

I/O в Java

Юрий Литвинов
yurii.litvinov@gmail.com

23.01.2019г

I/O в Java

- ▶ Пакет java.io
- ▶ Интерфейсы
 - ▶ InputStream, OutputStream — работа с байтами
 - ▶ Reader, Writer — работа с символами
- ▶ Реализации
 - ▶ File* — работа с файлами
 - ▶ Buffered* — оптимизация с помощью буферизации
 - ▶ Data* — работа с данными
 - ▶ ByteArray* — потоки в оперативной памяти
- ▶ Исключения
 - ▶ IOException — общий предок
 - ▶ FileNotFoundException, EOFException и т.д.
- ▶ Пример:
`new BufferedReader(new FileReader("path/to/file"));`
- ▶ Потоки надо закрывать (close(), try-with-resource)

Потоки ввода, основные операции

- ▶ **int read()** — чтение элемента
- ▶ **read(T[] v), read(T[] v, off, len)** — чтение элементов в массив
- ▶ **skip(n)** — пропуск n элементов
- ▶ **available()** — сколько элементов доступно
- ▶ **mark(limit)** — пометка текущей позиции
- ▶ **reset()** — возврат к помеченной позиции
- ▶ **close()** — закрытие потока
 - ▶ Интерфейс `Closeable` extends `AutoCloseable`

Потоки вывода, основные операции

- ▶ `write(int v)` — запись элемента
- ▶ `write(T[] v)` — запись массива элементов
- ▶ `write(T[] v, off, len)` — запись части массива
- ▶ `flush()` — запись буфера
 - ▶ Интерфейс `Flushable`
- ▶ `close()` — закрытие потока

Пример, блочное копирование

```
void copy(InputStream is, OutputStream os)
    throws IOException
{
    var b = new byte[1024];
    int c = 0;
    while ((c = is.read(b)) > 0) {
        os.write(b, 0, c);
    }
}
```

Пример, чтение из файла

```
FileInputStream in = null;
```

```
try {  
    in = new FileInputStream("test.txt");  
    int c;  
  
    while ((c = in.read()) != -1) {  
        doSomething(c);  
    }  
} finally {  
    if (in != null) {  
        in.close();  
    }  
}
```

Или (try-with-resource)

```
try (var in = new FileInputStream("test.txt")) {  
    int c;  
  
    while ((c = in.read()) != -1) {  
        doSomething(c);  
    }  
}
```

- ▶ Работает для классов, реализующих `java.lang.AutoCloseable`

Исключения

- ▶ `IOException`
 - ▶ Корень иерархии исключений ввода-вывода
 - ▶ Бросается всеми операциями ввода/вывода
- ▶ `EOFException` — достигнут конец потока
- ▶ `FileNotFoundException` — файл не найден
- ▶ `UnsupportedEncodingException` — неизвестная кодировка

Преобразование потоков

- ▶ При чтении возможно преобразование байтового потока в символьный, с указанием кодировки, класс `InputStreamReader`
 - ▶ `InputStreamReader(InputStream, encoding?)`
- ▶ Наоборот, из символьного в байтовый: `OutputStreamWriter`
 - ▶ `OutputStreamWriter(OutputStream, encoding?)`

Пример, перекодирование файла

```
Reader reader = new InputStreamReader(  
    new FileInputStream("input.txt"), "Cp1251");  
Writer writer = new OutputStreamWriter(  
    new FileOutputStream("output.txt"), "Cp866");  
int c = 0;  
while ((c = reader.read()) >= 0) {  
    writer.write(c);  
}  
  
reader.close();  
writer.close();
```

Ещё пример, буферизующий поток

```
static String readFirstLineFromFile(String path)
    throws IOException {
    try (var br = new BufferedReader(new FileReader(path))) {
        return br.readLine();
    }
}
```

Потоки для работы в памяти

- ▶ `ByteArrayInputStream` — чтение из массива байт
- ▶ `CharArrayReader` — чтение из массива символов
- ▶ `StringReader` — чтение из строки
- ▶ `ByteArrayOutputStream` — запись в массив байт (`toByteArray()`)
- ▶ `CharArrayWriter` — запись в массив символов (`toString()`, `toCharArray()`)
- ▶ `StringWriter` – запись в `StringBuffer` (`toString()`, `toStringBuffer()`)

Scanner

```
try (var s = new Scanner(  
    new BufferedReader(new FileReader("test.txt")))) {  
    while (s.hasNextDouble()) {  
        System.out.println(s.nextDouble());  
    }  
}
```

Работа с данными

- ▶ Интерфейс `DataOutput`, класс `DataOutputStream`
 - ▶ `writeT(T)` — запись примитивных типов
 - ▶ `writeUnsignedByte()` / `writeUnsignedShort()` — запись беззнаковых целых
 - ▶ `writeUTF()` — запись строки в кодировке UTF-8
- ▶ Интерфейс `DataInput`, класс `DataInputStream`
 - ▶ `T readT()` — чтение примитивных типов
 - ▶ `readUnsignedByte()` / `readUnsignedShort()` — чтение беззнаковых целых
 - ▶ `readUTF()` — чтение строки в кодировке UTF-8

Задача на остаток пары

- ▶ Реализовать консольную программу, работающую в трёх режимах:
 - ▶ архивирует каталог (путь до него – первый аргумент командной строки) с заданным именем (второй аргумент)
 - ▶ разархивирует заданный архив (первый аргумент) в заданное место (второй аргумент)
 - ▶ распечатывает содержимое архива (первый аргумент): имена файлов, размер сжатый/несжатый, метод архивации и т.п.
- ▶ Могут быть полезны классы `ZipFile`, `ZipEntry`, `ZipOutputStream`, `ZipInputStream`