

StringReader, StringWriter

Java Collections
2 уровень, 3 лекция

ОТКРЫТА

— Привет, Амиго!

— Привет, Элли!

— Сегодня я хочу тебе рассказать про классы **StringReader** и **StringWriter**. Принципиально нового тут для тебя будет мало, но иногда эти классы бывают очень полезны. И, как минимум, я хочу, чтобы ты знал, что они есть.

Эти классы – это простейшие реализации абстрактных классов Reader и Writer. И практически аналоги FileReader и FileWriter. Но, в отличие от них, они работают не с данными в файле на диске, а со строкой (String) находящейся в памяти Java-машины.

— А зачем нужны такие классы?

— Иногда нужны. **StringReader** – это, фактически, переходник между классом **String** и **Reader**. А **StringWriter** – это строка, которая унаследована от **Writer**. М-да. Сама вижу, что объяснение не очень. Давай лучше для начала рассмотрим пару примеров.

Например, ты хочешь проверить, как работает твой метод, который должен вычитывать данные из переданного в него объекта Reader. Вот как это можно сделать:

Чтение из объекта reader:

```
1 public static void main (String[] args) throws Exception
2 {
3     String test = "Hi!\n My name is Richard\n I'm a photographer\n";
4
5     //это строчка – ключевая: мы «превратили» строку в Reader
6     StringReader reader = new StringReader(test);
7
8     executor(reader);
9 }
10
11 public static void executor(Reader reader) throws Exception
12 {
13     BufferedReader br = new BufferedReader(reader);
14     String line;
15     while ((line = br.readLine()) != null)
16     {
17         System.out.println(line);
18     }
19 }
```

— Т.е. мы просто взяли строку, обернули ее в StringReader и передали вместо объекта Reader? И из нее все будет читаться, как и надо?

— Ага. Гм. А в этом есть смысл. А теперь проверим, как работают методы StringWriter. Для этого усложним пример. Теперь он будет не просто читать строки, и выводить их на экран, а разворачивать их задом наперед и выводить в объект writer. Пример:

```
1 public static void main (String[] args) throws Exception
2 {
3     //эту строку должен будет прочитать Reader
4     String test = "Hi!\n My name is Richard\n I'm a photographer\n";
5     //заворачиваем строку в StringReader
6     StringReader reader = new StringReader(test);
7
8     //Создаем объект StringWriter
9     StringWriter writer = new StringWriter();
10
11    //переписываем строки из Reader во Writer, предварительно развернув их
12    executor(reader, writer);
13
14    //получаем текст, который был записан во Writer
15    String result = writer.toString();
16
17    //выведем полученный из Writer'а текст на экран
18    System.out.println("Результат: "+result);
19 }
20
21 public static void executor(Reader reader, Writer writer) throws Exception
22 {
23     BufferedReader br = new BufferedReader(reader);
24     String line;
25
26     //читаем строку из Reader'а
27
28     while ((line = br.readLine()) != null)
29     {
30         //разворачиваем строку задом наперед
31         StringBuilder sb = new StringBuilder(line);
32         String newLine = sb.reverse().toString();
33
34         //пишем строку в Writer
35         writer.write(newLine);
36     }
37 }
```

Мы создали объект **StringWriter**, внутри которого есть строка, в которой хранится все, что в этот **writer** пишут. А чтобы ее получить, надо всего лишь вызвать метод **toString()**.

— Гм. Как-то все слишком просто получается. Метод **executor** работает с объектами потокового ввода **reader** и **writer**, а в методе **main** мы работаем уже со строками.

Все действительно так просто?

— Ага. Чтобы преобразовать строку в **Reader** достаточно написать:

Создание Reader из String	
1	String s = "data";
2	Reader reader = new StringReader(s);

А преобразовать **StringWriter** к строке еще проще:

```
1  Writer writer = new StringWriter();
2  /*тут пишем кучу данных во writer */
3  String result = writer.toString();
```

— Отличные классы, как по мне. Спасибо за рассказ, Элли.

< Предыдущая лекция

 ×22

 +85 

Комментарии (42)


популярные

новые

старые

JavaCoder


Введите текст комментария


comrade_b Уровень 33, Амстердам, Нидерланды 16 июля, 15:27 

Как обычно оттачиваем навыки гугления и чтения описания класса и его методов в IDEA.

1. StringReader и StringWriter - работают с потоками char.
2. Под капотом используют StringBuffer, который является синхронизированным классом.
3. StringBuilder чище в коде, но несинхронизированный класс.
4. Если строку будут использовать разные потоки, то лучше использовать StringReader/Writer из-за StringBuffer.

Ответить

 0 

Максим Дудин Уровень 38, Калининград 21 декабря 2021, 13:53 

А если бы строка была не одна, то в цикле мы бы первую затёрли второй..... а если мы знаем что она одна, зачем цикл while? а если он нужен, тогда.... объявление StringBuilder вынести за цикл, а в цикле sb.apend();

Ответить


 0 

Anonymous #2530593 Уровень 51 31 января, 16:16 

У нас 3 строки line будет, потому что br.readline() будет возвращать часть строки до \n. Затираться ничего не будет, потому что мы внутри цикла развернутую строку line записываем в writer.

Ответить

 +8 

Andrew Shemetov Уровень 27, Ростов-на-Дону, Россия 12 октября 2021, 14:49 

Как так получается что мы объявляем в коде

1 StringWriter writer = **new** StringWriter();

, передаём writer в метод

1 **executor**(reader, writer);

в качестве параметра.

Как данные попадают обратно из метода ehexutor в main?

Почему в моём примере ниже так не работает?

1 **public static void main**(String[] args) **throws** IOException {

2 Integer a = 1;

3 **plus100**(a);

4 System.out.**println**(a); // выводится 1, а не 101



5 }

6 **static void plus100**(Integer a) {

7 a+=100;

8 }

Ответить

 +1 

НАЧАТЬ ОБУЧЕНИЕ

1) в main создается объект Integer(1). Ссылка на него записывается в переменную a.
2) при входе в plus100 в параметр a записывается ссылка на тот же объект Integer(1). (две ссылки на один объект)
3) Далее делаем a+=100. По факту это тоже, что a = a + 100.
3.1) a + 100 создает новый объект Integer(101)
3.2) a = записывает ссылку на него в a.
Итого:
-переменная a из main содержит ссылку на Integer(1) тут ничего не поменялось,
-параметр a из plus100 содержит ссылку на новый объект Integer(101).

Вышли из plus100 обратно в main:
- ну что ж, имеем нашу локальную переменную a с ссылкой на Integer(1), тут нет изменений.
- параметр a с ссылкой на Integer(101) из plus100 вышел за пределы видимости и больше не существует (стек откатился на предыдущий фрейм). Из-за отсутствия ссылок на Integer(101) данный объект будет уничтожен сборщиком мусора.

В примере из лекции объект Writer мутабелен. Когда делают writer.write(newLine); меняется его внутреннее состояние, а вот ссылка в параметре как раз не меняется. Поэтому после выхода мы наблюдаем измененное состояние.

Ответить

−

+3

+

Max Zap

Уровень 41

20 октября 2021, 21:50

...

А если вместо Integer сделать StringBuilder, ну или, например, AtomicInteger, то 101 выведется.

```
1 public static void main(String[] args) throws Exception {
2     StringBuilder sb = new StringBuilder("1");
3     plus100(sb);
4     System.out.println(sb); // выводится 101
5 }
6 static void plus100(StringBuilder sb) {
7     sb.append("01");
8 }
```

Ответить

−

+1

+

Ilyas DzhaliIov

Уровень 48

6 октября 2021, 21:31

...

Так и зачем этот класс?

Ответить

−

+5

+

Lycurgus

Уровень 37, Казахстан

3 сентября 2021, 12:43

...

Конечно, зачем писать для чего их используют. Легче дать примеры, с задачей, выполнимой и без этих классов.
Этот класс считывает символы по отдельности - read(). Он может также пропускать символы благодаря методу skip().

Ответить

−

+6

+

Rock133

Java Intern

23 апреля 2021, 17:31

...

А куда StringWriter сохраняет свои данные?

Ответить

−

0

+

Данил

Уровень 27

10 июня 2021, 10:51

...

в оперативной памяти хранятся

Ответить

−

+2

+

Алексей

Уровень 37, Казань

1 августа 2021, 21:04

...

судя по всему в heap

Ответить

−

+2

+

Vladimir “Rain_Senpai1995” Soldatenko

Уровень 35, Киев, Украина

16 января 2021, 22:38

...

Я так полагаю, что StringReader/StringWriter - еще один способ очень "хитро" работать со строками. Хотя бы намекали под конец лекции, когда применять эти обёртки...

Ответить

−

+2

+

Anonymous #2489173

Уровень 35

22 марта 2021, 08:29

...

Нет, всё, что мы получим это "Гм. Это имеет смысл".

Ответить

−

+11

+

Акижан

Кручу хвосты

3 января 2021, 21:01

/ Комментарий удален */*

Ответить

👍

0

👎

НАЧАТЬ ОБУЧЕНИЕ

Так а в каком месте он создает его? Он принимает ссылку на объект *Writer* и ее использует. Создание идет только в *main*.

Ответить

0

Акижан

Кручу хвосты

9 января 2021, 16:21

да, ща посмотрел свежим взглядом. Не знаю к чему я тогда прицепился.

Ответить

0

Will Fight

Уровень 27, Россия

16 декабря 2020, 17:34

Вот прямо на предыдущей задаче нужна была эта лекция!

Ответить

0

Павел Павличенко

Уровень 35, Львов

3 августа 2020, 16:28

что это? верните темную материю, сичяже

Ответить

+26

Niqita)_

Уровень 26, Chelyabinsk, Россия

26 августа 2020, 14:30

++

Ответить

0

Показать еще комментарии

ОБУЧЕНИЕ

- Курсы программирования
- Курс Java
- Помощь по задачам
- Подписки
- Задачи-игры

СООБЩЕСТВО

- Пользователи
- Статьи
- Форум
- Чат
- Истории успеха
- Активности

КОМПАНИЯ

- О нас
- Контакты
- Отзывы
- FAQ
- Поддержка



JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

Русский

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ

Доступно в

Google Play

Загрузите в

App Store

