
[Лента постов](#) [Все группы](#) [Мои группы](#)

Professor Hans Noodles

41 уровень



2 мая 2018 17:20



5910



5

Класс Scanner

Пост из группы **Java Developer**

3785 участников

[Присоединиться](#)

Привет!

Наше сегодняшнее занятие будет особенным!

До этого при решении задач и написании программ алгоритм был простым: мы пишем какой-то код, запускаем метод `main()`, программа делает что от нее требуется, и завершает свою работу.

Но теперь всё изменится! Сегодня мы научимся по-настоящему взаимодействовать с программой: мы научим её реагировать на наши действия!

Перед тем, как мы перейдем к изучению кода, скажи, тебе когда-нибудь приходилось встречаться с таким устройством как сканер?



Наверняка да. Изнутри строение сканера достаточно сложное, но суть его работы довольно проста: он считывает те данные, которые пользователь в него вводит (например, паспорт или страховой полис) и сохраняет считанную информацию в памяти (например, в виде изображения).

Так вот, сегодня ты создашь свой собственный сканер!

С документами он, конечно, не справится, а вот с текстовой информацией — вполне:)

Поехали!

Java Scanner Class

Первое и главное, с чем нам нужно познакомиться — класс `java.util.Scanner`.

Его функциональность очень проста. Он, словно настоящий сканер, считывает данные из источника, который ты для него укажешь (например, из строки, из файла, из консоли). Далее, он распознает эту информацию и обрабатывает нужным образом.

Приведем самый простой пример:

```
1 public class Main {  
2  
3     public static void main(String[] args) {  
4  
5         Scanner scanner = new Scanner("Люблю тебя, Петра творенье,\n" +  
6             "Люблю твой строгий, стройный вид,\n" +  
7             "Невы державное течение,\n" +  
8             "Береговой ее гранит");  
9         String s = scanner.nextLine();  
10        System.out.println(s);  
}
```

```
11     }  
12 }
```

Мы создали объект сканера, и указали для него источник данных (строку с текстом).

Метод `nextLine()` обращается к источнику данных (нашему тексту с четверостишием), находит там следующую строку, которую он еще не считывал (в нашем случае — первую) и возвращает ее. После чего мы выводим ее на консоль:

Вывод в консоль:

Люблю тебя, Петра творенье,

Мы можем использовать метод `nextLine()` несколько раз и вывести весь кусок поэмы:

```
1  public class Main {  
2  
3      public static void main(String[] args) {  
4  
5          Scanner scanner = new Scanner("Люблю тебя, Петра творенье,\n" +  
6              "Люблю твой строгий, стройный вид,\n" +  
7              "Невы державное течение,\n" +  
8              "Береговой ее гранит");  
9          String s = scanner.nextLine();  
10         System.out.println(s);  
11         s = scanner.nextLine();  
12         System.out.println(s);  
13         s = scanner.nextLine();  
14         System.out.println(s);  
15         s = scanner.nextLine();  
16         System.out.println(s);  
17     }  
18 }
```

Каждый раз наш сканер будет делать один шаг вперед и считывать следующую строку.

Результат работы программы — вывод в консоль:

*Люблю тебя, Петра творенье,
Люблю твой строгий, стройный вид,
Невы державное течение,
Береговой ее гранит*

Как мы уже говорили, источником данных для сканера может быть не только строка, но и, например, консоль.

Важная новость для нас: если раньше мы только выводили туда данные, то теперь будем вводить данные с клавиатуры!

Посмотрим, что еще умеет класс `Scanner`:

```
1 public class Main {
2
3     public static void main(String[] args) {
4
5         Scanner sc = new Scanner(System.in);
6         System.out.println("Введите число:");
7
8         int number = sc.nextInt();
9
10        System.out.println("Спасибо! Вы ввели число " + number);
11
12    }
13 }
```

Метод `nextInt()` считывает и возвращает введенное число. В нашей программе он используется для того, чтобы присвоить значение переменной `number`.

Это уже больше похоже на настоящий сканер! Программа просит пользователя ввести в строку любое число. После того, как пользователь это сделал, программа благодарит его, выводит на консоль итог своей работы и завершается.

Но у нас осталась одна серьезная проблема.

Пользователь может ошибиться и ввести что-то не то.

Вот пример, когда наша текущая программа перестанет работать:

```
1 public class Main {
2
3     public static void main(String[] args) {
4
5         Scanner sc = new Scanner(System.in);
6         System.out.println("Введите число:");
7
8         int number = sc.nextInt();
```

```
9
10         System.out.println("Спасибо! Вы ввели число " + number);
11
12     }
13 }
```

Попробуем ввести вместо числа строку “JavaRush”:

Вывод в консоль:

Введите число:

JavaRush

```
Exception in thread "main" java.util.InputMismatchException
    at java.util.Scanner.throwFor(Scanner.java:864)
    at java.util.Scanner.next(Scanner.java:1485)
    at java.util.Scanner.nextInt(Scanner.java:2117)
    at java.util.Scanner.nextInt(Scanner.java:2076)
    at Main.main(Main.java:10)
```

Process finished with exit code 1

Ой-ой, все плохо -_-

Во избежание подобных ситуаций нам нужно придумать способ проверки данных, которые вводит пользователь. Например, пользователь вводит что угодно, кроме числа, хорошо бы вывести в консоль предупреждение, что введенная информация не является числом, а если все в порядке — вывести текст подтверждения.

Но для этого нам надо фактически “заглянуть в будущее” — узнать, что там дальше в нашем потоке. Умеет ли `Scanner` это делать?

Еще как умеет! И для этого у него есть целая группа методов:

`hasNextInt()` — метод проверяет, является ли следующая порция введенных данных числом, или нет (возвращает, соответственно, true или false)

`hasNextLine()` — проверяет, является ли следующая порция данных строкой.

`hasNextByte()`, `hasNextShort()`, `hasNextLong()`, `hasNextFloat()`,
`hasNextDouble()` — все эти методы делают то же самое для остальных типов данных.

Попробуем изменить нашу программу для чтения числа:

```
1  public class Main {  
2  
3      public static void main(String[] args) {  
4  
5          Scanner sc = new Scanner(System.in);  
6          System.out.println("Введите число:");  
7  
8          if (sc.hasNextInt()) {  
9              int number = sc.nextInt();  
10             System.out.println("Спасибо! Вы ввели число " + number);  
11         } else {  
12             System.out.println("Извините, но это явно не число. Перезапустите программу.");  
13         }  
14     }  
15 }  
16 }
```

Теперь наша программа проверяет, является ли следующий введенный символ числом, или нет. И только в случае, если является, выводит подтверждение. Если же ввод не прошел проверку, программа это замечает и просит попробовать снова.

По сути, ты можешь общаться с объектом `Scanner` и **заранее** узнавать, какой тип данных тебе ожидать. “Эй, сканер, что там дальше будет? Число, строка, или еще что? Число? А какое — `int`, `short`, `long`?”

Такая гибкость дает тебе возможность выстраивать логику своей программы в зависимости от поведения пользователя.

Еще один важный метод, на который стоит обратить внимание — `useDelimiter()`.

В этот метод передается строка, которую вы хотите использовать в качестве разделителя.

Например, мы неожиданно увлеклись японской поэзией, и решили считать с помощью сканера несколько хокку великого поэта Мацуо Басе.

Даже если три разных стиха переданы нам одной корявой строкой, мы легко можем их разделить и красиво отформатировать:

```
1  public class Main {  
2  
3      public static void main(String[] args) {
```

```
4
5
6      Scanner scan = new Scanner("На голой ветке\n" +
7          "Ворон сидит одиноко.\n" +
8          "Осенний вечер.\n\n***" +
9          " \n" +
10         " \n" +
11         "В небе такая луна,\n" +
12         "Словно дерево спилено под корень:\n" +
13         "Белеет свежий срез.\n\n***" +
14         " \n" +
15         " \n" +
16         "Как разлилась река!\n" +
17         "Цапля бредет на коротких ножках,\n" +
18         "По колено в воде.");
19
20     scan.useDelimiter("\n/*//*/");
21
22     while (scan.hasNext()) {
23         System.out.println(scan.nextLine());
24     }
25
26     scan.close();
27 }
28
29 }
```

Мы используем в качестве разделителя шаблон `"\n/*//*/"` (перенос строки и три звездочки).

В результате в консоли у нас появится красивый вывод, совсем как в книгах:

*На голой ветке
Ворон сидит одиноко.
Осенний вечер.*

*В небе такая луна,
Словно дерево спилено под корень:
Белеет свежий срез.*

Как разлилась река!

*Цапля бредет на коротких ножках,
По колено в воде.*

В этом же примере есть и еще один метод, на который нужно обязательно

обратить внимание — `close()`. Как и любой объект, работающий с потоками ввода-вывода, сканер должен быть закрыт по завершении своей работы, чтобы больше не потреблять ресурсы нашего компьютера.

Никогда не забывай про метод `close()`!

```
1 public class Main {
2
3     public static void main(String[] args) {
4
5         Scanner sc = new Scanner(System.in);
6         System.out.println("Введите число:");
7
8         int number = sc.nextInt();
9
10        System.out.println("Спасибо! Вы ввели число " + number);
11
12        sc.close();//вот теперь мы сделали все правильно!
13
14    }
15 }
```

Вот и все! Как видишь, при всей своей полезности, класс `Scanner` достаточно прост в использовании! :)



Комментарии (5)

популярные новые старые

Vitaly Morochenets

Введите текст комментария

Макс 19 уровень, Киев

22 октября, 21:05



Кто понял что конкретно делает этот метод ↓, плиз объясните, чтобы понять.

```
1 useDelimiter()
```

Ответить

0

Alexander Korznikov 6 уровень, Киев

26 октября, 04:33



<http://javatutorialhq.com/java/util/scanner-class-tutorial/usedelimiter-string-pattern-method-example/>

Вот гораздо более понятные объяснения. В этой лекции очень плохой пример.

<http://qaru.site/questions/1692/how-do-i-use-a-delimiter-in-java-scanner>

Ответить

+1

Farid 11 уровень

5 сентября, 13:57

/ Комментарий удален */*

Ответить

0

Mr. Alex 16 уровень, Киев

9 сентября, 00:11



Все стримы должен быть закрыты по завершении своей работы, вручную или используя try с ресурсами.

Ответить

+3

Макс 19 уровень, Киев

22 октября, 21:03



Не все. Эти не надо закрывать:

- ```
1 ByteArrayInputStream
2 ByteArrayOutputStream
```

Ответить

+2