Система ввода-вывода

- Класс File (на самом деле путь к файлу)
- Имя\набор файлов, хранящихся в каталоге, если каталог - list, чтобы получить массив строк. Если нужен конкретный файл нужно использовать фильтр каталогов - DirFilter - интерфейс
- Предоставляет метод accept для метода list, чтобы метод list мог вызвать accept для определения имен файлов, включаемых в список. Принимает в качестве аргументов File - каталог, в котором был найден файл
- Обход дерева каталогов можно осуществить следующим образом - вызвать метод listFiles, внутри фильтры - по строке.
- класс можно использовать для создания нового каталога или дерева каталогов
- Можно проверить характеристики файлов, проверить характеристики файлов.
- Доступен ли для чтения\записи, имя, путь, родительская папка, длина, последнее изменение
- В методе main вызывается метод renameTo(), позволяет переименовывать\перемещать файлы
- используя второй аргумент (место папки куда переместить)

- Ввод вывод
- Часто используется абстракция потока. Интерфейс для ввода вывода на основе inputStream/outputStream
- reader\writer.
- Типы InputStream - классы, получающие данные из различных источников
- массив байтов, строка, файл, посл потоков в одном потоке
- распр - FileInputStream - для чтения информации из файла
- Типы outputStream - классы, определяющие куда напр ваши данные - массив байтов\
- файл\канал
- распр - FileInputStream

- Добавление атрибутов и интерфейсов - Декораторы\надстройки. Обязательно - наследование от
- классов InputStream\OutputStream

- DataInputStream - чтение из потока простейших данных и строк

- DataOutPutStream - позволяет записывать в поток примитивы, не зависимо от платформы
- Reader и Writer - проведение операций символьно-ориентированного ввода-вывода
- в кодировке Юникод. Основная причина появлений Reader и Writer - для поддержки 16-ти
- битовых символов Юникода

- Источники и приемники данных - потомки потоковых классов, реализ функц reader и
- writer. Иногда нужно все-таки использовать reader и writer, работающие с с байтами

- Изменение поведения потока
- специальные файлы, изменяющие поведение потока в зависимости от ситуации, где он
- используется

- RandomAccesFile - предназначен для работы с файлами, содержащими записи известного
- размера, по которым можно перемещаться с помощью метода seek() с последующим
- чтением\записью данных.

- Не является частью иерархии InputStream\OutputStream.Позволяет свободно
- пермещаться по файлу в прямом и обратном напр, унаследован от Object

- RandomAccesFile похож на совмещение потокового чтения и вывода с доп методом getFilePointer?
- показывающим текущую позицию в файле + методом lenght()

- Из библиотеки ввода-вывода можно составить конфигурации
- Можно осуществить чтение из файла
- FileInputReader - извлечение строк или объекта File. Ускорение процесса чтения за счет буфферизации ввода
-
- Совмещение - есть конфигурация

- метод close() должен вызываться в завершающем методе finalize(), но недостоточно согласованно

- При работе с потоками лучше всегда вызывать метод close

- Чтение из памяти - StringReader, внутри BufferedInputFile.read(MainClass.java)

- Форматированное чтение из памяти
- Применяется класс DataInputStream, ориентированный на ввод-вывод байтов
- FileWriter - запись данных в файл

- PrintWriter - сокр. запись для вывода в текстовые файлы. (не нужно реализовывать
- буфферизацию)
- Сохранение и восстановление данных
- PrintWriter - форматирование данных, чтоб их мог прочитать человек
- Но если данные нужны для другого потока, то лучше dataoutput\datainput Stream

- Каналы - предназначены для связи между отдельными программными потоками

- Средства чтения и записи файлов - PrintWriter - позволяет лекго открыть вспомогательный конструктор PrintWriter

- метод write("текст", куда записывать) - читает и записывает

- Стандартный ввод-вывод - концепция единого потока информации, используемого программой

- Чтение из стандартного потока - System, читает построчно -> имеет смысл буфферизировать

- Замена System.out на PrintWriter -
- PrintWriter(System.out,true)
- Ориентиированный поток на запись
- true - автоматический сброс буфера на печать

- Перенаправление стандартного ввода-вывода
- BufferedStream in = new BufStr(new FileInputStream("classpath"))

- Увеличение скорости с помоью nio
- использование каналов(посылают извлекают данные) и буферов(посылаются в каналы)
- напрямую действуеттолько byteBuffer

- ByteBuffer - буфер для данных в виде байтов
- поддерживает методы, образующие любые значения

- Создается упаковкой массива из восьми байтов, который затем просматривается через
- представления для всех возможных примитивных типов

- Буферы и манипуляции данными - перемещать данные допустимо только с помощью байтовых буферов
- Для примитивов отдельный буфер -> буфер с примитивами не есть буфер с данными

- Индексы буфера - метка, позиция, предел, вместимость
- Быстродействие старого ввода вывода - улучшено, отображение файлов старое эффективнее
- Блокировка файлов (синхронизация доступа к файлу как к совместно используемому ресурсу)
- объект FileLock с методом tryLock
- Частичная блокировка - возможна с LockAndModify и указания размеров блокируеммых данных
- Сжатие данных
- Есть классы поддерживающие чтение сжатых ланных(zip, jar, gzip)
- Сериализация
- - используется для хранение объекта в виде данных где-то легковесна и долговременна
- Что бы было надо наседовать интерфейс + использовать ObjectOutputStream
- Для управления процессом сериализации необходимо реализовать в своем классе интерфейс Externalizablr
- вместо Serializable с методами writeExternal(), readExternal()
- Ключевое слово transient
- Слово transient - поле, которое не надо не сохранять не восстанавливать.
- Возможно возникновение ситуации, когда автоматическое сохранение и восставновление
- нежелательно. Если информация в объекте описана как закрытая - не спасает ее от сериализации->
- можно извлечь файлы из закрытого объекта.
- write External - явное указание какие части объекта должны сериализовываться

- Если информацию надо сохранить но без пароля (время входа пользователя в сеть).

- Альтернатива для Externalizable - можно реализовать Serializable и добавить методы с именами
- write и read Object
- Автоматически будут вызваны при сериализации и восстановлении объектов
- Если вы предоставите эти два метода, они будут исп вместо сериализации по умолчанию
- Можно вызвать стандартные а можно переопределить
- Метод read\writeObject - имеется ли в объекте свой собственный метод, если есть,то

- Если необходимо изменить версию объекта, то можно, но сложно и редко

- Долговременное хранение
- Прежде чем долговременно хранить объекты надо подумать
- Если нужно сохранить состояние системы - безопаснее всего в рамках "атомарной" операции
- Следует поместить все объекты в контейнер и сохранять контейнер.
- После можно восстановить его вызовом одного метода
- стат методы автоматически не сериализуются -> нужно самим или
- serializeStaticState(), deserializeStaticState() класса Line
- serializeStaticState
- deserializeStaticState

- преобразование сериализ данных в XML - для использования на разных платформах
- для этого бибилиотека XOM
- описание в документации
- содержит класс сериализации, который используется в методе format - преобразование xml
- в удобочитаемую форму.
- Необходимо заранее знать структуру файла XML

- Предпочтения

- более тесно связаны с долговременным хранением. Предназначены для хранения и получения
- информации о предпочтениях пользователя и конфигурации программы

- Набор пар ключ-значение образующих иерархию узлов. Создают один узел и хранят информацию в нем
- Данные хранятся используя системные возможности, различных, в различных опреационных системах
- Информация сохраняется автоматически

- Бибилиотека ввода-вывода подходит для чтения данных и запись на консоь, в файл, в буфер памяти, сетевое
- Интернета
- Применяя наследование можно создать новые типы объектов для ввода и вывода данных
- Можно расширить виды объектов, принимаемых потоком.
- Проверка существования файла возможна с помощью объекта File

- RandomAcsessFile - немалые усилия для ее изучения