1. Зарезервированные слова в Java

**Примитивы**  
  
1. byte  
2. short  
3. int  
4. long  
5. char  
6. float  
7. double  
8. boolean  
  
**Циклы и ветвления**  
  
9. if  
10. else  
11. switch  
12. case  
13. default  
14. while  
15. do  
16. break  
17. continue  
18. for  
  
**Исключения**  
  
19. try   
20. catch  
21. finally  
22. throw  
23. throws  
  
**Области видимости**  
  
24. Private Видимый только для класса   
25. Protected Видимый для пакета и всех подклассов   
26. public  
  
**Объявление \ Импорт**  
  
27. import  
28. package  
29. class  
30. interface  
31. Extends расширения реализации какого-то существующего класса. Создается новый класс на основе существующего, и этот новый класс расширяет (extends) возможности старого.  
32. implements предназначенное для реализации интерфейса   
33. static   
34. Final нельзя наследовать  
35. void  
36. abstract  
37. native / / сигнализирует о том, что метод реализован в платформо-зависимом коде, часто на языке С. Такой модификатор имеет, например, метод hashcode в Object.   
  
**Создание \ Возврат \ Вызов**  
  
38. new  
39. return  
40. this  
41. super обозначает **суперкласс**, т.е. класс, производным от которого является текущий класс  
  
**Многопоточность**  
42. synchronized  
43. volatile  
  
**Ключевые слова, которые есть (зарезервированы), но не используются в языке**  
  
44. const  
45. goto  
  
**Ключевые слова, которые не вошли ни в одну из групп**  
  
46. instanceof  
47. enum // перечисление  
48. assert // проверка данных  
49. transient // применяется к полям класса и означает, что данное поле не входит в персистентное состояние класса. Т.е. при сериализации данное поле не будет записываться. И, соответственно, при де-сериализации оно не будет восстанавливаться из потока байтов.

50. Strictfp

Ключевое слово языка программирования Java, используемое для  для гарантирования неизменности результатов операций с плавающей точкой на всех платформах. Используется для ограничения вычислений с плавающей точкой, чтобы гарантировать переносимость кода. strictfp используется как модификатор и введено  в JDK 1.2.

В предыдущих версиях JVM, вычисления с плавающей точкой были всегда ограничены, подразумевая что все значения, используемые в ходе вычислений с плавающей точкой были произведены по IEEE-стандарту, в float или double размерах. Это могло в результате иногда приводить к численному переполнению или потере значащих разрядов посреди вычисления, даже если в конечном результате должно было быть допустимое число. Начиная с версии 1.2 JVM, вычисления с плавающей точкой не требуют, чтобы все числа, используемые в вычислении,  ограничивали себя стандартной точностьютипов float или double.

Тем не менее, для некоторых приложений, программисту может потребоваться, чтобы на каждой платформе вычисления с плавающей точкой вели себя одинаково и с определенной точностью, даже если некоторые платформы могут производить вычисления с большей точностью. В этом случае, программист может использовать модификатор strictfp чтобы быть уверенным что вычисления будут выполняться как в прежних версиях — только с float и double.

Модификатор может быть использован в комбинации с классами, интерфейсами и методами. Константы и выражения, состоящие из них во время компиляции должны всегда использовать поведение ограничения чисел с плавающей-точкой. Другие выражения используют подобное ограничение, если они содержатся в методе, интерфейсе или классе который использует модификатор strictfp в объявлении.

1. **Зарезервированные имена методов Java**

clone

equals

finalize

getClass

hashCode

notify

notifyAll

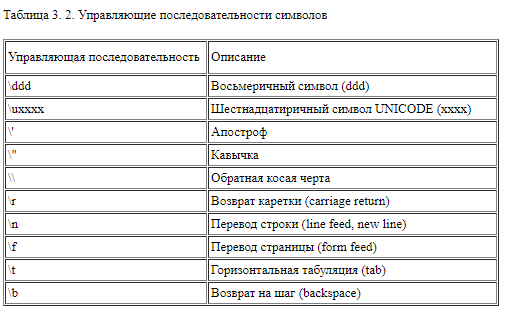
toString

wait

1. Идентификаторы  
     
   Имена переменных, классов, объектов, интерфейсов, методов называются идентификаторами. Названия идентификаторов выбираются по следующим правилам:  
   • они должны начинаться с буквы или символа подчеркивания;  
   • они могут содержать латинские буквы, символы подчеркивания или цифры без пробелов;  
   • названия идентификаторов не должны совпадать с ключевыми словами.
2. <http://developer.alexanderklimov.ru/android/java/literals.php>

**Литерал** — представление исходного кода как фиксированное значение. Оно представлено непосредственно в коде без каких-либо вычислений.

Литерал в Java можно назначить любой переменной из основного типа.

1. Управляющие символы  
   



**byte** — используется для передачи данных по сети, записи и чтения из файла. В математических операциях, как правило, не используется.  
https://midgardabc.wordpress.com/2015/04/23/primitives/  
  
7) Разделители

|  |  |  |
| --- | --- | --- |
| **Символ** | **Название** | **Назначение** |
| ( ) | Круглые скобки | Используются для передачи списков параметров в определениях и вызовах методов. Их применяют также для определения приоритета в выражениях, указания выражений в управляющих операторах и указания преобразования типов. |
| { } | Фигурные скобки | Используются для указания значений автоматически инициализируемых массивов. Их применяют также для определения блоков кода, классов, методов и локальных областей определения. |
| [ ] | Квадратные скобки | Используются для объявления типов массивов, а также при разыменовании значений массивов. |
| ; | Точка с запятой | Завершает операторы. |
| , | Запятая | Разделяет последовательные идентификаторы в объявлениях переменных. Этот символ-разделитель используют также для создания цепочек операторов внутри оператора for. |
| . | Точка | Используется для разделения имен пакетов от подпакетов и классов, а также для отделения переменной или метода от ссылочной переменной. |

## ****Правила именования переменных в java****

1. Имя переменной должно **начинаться с буквы (маленькой) и состоять из букв (Unicode) цифр и символа подчеркивания «\_»**. Технически возможно начать имя переменной также с «$» или «\_», однако это запрещено соглашением по оформлению кода в Java (Java Code Conventions). Кроме того, символ доллара «$», по соглашению, никогда не используется вообще. В соответствии с соглашением имя переменной должно начинаться именно с маленькой буквы (с заглавной буквы начинаются имена классов). Пробелы при именовании переменных не допускаются.
2. Имя переменной **не должно быть** [ключевым или зарезервированным словом языка Java](http://study-java.ru/spravochnik-java/klyuchevy-e-slova-yazy-ka-java/).
3. Имя переменной **чувствительно к регистру**.  newVariable и newvariable — разные имена.
4. При выборе имени переменных, следует **использовать полные слова** вместо загадочных аббревиатур. Это сделает ваш код более удобным для чтения и понимания. Во многих случаях это также сделает ваш код самодокументируемым.
5. Если выбранное вами имя переменной состоит только из одного слова — запишите его маленькими буквами. Если оно состоит из более чем одного слова, то **отделяйте каждое последующее слово в имени переменной заглавной буквой.** Например: superCounter, myDomesticAnimal
6. Если переменная сохраняет постоянное значение, то каждое слово следует писать заглавными буквами и отделять при помощи символа подчеркивания. Пример: static final int NUMBER\_OF\_HOURS\_IN\_A\_DAY = 24

ОПЕРАТОРЫ

<http://proglang.su/java/operators>

# Классы-оболочки в Java <https://javadevblog.com/klassy-obolochki-v-java-rukovodstvo-s-primerami.html>

# Массивы в Java

Массив — это конечная последовательность упорядоченных элементов одного типа, доступ к каждому элементу в которой осуществляется по его индексу.

Возможные следующие варианты объявления массива:

тип[] имя;  
тип  имя[];

int[] cats = {2, 5, 7, 8, 3, 0};

int[] cats = new int[10];

int[][][] b = new int[2][4][4];

int[][] a = {

{ 1, 2, 3 },

{ 4, 5, 6 }

}

<http://developer.alexanderklimov.ru/android/java/array.php>

# Строки и методы класса String

<http://proglang.su/java/strings>

<http://developer.alexanderklimov.ru/android/java/string.php>

IO:

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

String str = reader.readLine();

// Пример получения числа

Integer intSample = Integer.parseInt(reader.readLine());

System.out.print(str);

9

|  |  |
| --- | --- |
|  | //считываем  Scanner scanner=new Scanner(System.in);  int value=scanner.nextInt();  //выводим  System.out.println(value); |
| Scanner in = new Scanner(System.in);          System.out.print("Введите имя: "); |  |

**Javadoc** — стандартный генератор документации в HTML-формате из комментариев исходного кода.  
  
Для создания описания к элементу(поле, класс, метод) используются специальный комментарий, расположенный выше этого элемента:

/\*\* Описание \*/

Для документирования можно использовать дескрипторы, вот некоторые из них:  
**@author** — автор  
**@version** — версия  
**@since** — указывает с какой версии появился этот блок кода  
**@see** — ссылка на другое место в документации  
**@param** — передаваемый параметр методу  
**@return** — описание возвращаемого значения метода  
**@exception** и **@throws** — описание исключений  
**@deprecated** — документирование устаревших частей кода  
**{@link}** — создание ссылки, можно вставлять в любое место  
**{@value}** — описание значения переменной

# **Рекомендации к стилю кода**

<https://habrahabr.ru/post/112042/>

Целые константы можно записывать в трех системах счисления

<https://yougoup.wordpress.com/2015/11/05/java-%D1%86%D0%B5%D0%BB%D1%8B%D0%B5-%D0%BA%D0%BE%D0%BD%D1%81%D1%82%D0%B0%D0%BD%D1%82%D1%8B/>

Задайте две **константы**в классе

**final int**

**public final int**

**public static final int**

Объясните разницу между ними

private final static -> создать эту переменную только один раз. private final -> создать эту переменную для каждого объекта

Если бы индекс превысил максимально возможное для такого массива значение, то появилась бы ошибка времени исполнения. Проверка, не выходит ли индекс за допустимые пределы, происходит только во время исполнения программы, т.е. компилятор не пытается выявить эту ошибку даже в таких явных случаях, как:

int i[]=new int[5];

i[-2]=0; // ошибка! индекс не может

// быть отрицательным

Ошибка возникнет только на этапе выполнения программы.

Хотя при создании массива необходимо указывать его длину, это значение не входит в определение типа массива, важна лишь размерность. Таким образом, одна переменная может ссылаться на массивы разной длины:

int i[]=new int[5];

...

i=new int[7]; // переменная та же, длина

// массива другая

Однако для объекта массива длина обязательно должна указываться при создании и уже никак не может быть изменена.

**Java-аннотация** — в языке [Java](https://ru.wikipedia.org/wiki/Java) специальная форма синтаксических [метаданных](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5), которая может быть добавлена в [исходный код](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4).

Аннотации используются для анализа кода, [компиляции](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) или [выполнения](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80). Аннотируемы пакеты, классы, методы, переменные и параметры.

Выглядит как @ИмяАннотации, предваряющее определение переменной, параметра, метода, класса, пакета