Урок 10.
Потоки ввода-вывода,
работа с файловой системой,
NIO

Гарбузов Николай
Donriver Inc.

DonRiver

Agenda

- Понятие потока ввода-вывода
- Байтовые потоки ввода-вывода
- Символьные потоки ввода-вывода
- Сериализация
- Работа с файловой системой Java NIO
- try-with-resources

Понятие потока ввода-вывода

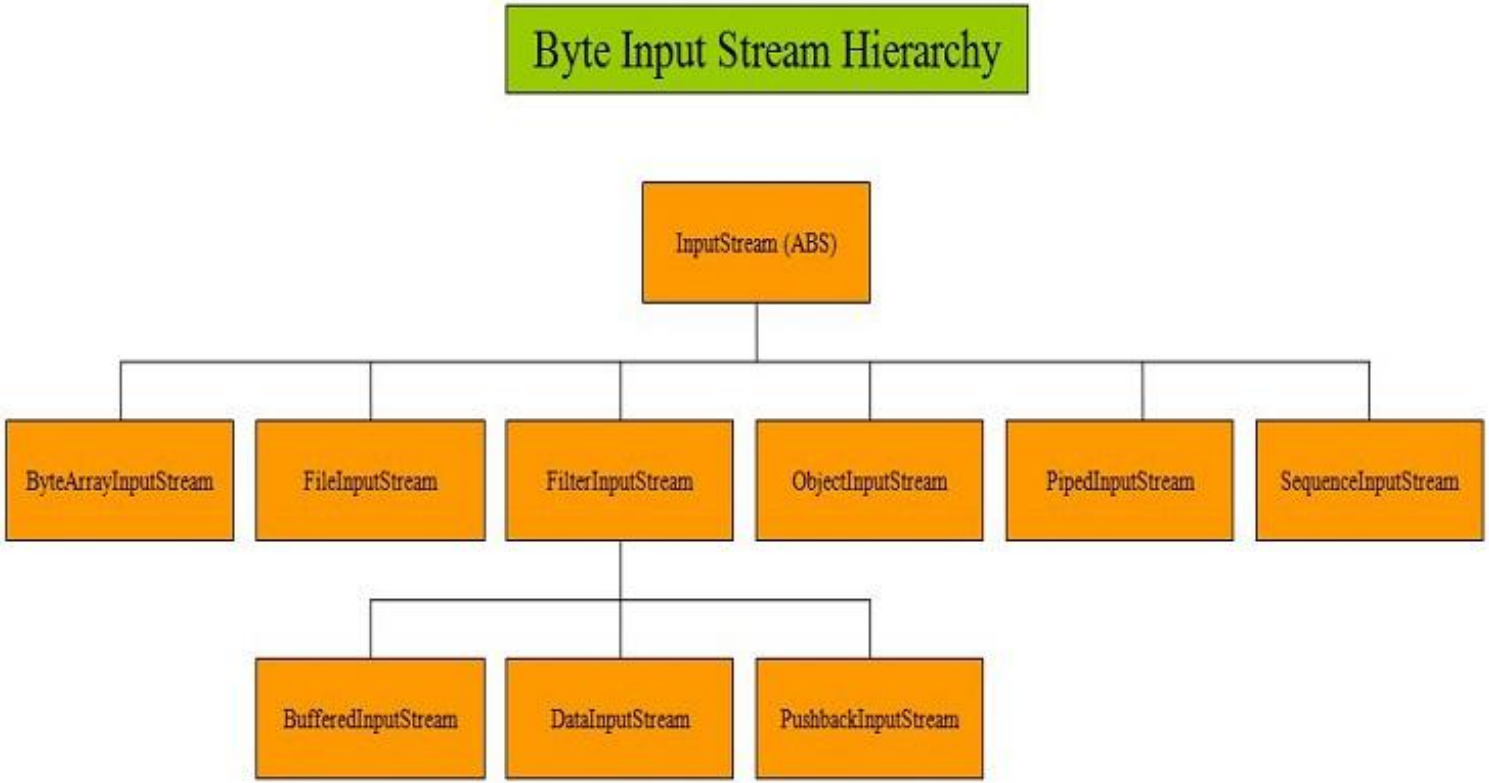
- **Поток ввода-вывода** - поставляет и потребляет информацию
  - “сырые данные” (байты)
  - строки
  - типизированные данные

Понятие потока ввода-вывода

- **Поток ввода-вывода** - поставляет и потребляет информацию
  - “сырые данные” (байты)
  - строки
  - типизированные данные

- **Поток ввода-вывода** - абстракция над физическими устройствами ввода-вывода!
  - файлы
  - сокеты
  - com & usb порты

Байтовые потоки ввода-вывода



Байтовые потоки ввода-вывода

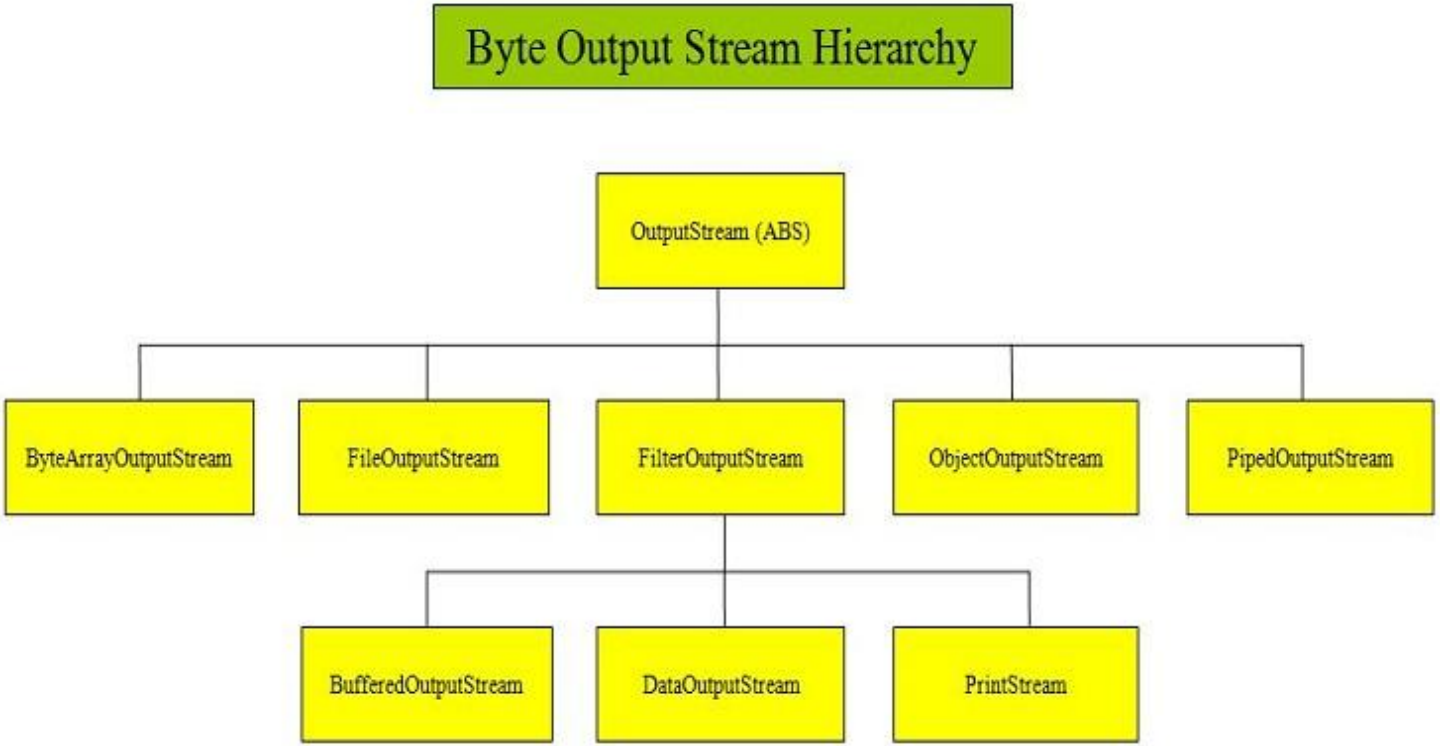
InputStream - абстрактный поток ввода

- int read() - читает байт из потока
- int read(byte[] buffer) - читает массив байт из потока и возвращает количество прочитанных байт

- int available() - возвращает количество байт, доступных к чтению
- long skip(long bytesAmount) - пропускает заданное количество байт

- void close() - закрывает поток

Байтовые потоки ввода-вывода



Байтовые потоки ввода-вывода

OutputStream - абстрактный поток вывода

- void write(int bt) - записывает байт в поток
- void write(byte[] buffer) - записывает массив байт в поток

- int flush() - “сбрасывает” внутренний буфер потока получателю

- void close() - “сбрасывает” внутренний буфер потока получателю и закрывает поток

Байтовые потоки ввода-вывода

File - описывает файл либо директорию

- File(dir_path, file_path)

Байтовые потоки ввода-вывода

File - описывает файл либо директорию

- File(dir_path, file_path)

- File getParentFile() - возвращает директорию файла
- boolean exists() - проверяет файл на существование
- boolean isFile() - проверяет является ли файл файлом или директорией
- File[] listFiles() - возвращает содержимое директории

Байтовые потоки ввода-вывода

File - описывает файл либо директорию

- File(dir_path, file_path)

- File getParentFile() - возвращает директорию файла
- boolean exists() - проверяет файл на существование
- boolean isFile() - проверяет является ли файл файлом или директорией
- File[] listFiles() - возвращает содержимое директории

- boolean createNewFile() - создает новый файл
- boolean mkdir() - создает новую директорию
- boolean delete() - удаляет файл
- boolean renameTo(File fl) - переименовывает файл

Байтовые потоки ввода-вывода

File Input/Output Stream - файловые потоки ввода-вывода

- в конструкторе можно передать путь к файлу либо объект File

Байтовые потоки ввода-вывода

File Input/Output Stream - файловые потоки ввода-вывода

- в конструкторе можно передать путь к файлу либо объект File
- с ними удобно работать через прокси Buffered Input/Output Stream
- либо использовать через Data Input/Output Stream

Байтовые потоки ввода-вывода

PrintStream - поток вывода предоставляющий операции форматирования

- void print - пишет строку или объект в поток
- void println - пишет строку или объект в поток и переходит на новую строку
- void printf (String formatStr, Object… args) - пишет отформатированную строку в поток
-
- PrintStream format (String formatStr, Object… args) - пишет отформатированную строку в поток и возвращает его

Байтовые потоки ввода-вывода

PrintStream - поток вывода предоставляющий операции форматирования

- void print - пишет строку или объект в поток
- void println - пишет строку или объект в поток и переходит на новую строку
- void printf (String formatStr, Object… args) - пишет отформатированную строку в поток
-
- PrintStream format (String formatStr, Object… args) - пишет отформатированную строку в поток и возвращает его

Работает и с байтами, и с символами,
но не позволяет задать кодировку!

Можно считает его устаревшим!

Байтовые потоки ввода-вывода

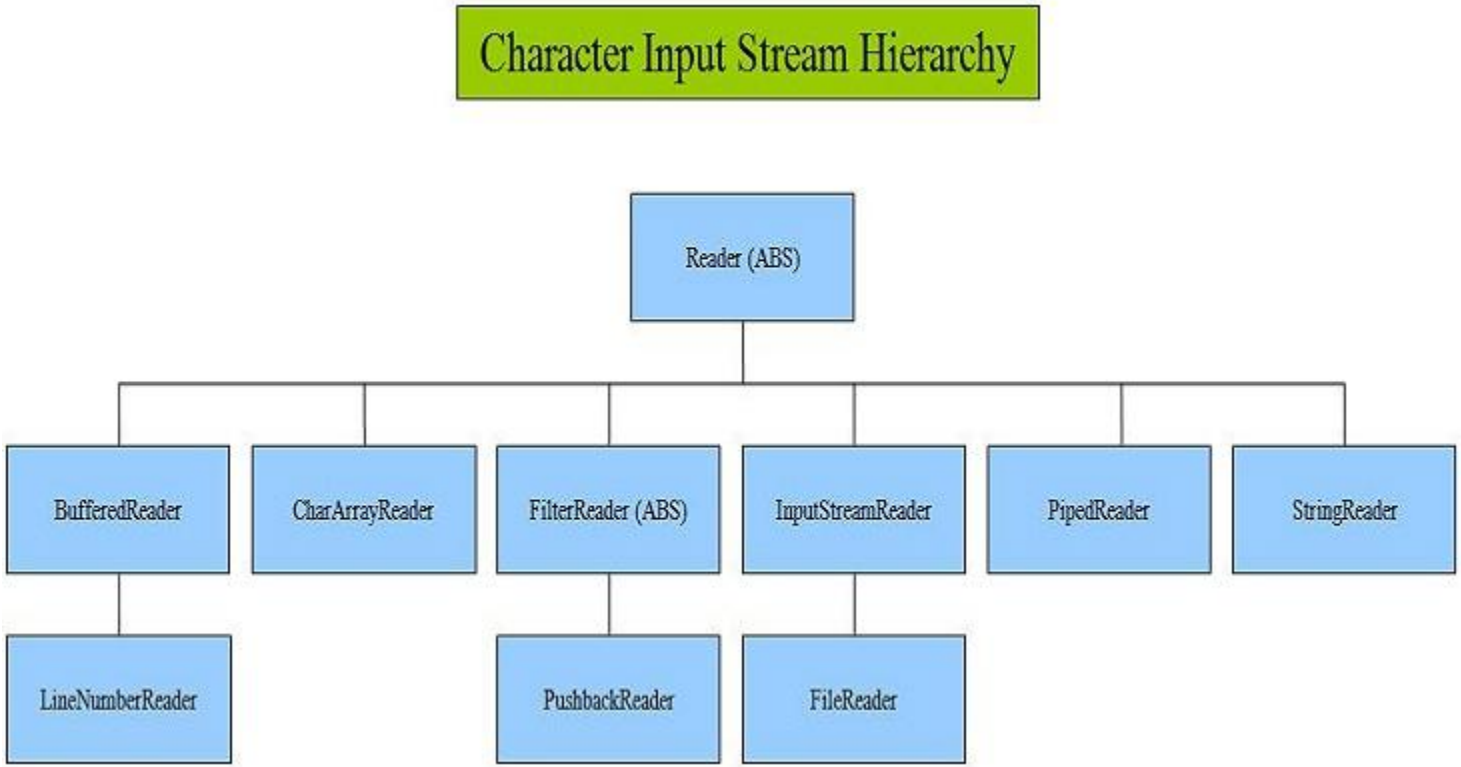
PrintStream - поток вывода предоставляющий операции форматирования

- void print - пишет строку или объект в поток
- void println - пишет строку или объект в поток и переходит на новую строку
- void printf (String formatStr, Object… args) - пишет отформатированную строку в поток
-
- PrintStream format (String formatStr, Object… args) - пишет отформатированную строку в поток и возвращает его

Работает и с байтами, и с символами,
и позволяет задать кодировку с JDK 1.4!

Но все равно его лучше не использовать!

Символьные потоки ввода-вывода



Символьные потоки ввода-вывода

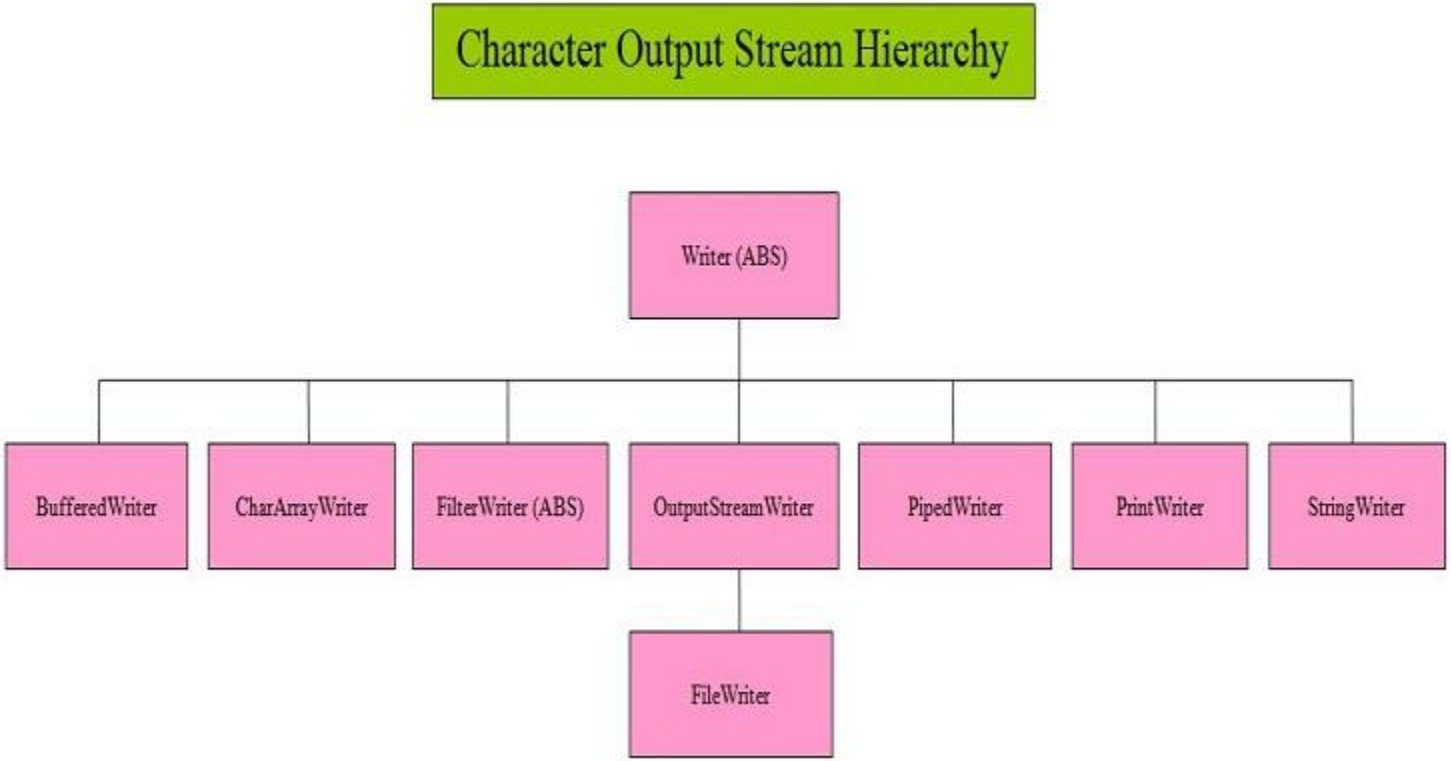
Reader - абстрактный поток символьный ввода

- int read() - читает символ из потока
- int read(char[] buffer) - читает массив символов из потока и возвращает количество прочитанных символов
- int read(CharBufffer buffer) - читает символы в буфер и возвращает количество прочитанных символов

- int available() - возвращает количество символов, доступных к чтению
- long skip(long charAmount) - пропускает заданное количество символов

- void close() - закрывает поток

Символьные потоки ввода-вывода



Символьные потоки ввода-вывода

Writer - абстрактный символьный поток вывода

- void write(String str) - пишет строку в поток
- void write(char[] buffer) - пишет массив символов в поток

- int flush() - “сбрасывает” внутренний буфер потока получателю

- void close() - закрывает поток

Символьные потоки ввода-вывода

File Reader/Writer - файловые символьные потоки ввода-вывода

- в конструкторе можно передать путь к файлу либо объект File

Символьные потоки ввода-вывода

File Reader/Writer - файловые символьные потоки ввода-вывода

- в конструкторе можно передать путь к файлу либо объект File
- с ними удобно работать через прокси Buffered Reader/Writer

Сериализация

Сериализация это процесс записи состояния объектов в байтовый поток вывода

Используется для сохранения в

- файл
- массив байт (для пересылки состояния объекта по сети, напр для RMI)

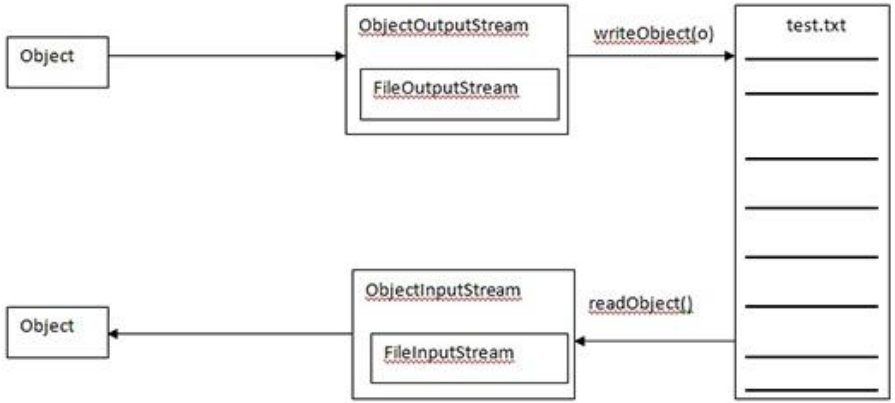


Fig: 1.2

Сериализация

Сериализация это процесс записи состояния объектов в байтовый поток вывода

- Класс должен имплементировать интерфейс-маркер Serializable
- При (де)сериализации объекта будут (де)сериализованы (почти) все его поля.

Сериализация

Сериализация это процесс записи состояния объектов в байтовый поток вывода

- Класс должен имплементировать интерфейс-маркер Serializable
- При (де)сериализации объекта будут (де)сериализованы (почти) все его поля.
- Можно определить private методы write/readObject для кастомизации записи и чтения в поток

Сериализация

Сериализация это процесс записи состояния объектов в байтовый поток вывода

- Класс должен имплементировать интерфейс-маркер Serializable
- При (де)сериализации объекта будут (де)сериализованы (почти) все его поля.
- Можно определить private методы write/readObject для кастомизации записи и чтения в поток

- Для “ручного” контроля сериализации используют Externalizable и реализуют методы write/readExternal

Сериализация

ObjectOutputStream используется для сериализации объекта в поток

- void writeObject(Object obj) сериализует и записывает объект в поток
- void writeXXX(XXX var) сериализует и записывает примитив XXX в поток

Сериализация

ObjectInputStream используется для десериализации объекта в поток

- Object readObject() читает и десериализует объект из потока
- XXX readXXX() читает и десериализует примитив XXX из потока

Сериализация

Сериализация сложный процесс, работающий с состоянием класса и его структурой

Q:
Что произойдет, если между и сериализаций и десериализацией изменилась структура класса?

Сериализация

Сериализация сложный процесс, работающий с состоянием класса и его структурой

Q:
Что произойдет, если между и сериализаций и десериализацией изменилась структура класса?

A:
Можно получить неконсистентные данные после десериализации!

Сериализация

Сериализация сложный процесс, работающий с состоянием класса и его структурой

Нужно использовать static final поле serialVersionUUID в классе, для определения версии

- Если изменения в классе “не ломающие” (напр, добавление нового поля), то версию не увеличиваем
- Если изменения в классе “ломающие” (напр, удаление поля, изменение типов полей), то увеличиваем версию

Сериализация

Сериализация сложный процесс, работающий с состоянием класса и его структурой

Нужно использовать static final поле serialVersionUUID в классе, для определения версии

- Если изменения в классе “не ломающие” (напр, добавление нового поля), то версию не увеличиваем
- Если изменения в классе “ломающие” (напр, удаление поля, изменение типов полей), то увеличиваем версию

InvalidClassException

кидается в случае, если версия была изменена,
но код, производящий десериализацию, не был обновлен!

Работа с файловой системой Java NIO

NIO (New I/O) новая система ввода-вывода в Java (введена в v 1.4 и улучшенна в v 1.7)

NIO решает следующие задачи

- операции в файловой системе
- канальный ввод-вывод
- потоковый ввод-вывод

Работа с файловой системой Java NIO

- **Path -** инкапсулирует путь к файлу и предоставляет API для работы с ним (с путем). Создается фабрикой Paths
- **Files -** предоставляет методы для работы с файлом, определенным через Path

Работа с файловой системой Java NIO

- **Path -** инкапсулирует путь к файлу и предоставляет API для работы с ним (с путем). Создается фабрикой Paths
- **Files -** предоставляет методы для работы с файлом, определенным через Path

Q:
Зачем вводить новый средства,
когда уже есть File из Java IO?

Работа с файловой системой Java NIO

- **Path -** инкапсулирует путь к файлу и предоставляет API для работы с ним (с путем). Создается фабрикой Paths
- **Files -** предоставляет методы для работы с файлом, определенным через Path

Q:
Зачем вводить новый средства,
когда уже есть File из Java IO?

A:
File имеет ряд проблем:

- не достаточная поддержка Exception
- проблемы с кроссплатформенностью
- нет поддержки символических ссылок
- и тд

Работа с файловой системой Java NIO

- **Path.getName()** получение имени файла
- **Path.toAbsolutePath()** получение абсолютного пути
- **Path.getParent()** получение родительской директории

- **Files.exists(path)** проверка существования файла
- **Files.isHidden/isWritable/isReadable** без комментариев ;)
- **Files.walkFileTree (path, fileVisitor)** обход всех файлов в директории

- **Files.createFile/Directory/Link** создание файла, директории, ссылки
- **Files.delete(path)** удаление файла
- **etc...**

Работа с файловой системой Java NIO

NIO (New I/O) новая система ввода-вывода в Java (введена в v 1.4 и улучшенна в v 1.7)

NIO решает следующие задачи

- операции в файловой системе
- канальный ввод-вывод
- потоковый ввод-вывод

Работа с файловой системой Java NIO

NIO (New I/O) новая система ввода-вывода в Java (введена в v 1.4 и улучшенна в v 1.7)

Основные элементы NIO это

- **буфер** хранит данные
- **канал** представляет соединение с физ. устройством ввода-вывода

Работа с файловой системой Java NIO

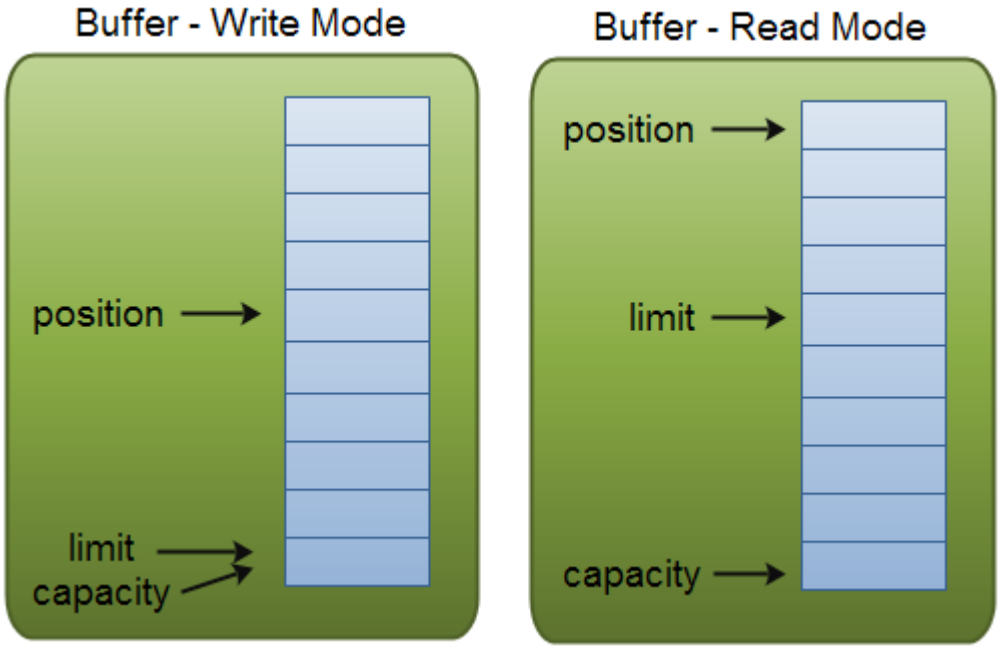
Буфер (Buffer) конечная последовательность элементов примитивного типа

Ключевые характеристики

- Текущая позиция (position)
- Предел (limit) - реальное количество элементов в буфере
- Емкость (capacity) - максимально возможное количество элементов в буфере

Работа с файловой системой Java NIO

Буфер (Buffer) конечная последовательность элементов примитивного типа



DonRiver

Работа с файловой системой Java NIO

Буфер (Buffer) конечная последовательность элементов примитивного типа

Ключевые методы записи и чтения

- **РеализБуфера put(ПримТип val)** - записывает значение в буфер и возвращает сам буфер
- **ПримТип get(int position)** возвращает значение относительно указанной позиции
- **ПримТип get()** возвращает значение относительно текущей позиции

Работа с файловой системой Java NIO

Буфер (Buffer) конечная последовательность элементов примитивного типа

Ключевые методы

- **Buffer clear()** очищает буфер, устанавливает позицию на 0 и лимит равный емкости буфера
- **Buffer reset()** устанавливает позицию на текущую метку и не меняет лимит. Метку можно на некоторую позицию в буфере задать специально.

- **Buffer rewind()** устанавливает позицию на 0 и не меняет лимит
- **Buffer flip()** устанавливает позицию на 0 и лимит равным текущей позиции

Работа с файловой системой Java NIO

Канал (Channel) представляет открытое соединение с источником или получателем данных

- Получается методом getChannel() из следующих классов
  - FileInput/OutputStream, RandomAccessFile вернут FileChannel
  - Socket, ServerSocket вернут SocketChannel
  - DatagramSocket вернут DatagramChannel
- Работает с буфером

Работа с файловой системой Java NIO

Q:

Зачем добавлять канальный IO?
Чем не устраивает потоковый?

A:
Канальный IO

- работает быстрее, поскольку использует буферы
- есть поддержка не-блокирующего ввода-вывода (актуально для работы с сокетами)

Работа с файловой системой Java NIO

NIO (New I/O) новая система ввода-вывода в Java (введена в v 1.4 и улучшенна в v 1.7)

NIO решает следующие задачи

- операции в файловой системе
- канальный ввод-вывод
- потоковый ввод-вывод

Работа с файловой системой Java NIO

- Потоковый ввод-вывод в NIO такой же как и в Java IO
  Для потокового ввода-вывода используются Input/Output Stream

- **FIles.newInputStream(Path, ...)** возвращает поток ввода
- **Files.newOutputStream(Path,...)** возвращает поток вывода

try-with-resources

- конструкция try-with-resources была введена в Java 1.7
- обеспечивает удобную и безопасную работу с потоками ввода-вывода и соединениями с ресурсами, которые реализуют интерфейс AutoCloseable

try-with-resources

- конструкция try-with-resources была введена в Java 1.7
- обеспечивает удобную и безопасную работу с потоками ввода-вывода и соединениями с ресурсами, которые реализуют интерфейс AutoCloseable

```
try(создание_одной_или_неск_реализаций_AutoCloseable){
        // работа с подключениями к ресурсам
} cath(список_исключений_1){
...
 cath(список_исключений_n){
} finally {
        // не нужно закрывать подключения к ресурсам вручную!
}
```

Home work

- Добавить в чат возможность сохранять переписку, путем ввода спец команды. Определить синтаксис команды самостоятельно
- Добавить возможность работы с сокетами используя Java NIO и не-блокирующий механизм Selector