает параметр — строку (должна содержать число); возвращает значение `Byte`, получает его из параметра; возвращает `Byte`;
этот метод статический;

`parseByte()`

используется для возврата числа; принимает параметр — строку (должна содержать число); возвращает число, полученное из параметра; возвращает `byte`;
этот метод статический;

`toUnsignedInt()`

используется для возрата указанного числа в соответствующем представлении; принимает параметр — `byte`; возвращает указанное число в целевом представлении; возвращает `int`;
этот метод статический;

`toUnsignedLong()`

используется для возрата указанного числа в соответствующем представлении; принимает параметр — `byte`; возвращает указанное число в целевом представлении; возвращает `long`;
этот метод статический;

Переменные класса Byte

`TYPE`

объект класса `Class` для соответствующего класса;

`SIZE`

константа; число; обозначает длину числа в битах;

`MAX_VALUE`

константа; число; обозначает максимальное значение;

`MIN_VALUE`

константа; число; обозначает минимальное значение;

Класс Short

Конструктор класса Short

`Short()`

конструктор класса `Short`; принимает параметр — `short` или строку (должна содержать запись числа); создает объект своего класса; возвращает ничтоже;

Методы класса Short

`valueOf()`

используется для возврата инкапсуллируемого значения; параметров не принимает; возвращает свое инкапсуллируемое значение; возвращает `short`;
перегруженный вариант этого метода принимает параметр — строку (содержит число), или строку и число (основание системы счисления результата), возвращает число получая его из параметра, эта версия метода статическая;

`hashCode()`

используется для возврата хэша своего объекта; параметров не принимает; возвращает хэш своего объекта; возвращает `int`;
этот метод статический;

`toString()`

используется для возврата строкового представления инкапсуллированного значения; параметров не принимает; возвращает строковое представление инкапсуллированного значения; возвращает строку;
перегруженная версия этого метода принимает параметр — `short`, и преобразует его в строку, эта версия метода статическая;

`Методы byteValue(), shortValue(), intValue(), longValue(), floatValue(), doubleValue()`

используются для возврата инкапсуллируемого значения в соответствующем представлении; параметров не принимают; возвращают инкапсуллируемое значение в соответствующем представлении; возвращают число;

`compare()`

используется для сравнения чисел; принимает параметры — `short` (указанное число), `short` (указанное число); сравнивает указанные числа; возвращает `int` (ноль — если равны, отрицательное — если первое меньше, положительное — если первое больше);
этот метод статический;

`compareTo()`

используется для сравнения своего значения с указанным значением; принимает параметр — `Short` (указанное значение); сравнивает свое инкапсуллируемое значение с указанным значением; возвращает `int` (ноль — если равны, отрицательное значение — если свое значение меньше, положительное значение — если больше);

`equals()`

используется для сравнения инкапсуллируемого значения с указанным значение; принимает параметр — `Short`; сравнивает свое инкапсуллируемое значение с указанным значением; возвращает логическое значение (`true` – равны, `false` – не равны);

`decode()`

используется для возврата значения `Short`; принимает параметр — строку (должна содержать число); возвращает значение `Short`, получает его из параметра; возвращает `Short`;

этот метод статический;

`parseShort()`

используется для возврата числа; принимает параметр — строку (должна содержать число); возвращает число, полученное из параметра; возвращает `short`;
этот метод статический;

`toUnsignedInt()`

используется для возврата указанного числа в соответствующем представлении; принимает параметр — `short`; возвращает указанное число в целевом представлении; возвращает `int`;
этот метод статический;

`toUnsignedLong()`

используется для возврата указанного числа в соответствующем представлении; принимает параметр — `short`; возвращает указанное число в целевом представлении; возвращает `long`;
этот метод статический;

`reverseBytes()`

используется для реверса порядка байтов указанного значения; принимает параметр — `short`; реверсирует порядок байтов указанного значения; возвращает `short`;

Переменные класса Short

Это в точности те же переменные, что у класса `Byte`.

Класс Integer

Конструктор класса Integer

`Integer()`

конструктор класса `Integer`; принимает параметр — `int`/ строку (должна содержать запись числа); создает объект своего класса; возвращает ничтоже;

Методы класса Integer

`valueOf()`

используется для возврата инкапсуллируемого значения; параметров не принимает; возвращает свое инкапсуллируемое значение; возвращает `byte`;
перегруженный вариант этого метода принимает параметр — строку (содержит число), или строку и число (основание системы счисления результата), возвращает число получая его из параметра, эта версия метода статическая;

`hashCode()`

используется для возврата хэша своего объекта; параметров не принимает; возвращает хэш своего объекта; возвращает `int`;
этот метод статический;

`toString()`

используется для возврата строкового представления инкапсуллированного значения; параметров не принимает; возвращает строковое представление инкапсуллированного значения; возвращает строку;
перегруженная версия этого метода принимает параметр — `int`, и преобразует его в строку, эта версия метода статическая;

`toUnsignedString()`

используется возврата строкового представления указанного числа; принимает параметры — `int` (указанное число), `int` (основание системы счисления результата); возвращает строковое представление указанного числа (число берется без знака); возвращает строку;
этот метод статический;

`Методы byteValue(), shortValue(), intValue(), longValue(), floatValue(), doubleValue()`

используются для возврата инкапсуллируемого значения в соответствующем представлении; параметров не принимают; возвращают инкапсуллируемое значение в соответствующем представлении; возвращают число;

`compare()`

используется для сравнения чисел; принимает параметры — `int` (указанное число), `int` (указанное число); сравнивает указанные числа; возвращает `int` (ноль — если равны, отрицательное — если первое меньше, положительное — если первое больше);
этот метод статический;

`compareTo()`

используется для сравнения своего значения с указанным значением; принимает параметр — `Integer` (указанное значение); сравнивает свое инкапсуллируемое значение с указанным значением; возвращает `int` (ноль — если равны, отрицательное значение — если свое значение меньше, положительное значение — если больше);

`compareUnsigned()`

используется для сравнения чисел как беззнаковых; принимает параметры — `int` (указанное число), `int` (указанное число); сравнивает указанные числа как беззнаковые; возвращает `int` (ноль — если равны, отрицательное — если первое меньше, положительное — если больше);

`equals()`

используется для сравнения инкапсуллируемого значения с указанным значение; принимает параметр — `Integer` (указанное значение); сравнивает свое инкапсуллируемое значение с указанным значением; возвращает логическое значение (`true` – равны, `false` – не равны);

`decode()`

используется для возврата значения `Integer`; принимает параметр — строку (должна содержать число); возвращает значение `Integer`, получает его из параметра; возвращает `Integer`;
этот метод статический;

`parseInt()`

используется для возврата числа; принимает параметр — строку (должна содержать число); возвращает число, полученное из параметра; возвращает `int`;
этот метод статический;

`parseUnsignedInt()`

используетсядля возврата числа; принимает параметр — строку; возвращает число получая его из строки (число получает без знака); возвращает `int`;
этот метод статический;

`toUnsignedLong()`

используется для возрата указанного числа в соответствующем представлении; принимает параметр — `int`; возвращает указанное число в целевом представлении; возвращает `long`;
этот метод статический;

`reverse()`

используется для реверса битов числа; принимает параметр — `int` (указанное число); реверсирует биты указанного числа; возвращает `int`;
этот метод статический;

`reverseBytes()`

используется для реверса байтов указанного числа; принимает параметр — `int` (указанное число); реверсирует байты указанного числа; возвращает `int`;
этот метод статический;

`rotateLeft()`

используется для сдвига числа влево; принимает параметры — `int` (сдвигаемое число), `int` (число позиций сдвига); сдвигает сдвигаемое число влево; возвращает `int`;
этот метод статический;

`rotateRight()`

используется для сдвига числа вправо; принимает параметры — `int` (сдвигаемое число), `int` (число позиций сдвига); сдвигает сдвигаемое число вправо; возвращает `int`;
этот метод статический;

`signum()`

используется для возврата знака числа; принимает параметр — `int` (указанное число); возвращает знак указанного числа в виде числа (`0` — если число ноль, `-1` — если число отрицательное, `1` — если положительное);
этот метод статический;

`sum()`

используется для возврата суммы чисел; принимает параметры — `int` (указанное число), `int` (указанное число); складывает указанные числа; возвращает `int`;
этот метод статический;

`toBinaryString()`

используется для преобразования числа к строке (содержащей соответствующее представление числа); принимает параметр — `int` (указанное число); преобразует указанное число к строке (содержащей соответствующее представление числа); возвращает строку;
этот метод статический;

`toHexString()`

используется для преобразования числа к строке (содержащей соответствующее представление числа); принимает параметр — `int` (указанное число); преобразует указанное число к строке (содержащей соответствующее представление числа); возвращает строку;
этот метод статический;

`toOctalString()`

используется для преобразования числа к строке (содержащей соответствующее представление числа); принимает параметр — `int` (указанное число); преобразует указанное число к строке (содержащей соответствующее представление числа); возвращает строку;
этот метод статический;

`remainderUnsigned()`

используется для возрата остатка от деления; принимает параметры — `int` (делимое), `int` (делитель); возвращает остаток от целочисленного деления указанных чисел; возвращает `int`;

этот метод статический;

Переменные класса Integer

Это в точности те же переменные, что у класса `Byte`.

Класс Long

Конструктор класса Long

`Long()`

конструктор класса `Long`; принимает параметр — `long` или строку (должна содержать запись числа); создает объект своего класса; возвращает ничтоже;

Методы класса Long

`valueOf()`

используется для возврата инкапсуллируемого значения; параметров не принимает; возвращает свое инкапсуллируемое значение; возвращает `long`;
перегруженный вариант этого метода принимает параметр — строку (содержит число), или строку и число (основание системы счисления результата), возвращает число получая его из параметра, эта версия метода статическая;

`hashCode()`

используется для возврата хэша своего объекта; параметров не принимает; возвращает хэш своего объекта; возвращает `int`;
этот метод статический;

`toString()`

используется для возврата строкового представления инкапсуллированного значения; параметров не принимает; возвращает строковое представление инкапсуллированного значения; возвращает строку;
перегруженная версия этого метода принимает параметр — `long`, и преобразует его в строку, эта версия метода статическая;

`toUnsignedString()`

используется возврата строкового представления указанного числа; принимает параметры — `long` (указанное число), `long` (основание системы счисления результата); возвращает строковое представление указанного числа (число берется без знака); возвращает строку;
этот метод статический;

`Методы byteValue(), shortValue(), intValue(), longValue(), floatValue(), doubleValue()`

используются для возврата инкапсуллируемого значения в соответствующем представлении; параметров не принимают; возвращают инкапсуллируемое значение в соответствующем представлении; возвращают число;

`compare()`

используется для сравнения чисел; принимает параметры — `long` (указанное число), `long` (указанное число); сравнивает указанные числа; возвращает `int` (ноль — если равны, отрицательное — если первое меньше, положительное — если первое больше);
этот метод статический;

`compareTo()`

используется для сравнения своего значения с указанным значением; принимает параметр — `Long` (указанное значение); сравнивает свое инкапсуллируемое значение с указанным значением; возвращает `int` (ноль — если равны, отрицательное значение — если свое значение меньше, положительное значение — если больше);

`compareUnsigned()`

используется для сравнения чисел как беззнаковых; принимает параметры — `long` (указанное число), `long` (указанное число); сравнивает указанные числа как беззнаковые; возвращает `int` (ноль — если равны, отрицательное — если первое меньше, положительное — если больше);

`equals()`

используется для сравнения инкапсуллируемого значения с указанным значение; принимает параметр — `Long`; сравнивает свое инкапсуллируемое значение с

указанным значением; возвращает логическое значение (`true` – равны, `false` – не равны);

`decode()`

используется для возврата значения `Long`; принимает параметр — строку (должна содержать число); возвращает значение `Long`, получает его из параметра; возвращает `Long`;
этот метод статический;

`parseLong()`

используется для возврата числа; принимает параметр — строку (должна содержать число); возвращает число, полученное из параметра; возвращает `long`;
этот метод статический;

`parseUnsignedLong()`

используетсядля возврата числа; принимает параметр — строку; возвращает число получая его из строки (число получает без знака); возвращает `long`;
этот метод статический;

`toUnsignedLong()`

используется для возрата указанного числа в соответствующем представлении; принимает параметр — `long`; возвращает указанное число в целевом представлении; возвращает `long`;
этот метод статический;

`reverse()`

используется для реверса битов числа; принимает параметр — `long` (указанное число); реверсирует биты указанного числа; возвращает `long`;
этот метод статический;

`reverseBytes()`

используется для реверса байтов указанного числа; принимает параметр — `long` (указанное число); реверсирует байты указанного числа; возвращает `long`;
этот метод статический;

`rotateLeft()`

используется для сдвига числа влево; принимает параметры — `long` (сдвигаемое число), `long` (число позиций сдвига); сдвигает сдвигаемое число влево; возвращает `long`;
этот метод статический;

`rotateRight()`

используется для сдвига числа вправо; принимает параметры — `long` (сдвигаемое число), `long` (число позиций сдвига); сдвигает сдвигаемое число вправо; возвращает `long`;
этот метод статический;

`signum()`

используется для возврата знака числа; принимает параметр — `int` (указанное число); возвращает знак указанного числа в виде числа (`0` — если число ноль, `-1` — если число отрицательное, `1` — если положительное);
этот метод статический;

`lowestOneBit()`

используется для возврата позиции младшего установленного бита числа; принимает параметр — `long` (указанное число); возвращает позицию самого младшего из установленных битов указанного числа; возвращает `int`;
этот метод статический;

`numberOfLeadingZeros()`

используется для возврата количества старших сброшенных битов числа; принимает параметр — `long` (указанное число); возвращает количество старших сброшенных битов указанного числа; возвращает `int`;
этот метод статический;

`numberOfTrailingZeros()`

используется для возврата количества младших сброшенных битов числа; принимает параметр — `long` (указанное число); возвращает количество младших сброшенных битов указанного числа; возвращает `int`;
этот метод статический;

`max()`

используется для определения максимального числа из указанных; принимает параметры — `long` (указанное число), `long` (указанное число); возвращает максимальное число из указанных; возвращает `long`;
этот метод статический;

`min()`

используется для определения минимального числа из указанных; принимает параметры — `long` (указанное число), `long` (указанное число); возвращает минимальное число из указанных; возвращает `long`;
этот метод статический;

`sum()`

используется для возврата суммы чисел; принимает параметры — `long` (указанное число), `long` (указанное число); складывает указанные числа; возвращает `long`;
этот метод статический;

`toBinaryString()`

используется для преобразования числа к строке (содержащей соответствующее представление числа); принимает параметр — `long` (указанное число); преобразует указанное число к строке (содержащей соответствующее представление числа); возвращает строку;
этот метод статический;

`toHexString()`

используется для преобразования числа к строке (содержащей соответствующее представление числа); принимает параметр — `long` (указанное число); преобразует указанное число к строке (содержащей соответствующее представление числа); возвращает строку;
этот метод статический;

`toOctalString()`

используется для преобразования числа к строке (содержащей соответствующее представление числа); принимает параметр — `long` (указанное число); преобразует указанное число к строке (содержащей соответствующее представление числа); возвращает строку;

этот метод статический;

`remainderUnsigned()`

используется для возрата остатка от деления; принимает параметры — `long` (делимое), `long` (делитель); возвращает остаток от целочисленного деления указанных чисел; возвращает `long`;
этот метод статический;

Переменные класса Long

Это в точности те же переменные, что у класса `Byte`.

Классы - обертки нецелочисленных

Класс Float

Конструктор класса Float

`Float()`

конструктор класса `Float`; принимает параметры — число с плавающей точкой (`float/ double`), либо строку (предполагается содержащей запись числа с плавающей точкой); создает объект своего класса; возвращает ничтоже;

Методы класса Float

`Методы byteValue(), shortValue(), intValue(), floatValue(), doubleValue()`

каждый из этих методов используется для возврата инкапсуллируемого значения в соответствующем представлении; параметров не принимает; возвращает инкапсуллируемое значение в соответствующем представлении; возвращает число (соответствующего типа);

`floatToRowIntBits()`

используется для возврата представления числа (в интерпретаторе) в целочисленном виде; принимает параметр — `float` (указанное число, оно не должно быть инициализировано нечисленным значением); возвращает

представление указанного числа в интерпретаторе в целочисленном виде; возвращает `int`;
этот метод статический;

`intBitsToFloat()`

используется для возврата числа с плавающей точкой; принимает параметры — `int` (указанное число); возвращает число с плавающей точкой (интерпретирует параметр как содержащий fp- число в представлении интерпретатора); возвращает `float`;
этот метод статический;

`hashCode()`

используется для возврата хэша своего объекта; параметров не принимает; возвращаетхэш своего объекта; возвращает `int`;

`isInfinite()`

используется для проверки на бесконечность; параметров не принимает; проверяет, является ли инкапсуллируемое число бесконечностью; возвращает логическое значение (`true` – да, `false` – нет);

`isNaN()`

используется для проверки на нечисленное значение; параметров не принимает; проверяет, является ли инкапсуллируемое число нечисленным значением; возвращает логическое значение (`true` – да, `false` – нет);

`max()`

используется для возврата максимального числа из указанных; принимает параметры — `float` (указанное число), `float` (указанное число); возвращает наибольшее число из указанных; возвращает `float`;
этот метод статический;

`min()`

используется для возврата минимального числа из указанных; принимает параметры — `float` (указанное число), `float` (указанное число); возвращает наименьшее число из указанных; возвращает `float`;
этот метод статический;

`parseFloat()`

используется для возврата числа с плавающей точкой; принимает параметр — строку (должна содержать запись числа с плавающей точкой в десятичном виде); возвращает число с плавающей точкой (получая его из параметра); возвращает `float`;
этот метод статический;

`sum()`

используется для получения суммы чисел; принимает параметры — `float` (указанное число), `float` (указанное число); возвращает сумму указанных чисел; возвращает `float`;
этот метод статический;

`toHexString()`

используется для возврата шестнадцатеричного представления указанного числа; принимает параметр — `float` (указанное число); возвращает шестнадцатеричное представление указанного числа (в строковом представлении); возвращает строку;
этот метод статический;

`toString()`

используется для возврата строкового представления своего объекта; параметров не принимает; возвращает строковое представление своего объекта; возвращает строку;
есть перегруженный вариант — он принимает параметр `float` (указанное число), и возвращает его представление, этот вариант метода статический;

`valueOf()`

используется для возврата объектного представления указанного числа; принимает параметр — `float` или строка (указаное число); возвращает значение указанного числа в представлении обертки `Float`; возвращает `Float`;
этот метод статический;

Переменные класса Float

`SIZE`

константа; число; обозначает размер числа в битах;

`TYPE`

константа; объект типа `Class` для соответствующего класса;

`BYTES`

константа; число; обозначает длину числа в байтах;

`MAX_EXPONENT`

константа; число; обозначает максимальный показатель степени;

`MIN_EXPONENT`

константа; число; обозначает минимальный показатель степени;

`MAX_VALUE`

константа; число; обозначает максимальное положительное значение;

`MIN_VALUE`

константа; число; обозначает минимальное положительное значение;

`MIN_NORMAL`

константа; число; обозначает минимальное положительное нормализованное;

`POSITIVE_INFINITY`

константа; число; обозначает плюс бесконечность;

`NEGATIVE_INFINITY`

константа; число; обозначает минус бесконечность;

Класс Double

Конструктор класса Double

`Double()`

конструктор класса `Double`; принимает параметры — `double` или строка (указанное число); создает объект своего класса; возвращает ничтоже;

Методы класса Double

`Методы byteValue(), shortValue(), intValue(), floatValue(), doubleValue()`

каждый из этих методов используется для возврата инкапсуллируемого значения в соответствующем представлении; параметров не принимает; возвращает инкапсуллируемое значение в соответствующем представлении; возвращает число (соответствующего типа);

`compare()`

используется для сравнения чисел; принимает параметры — `double` (указанное число), и `double` (указанное число); сравнивает указанные числа; возвращает `int` (ноль — равны, отрицательное — если первое меньше, и положительное — если больше);
этот метод статический;

`compareTo()`

используется для сравнения инкапсуллируемого значения с указанным числом; принимает параметр — `Double` (указанное число); сравнивает свое инкапсуллируемое значение с указанным; возвращает `int` (ноль — равны, отрицательное — если свое значение меньше, и положительное — если больше);

`doubleToLongBits()`

используется для возврата представления указанного числа в интерпретаторе; принимает параметр — `double` (указанное число); возвращает представление указанного числа в интерпретаторе; возвращает `long`;
этот метод статический;

`doubleToRowLongBits()`

используется для возврата представления указанного числа в интерпретаторе; принимает параметр — `double` (указанное число, нечисленное значение не допускается); возвращает представление указанного числа в интерпретаторе; возвращает `long`;
этот метод статический;

`longBitsToDouble()`

используется для возврата числа с плавающей точкой; принимает параметр — `long` (содержит представление fp- числа в интерпретаторе); возвращает число с лавающей точкой (получая его представление из параметра); возвращает `double`;

`equals()`

используется для сравнения инкапсуллируемого значения с указанным; принимает параметр — `Double` (указанное значение); сравнивает свое значение с указанным; возвращает логическое значение (`true` – равны, `false` – не равны);

`isInfinite()`

используется для проверки на бесконечность; параметров не принимает; проверяет инкапсуллируемое значение на бесконечность; возвращает логическое значение (`true` – да, `false` – нет);
перегруженная версия этого метода принимает параметр — `double`, это число и будет проверяться на бесконечность, эта версия метода статическая;

`isNaN()`

используется для проверки на нечисленное значение; параметров не принимает; проверяет, является ли инкапсуллируемое число нечисленным значением; возвращает логическое значение (`true` – да, `false` – нет);

`hashCode()`

используется для возврата хэша своего объекта; параметров не принимает; возвращает хэш своего объекта; возвращает `int`;

`max()`

используется для возврата максимального числа из указанных; принимает параметры — `double` (указанное число), `double` (указанное число); возвращает наибольшее число из указанных; возвращает `double`;
этот метод статический;

`min()`

используется для возврата минимального числа из указанных; принимает параметры — `double` (указанное число), `double` (указанное число); возвращает наименьшее число из указанных; возвращает `double`;
этот метод статический;

`parseDouble()`

используется для возврата числа с плавающей точкой; принимает параметр — строку (должна содержать запись числа с плавающей точкой в десятичном виде); возвращает число с плавающей точкой (получая его из параметра); возвращает `double`;
этот метод статический;

`sum()`

используется для получения суммы чисел; принимает параметры — `double` (указанное число), `double` (указанное число); возвращает сумму указанных чисел; возвращает `double`;
этот метод статический;

`toHexString()`

используется для возврата шестнадцатеричного представления указанного числа; принимает параметр — `double` (указанное число); возвращает шестнадцатеричное представление указанного числа (в строковом представлении); возвращает строку;
этот метод статический;

`toString()`

используется для возврата строкового представления своего объекта; параметров не принимает; возвращает строковое представление своего объекта; возвращает строку;

есть перегруженный вариант — он принимает параметр `double` (указанное число), и возвращает его представление, этот вариант метода статический;

`valueOf()`

используется для возврата объектного представления указанного числа; принимает параметр — `double` или строка (указаное число); возвращает значение указанного числа в представлении обертки `Float`; возвращает `double`;
этот метод статический;

Переменные класса Double

Это в точности те же переменные, что у класса `Float`.

Класс Math

Этот класс инкапсуллирует различные математические функции.

Тригонометрические функции

`sin()`

используется для вычисления синуса; принимает параметр — `double` (значение угла в радианах); вычисляет синус заданного угла; возвращает `double`;
этот метод статический;

`cos()`

используется для вычисления косинуса; принимает параметр — `double` (значение угла в радианах); вычисляет косинус заданного угла; возвращает `double`;
этот метод статический;

`tan()`

используется для вычисления тангенса; принимает параметр — `double` (значение угла в радианах); вычисляет тангенс заданного угла; возвращает `double`;
этот метод статический;

`asin()`

используется для вычисления арксинуса; принимает параметр — `double` (значение синуса угла); вычисляет арксинус заданного угла; возвращает `double`;
этот метод статический;

`acos()`

используется для вычисления арккосинуса; принимает параметр — `double` (значение косинуса угла); вычисляет арккосинус заданного угла; возвращает `double`;
этот метод статический;

`atan()`

используется для вычисления арктангенса; принимает параметр — `double` (значение тангенса); вычисляет арктангенс заданного угла; возвращает `double`;
этот метод статический;

`atan2()`

используется для вычисления арктангенса2; принимает параметры — `double` (абсцисса), и `double` (ордината); вычисляет арктангенс2 заданного угла; возвращает `double`;
этот метод статический;

Прочие тригонометрические функции

Прочие тригонометрические функции — в других источниках.

Экспоненциальные функции

`sqrt()`

используется для вычисления квадратного корня; принимае параметр — `double` (указанное значение); вычисляет квадратный корень указанного значения; возвращает `double`;
этот метод статический;

`cbrt()`

используется для вычисления кубического корня; принимае параметр — `double` (указанное значение); вычисляет кубический корень указанного значения; возвращает `double`;
этот метод статический;

`exp()`

используется для возврата экспоненты; принимает параметр — `double` (указанное значение); возвращает показатель степени указанного числа; возвращает `double`;
этот метод статический;

`log()`

используется для вычисления логарифма; принимает параметр — `double` (указанное значение); вычисляет логарифм указанного числа; возвращает `double`;
этот метод статический;

`log10()`

используется для вычисления десятичного логарифма; принимает параметр — `double` (указанное значение); вычисляет десятичный логарифм указанного числа; возвращает `double`;
этот метод статический;

`pow()`

используется для возведения в степень; принимает параметры — `double` (основание степени), и `double` (показатель степени); вычисляет — основание в степени показателя; возвращает `double`;
этот метод статический;

`scalb()`

используется для вычисления значения `scalb()`; принимает параметры — нецелочисленное, и `int`; вычисляет «первый аргумент умножить на два (в степени второй аргумент)»; возвращает `double`;
этот метод статический;

Прочие экспоненциальные функции

прочие экспоненциальные функции — в других источниках.

Функции округления

abs()

используется для возврата абсолютного значения; принимает параметр — `int/ long/ float/ double`; возвращает абсолютное значение параметра; возвращает `int`;
этот метод статический;

ceil()

используется для округления в большую сторону; принимает параметр — `double`; округляет параметр в большую сторону; возвращает `double`;
этот метод статический;

floor()

используется для округления в меньшую сторону; принимает параметр — `double`; округляет параметр в меньшую сторону; возвращает `double`;
этот метод статический;

floorDiv()

используется для возврата частного; принимает параметры — `int/ long/ float/ double` (делимое), и `int` (делитель); возвращает частное от целочисленного деления; возвращает `int`;
этот метод округляет результат соответствующим способом;
этот метод статический;

floorMod()

используется для возврата остатка от деления; принимает параметры — `int/ long/ float/ double` (делимое), и `int` (делитель); возвращает остаток от целочисленного деления; возвращает `int`;
этот метод округляет результат соответствующим способом;
этот метод статический;

`max()`

используется для возврата наибольшего из указанных чисел; принимает параметры — `int` и `int`, либо `long` и `long`, либо `float` и `float`, либо `double` и `double`; возвращает наибольшее из указанных числе; возвращает `int` или `long` или `float` или `double` соответственно;
этот метод стантический;

`min()`

используется для возврата наименьшего из указанных чисел; принимает параметры — `int` и `int`, либо `long` и `long`, либо `float` и `float`, либо `double` и `double`; возвращает наименьшее из указанных числе; возвращает `int` или `long` или `float` или `double` соответственно;
этот метод стантический;

`nextAfter()`

используется для возврата числа (от параметра до пункта направления); принимает параметры — `float` (указанное число) и `float` (число, пункт направления), либо `double` и `double`; возвращает число, следующее за указанным в направлении пункта направления; возвращает `float` или `double` соответственно;
этот метод статический;

`nextDown()`

используется для возврата числа (следующее за указанным в меньшую сторону); принимает параметр — `float` (указанное число), либо `double`; возвращает число, следующее за указанным в меньшую сторону; возвращает `float` или `double` соответственно;
этот метод статический;

`nextUp()`

используется для возврата числа (следующее за указанным в большую сторону); принимает параметр — `float` (указанное число), либо `double`; возвращает число, следующее за указанным в большую сторону; возвращает `float` или `double` соответственно;
этот метод статический;

`rint()`

используется для округления в сторону ближайшего целого; принимает параметр — `double` (указанное число); округляет указанное число в сторону ближайшего целого; возвращает `double`;
этот метод статический;

`round()`

используется для округления в сторону ближайшего целого; принимает параметр — `float` или `double` (указанное число); округляет указанное число в сторону ближайшего целого; возвращает `int` или `long` соответственно;
этот метод статический;

Прочие функции округления

Прочие функции округления — в других источниках.

Прочие методы

`copySign()`

используется для назначения числу знака; принимает параметры — `float/ double` (указанное число), и `float/ double` (знак); назначает указанному числу знак; возвращает `float` или `double` соответственно;
этот метод статический;

`addExact()`

используется для складывания чисел; принимает параметры — `int/ long` (указанное число), и `int/ long` (указанное число); складывает указанные числа; возвращает `int` или `long` соответственно;
этот метод статический;

`incrementExact()`

используется для инкремента числа; принимает параметры — `int/ long` (указанное число); производит инкремент указанного числа; возвращает `int` или `long` соответственно;
этот метод статический;

`decrementExact()`

используется для декремента числа; принимает параметры — `int/ long` (указанное число); производит декремент указанного числа; возвращает `int` или `long` соответственно;
этот метод статический;

`getExponent()`

используется для возврата степени числа; принимает параметры — `float/ double` (указанное число); возвращает степень числа (конкретно, его представления в интерпретаторе); возвращает `int`;
этот метод статический;

`hypot()`

используется для вычисления гипотенузы; принимает параметры — `double` (катет), и `double` (катет); производит вычисление гипотенузы (по катетам); возвращает `double`;
этот метод статический;

`IEEERemainder()`

используется для вычисления остатка от деления; принимает параметры — `double` (делимое), и `double` (делитель); производит вычисление остатка от деления; возвращает `double`;
этот метод статический;

`multiplyExact()`

используется для умножения; принимает параметры — `int/ long` (множитель), и `int/ long` (множитель); производит умножение указанных значений; возвращает `int/ long`;
этот метод статический;

`negateExact()`

используется для отрицания числа; принимает параметры — `int/ long` (указанное число); производит отрицание указанного числа; возвращает `int/ long`;
этот метод статический;

`random()`

используется для возврата случайного числа; параметров не принимает; возвращает случайное число (от нуля до единицы); возвращает `double`;
этот метод статически;

`signum()`

используется для возврата знака числа; принимает параметр – `float/ double` (указанное число); возвращает знак указанного числа (в виде числа с плавающей точкой: ноль — если указанное число ноль, минус один — если отрицательное, один — если положительное); возвращает `float`;
этот метод статический;

`subtractExact()`

используется для вычисления разности; принимает параметры – `int/ int` (уменьшаемое), `и int/ long` (вычитаемое); вычисляет разность; возвращает `int/ long`;
этот метод статический;

`toIntExact()`

используется для преобразования к `int`; принимает параметр – `long` (указанное значение); преобразует указанное значение к `int`; возвращает `int`;
этот метод статический;

`toRadians()`

используется для преобразования градусов в радианы; принимает параметр — `double` (значение угла в градусах); производит преобразование значения угла из градусов в радианы; возвращает `double`;
этот метод статический;

`toDegrees()`

используется для преобразования радиан в градусы; принимает параметр — `double` (значение угла в радианах); производит преобразование значения угла из радиан градусы; возвращает `double`;
этот метод статический;

Класс String

Это класс, представляющий собой строки. Все строки в языке Java представлены в кодировке Unicode.
Строки являются не изменяемыми (с точки зрения их реализации в интерпретаторе). Программист, конечно же, может присвоить уже инициализированной значением строке новое значение — но, с точки зрения интерпретатора, это просто перезапись новым значением, а не изменение уже существующего значения. Такая реализация строк в интерпретаторе была принята из соображений эффективности с точки зрения реализации.
Классы StringBuilder, и StringBuffer используются для формирования строк и буферизованного ввода- вывода строк соответственно. Эти классы эмулируют изменяемость строк в интерпретаторе.
Классы String, StringBuilder, и StringBuffer являются завершенными.
Операции со строками выполняются со следующими особенностями:
литералы строк (прописываются так « … ») автоматически преобразуются к объектам- строкам (это касается э.т.д. и Object);
оператор + перегружается языком для строк, и обозначает их конкатенацию;
операция конкатенации (+) возможна и с нестроковыми типами (тогда они приводятся к строке);
операция сравнения (==) по прежнему сравнивает только ссылки на объекты;
некоторые методы работают с позицией символа — позиции стартуют с нуля;

необходимо помнить об этих особенностях.

Конструктор класса String

```
String()
```

конструктор класса String; параметров не принимает; создает объект своего класса; возвращает ничтоже;

Методы класса String

```
lenght()
```

используется для возврата длины строки; параметров не принимает; возвращает длину своей строки; возвращает int;

`charAt()`

используется для возврата символа; принимает параметр — `int` (позиция символа); возвращает символ по указанной позиции; озвращает `char`;

`getChars()`

используется для возврата символов; принимает параметры – `int` (стартовая позиция), `int` (финальная позиция), `char[]` (ссылка на целевой массив, принимает символы из строки), `int` (целевая позиция целевого массива); возвращает символы из своей строки (со стартовой позиции по финальную) в целевой массив (начиная пополнять его с уелевой позиции); возвращает ничтоже;

`getBytes()`

используется для возврата байтов; параметров не принимает; возвращает массив байтов, составляющих контент своей строки; возвращает `byte[]`;

`Методы сравнения строк`

будут рассмотрены наиболее употребимые из этих методов;
это следующие методы;

`equals()`

используется для сравнения строк; принимает параметр — строку; сравнивает свою строку с указанной (с учетом регистра символов); возвращает логическое значение (`true` – равны, `false` – не равны);

`equalsIgnoreCase()`

используется для сравнения строк; принимает параметр — строку; сравнивает свою строку с указанной (без учета регистра символов); возвращает логическое значение (`true` – равны, `false` – не равны);

`regionMatches()`

используется для сравнения частей строк; принимает параметры – `int` (стартовая позиция своей строки), `String` (указанная строка), `int` (стартовый индекс указанной строки), `int` (количество символов с указанных позиций обеих строк, которые участвуют в сравнении); руководстуясь переданными

параметрами, выполняет сравнение подстрок; возвращает логическое значение (`true` – равны, `false` – не равны);
перегруженный вариант этого метода принимает параметры — `boolean` (`true` – учитывать регистр, `false` – нет), и те же параметры, что неперегруженная версия; перегруженная версия производит сравнение с учетом регистра символов (в зависимости от переданных параметров); перегруженная версия не отличается типом возвращаемого значения;

`startsWith()`

используется для проверки, стартует ли своя строка с указанной строки; принимает параметр — строку (указанная строка); проверяет, стартует ли своя строка с указанной строки; возвращает логическое значение (`true` – да, `false` – нет);

`endsWith()`

используется для проверки, финиширует ли своя строка указанной строкой; принимает параметр — строку (указанная строка); проверяет, финиширует ли своя строка указанной строкой; возвращает логическое значение (`true` – да, `false` – нет);

`compareTo()`

используется для сравнения строк; принимает параметр — строку (указанная строка); сравнивает свою строку с указанной; возвращает `int` (ноль — если равны, отрицательное число — если своя строка меньше, положительное число — если больше);
этот метод производит сравнение с учетом регистра;

`compareToIgnoreCase()`

используется для сравнения строк; принимает параметр — строку (указанная строка); сравнивает свою строку с указанной; возвращает `int` (ноль — если равны, отрицательное число — если своя строка меньше, положительное число — если больше);
этот метод производит сравнение без учета регистра;

`Методы поиска в строках`

это следующие методы;

`indexOf()`

используется для возврата позиции символа в строке; принимает параметр — `char` (указанный символ); возвращает позицию указанного символа в строке; возвращает `int`;
при ненахождении позиции возвращает минус единицу;
перегруженная версия метода принимает параметр — строку и `int` (стартовая позция поиска, этот параметр можно опустить), и ищет ее позицию в своей строке;

`lastIndexOf()`

используется для возврата последней позиции символа в строке; принимает параметр — `char` (указанный символ); возвращает последнюю позицию указанного символа в строке; возвращает `int`;
при ненахождении позиции возвращает минус единицу;
перегруженная версия метода принимает параметр — строку и `int` (стартовая позция поиска, этот параметр можно опустить), и ищет ее позицию в своей строке;

`Методы перезаписи строк`

это следующие методы;

`substring()`

используется для возврата подстроки; принимает параметры — `int` (стартовая позиция, с которой возвращается подстрока), и `int` (финальная позиция, этот параметр можно опустить); возвращает подстроку своей строки; возвращает строку;

`concat()`

используется для конкатенации строк; принимает параметр — строку (указанная строка); выполняет конкатенацию своей строки с указанной; возвращает строку;

`replace()`

используется для замены символов своей строки; принимает параметры — `char` (заменяемый символ), и `char` (замещающий символ); заменяет заменяемый символ воей строки (во всех позициях) заменяющим символом; возвращает строку;

`trim()`

используется для устрижения пробельных символов своей строки; параметров не принимает; устригает пробельные символы (из начала и конца) своей строки; возвращает строку;

`valueOf()`

используется для возврата строкового представления указанного значения (и, таким образом, используется в задачах перезаписи строк); принимает параметр — объект или `char[]/ long/ double` (указанное значение); возвращает строковое представление указанного значения; возвращает строку;
этот метод статический;

`toLowerCase()`

используется для преобразования к нижнему регистру; параметров не принимает; преобразует символы своей строки к нижнему регистру; возвращает строку;

`toUpperCase()`

используется для преобразования к верхнему регистру; параметров не принимает; преобразует символы своей строки к верхнему регистру; возвращает строку;

`join()`

используется для соединения ряда строк; принимает параметры — строку (используется как разделитель, употребляемый между соединяемых строк), и спиок строк (строки этого списка прописываются просто через запятую); соединяет указанные строки, используя указанный разделитель; возвращает строку;
этот метод статический;

`Прочие методы строк`

это следующие методы;

`codePointAt()`

используется для возврата кодовой точки в Unicode; принимает параметр – `int` (указанная позиция); возвращает кодовую точку в Unicode на указанной позиции; возвращает `int`;

`codePointBefore()`

используется для возврата кодовой точки в Unicode; принимает параметр – `int` (указанная позиция); возвращает кодовую точку в Unicode до указанной позиции; возвращает `int`;

`codePointCount()`

используется для возврата количества кодовых точек; принимает параметры — `int` (стартовая позиция своей строки), `int` (финальная позиция своей строки); возвращает количество кодовых точек своей строки со стартовой позиции по финальную; возвращает `int`;

`contains()`

используется для проверки, содержит ли своя строка указанную подстроку; принимает параметр — строку; проверяет, содержит ли своя строка указанную подстроку; возвращает логическое значение (`true` – да, `false` – нет);

`conentEquals()`

используется для проверки, содержит ли своя строка указанную подстроку; принимает параметр — строку; проверяет, содержит ли своя строка указанную подстроку; возвращает логическое значение (`true` – да, `false` – нет);
перегруженная версия метода работает с параметро `StringBuffer`;

`isEmpty()`

используется для проверки, пуста ли строка; параметров не приинимает; проверяет, пуста ли своя строка (нулевая ли у нее длина); возвращает логическое значение (`true` – да, `false` – нет);

`matches()`

используется для проверки своей строки на совпадение (полное/ частичное) с указанной строкой; принимает параметр — строку; возвращает логическое значение (`true` – да, `false` – нет);

`offsetByCodefirst()`

используется для возврата позиции в строке; принимает параметры — int (стартовая позиция поиска), и int (дистанция от стартовой позиции); возвращает

идекс позиции, отстоящей от указанной стартовой позиции на указанную дистанцию; возвращает `int`;

`replaceFirst()`

используется для замещения части строки; принимает параметры — строку (замещаемая строка), и строка (замещающая строка); первую найденную замещаемую строку в своей строке замещает указанной строкой; возвращает строку;

`replaceAll()`

используется для замещения части строки; принимает параметры — строку (замещаемая строка), и строка (замещающая строка); все найденные замещаемые строки в своей строке замещает указанной строкой; возвращает строку;

`split()`

используется для разделения строки; принимает параметр — строку (используется как разделтель); разделяет свою строку на ряд строк (при обнаружении разделителя); возвращает `String[]`;
перегруженная версия этого метода принимает параметры — строку и `int` (положительное число — разбивать строку не более чем на это число подстрок, ноль или отрицательное число — разбовать строку полностью), перегруженная версия типом возвращаемого значенияне отличается;

`subSequence()`

используется для возврата подстроки; принимает параметры – `int` (стартовая позиция), `int` (финальная позиция); возвращает подстроку из своей строки со стартовой позиции по финальную; возвращает строку;

Переменные класса String

Не представляют интереса.

Класс StringBuffer

Этот класс используется для эмуляции модифицируемости строк. Объекты этого класса автоматически приводимы к обычным строкам.
Кроме сказанного о нем выше, этот класс содержит дополнительные методы.
Метод `capacity()` возвращает объем памяти, зарезервированный для строки (параметров не принимает, и возвращает `int`).

Метод `ensureCapacity()` задает заведомо определенный объем памяти, зарезервированный для строки — но при необходимости этот объем увеличивается (принимает параметры — `int` (объем памяти), и возвращает ничтоже), этот метод полезен тем, что позволяет заведомо урегулировать зарезервированный объем памяти.
Метод `setLenght()` задает длину хранимой в выделенной памяти строки (принимает параметр — `int` (длина строки), и возвращает ничтоже), этот метод полезен тем что позволяет усечь/ расширить хранимую строку.
Метод `setCharAt()` инициализирует указанный символ целевым симолом (приинимает параметры — `int` (указанная позиция символа), `char` (целевой символ), и возвращает ничтоже).
Метод `append()` приписывает к своей строке указанное значение (принимает параметр – `String/ int/ Object` (указанное значение), и возвращает `StringBuffer`).
Метод `insert()` вставляет в свою строку указанное значение (принимает параметры – `int` (позиция вставки) и `String/ char/ Object` (указанное значение), и возвращает `StringBuffer`).
Метод `delete()` удаляет секвенцию символов из своей строки со стартовой позиции (принимает параметр – `int` (стартовая позиция), и возвращает `StringBuffer`).
Метод `delete()` удаляет секвенцию символов из своей строки со стартовой позиции по финальную (принимает параметры – `int` (стартовая позиция) и `int` (финальная позиция), и возвращает `StringBuffer`).
Прочие методы не представляют собой интереса.
Переменные не представляют собой интереса.

Класс StringBuilder

Это устаревший суррогат вышеописанного класса. Он почти полностью идентичен ему (за исключением того, что механизмы синхронизации не работают с этим суррогатом).

Класс Enum

Это специфический класс, и его объекты создаются не конструктором, а специфической записью. Однако, у этого класса есть методы.

Методы класса Enum

`clone()`

используется для клонирования; параметров не принимает; по сути, не делает своей работы, так как клонирование перечислениям не доступно; возвращает перечисление;
этот метод завершенный;
этот метод описан здесь лишь потому, что он обычно описывается в справ.;

`compareTo()`

используется для сравнения своего перечисления с указанной константой перечисления; принимает параметр — константу перечисления; сравнивает свое перечисление с указанной константой (своего) перечисления; возвращает `int` (ноль — равны, отрицательное — свое меньше, положительное — больше);
этот метод завершенный;

`equals()`

используется для сравнения своего перечисления с указанной константой перечисления; принимает параметр — константу перечисления; сравнивает свое перечисление с указанной константой (своего) перечисления; возвращает логическое значение (`true` – равны, `false` – нет);
этот метод завершенный;

`getDeclaringClass()`

используется для возврата типа перечисления; параметров не принимает; возвращает тип своего перечисления; возвращает объект класса `Class`;
этот метод завершенный;

`hashCode()`

используется для возврата хэша своего объекта; параметров не принимает; возвращает хэш своего объекта; возвращает `int`;
этот метод завершенный;

`name()`

используется для возврата имени константы своего перечисления; параметров не принимает; возвращает имя константы своего перечисления; возвращает строку;
этот метод завершенный;

`ordinal()`

используется для возврата позиции константы своего перечисления в списке констант перечисления; параметров не принимает; возвращает позицию константы своего перечисления в списке констант перечисления; возвращает `int`;
этот метод завершенный;

`toString()`

используется для возврата строкового представления своего объекта; параметров не принимает; возвращает имя константы своего перечисления; возвращает строку;
этот метод завершенный;

`valueOf()`

используется для возврата значения константы перечисления; принимает параметры — объект класса `Class` (указывает класс перечисления), и строку (указывает имя константы из списка перечисления указанного класса); возвращает целевую константу целевого класса перечисления; возвращает объект (класса из списка перечисления);
этот метод статический;

Класс System

Этот класс содержит различные методы и переменные, предназначенные для работы с системой, и более общего назначения.
Члены этого класса статические.

Методы класса System

`arraycopy()`

используется для копирования массивов; принимает параметры — массив (источник копируемых элементов), `int` (стартовая позиция в массиве-источнике), массив (приемник копируемых элементов), `int` (стартовая позиция в массиве приемнике), `int` (количество копируемых элементов); руководствуясь переданными параметрами, выполняет копирование элементов; возвращает ничтоже;
этот метод статический;

`currentTimeMillis()`

используется для возврата системного времени; параметров не принимает; возвращает системное время в миллисекундах (это количество миллисекунд, прошедших с `01.01.1970`); возвращает `long`;
этот метод статический;

`nanoTime()`

используется для возврата значения времени; параметров не принимает; возвращает значение точного системного времени; возвращает `long`;
этот метод статический;
время в наносекундах обновляется достаточно быстро (даже для машины), так что на момент получения такого времени оно уже будет не актуально (по крайней мере, в наносекундах);

`lineSeparator()`

используется для возврата ряда символов- разделителей; параметров не принимает; возвращает ряд символов — разделителей строк; возвращает строку;
этот метод статический;

`console()`

используется для возврата консоли; параметров не принимает; возвращает текущую консоль; возвращает ссылку на объект;
этот метод статический;

`setErr()`

используется для установки целевого потока вывода ошибок; принимает параметр — поток; устанавливает указанный поток потоком вывода ошибок; возвращает ничтоже;
этот метод статический;

`setIn()`

используется для установки целевого потока ввода; принимает параметр — поток; устанавливает указанный поток потоком ввода; возвращает ничтоже;
этот метод статический;

`setOut()`

используется для установки целевого потока вывода; принимает параметр — поток; устанавливает указанный поток потоком вывода; возвращает ничтоже; этот метод статический;

`Прочие методы`

о них — в справочнике;

`Переменные класса System`

О них — в справочнике.

Многопоточность (Multithreading)

Многопоточное программирование

О многозадачности

Многозадачность на основе процессов позволяет контроллировать запуск нескольких субстанций программы на виртуальной машине. Многозадачность на основе потоков позволяет контроллировать исполнение одной субстанции процесса в нескольких его потоках. Многозадачность на основе потоков позволяет более тонко управлять вычислительными ресурсам компьютера, чем многозадачность на основе процессов.
Об однопоточных программах необходимо помнить следующее — программа не может выполняться далее, пока не будет исполнена ее текущая исполняющаяся часть (если эта часть заведомо долгая (например, пользовательский ввод), то программа будет фактически простаивать).
Современные ОС поддерживают многозадачность на основе процессов, и на основе потоков внутри каждого процесса.
Язык программирования `Java` позволяет управлять многозадачностью на основе потоков (и оставляет управление процессами на откуп виртуальной машине).

Многопоточное программирование на ЯПВУ Java

`О потоках`

Многопоточная программа исполняется в ряде модулей — потоков.

Жизненный цикл потоков

Жизненный цикл потока проистекает следующим образом. Новый поток создается (состояние потока – NEW). Затем он вызывается, и диспетчеризуется планировщиком потоков (состояние потока – RUNNABLE). Затем, в ходе исполнения поток может обнаружить необходимость доступа к разделяемому ресурсу — если ресурс используется, то поток ожидает его освобождения (тогда состояние потока – BLOCKED); либо, поток уступает свое время другому потоку (тогда его состояние – WAITING); либо, поток диспетчерезуется планировщиком и вытесняется на время исполнения другого потока (тогда его состояние – TIME_WAITING). В конце жизненного цикла поток завершается (тогда его состояние – TERMINATED).

Приоритеты потоков

Потоки имеют приоритеты. Приоритеты имеют значение для диспетчерзации потоков. Чем выше приоритет потока, тем первее он в очереди потоков, готовых к исполнению. Приоритету задается значение [1 … 10], 5 by default.
Если исполняющийся поток имеет более низкий приоритет, чем стоящий за ним в очереди, то этот (более высокоприоритетный) поток вытесняет исполняющийся поток.

Механизмы синхронизации

Потоки могут использовать разделяемые ресурсы (какие- либо данные, разделяемые между потоками в использовании). Это обозначает необходимость синхронизации доступа к разделяемым ресурсам. Механизмом синхронизации в языке Java является монитор.
Монитор есть следующий механизм. Это механизм, имеющий следующие составляющие: мьютекс — используется для осуществления блокировки разделяемого ресурса; и код, выполняющий работу с мьютексом.
В языке Java монитор используется неявно. Когда поток программы вызывает какой- либо синхронизированный метод, задействуется монитор. При исполнении тела такого метода остальные потоки вынуждены ожидать освобождения разделяемого ресурса. Соответственно, при освобождении этого ресурса ожидающиеего потоки могут приступить к работе с ним.

Механизмы межпоточного взаимодействия

Чтобы потоки, ожидающие освобождения ресурса, своевременно обнаружили его освобождение, они извещаются об освобождении ресурса.
Для извещения применяется обмен сообщениями между потоками. Поток, освободивший ресурс, отправляет сообщение об освобождении ресурса.

Класс Thread

Класс `Thread` реализует поддержку потоков. Он реализует интерфейс `Runnable`. Этот интерфейс инкапсуллирует общие для потоков методы (что логично для интерфейса).
Объект класса `Thread` инкапсуллирует поток. Новый поток создается наследованием от класса `Thread`, или же реализацией интерфейса `Runnable`.

Потоки программы

Программа на языке `Java` стартует исполнение с точки входа. Поток, инкапсуллирующий точку входа, называется «главный поток». Прочие («дочерние») потоки программы порождаются при ее исполнении, и это производится главным потоком. В устоявшейся практике программирования главный поток отвечает за финализацию цикла работы программы, и содержит необходимый для этого код.
Создание потоков реализацией интерфейса `Runnable` производится следующим образом. Пользовательский класс должен объявлять метод `run()` – при этом, в этом классе должен быть единственный метод с этим именем. Этот метод должен быть объявлен как публичный, и возвращающий ничтоже. В теле этого метода должен быть прописан код, составляющий точку входа в поток. В пользовательском классе также должен быть получен экземпляр класса `Thread`, этот экземпляр (при данном способе создания потока) создается конструктором `Thread(this, " … ")`, где `this` – это экземпляр этого пользовательского класса, а строка — это присваиваемое потоку имя. Созданный поток запускается методом `start()` класса `Thread`; тот метод вызывает метод `run()` потока программы. Код создания потока программы принимает вид:

```
// объявление пользовательского класса
class UserClassName implements Runnable {
Thread ThreadName;

// конструктор пользовательского класса,
// запускающий создаваемый им поток
        UserClassName() {
                ThreadName = new Thread(this, "Name");
                ThreadName.start();
        }

// объявляем точку входа в поток
        public void run() {
         …
        }
}

// точка входа в программу
        class ClassName {
```

```
                public static void main(String args[]) {
                new UserClassName();
                }
        }
```

при этом, необходимо помнить, что точка входа в программу исполняется в главном потоке программы. Соответственно, использование в этом потоке пользовательского потока приводит к порождению дочернего потока. Это всегда два разных потока (хотя это и не очевидно из записи — так как явно создавать главный поток не требуется, он и так создается JVM).

Как и говорилось выше, поток может быть создан и созданием класса, наследующего от `Thread`. Этот класс должен переопределять метод `run()`, и вызывать метод `start()`. Код создания потока программы принимает вид:

```
// объявление пользовательского класса
class UserClassName {

// конструктор пользовательского класса,
// запускающий поток
        UserClassName() {
                super( … );
                start();
        }

// переопределение метода run()
        public void run() {
         …
        }
}

// точка входа в программу
class ClassName {
        public static void main(String args[]) {
        new UserClassName();
                 …
}
```

в этом коде конструктору `super()` следует передать строку — имя создаваемого им потока.

Выбор конкретного способа создания потоков остается на откуп программисту.

Если необходимо создать большее количество потоков, то в точке входа просто используется ряд соответствующих записей (`new UserClassName();`).

Как говорилось выше, согласно устоявшейся традиции программирования, главный поток завершается последним. Это организуется применением методов потоков.

Работа с потоками осуществляется методами потоков.

Методы потоков

`start()`

используется для инкапсуллирования старта потока; параметров не принимает; этот метод вызывает метод `run()`, и исполнение потока стартует; возвращает ничтоже;

`run()`

используется для инкапсуллирования точки входа в поток; параметров не принимает; этот метод вызывается в результате вызова метода `start()`, и исполнение потока стартует с того метода — метода `run()` (поэтому этот метод должен инкапсуллировать точку входа в поток); возвращает ничтоже;

`sleep()`

используется для переведение потока в состояние `TIMED_WAITING`; принимает параметры – `long`, мс (время пребывания в спящем состоянии); переводит поток в состояние `TIMED_WAITING`; возвращает ничтоже;
этот метод может принять и два параметра (того же типа) — мс и нс;

`join()`

используется для присоединения к потоку (в плане условия завершения); параметров не принимает; выполняет присоединение потока (в контексте которого этот метод вызван) к потоку (относительно которого этот метод вызван как его метод); возвращает ничтоже;
условие завершения — поток (вызвавший этот метод) завершится не раньше, чем поток (относительно которого он вызван);

`isAlive()`

используется чтобы определить, исполняется ли поток; параметров не принимает; определяет, исполняется ли поток; возвращает логическое значение (`true` – alive, `false` – no);

`getName()`

используется для получения имени потока исполнения; параметров не принимает; возвращает имя потока исполнения; возвращает строку;
этот метод объявлен как `final`;

`setName()`

используется для установки имени потока исполнения; принимает параметр — строку (имя потока); устанавливает имя потока исполнения; возвращает ничтоже;
этот метод объявлен как `final`;

`getPriority()`

используется для получения приоритета потока; параметров не принимает; возвращает приоритет потока; возвращает целочисленное;
этот метод объявлен как `final`;

`setPriority()`

используется для установки приоритета потока; принимает параметр — `int` (значение приоритета); устанавливает значение приоритета потока; возвращает ничтоже;
этот метод объявлен как `final`;

`currentThread()`

используется для получения ссылки на текущий поток; параметров не принимает; возвращает ссылку на текущий поток (из которого этот метод вызван); возвращает объект;

Работа с потоками должна осуществляться в рамках устоявшейся практики программирования. Подробнее об этом — в других источниках.

Примечание

Также следует дополнить этот материал следующим.
Синхронизация, как и говорилось выше, задействует такой механизм, как мониторы. Мониторы задействуются неявно. Они задействуются, когда поток программы вызывает синхронизированный метод. И освобождаются, когда синхронизированный метод возвращает управление. Поэтому, в интересах синхронизации, для доступа к разделяемым ресурсам должны использоваться синхронизированные методы. Синхронизированные методы объявляются так:
```
synchronized RetType FuncName();
```

по входу в синхронизированный метод, он захватывает разделяемый ресурс. Никакие другие потоки (до возврата из этого метода) не могут вызвать ни этот метод, ни какие- либо другие синхронизированные методы объекта (своего для этого метода). Этот факт позволяет использовать этот объект как хранилище разделяемых ресурсов.

Необходимо помнить, что несинхронизированные методы этого обекта по прежнему доступны, и поэтому не должны использоваться для доступа к разделяемым ресурсам.
Если же для доступа к разделяемым ресурсам используется унаследованный код, который не доступен для модификации, то использование соответствующих методов этого кода следует заключить в блок `synchronized`:
`synchronized(ObjectName) { … }`

`ObjectName` – это ссылка на объект, содержащий разделяемые ресурсы (в данном случае предполагается, что это объект из унаследованного кода, методы которого и используются для доступа к разделяемым ресурсам).
Так же, при использовании многих потоков, может обнаружиться целесообразность применения методов `wait()`, `notify()`, `notifyAll()`, - эти методы класса `Object` открыты для наследования, и доступны всем пользовательским объектам. Эти методы позволяют регулировать взаимодействие потоков без формирования каких- либо объектов- сообщений.
При использовании методов (вышеопианных) необходимо помнть, что потоки все же могут пребвать в состоянии зависимости от результатов работы друг друга. Эта зависимость возможна, даже если потоки ипользуют ряд объектов, содержащих ресурсы (разделяемые между ними). Таким образом, возможна взаимная блокировка потоков. Эту возможность следует оценивать исходя из логики работы потоков.

Часть 2 (Прочие вопросы)

Файл с точкой входа в программу

Файл с точкой входа в программу должен иметь следующую структуру:

```
class ClassName0 { … }
 …
public class ClassNameN { … }
public class PublicClassName {
        public static void main (String args[]) {
         …
        }
}
```

точка входа — `main()` должна быть объявлена как `public`, так как она должна быть доступна не только из своего пакета, но и из самой среды исполнения, дабы быть из нее вызванной. Сам класс, в котором она объявляется тоже должен быть объявлен как `public` (так как и сам этот класс должен быть так же доступен). Функции `main()` должны быть присущи прописанные в образце аргументы — чтобы ей можно было передать параметры командной строки (и тогда эта функция должна предусматривать логику их обработки), этим не следует пренебрегать в интересах соблюдения соглашений об оформлении кода (при подробном их рассмотрении, они предусматривают эту деталь). Само тело этой функции — точка входа в программу, и программа начинает исполняться с этой функции. Функция должна иметь прописанное в образце имя (в интересах соблюдения соглашений об оформлении кода). Она не должна возвращать значений (так как нет смысла передавать их среде исполнения; дейтвительно, вся полезная работа программы должна выполняться ей самой). Функция — точка входа должна объявляться как `static`, так как загрузчик вызывает эту функцию еще до создания экземпляров каких- либо классов (проще говоря, это требование загрузчика).

Компиллируемому файлу следует присвоить имя класса, инкапсуллирующего точку входа в программу (это требование загрузчика). Расширением этого файла должно быть расширение `.java` (это требование компплятора).

Компилляция этого файла (из самой JDK) задается командой:

```
javac PublicClassName.java
```

результатом компилляции будет секвенция файлов. Каждый класс исходного кода (объявленный на уровне файла исходного кода) будет располагаться в отдельном файле, в том числе и класс с точкой входа. Каждый файл будет иметь расширение `.class`, и имя лежащего в нем класса.

Исполнение скомпиллированной программы (из самой JDK) задается командой:

```
java PublicClassName
```

результатом будет собственно исполнение этой программы.

Проверка предикатов

Проверка предикатов включается/ отключается при выполнении скомпиллированного кода. При выполнении из JRE проверка включается так:

```
-ea:PacketName…        // для целевого пакета
-ea:ClassName  // для целевого класса
```

и выключается аналогичными записями (разве что, используется ключ не `-ea`, а ключ `-da`).

Специфическая форма конструкторов

Есть следующая специфическая форма вызова перегруженных конструкторов:

```
ClassName( … ) {        // запись конструктора класса
this( … )               // вызов одного из ранее определенных конструкторов этого класса
 … ;                    // прочие операторы этого конструктора
}
```

в этой записи вызов уже существующего перегруженного конструктора должен быть первым. Эта запись позволяет создавать прочие конструкторы класса проще — по сути, используя и просто дополняя уже существующий конструктор. Стоит помнить о существовании этой формы записи конструкторов. При этом стоит помнить, что одновременное использование конструкторов `this()`, и `super()` не допускается — так как каждый из них непременно должен быть первым прописан в коде создаваемого конструктора класса.