Урок J-15. Форматирование чисел и текста в Java.

31 марта 2015 Мария (admin)

В предыдущих уроках для вывода информации на консоль мы использовали методы ***print()*** и ***println(),*** которые выводили строки или числа без какого-либо форматирования. Элементарное форматирование мы могли реализовать при помощи добавления к строкам дополнительных пробелов или других символов. В этом уроке мы познакомимся с методами, которые откроют нам новые возможности для **форматирования строк в Java**.

Краткое содержание урока:

Для начала немного мотивации. Рассмотрим пример, в котором рассчитывается и выводится на консоль таблица умножения:

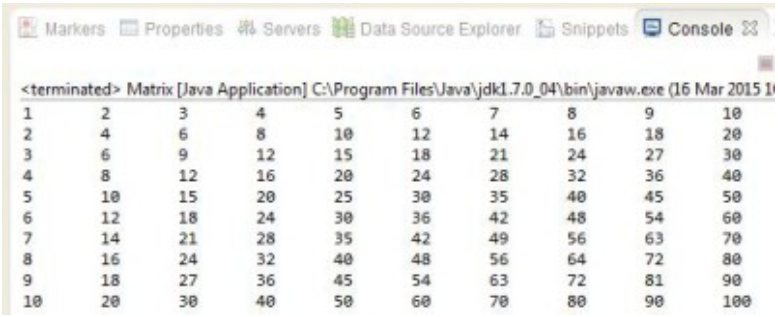
```
int[][] multiplyTab  = new int[10][10];

for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
        multiplyTab[i][j] = (i+1)*(j+1);
        //вывод ряда чисел разделенных знаком табуляции
        System.out.print(multiplyTab[i][j] + "\t");
   }
   System.out.println();
}
```

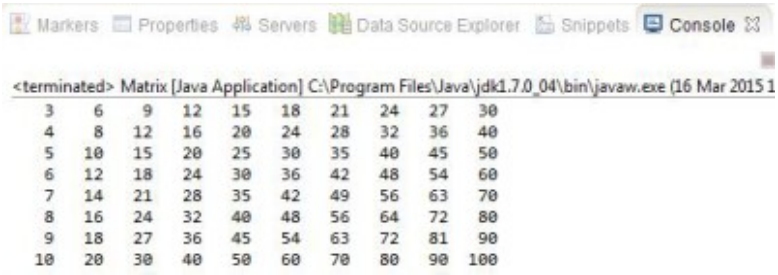
В данном случае для разделения чисел мы использовали знак табуляции, это дало следующий результат:

(http://study-java.ru/wp-content/uploads/2015/03/multiply_tab.jpg)

Таблица выглядит ровно, но она слишком широкая. Для того, чтобы сделать таблицу более компактной, будем использовать метод ***printf()***. Заменим в предыдущем коде строку

```
System.out.print(multiplyTab[i][j] + "\t");
```





на строку

```
System.out.printf("%4d", multiplyTab[i][j]);
```

Получим следующий результат:

[(http://study-java.ru/wp-content/uploads/2015/03/multiply_printf.jpg)](http://study-java.ru/wp-content/uploads/2015/03/multiply_printf.jpg)

Как мы видим, таблица стала компактнее. Более того, теперь мы можем уменьшать или увеличивать промежутки между числами по нашему желанию. Для этого нужно лишь заменить число `4` в выражении **«%4d»**.

Далее рассмотрим метод ***printf()*** и его возможности подробнее.

Использование printf для форматирования в Java

Метод ***printf()*** принадлежит классу **PrintStream**, который отвечает за вывод информации. Уже знакомые нам методы ***print()*** и ***println()*** также являются методами класса **PrintStream.**

Метод **printf** определен следующим образом:

```
printf(String format, Object... args)
```

Первый аргумент **format** это строка, определяющая шаблон, согласно которому будет происходить форматирование. Для ее записи существуют определенные правила.

В предыдущем примере формат был **«%4d»,** где **d** означает вывод десятичного целого числа, а **4** — означает то, что если количество знаков в числе меньше, чем 4, то оно будет спереди дополнено пробелами на недостающее (до 4-х) количество знаков (тем самым подвинуто вправо).

Для наглядности приведем еще один пример, который выводит столбиком несколько чисел.

```
System.out.printf("%6d%n%6d%n%6d%n%6d%n%6d%n%6d", 666666, 55555, 4444, 333, 22, 1);
```

На консоль будет выведено:

```
666666
 55555
  4444
   333
    22
     1
```

Здесь в строке форматирования мы использовали несколько раз **%6d%n,** где каждое **%6d** задает формат для одного из чисел, указанных далее в качестве аргументов. Первое **%6d** форматирует число **666666**, второе **%6d** форматирует **55555** и т.д. **%n** означает перевод строки. Поскольку ко всем числам было применено форматирование **%6d**, то числа, которые содержат менее 6 знаков подвинуты вправо на недостающее количество знаков и тем самым красиво выровнены.

Данный пример иллюстрирует также то, что метод **printf** может содержать несколько аргументов после строки с форматом. На что указывает **Object… args** в сигнатуре метода. Эти аргументы должны соответствовать ссылкам на них в строке формата. Например, если в строке формата стоит символ **d**, отвечающий за вывод целого десятичного числа, а далее в аргументе вы укажете строку, при компиляции произойдет ошибка преобразования формата java.util.IllegalFormatConversionException. Если аргументов больше, чем определено в формате, то лишние аргументы будут игнорироваться.

Общий вид инструкции форматирования следующий:

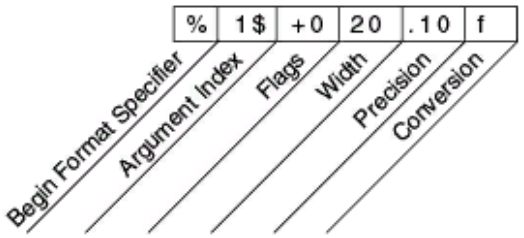
`%[argument_index$][flags][width][.precision]conversion`

- **%** — специальный символ, обозначающий начало инструкций форматирования.
- **[argument_index$]** — целое десятичное число, указывающее позицию аргумента в списке аргументов. Ссылка на первый аргумент `"1$"`, ссылка на второй аргумент `"2$"` и т.д. Не является обязательной частью инструкции. Если позиция не задана, то аргументы должны находиться в том же порядке, что и ссылки на них в строке форматирования.
- **[flags]** — специальные символы для форматирования. Например, флаг `"+"` означает, что числовое значение должно включать знак +, флаг `"-"` означает выравнивание результата по левому краю, флаг «,» устанавливает разделитель тысяч у целых чисел. Не является обязательной частью инструкции.
- **[width]** — положительное целое десятичное число, которое определяет минимальное количество символов, которые будут выведены. Не является обязательной частью инструкции.
- **[.precision]** — не отрицательное целое десятичное число с точкойперед ним. Обычно используется для ограничения количества символов. Специфика поведения зависит от вида преобразования. Не является обязательной частью инструкции.
- **conversion** — это символ, указывающий, как аргумент должен быть отформатирован. Например **d** для целых чисел, **s** для строк, **f** для чисел с плавающей точкой. Является обязательной частью инструкции.

Все возможные флаги и преобразования (**conversion**) указаны в **официальной документации (http://docs.oracle.com/javase/8/docs/api/java/util/Formatter.html)**. Здесь мы не ставим цель изучить их все, наша цель — научиться применять форматирование. Поэтому рассмотрим несколько примеров.

Пример 1. Наглядный пример инструкции форматирования в ее полном виде приведен на следующей картинке:

(http://study-java.ru/wp-content/uploads/2015/03/format.png)



Если мы применим формат с картинки к числу Pi

```
System.out.printf("%1$+020.10f", Math.PI);
```

Получим следующий результат на консоли:

```
+00000003,1415926536
```

Разберем данную инструкцию с конца:

- **f** — указывает на то, что выводим число с плавающей точкой.
- **.10** — выведенное число будет содержать 10 знаков после запятой.
- **20** — всего выведенное число будет иметь ширину в 20 символов
- **+0** — недостающие (до 20-ти) символы будут заполнены нулями, перед числом будет указан знак (+)
- **1$** — данный формат применяется к первому аргументу, который находится после строки форматирования. В данном примере это было указывать не обязательно.

Пример 2. Рассмотрим пример демонстрирующий вывод на консоль **до** знакомства с форматированием и **после**.

Без форматирования

```
Integer i=675;
double root;
root = Math.sqrt(i);
System.out.println("Корень числа " + i + " равен " + root );
```

На консоль будет выведено:

```
Корень числа 675 равен 25.98076211353316
```

В этом случае преобразование кода в строку автоматически генерируется компилятором Java. Этот способ плох тем, что при большом количестве переменных и текста для вывода, легко потерять контроль над результатами.

С форматированием

```
Integer i=675;
double root;
root = Math.sqrt(i);
System.out.printf("Корень числа %d равен %f", i, root );
```

Где **%d** отвечает за вывод значения переменной **i**, а **%f** за вывод значения переменной **root**. При этом мы уже не используем конкатенацию.

На консоль будет выведено:

```
Корень числа 675 равен 25,980762
```

Как мы видим, формат автоматически округляет число до 6 знаков после запятой. Однако, при форматировании мы можем устанавливать такое количество знаков после запятой, которое нам нужно, например:

```
System.out.printf("Корень числа %d равен %.2f", i, root );
```

Выведет число с двумя знаками после запятой.

Далее на примерах рассмотрим наиболее популярные правила форматирования.

Форматирование целых чисел

Вывод целого числа

```
System.out.printf("%d", 7845); // --> "7845"
```

Вывод целого числа с разделением тысяч

```
System.out.printf("%,d", 7845); // --> "7 845"
```

Число менее 7 знаков будет «подвинуто» вправо на недостающее количество знаков.

```
System.out.printf("%7d", 7845); // -->  "   7845"
```

Число менее 7 знаков будет дополнено нулями слева на недостающее количество знаков.

```
System.out.printf("%07d", 7845); // --> "0007845"
```

Число будет дополнено знаком + и, если оно менее 7 знаков, то будет дополнено нулями на недостающее количество знаков.

```
System.out.printf("%+07d", 7845); //--> "+007845"
```

Число будет выровнено по левому краю и, если оно менее 7 знаков, то будет дополнено пробелами справа на недостающее количество знаков.

```
System.out.printf("%-7d", 7845); // --> "7845   "
```

Разумеется вместо 7 можно использовать любое другое положительное целое число.

Форматирование чисел с плавающей точкой

Вывод числа **e**. Автоматическое округление до 6 знаков после запятой.

```
System.out.printf("%f", Math.E); // --> "2,718282"
```

Число менее 10 знаков будет «подвинуто» вправо на недостающее количество знаков.

```
System.out.printf("%10f", Math.E); // -->  "  2,718282"
```

Число менее 10 знаков будет дополнено нулями слева на недостающее количество знаков.

```
System.out.printf("%010f", Math.E); // --> "002,718282"
```

Число будет дополнено знаком + и, если оно менее 10 знаков, то будет дополнено нулями на недостающее количество знаков.

```
System.out.printf("%+010f", Math.E); // --> "+02,718282"
```

Число будет выведено с 15 знаками после запятой.

```
System.out.printf("%.15f", Math.E); // --> "2,718281828459045"
```

Число будет выведено с 3-мя знаками после запятой и, если оно менее 8 символов, то будет «подвинуто» вправо на недостающее количество знаков.

```
System.out.printf("%8.3f", Math.E); // --> "   2,718"
```

Число будет выровнено по левому краю, выведено с 3-мя знаками после запятой и, если оно менее 8 знаков, то будет дополнено пробелами справа на недостающее количество знаков.

```
System.out.printf("%-8.3f", Math.E); // --> "2,718   "
```

Форматирование строк

Вывод строки.

```
System.out.printf("%s%n", "Hello"); // --> "Hello"
```

Если строка содержит менее 10 символов, то «подвинуть » ее вправо на недостающее количество символов.

```
System.out.printf("%10s%n", "Hello"); // --> "     Hello"
```

Строка будет выровнена по левому краю и, если она менее 10 символов, то будет дополнена справа пробелами на недостающее количество символов.

```
System.out.printf("%-10s%n", "Hello"); // --> "Hello     "
```

Выведет первые 3 символа строки.

```
System.out.printf("%.3s%n", "Hello"); // --> "Hel"
```

Выведет первые 3 символа строки и подвинет их вправо на недостающее до 8 количество символов.

```
System.out.printf("%8.3s%n", "Hello"); // --> "     Hel"
```

Локализация

В разных странах некоторые записи принято производить по-разному. Например, в одних странах дробное число принято писать с точкой «3.68», а в других с запятой «3,68». Java нам позволяет соблюдать эти традиции. Метод **printf** имеет еще одну сигнатуру:

printf(Locale l, String format, Object… args)

Первым аргументом стоит **Locale l**, который и определяет локализацию**.** Для того, чтобы использовать локализацию необходимо вначале файла с вашим кодом импортировать библиотеку java.util.Locale.

```
import java.util.Locale;
```

Рассмотрим несколько примеров применения:

```
System.out.printf(Locale.ENGLISH,"%,d%n", 1000000 );// 1,000,000
System.out.printf(Locale.GERMAN,"%,d%n", 1000000 ); // 1.000.000
System.out.printf(Locale.FRANCE,"%,d%n", 1000000 ); // 1 000 000
```

В зависимости от указанной страны используются разные разделители для тысяч.

```
System.out.printf(Locale.ENGLISH,"%.2f%n", 9.87 ); //9.87
System.out.printf(Locale.FRANCE,"%.2f%n", 9.87 ); //9,87
```

В зависимости от указанной страны используются разные разделители у дробных чисел.

Использование String.format

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

Пример использования:

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

Это далеко не все возможности форматирования в Java. Существуют несколько классов, предназначенных для более сложного форматирования, но их оставим для дальнейших уроков.

Закончить урок хочется примером, в котором используются форматирование всех изученных в этом уроке типов данных: целых чисел, чисел с плавающей точкой и строк.

Пример: Таблица курсов валют

```
System.out.printf("%-5s%-11s%-25s%-11s%n","Код","За единиц","Валюты","Рублей РФ");
System.out.println("-------------------------------------------------");
System.out.printf("%-5s%-11d%-25s%-11.4f%n","AUD",1,"Австралийский доллар",44.9883);
System.out.printf("%-5s%-11d%-25s%-11.4f%n","GBP",1,"Фунт стерлингов",86.8429);
System.out.printf("%-5s%-11d%-25s%-11.4f%n","BYR",10000,"Белорусский рубль",39.7716);
System.out.printf("%-5s%-11d%-25s%-11.4f%n","DKK",10,"Датская крона",84.9192);
System.out.printf("%-5s%-11d%-25s%-11.4f%n","USD",1,"Доллар США",58.4643);
System.out.printf("%-5s%-11d%-25s%-11.4f%n","EUR",1,"Евро",63.3695);
System.out.printf("%-5s%-11d%-25s%-11.4f%n","KZT",100,"Казахский тенге",31.4654);
```

Результат вывода на консоль:

```
Код  За единиц  Валюты                    Рублей РФ
-------------------------------------------------
AUD  1          Австралийский доллар      44,9883
GBP  1          Фунт стерлингов           86,8429
BYR  10000      Белорусский рубль         39,7716
DKK  10         Датская крона             84,9192
USD  1          Доллар США                58,4643
EUR  1          Евро                      63,3695
KZT  100        Казахский тенге           31,4654
```

Полезные ссылки из официальной документации:

- **http://docs.oracle.com/javase/8/docs/api/java/util/Formatter.html (http://docs.oracle.com/javase/8/docs/api/java/util/Formatter.html)**
- **https://docs.oracle.com/javase/tutorial/java/data/numberformat.html (https://docs.oracle.com/javase/tutorial/java/data/numberformat.html)**
- **http://docs.oracle.com/javase/tutorial/essential/io/formatting.html (http://docs.oracle.com/javase/tutorial/essential/io/formatting.html)**

Категория: **Уроки Java (http://study-java.ru/category/uroki-java/)**
« **Урок J-14. Работа со строками в Java. Основные методы класса String. (http://study-java.ru/uroki-java/rabota-so-strokami-v-java-osnovnye-metody-klassa-string/)**
Комментарии и пинги к записи запрещены.

Комментариев к записи: 5

1. *Лев*:
   Апрель 5, 2015 в 3:30 пп (http://study-java.ru/uroki-java/formatirovanie-chisel-i-texta-v-java/#comment-47955)

   Толковые уроки! Самое то для начала погружения в Java! Спасибо!

   - *Мария (admin)*:
     Апрель 5, 2015 в 7:43 пп (http://study-java.ru/uroki-java/formatirovanie-chisel-i-texta-v-java/#comment-47956)

     спасибо!

2. *Владимир*:
   Апрель 27, 2015 в 5:45 пп (http://study-java.ru/uroki-java/formatirovanie-chisel-i-texta-v-java/#comment-47957)

Здравствуйте! У Вас пожалуй, самые лучшие уроки по Java на русском языке — все последовательно и изложено доступным языком. Очень волнует дальнейшее развитие цикла уроков по java для Android. Последняя запись была 8 месяцев назад=(((

- *Мария (admin)*:
  
  Апрель 27, 2015 в 11:36 пп (http://study-java.ru/uroki-java/formatirovanie-chisel-i-texta-v-java/#comment-47958)

  Спасибо! По андроиду готовится блок уроков. Этот проект является скорее хобби, авторы пишут в свободное всемя и когда есть вдохновение:) Поэтому все долго.

  - *Владимир*:
    Апрель 28, 2015 в 3:57 дп (http://study-java.ru/uroki-java/formatirovanie-chisel-i-texta-v-java/#comment-47959)

    Спасибо! Ждем с нетерпением)

(#) (#) (#) (#) (#) (#)

Новое на сайте

- Урок J-15. Форматирование чисел и текста в Java. (http://study-java.ru/uroki-java/formatirovanie-chisel-i-texta-v-java/)
- Урок J-14. Работа со строками в Java. Основные методы класса String. (http://study-java.ru/uroki-java/rabota-so-strokami-v-java-osnovnye-metody-klassa-string/)
- Урок J-13. Арифметические операторы и математика в Java. (http://study-java.ru/uroki-java/arifmeticheskie-operatory-i-matematika-v-java/)

Партнеры

(http://vk.com/club98704306)

(https://www.change.org/p/%D0%BC%D1%8B-%D0%BF%D1%80%D0%BE%D1%82%D0%B8%D0%B2-%D1%8F%D0%B4%D0%B5%D1%80%D0%BD%D0%BE%D0%B3%D0%BE-%D0%BC%D0%BE%D0%B3%D0%B8%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA%D0%B0)

- [+] Развернуть

УРОКИ JAVA

Мы в социальных сетях