# Nio мой Nio

Практика 40 - Айрат Хасьянов

# NIO

- Появился в Java 1.4
- Создан быть быстрым
- Старые классы тоже переписаны
- Ускорение за счет каналов и буферов
- Мы используем буфер; буфер взаимодействует с каналом.
- **FileInputStream**, **FileOutputStream** и **RandomAccessFile** были изменены, чтобы возвращать канал **FileChannel**
- **java.nio.channels.Channels** позволяет получать объекты **Reader** и **Writer** для каналов.

# Получение каналов из потоков

**//OutputStream**
FileChannel fc=new FileOutputStream("neo.txt").getChannel();
fc.write(ByteBuffer.wrap("Knock, knock...\n".getBytes()));
fc.close();

**//RandomAccess**
fc = new RandomAccessFile("neo.txt", "rw").getChannel();
fc.position(fc.size());
fc.write(ByteBuffer.wrap("The Matrix has you\n".getBytes()));
fc.close();

**//InputStream fc** = new FileInputStream("neo.txt").getChannel();
ByteBuffer buf = ByteBuffer.allocate(512);
fc.read(buf);
buf.flip();
while(buf.hasRemaining()){
System.out.print((char)buf.get()); }

# Копирование файлов

```
FileChannel in=new FileInputStream("neo.txt").getChannel(),
            out=new FileOutputStream("matrix.txt").getChannel();
ByteBuffer buffer = ByteBuffer.allocate(2048);
while(in.read(buffer) != -1) {
    buffer.flip(); // Готовимся к записи
    out.write(buffer);
    buffer.clear(); // Готовимся к чтению
}
```

# Присоединение каналов

```
FileChannel in=new FileInputStream("neo.txt").getChannel(),
out=new FileOutputStream("reality.txt").getChannel();
in.transferTo(0, in.size(), out);
//или так:
//out.transferFrom(in, 0, in.size());
```

# Задачи

1. Напечатайте содержимое файла, используя присоединение каналов
2. Запишите в файл набор случайных чисел, используя Stream API

# Кодировки, кодировки

***Буфер содержит байты***, следовательно для превращения их в символы ***кодируем*** их либо при ***помещении***, либо при ***извлечении*** из буфера.

```
FileChannel fc = new FileOutputStream("java.txt").getChannel();
fc.write(ByteBuffer.wrap(TEXT.getBytes()));
fc.close();

fc = new FileInputStream("java.txt").getChannel();
ByteBuffer buf = ByteBuffer.allocate(CAPACITY);
fc.read(buf);
buf.flip();
//Декодирование в кодировку по-умолчанию
String encoding = System.getProperty("file.encoding");
System.out.println("["+encoding+"] "+Charset.forName(encoding).decode(buf));
```

[UTF-8] Я помню чудное мгновенье,
передом

# Можно проще? CharBuffer!

```
FileChannel fc = new FileOutputStream("anotherjava.txt").getChannel();
ByteBuffer buf = ByteBuffer.allocate(2*TEXT.length());
buf.asCharBuffer().put(TEXT);
fc.write(buf);
fc.close(); //Читаем записанное
fc = new FileInputStream("anotherjava.txt").getChannel();
buf.clear();
fc.read(buf);
buf.flip();
System.out.println(buf.asCharBuffer());
```

Я помню чудное мгновенье,
передомной явилась ты
- как мимолетное виденье,
как гений чистой красоты...

# Задачи 2

1. Прочитайте из файла исходный код. Используя Stream API, запишите код в новый файл используя только заглавные буквы.

# Примитивы, опять…

```
    ByteBuffer bb = ByteBuffer.allocate(CAPACITY);
// Выделение памяти обнуляет буфер. Запишем массив символов:

    bb.asCharBuffer().put("Howdy!");
    char c;

    while((c = bb.getChar()) != 0)
        System.out.print(c + " ");

    System.out.println();
    bb.rewind();

// Запишем и прочтем short:
    bb.asShortBuffer().put((short)471142);
    System.out.println(bb.getShort());
    bb.rewind();
```

```
H o w d y !
12390
```

# Изменение представления буфера

```
ByteBuffer bb = ByteBuffer.allocate(CAPACITY);
IntBuffer ib = bb.asIntBuffer(); // Сохраним в буфер массив целых
ib.put(new int[]{ 11, 42, 47, 99, 143, 811, 1016 });

// Буфер можно индексировать:
System.out.println(ib.get(3)+"\n");

ib.put(3, 1811);

// Готовимся к чтению из буфера
ib.flip();
while(ib.hasRemaining()) {
      int i = ib.get();
      System.out.println(i);
}
```

```
99

11
42
47
1811
143
811
1016
```

# Задачи 3

1. Заполните буфер примитивами используя Stream API,
2. измените представление буфера,
3. извлеките примитивы из буфера

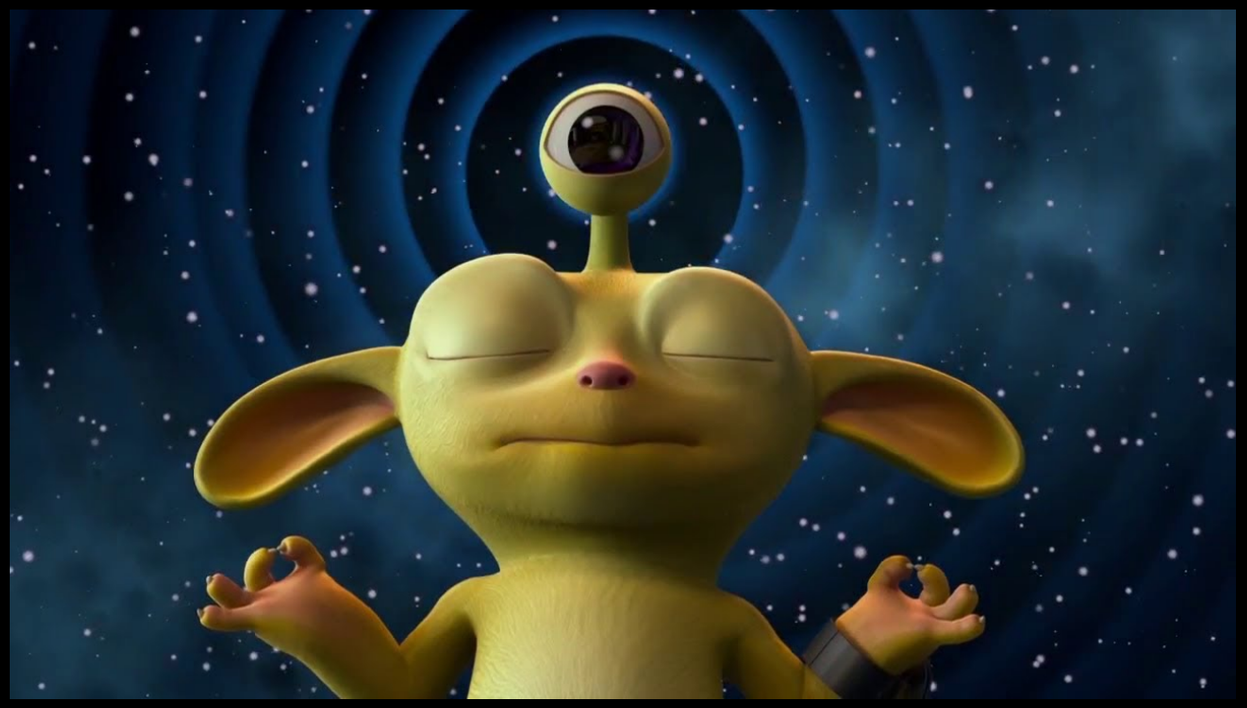
# Отображаемый в память файл

Если файл слишком велик (до 2 Гб),
для прямого размещения в памяти,
можно хотя бы представить, как будто
он поместился в памяти целиком :)

```
final int length = 0x8FFFFFF; // 128 MB
MappedByteBuffer out = new RandomAccessFile("test.dat","rw")
    .getChannel().map(FileChannel.MapMode.READ_WRITE, 0, length);
for (int i = 0; i < length; i++)
    out.put((byte) 'x');
System.out.println("Finished writing");
for (int i = length / 2; i < length / 2 + 6; i++)
    System.out.println((char) out.get(i));
```

```
X
X
X
X
X
X
```

# Задачи 4



1. Давайте заполним 256мб полной ерунды (кириллицей) в файл oracul.txt, а потом, посмотрим, что там с 128-138 байты - гадание по-программистски. Используем Strream API.

# Блокировка файлов

- Блокировка файлов позволяет синхронизировать доступ к файлу.
- Блокировка видна другим процессам операционной системы!
- Блокировать файл целиком можно при помощи объекта **FileLock**, который возвращают методы
  - **FileChannel.tryLock()**;
  - **FileChannel.lock()**;
- Сетевые каналы **SocketChannel, DatagramChannel** и **ServerChannel** не нуждаются в блокировании, т.к. доступны только в пределах одного потока.

# Блокировка (прод.)

- Отличие методов tryLock и lock:
  - **tryLock()** в случае невозможности блокировки просто возвращает управление;
  - **lock()** приостанавливает поток до тех пор, пока блокировка не станет возможной.
- Можно заблокировать часть файла: **tryLock(long position, long size, boolean shared)**; Блокируется участок файла размером **size**, начиная с позиции **position**. Параметр **shared** указывает, будет ли блокировка совместной.

- Поддержка блокировок с эксклюзивным или разделяемым доступом должна поддерживаться операционной системой.

# Пример блокировки

```
FileOutputStream fos= new FileOutputStream("java.txt");
FileLock fl = fos.getChannel().tryLock();
if(fl != null) {
    System.out.println("Locked File");
    TimeUnit.MILLISECONDS.sleep(100);
    fl.release();
    System.out.println("Released Lock");
}
fos.close();
```

# Вопросы

- Какой класс обеспечивает произвольный доступ к файлу?
- Что такое nio, зачем его стоило придумать?
- Что такое каналы и буферы?
- Какие классы используются для работы с архивами?
- Поддерживает ли Java многофайловые архивы?
- Поддерживает ли Java работу с JAR архивами?
- Может ли поток, выполняемый вне JVM получить доступ к заблокированному файлу?
- Что такое эксклюзивный, совместный доступ?
- Как заблокировать сетевое соединение?

# Домашнее задание

1. Прочитайте из файла исходный код. Используя Stream API, удалите все комментарии и аннотации.

2. Напишите многопоточное приложение, которое читает наш большой файл, и преобразует его в заглавные буквы. Используйте блокировки.

3. Используя возможности nio заархивировать полученный файл.