



Потоки для ввода/вывода файлов

Java Core (/quests/QUEST_JAVA_CORE)
8 уровень (/quests/lectures/?quest=QUEST_JAVA_CORE&level=8), 2 лекция (/quests/lectures/questcore.level08.lecture02)

[ОТКРЫТА](#)

— А начнем мы с потоков для ввода/вывода файлов. Но обо всем по порядку.

Для чтений и записи файлов есть два класса: `FileInputStream` и `FileOutputStream`. Как ты уже, наверное, догадался, `FileInputStream` позволяет последовательно читать из файла байты, а `FileOutputStream` – записывать в файл байты. Вот какие методы есть у этих классов:

Метод

Что метод делает

```
1 FileInputStream(String fileName);
```

— это конструктор. Позволяет указать имя файла на диске, из которого созданный объект будет читать данные.

```
1 int read();
```

— метод читает один байт из файла и возвращает его как результат. Тип результата расширяется до `int`.

```
1 int available();
```

— метод возвращает количество непрочитанных (доступных) байт.

```
1 void close();
```

— метод «закрывает» поток, вызывается после окончания работы с потоком.
Объект выполняет служебные операции, связанные с закрытием файла на диске и т.д.
Из потока больше нельзя читать данные.

Давай ради интереса посчитаем сумму всех байт в файле на диске. Вот как будет выглядеть этот код:

Подсчет суммы всех байт файла на диске

```
1 public static void main(String[] args) throws Exception
2 {
3     //создаем объект FileInputStream, привязанный к файлу «c:/data.txt».
4     FileInputStream inputStream = new FileInputStream("c:/data.txt");
5     long sum = 0;
6
7     while (inputStream.available() > 0) //пока остались непрочитанные байты
8     {
9         int data = inputStream.read(); //прочитать очередной байт
10        sum += data; //добавить его к общей сумме
11    }
12    inputStream.close(); // закрываем поток
13
14    System.out.println(sum); //выводим сумму на экран.
15 }
```

— Мы уже раньше что-то подобное разбирали. А как устроен FileOutputStream?

— Ок. Вот, смотр

и:

Метод

Что метод делает

```
1 FileOutputStream (String fileName);
```

— это конструктор. Позволяет указать имя файла на диске, в который созданный объект будет писать данные.

```
1 void write(int data);
```

— метод записывает очередной байт, обрезая переменную data до одного байта.

```
1 void flush();
```

— часто данные для записи сначала собираются в большие блоки в памяти, а потом только пишутся на диск.

Команда flush требует немедленно записать всю несохраненную информацию на диск.

```
1 void close();
```

— метод «закрывает» поток, вызывается после окончания работы с потоком. Объект выполняет служебные операции, связанные с закрытием файла на диске и т.д.

В поток больше нельзя писать данные, flush при этом вызывается автоматически.

— И все?

— Да, тут фактически только один метод для записи – write, который записывает только один байт за раз. Но благодаря ему можно записать в файл сколько угодно информации.

Программирование – это процесс разбиения одной большой и сложной задачи на много маленьких. Тут происходит практически тот же процесс: чтение и запись больших данных маленькими порциями – по кусочкам – по одному байту.

Вот как можно скопировать файл на диске, пользуясь этими классами:

Копируем файл на диске

```
1 public static void main(String[] args) throws Exception
2 {
3     //Создаем поток-чтения-байт-из-файла
4     FileInputStream inputStream = new FileInputStream("c:/data.txt");
5     // Создаем поток-записи-байт-в-файл
6     FileOutputStream outputStream = new FileOutputStream("c:/result.txt");
7
8     while (inputStream.available() > 0) //пока есть еще непрочитанные байты
9     {
10         int data = inputStream.read(); // прочитать очередной байт в переменную data
11         outputStream.write(data); // и записать его во второй поток
12     }
13
14     inputStream.close(); //закрываем оба потока. Они больше не нужны.
15     outputStream.close();
16 }
```

— Спасибо, Риша. Наконец-то понял, как на самом деле работает этот код.

[< \(/quests/lectures/questcore.level08.lecture01\)](/quests/lectures/questcore.level08.lecture01)

[×18 > \(/quests/lectures/questcore.level08.lecture03\)](/quests/lectures/questcore.level08.lecture03)



0



0



0



0



0

+24

Комментарии (55)

популярные

новые

старые

Никита

Maxim Sivov 20 уровень, Taraz

30 марта, 19:11



Насколько понял:
Каждый символ в Юникод занимает 2 байта (16 бит), символ в ANSI - 1 байт.
С этим связано как метод `.read()` будет считывать побайтово файл

Ответить

0

Assanali 19 уровень

22 марта, 06:30



спасибо за `flush()`
в остальном читайте комментарии. и это не рекомендации.
это реально `MUST_read`.

Ответить

+1

Александр Немерицкий 20 уровень

26 февраля, 16:45



— Ок. Вот, смотр
и:
зачем тут перенос строки?

Ответить

0

Игорь Петров 26 уровень

2 февраля, 14:15



А как байты считаются из файла? без разбиение на строки? то есть подряд, и записываются в другой файл подряд или с переносом строки. Как определить где конец строки?

Ответить

0

rotarru 20 уровень, Минск

7 февраля, 09:19



`/r` – число 13 Это возврат каретки
`/n` – число 10 Это перенос строки

Ответить

+2

Игорь Петров 26 уровень

9 февраля, 11:35



ок! Спасибо! т.е я правильно понял мы проверяем если

```
1 if ( data.equals("/r"))
```

то пишем с новой строки , то есть

```
1 outputStream.write(data+"\n"); // и записать его во второй поток
2 else outputStream.write(data);
```

Ответить

0

Алексей 20 уровень

13 марта, 16:27



Нет, т.к. `read` читает именно байт, а не символ, то в данном случае в `data` будет байтовое представление символа `/r` или `/n` - т.е `data = 13` или `data = 10`.

для того чтобы получить именно символьное представление прочитанного байта, можно сделать вот так `System.out.println(char)data`). Но тут может быть подстава в случае многосимвольных кодировок. Поэтому лучше воспользоваться `BufferedReader`

Ответить

+1

Артем 22 уровень

15 января, 01:57



А ктонибудь знает какая кодировка используется по умолчанию в методе `read()`; ? С английскими буквами более менее понятно на букву Q выдает 81(UTF8, UTF16) А на русскую О выдает (-50). Я не в одной таблице не могу найти это соответствие. Может ктонибудь объяснить?

Ответить

0

Дмитрий 31 уровень, Москва

23 января, 15:03



После 20 уровня будут лекции и задачи с кодировкой и чтением/записью в файл

Ответить

0

George Matua 20 уровень

26 декабря 2017, 23:21

Советую прослушать лекции про IO от Головача - я только после их прослушивания понял эту статью полностью и написал код сам из примеров, понимая уже насколько он прост..

Качество видео любительское - отсюда, видимо, мало просмотров, зато материал доходчиво и понятно разжевывается, с примерами.

Начинать тут - [Java Core October: IO. Лекция #9 \(Часть 1\)](#)

Ответить

+12

Джонни 23 уровень

2 января, 00:05

Спасибо, выключило на неделю.

Ответить

0

Юрий Кузнецов 19 уровень

7 апреля, 13:23

да отличный материал у Головача! Советую всем просмотр 9 лекции и 10 как минимум. (это примерно часов 5-6)

Ответить

0

Джонни 23 уровень

25 декабря 2017, 12:41

Очень занятно.
Вбил я первый пример в IDE

```
1 public class Main {
2     public static void main(String[] args) throws Exception {
3         FileInputStream fileInputStream = new FileInputStream("Test1");
4         long sum = 0;
5         while (fileInputStream.available() > 0) {
6             int data = fileInputStream.read();
7             sum +=data;
8         }
9         System.out.println(sum);
10    }
11 }
```

Вывод: 2386

Затем я посчитал сумму по-другому:

```
1 public class Main {
2     public static void main(String[] args) throws Exception {
3         FileInputStream fileInputStream = new FileInputStream("Test1");
4         long sum = 0;
5         while (fileInputStream.available() > 0) {
6             fileInputStream.read();
7             sum ++;
8         }
9         System.out.println(sum);
10    }
11 }
```

Вывод: 27

На диске файл занимает 27 байт.

Либо я путаю сумму всех байт в файле на диске с размером файла на диске... Либо одно из двух. В чём дело?

Ответить

+1

Андрей Попонин 20 уровень, Санкт-Петербург

27 декабря 2017, 22:58

В первом случае вы складываете друг с другом именно байты, считанные с файла. Например 00000001 + 0001010 +... и так далее. А во втором случае вы просто считаете количество этих самых байт, которые складывали между собой в первом примере. И логично получается, что размер файла как раз равен количеству байт в файле, посчитанных вами. Если не прав, поправьте

Ответить

+2

Джонни 23 уровень

28 декабря 2017, 22:13

Спасибо. А зачем нужно складывать эти байты? В чём смысл и где это нужно?

Ответить

0

Андрей Попонин 20 уровень, Санкт-Петербург

30 декабря 2017, 01:31

Могу предположить, что для вычисления контрольной суммы файла, можете почитать про это в интернете. С помощью этого проверяется целостность переданного файла.

Ответить

+3

Джонни 23 уровень

1 января, 23:40

Хм. Очень интересно, спасибо!

Ответить

0

Starodubtsev Sergey 22 уровень, Минск

26 октября 2017, 19:27

Я или что-то упустил, но в край перестал понимать с этим байтом.
Пример: есть текстовый документ в нем слово "text". Метод read считывает "t" за два раза?
Или как? Объясните мне! MaYDaY

Ответить

+2

Наталка Чернобай 35 уровень, Киев

13 декабря 2017, 15:50

Извините, хотелось откомментировать Ваше сообщение, а получилось пожаловаться((. По Вашему вопросу: из <https://stackoverflow.com/questions/9648811/specific-difference-between-bufferedReader-and-FileReader> складывается мнение, что если файл в ASCII, то за раз. А если "из других алфавитов, поддерживаемых форматом UTF" то за три))

Ответить

+2

Джонни Мнемоник 24 уровень

28 августа 2017, 07:38

А я не понял, что это за не нужная избыточность обращения к дисковому пространству за каждым байтом? Если разве только для проверки целостности.

Ответить

0

максим захаренко 35 уровень, Москва

18 ноября 2017, 14:06

скоро расскажут про буфер

Ответить

0

Джонни Мнемоник 24 уровень

19 ноября 2017, 20:37

)

Ответить

0

Евгений Орлянкин 26 уровень, Харьков

27 июня 2017, 20:30

Спасибо всем кто дает полезные ссылки и пояснения(дополнения) в комментариях!!!

Ответить

+9

Загрузить еще

javarush.ru/, [G+ \(https://plus.google.com/114772402300089087607/\)](https://plus.google.com/114772402300089087607/), [Twitter \(https://twitter.com/javarush_ru\)](https://twitter.com/javarush_ru), [Facebook \(https://www.facebook.com/javarush\)](https://www.facebook.com/javarush)



Программистами не рождаются
© 2018