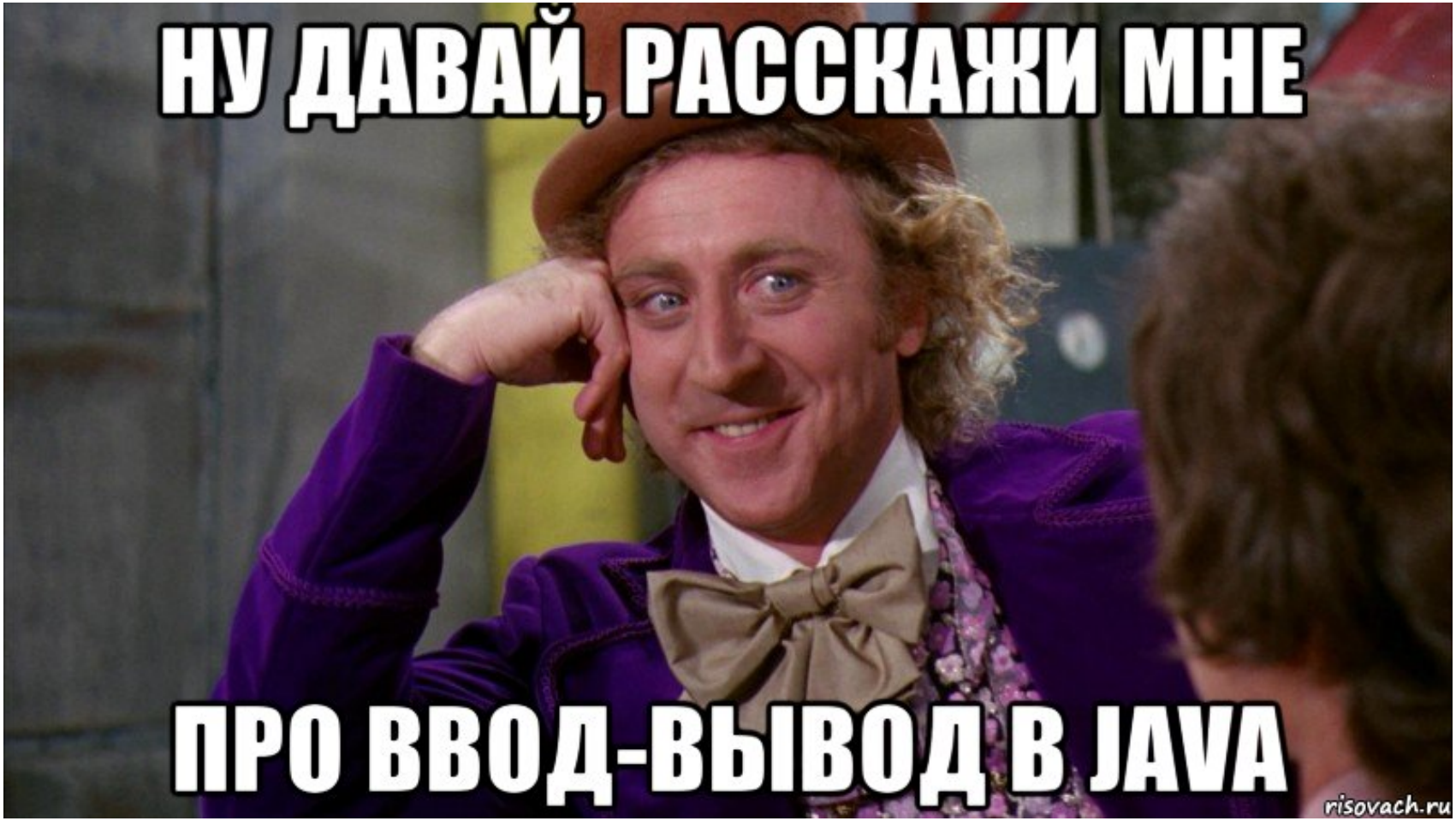
Стандартная библиотека Java: ввод/вывод

Алексей Владыкин

14 октября 2013

1. Доступ к файловой системе
2. Потоки байт
3. Потоки символов
4. NIO
5. Когда Java не хватает


НУ ДАВАЙ, РАССКАЖИ МНЕ
ПРО ВВОД-ВЫВОД В JAVA
risovach.ru

java.io.File

- API к файловой системе до Java 6 включительно
- Представляет файл *или директорию*
- Файл идентифицируется путем, специфичным для ОС/ФС
  - `\\server\share\`
  - `C:\Program Files\Java`
  - `/usr/bin/ls`

Работа с путями

```
File java = new File("/usr/bin/java");
java.isAbsolute();   // true
java.getPath();      // "/usr/bin/java"
java.getName();      // "java"
java.getParent();    // "/usr/bin"
```

- Файла по этому пути может не быть
- Поддерживаются абсолютные и относительные пути

Работа с путями

- `String getPath()`
- `String getAbsolutePath()`
- `String getCanonicalPath()`

Работа с файлами

```
File java = new File("/usr/bin/java");
java.exists();          // true
java.isFile();          // true
java.canRead();         // true
java.length();          // 1536
java.lastModified();// 1231914805000
```

- Если файла нет, `length()` и `lastModified()` возвращают 0

Работа с директориями

```
File usrbin = new File("/usr/bin");
usrbin.exists();       // true
usrbin.isDirectory(); // true
usrbin.list();         // array of Strings
usrbin.listFiles();    // array of Files
```

- Если директория не существует, `list()` вернет `null`
- Есть `list()` и `listFiles()` с фильтром

Модификация файловой системы

- `boolean createNewFile()`
- `boolean delete()`
- `boolean renameTo(File dest)`
- `boolean mkdir()`
  `boolean mkdirs()`

java.nio.file.*

- Новый API для работы с файловой системой (aka NIO.2)
- Добавлен в Java 7
- Покрывает всю функциональность `java.io.File`
- Добавляет то, чего не хватало:
  - Работа с расширенными атрибутами файлов (права доступа, владелец и т. д.)
  - Отслеживание изменений файловой системы

- Центральная сущность — `java.nio.file.Path` — представляет путь в файловой системе
- Доступ к файловой системе обеспечивает класс `java.nio.file.Files`

Работа с путями

```
Path java = Paths.get("/usr/bin/java");
java.isAbsolute();    // true
java.toString();      // /usr/bin/java
java.getFileName();   // java
java.getParent();     // /usr/bin
java.getNameCount();  // 3
java.getName(1);      // bin
```

- Основные операции с путями реализованы, не надо вручную возиться с разными разделителями на разных ОС/ФС

Работа с файлами

```
Path java = Paths.get("/usr/bin/java");
Files.exists(java);                 // true
Files.isRegularFile(java);          // true
Files.isReadable(java);             // true
Files.size(java);                   // 1536
Files.getLastModifiedTime(java)
        .toMillis();                // 1231914805000
```

- Если файла нет, size() и getLastModifiedTime() бросают NoSuchFileException

Работа с директориями

```
Path usrbin = Paths.get("/usr/bin");
Files.exists(usrbin);          // true
Files.isDirectory(usrbin);     // true

try (DirectoryStream<Path> dirStream =
         Files.newDirectoryStream(usrbin)) {
    for (Path child : dirStream) {
        System.out.println(child);
    }
}
```

- Если директория не существует, newDirectoryStream() бросит NoSuchFileException
- Есть вариант с фильтром

Модификация файловой системы

- `Path createFile(Path path)`
- `void delete(Path path)`
- `Path move(Path source, Path target)`
  `Path copy(Path source, Path target)`
- `Path createDirectory(Path dir)`
  `Path createDirectories(Path dir)`


SUPERCODER 2000
Air cooled coding keyboard for professional use.
Done
0
1

- Ввод данных
  `java.io.InputStream`
- Вывод данных
  `java.io.OutputStream`
- Производные классы для конкретных случаев
- Бросают `java.io.IOException` в случае ошибок

java.io.InputStream

- `int read()`
- `int read(byte b[])`
  `int read(byte b[], int off, int len)`
- `void close()`

java.io.InputStream — подклассы

- java.io.FileInputStream

```
new FileInputStream(new File("input.data"))
```

- java.io.ByteArrayInputStream

```
new ByteArrayInputStream(new byte[] {1, 2, 3})
```

- java.io.DataInputStream

```
new DataInputStream(anotherInputStream)
```

- java.util.zip.DeflaterInputStream

```
new DeflaterInputStream(anotherInputStream)
```

java.io.OutputStream

- `void write(int b)`
- `void write(byte b[])`
  `void write(byte b[], int off, int len)`
- `void flush()`
- `void close()`

java.io.OutputStream — подклассы

- java.io.FileOutputStream
  ```
  new FileOutputStream(new File("output.data"))
  ```
- java.io.ByteArrayOutputStream
  ```
  new ByteArrayOutputStream()
  ```
- java.io.DataOutputStream
  ```
  new DataOutputStream(anotherOutputStream)
  ```
- java.util.zip.DeflaterOutputStream
  ```
  new DeflaterInputStream(anotherOutputStream)
  ```

Копирование потока

```
byte[] buf = new byte[1024];
int bytesRead;
while ((bytesRead = inputStream.read(buf)) > 0) {
    outputStream.write(buf, 0, bytesRead);
}
```


НЕТ!
н е н у
ж н ы е
к н о
ненужная кнопка
Для больших букв
Для одной большой буквы
Ненужная кнопка
Флажок какой-то
Ненужная кнопка
самая длинная клавиша
Еще одна ненужная кнопка
Опять флажок
Ваще не понимаю
Лишняя кнопка
OK!
Самая большая кнопка
С этой кнопки нужно начинать писать мое имя

- Ввод данных
  `java.io.Reader`
- Вывод данных
  `java.io.Writer`
- Производные классы для конкретных случаев
- Бросают `java.io.IOException` в случае ошибок

java.io.Reader

- `int read()`
- `int read(char cbuf[])`
  `int read(char cbuf[], int off, int len)`
- `void close()`

java.io.Reader — подклассы

- java.io.InputStreamReader

  ```
  new InputStreamReader(inputStream, "UTF-8")
  ```

- java.io.CharArrayReader

  ```
  new CharArrayReader(new char[] {'a', 'b', 'c'})
  ```

- java.io.BufferedReader

  ```
  new BufferedReader(anotherReader)
  ```

  добавляет буферизацию и String readLine()

java.io.Writer

- `void write(int c)`
- `void write(char cbuf[])`
  `void write(char cbuf[], int off, int len)`
- `void flush()`
- `void close()`

java.io.Writer — подклассы

- java.io.OutputStreamWriter

  ```
  new OutputStreamWriter(outputStream, "UTF-8")
  ```

- java.io.CharArrayWriter

  ```
  new CharArrayWriter()
  ```

- java.io.BufferedWriter

  ```
  new BufferedWriter(anotherWriter)
  ```

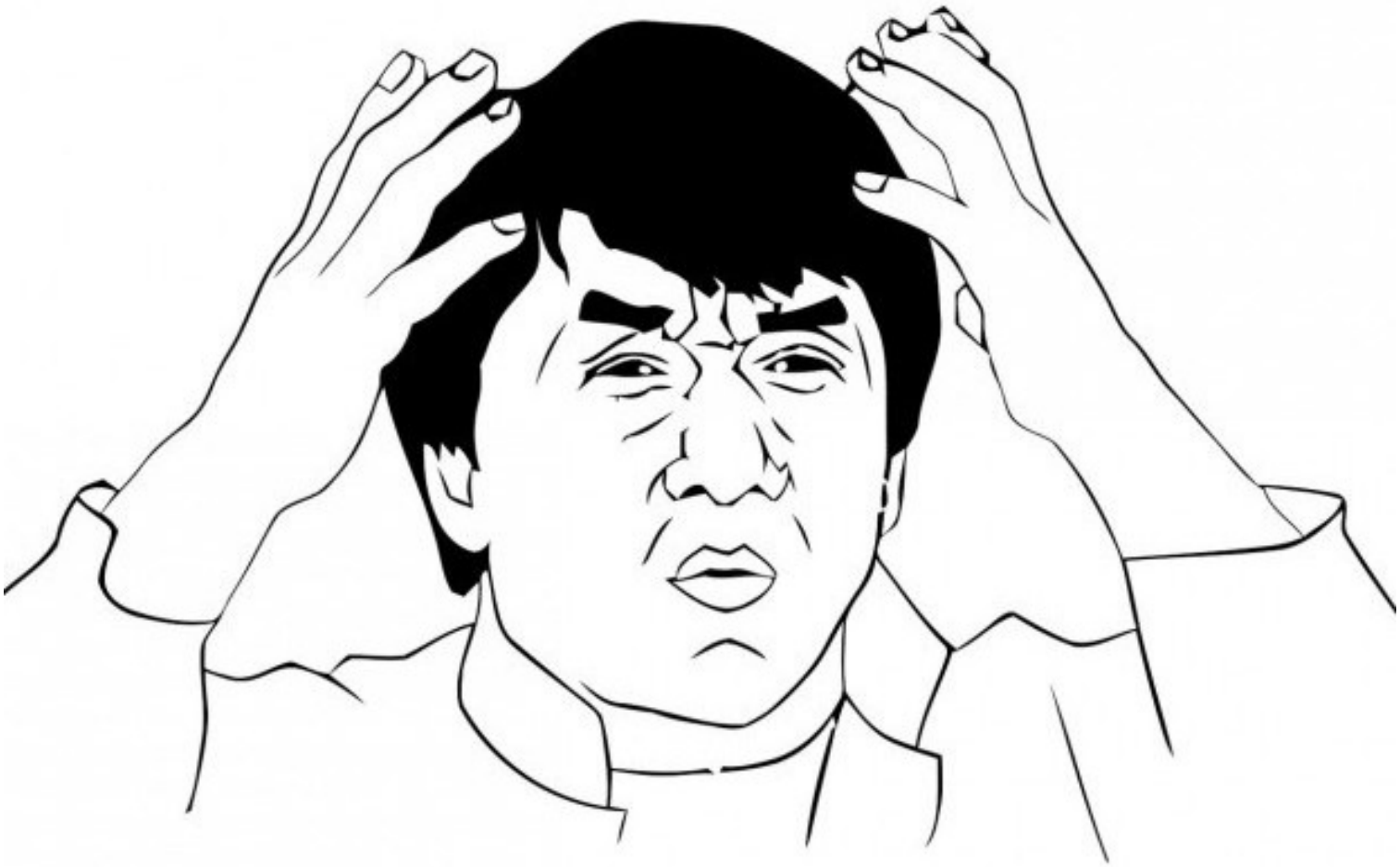
  добавляет буферизацию и `void newLine()`

Форматированный вывод

- `java.io.PrintStream` и `java.io.PrintWriter`
- Добавляют методы `print()`, `println()`, `printf()`
- Вместо исключения устанавливают флаг ошибки

Разбор текста на токены

- `java.io.StreamTokenizer`
  - умеет разбирать текст на «слова» и «числа»
- `java.util.Scanner`
  - добавлен в Java 5
  - умеет разбирать все примитивные типы, а также искать токены по произвольному регулярному выражению
  - поддерживает локали



Что такое NIO?

- Расширение `java.io`, добавлено в 1.4
- Уровень абстракции ближе к ОС
- Поддерживается неблокирующий ввод-вывод
- Более высокая производительность

Основные понятия NIO

- Buffer
  Аналог массива, но может представлять области памяти ОС
- Channel
  Аналог потока, поддерживает операции чтения/записи
- Selector
  Сервис для отслеживания событий в каналах

Пример NIO

```
try (FileChannel src =
        new FileInputStream(in).getChannel();
     FileChannel dst =
        new FileOutputStream(out).getChannel()) {

    src.transferTo(0, src.size(), dst);
}
```


Камасутра компа
DEMOTIVATORS.RU

Запуск внешних процессов

```
ProcessBuilder processBuilder =
    new ProcessBuilder("cmd", "/c", "dir");
Process process = processBuilder.start();

InpuStream inputStream = process.getInputStream();
BufferedReader reader = new BufferedReader(
    new InputStreamReader(inputStream, "cp866"));

String line;
while ((line = reader.readLine()) != null) {
    System.out.println(line);
}

int exitCode = process.waitFor();
```

JNI

- Java Native Interface
- Возможность вызова нативного кода из JVM
- Теряется переносимость кода (либо надо подкладывать нативные библиотеки под все поддерживаемые платформы)

Что сегодня узнали

- Как ходить по файловой системе, получать списки файлов и директорий, читать их атрибуты
- Как читать и писать двоичные данные при помощи потоков байт
- Как читать и писать текстовые данные при помощи потоков символов