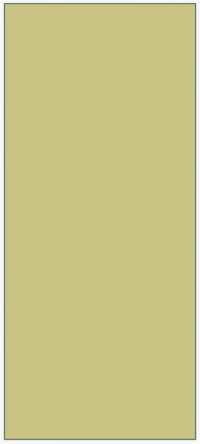
Interface again

JAVA ADVANCED (NOT SO)

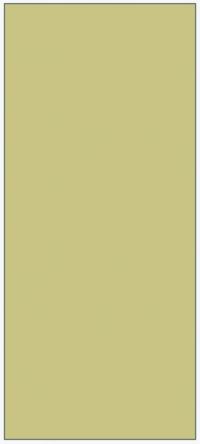


INTERFACE

- interface Instrument {
  int VALUE = 5; // public, static и final!
  void play(Note n); // public and abstract!
  void adjust(); // static methods are forbidden!
  }
- class Percussion implements Instrument {
  public void play(Note n) {
  …
  }
  public void adjust() {
  …
  }
  }

ВВОД-ВЫВОД

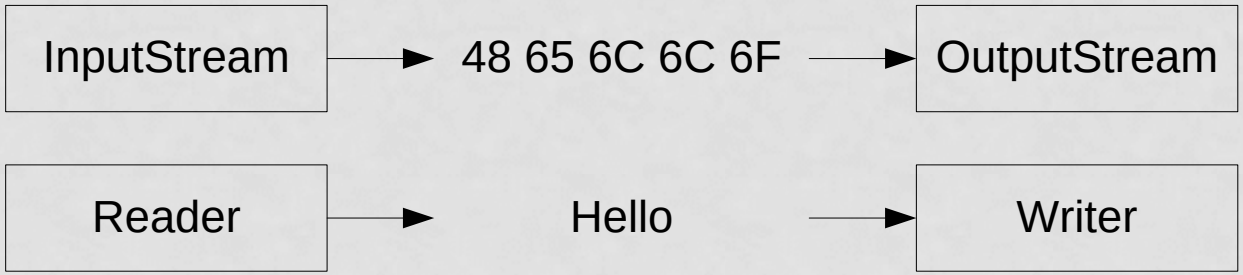
JAVA ADVANCED (NOT SO)



СОДЕРЖАНИЕ

ВВОД-ВЫВОД В JAVA

- Потоки ввода-вывода
- Пакет java.io



ПОТОКИ ВВОДА-ВЫВОДА

ЧАСТЬ 1

ВИДЫ ПОТОКОВ

- Направление
  - Ввод
  - Вывод
- Содержимое
  - Байтовые
  - Символьные

КЛАССЫ ПОТОКОВ

	Байтовый	**Символьный**
Ввод	InputStream	Reader
Вывод	OutputStream	Writer

ИСКЛЮЧИТЕЛЬНЫЕ СИТУАЦИИ

- Класс IOException
  - Корень иерархии исключений ввода-вывода
  - Бросается всеми операциями ввода/вывода
- Класс EOFException
  - Достигнут конец потока
- Класс FileNotFoundException
  - Файл не найден
- Класс UnsupportedEncodingException
  - Неизвестная кодировка

ПОТОКИ ВВОДА

- Основные операции
  - int read() — чтение элемента
  - read(T[] v), read(T[] v, off, len) — чтение элементов в массив
- Дополнительные операции
  - skip(n) — пропуск n элементов
  - close() — закрытие потока
- Пометки и возвраты
  - mark(limit) — пометка текущей позиции
  - reset() — возврат к помеченной позиции

ПОТОКИ ВЫВОДА

- Основные операции
  - write(int v) — запись элемента
  - write(T[] v) — запись массива элементов
  - write(T[] v, off, len) — запись части массива
- Дополнительные операции
  - flush() — запись буфера
  - close() — закрытие потока

ПРИМЕР: БЛОЧНОЕ КОПИРОВАНИЕ

- Процедура копирования

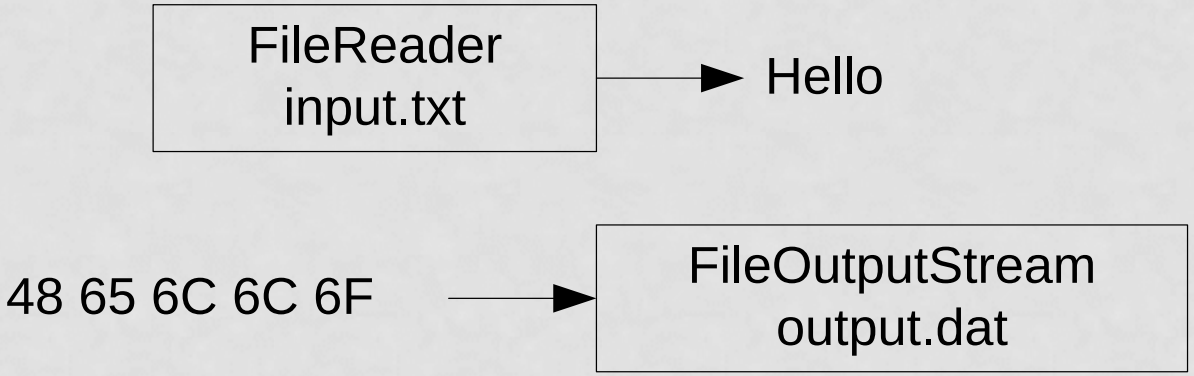
```
void copy(InputStream is, OutputStream os)
    throws IOException
{
    byte[] b = new byte[1024];
    int c = 0;
    while ((c = is.read(b)) >= 0) {
        os.write(b, 0, c);
    }
}
```

ФАЙЛОВЫЙ ВВОД-ВЫВОД И КОНВЕРТАЦИЯ ПОТОКОВ

ЧАСТЬ 2

КЛАССЫ ФАЙЛОВОГО ВВОДА-ВЫВОДА

- Классы File*
  - FileInputStream
  - FileOutputStream
  - FileReader
  - FileWriter



СОЗДАНИЕ ФАЙЛОВЫХ ПОТОКОВ

- Для символьных потоков используется кодировка по умолчанию
- Для ввода/вывода
  - File*(File file) — по дескриптору
  - File*(String file) — по имени
- Для дописывания
  - File*(File file, boolean append) — по дескриптору
  - File*(String file, boolean append) — по имени

ПРИМЕР: ПРЕОБРАЗОВАНИЯ РЕГИСТРА

- Файл input.txt копируется в output.txt с изменением регистра

```
Reader reader = new FileReader("input.txt");
Writer writer = new FileWriter("output.txt");
int c = 0;
while ((c = reader.read()) >= 0) {
    writer.write(Character.toUpperCase((char) c));
}
reader.close();
writer.close();
```

БАЙТОВЫЙ ПОТОК → СИМВОЛЬНЫЙ

- При чтении возможно преобразование байтового потока в символьный, с указанием кодировки
- Класс InputStreamReader
  - InputStreamReader(InputStream, encoding?)

СИМВОЛЬНЫЙ ПОТОК → БАЙТОВЫЙ

- При записи возможно преобразование символьного потока в байтовый, с указанием кодировки
- Класс OutputStreamWriter
  - OutputStreamWriter(OutputStream, encoding?)

ПРИМЕР: ПЕРЕКОДИРОВАНИЕ ФАЙЛА

- Файл input.txt копируется в output.txt с изменением кодировки с Cp1251 на Cp866

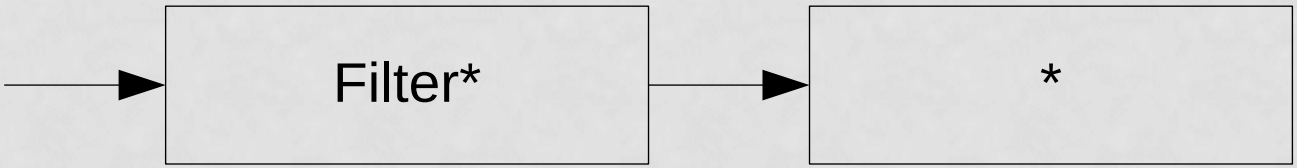
```
Reader reader = new InputStreamReader(
        new FileInputStream("input.txt"), "Cp1251");
Writer writer = new OutputStreamWriter(
        new FileOutputStream("output.txt"), "Cp866");
int c = 0;
while ((c = reader.read()) >= 0) writer.write(c);
reader.close();
writer.close();
```

ФИЛЬТРУЮЩИЕ ПОТОКИ

ЧАСТЬ 3

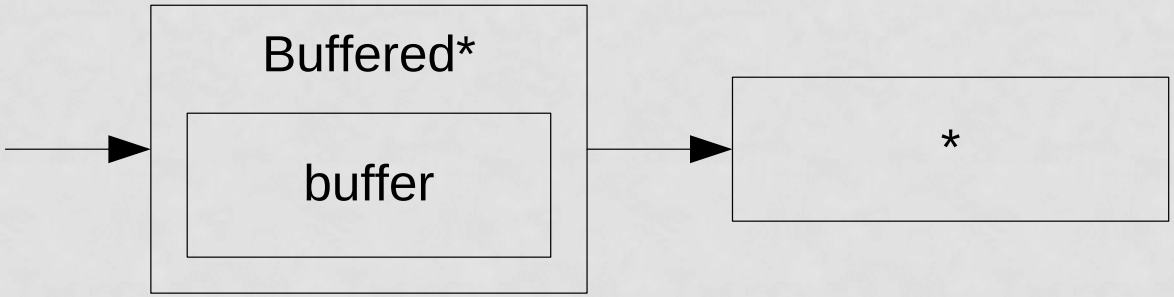
ФИЛЬТРУЮЩИЕ ПОТОКИ

- Направляют все вызовы вложенному потоку
- Классы Filter*



БУФЕРИЗУЮЩИЕ ПОТОКИ

- Содержат буфер, который считывают / записывают целиком
- Классы Buffered*



ПРИМЕР: ШИФРУЮЩИЙ ПОТОК

```
public class EncodingOutputStream extends
        FilterOutputStream {
    private final int key;

    public EncodingOutputStream(OutputStream os, int key) {
        super(os);
        this.key = key;
    }

    public void write(int b) throws IOException {
        super.write(b ^ key);
    }
}
```

ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ ПОТОКОВ

ЧАСТЬ 4

ЭМУЛЯЦИЯ ЧТЕНИЯ

- Чтение производится из буфера в памяти, передаваемого конструктору
- Классы
  - ByteArrayInputStream – чтение из массива байт
  - CharArrayReader – чтение из массива символов
  - StringReader – чтение из строки

ЭМУЛЯЦИЯ ЗАПИСИ

- Запись производится в буфер в памяти, который доступен в любое время
- Классы
  - ByteArrayOutputStream – запись в массив байт (toByteArray())
  - CharArrayWriter – запись в массив символов (toString(), toCharArray())
  - StringWriter – запись в StringBuffer (toString(), toStringBuffer())

КОНКАТЕНАЦИЯ ПОТОКОВ

- Несколько байтовых потоков можно конкатенировать
- Если первый из потоков закончился, производится чтение из второго и т.д.
- Класс SequenceInputStream
  - SequenceInputStream(InputStream, InputStream) – конкатенация двух потоков
  - SequenceInputStream(Enumeration) – конкатенация нескольких потоков

ВЫВОД С ПОДАВЛЕНИЕМ ОШИБОК

- Вывод осуществляется построчно, с подавлением ошибок
- Класс PrintWriter
  - checkError() – проверить, была ли ошибка
  - print(…) – запись без перевода строки
  - println(…) – запись с переводом строки

ВВОД С ПОДСЧЕТОМ СТРОК

- Ввод осуществляется построчно, с подсчетом количества строк
- Класс LineNumberReader
  - lineNumber() – текущий номер строки

РАСШИРЕННЫЙ ВВОД-ВЫВОД

ЧАСТЬ 5

РАСШИРЕННАЯ ЗАПИСЬ ДАННЫХ

- Платформонезависимая запись примитивных типов и строк
- Интерфейс DataOutput
  - writeT(T) – запись примитивных типов
  - writeUnsignedByte() / writeUnsignedShort() – запись беззнаковых целых
  - writeUTF() – запись строки в кодировке UTF-8
- Реализация
  - DataOutputStream

РАСШИРЕННОЕ ЧТЕНИЕ ДАННЫХ

- Платформонезависимое чтение примитивных типов и строк
- Интерфейс DataInput
  - T readT() – чтение примитивных типов
  - readUnsignedByte() / readUnsignedShort() – чтение беззнаковых целых
  - readUTF() – чтение строки в кодировке UTF-8
- Реализация
  - DataInputStream

ФАЙЛЫ С ПРОИЗВОЛЬНЫМ ДОСТУПОМ

- Класс RandomAccessFile
  - Реализует DataInput, DataOutput
- Конструктор
  - RandomAccessFile(file, mode) – открыть файл в заданном режиме

Строка	Режим
r	Чтение
w	Запись
rw	Чтение и запись
rws	Синхронное чтение и запись

ДОПОЛНИТЕЛЬНЫЕ ОПЕРАЦИИ

- Методы
  - length() – получить размер файла
  - setLength() – установить размер файла
  - getFilePointer() – получить положение указателя
  - seek(long) – установить положения указателя

КЛАСС SYSTEM

- out – PrintStream для stdout
- err – PrintStream для stderr
- in – InputStream для stdin
- setOut(.) / setErr(.) / setIn(.) – установка консольных потоков
- console() – символьный доступ к консоли

КЛАСС CONSOLE

- format/printf(format, args) – Форматированный вывод
- reader() – Reader для stdin
- writer() – Writer для stdout
- readLine() – чтение строки текста
- readPassword(format?, args?) – чтение пароля

ДЕСКРИПТОРЫ ФАЙЛОВ

ЧАСТЬ 6

ДЕСКРИПТОРЫ ФАЙЛОВ

- Позволяют осуществлять манипуляции с файлами
- Класс File
- Создание дескриптора по имени
  - File(pathname) – абсолютный или относительный путь
- В дескриптора по имени и директории
  - File(File dir, name)
  - File(String dir, name)

ОПЕРАЦИИ С ДЕСКРИПТОРАМИ

- Получение информации
  - getName() – получить имя
  - getPath() – получить имя и путь
  - getAbsolutePath() – получить абсолютный путь
  - getAbsoluteFile() – получить абсолютный дескриптор
- Определение родителя
  - String getParent() – как строки
  - File getParentFile() – как дескриптора

ОПЕРАЦИИ С ФАЙЛАМИ (1)

- Проверка типа
  - isFile() – является ли файлом
  - isDirectory() – является ли директорией
  - isHidden() – является ли скрытым
- Получение информации о файла
  - exist() – проверка существования
  - length() – длина файла
  - lastModified() – время последней модификации

ОПЕРАЦИИ С ФАЙЛАМИ (2)

- Создание
  - mkdir() – создать одну директорию
  - mkdirs() – создать все директории
  - createNewFile() – создать пустой файл
- Удаление
  - delete() – удалить немедленно
  - deleteOnExit() – удалить после завершения
- Переименование / перенос
  - renameTo(file) – переименовать / перенести в заданное место

ЛИСТИНГ ДИРЕКТОРИИ

- Листинг всех файлов
  - String[] list() – получить имена файлов
  - File[] listFiles() – получить дескрипторы файлов
- Листинг по критерию
  - String[] list(FileNameFilter) – получить имена файлов
  - File[] listFiles(FileFilter) – получить дескрипторы файлов

ПРОВЕРКА ДОСТУПА К ФАЙЛАМ

- canReadFile() – проверка возможности чтения
- canWriteFile() – проверка возможности записи
- canExecuteFile() – проверка возможности исполнения

ВВОД-ВЫВОД И ИСКЛЮЧЕНИЯ

ЧАСТЬ 7

ОБЫЧНАЯ ОБРАБОТКА ИСКЛЮЧЕНИЙ

```
Reader reader = new FileReader("input.txt");
try {
    // Операции с файлом
} finally {
    reader.close();
}
```

НАДЕЖНАЯ ОБРАБОТКА ИСКЛЮЧЕНИЙ

```
Reader reader = new FileReader("input.txt");
try {
    // Операции с файлом
    reader.close();
} catch (IOException e) {
    try {
        reader.close();
    } catch (IOException e) {  /* Ignoring */ }
    throw e;
}
```

АЛЬТЕРНАТИВНЫЙ МЕТОД

```
Reader reader = null;
try {
    reader = new FileReader("input.txt");
    …
} finally {
    if (reader != null) {
        reader.close();
    }
}
```

СЛУЧАЙ НЕСКОЛЬКИХ ПОТОКОВ

```
Reader reader = new FileReader("input.txt");
try {
    Writer writer = new FileWriter("output.txt");
    try {
        // Операции ввода-вывода
        ...
    } finally {
        writer.close();
    }
} finally {
    reader.close();
}
```

JAVA 7 STYLE

```
try (
    Reader reader = new FileReader("input.txt");
    Writer writer = new FileWriter("output.txt");
) {
    // Операции ввода-вывода
}
```

ПОДАВЛЕНИЕ ИСКЛЮЧЕНИЙ

- PrintWriter
  - checkError()
- PrintStream
  - checkErrors()
- Scanner
  - ioException()