Javaにおける入出力とXML

2011年5月15日

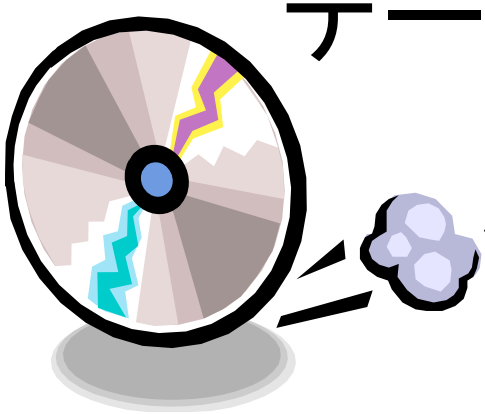
海谷 治彦

目次

- 入出力の抽象化
- ファイル入出力の概要
- XMLEncoder/Decoder
  - 永続化とは？

データと情報

データ



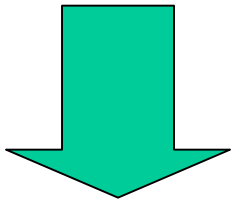
情報

人間による意味付けの無いもの.
CDの上の0/1列等.
磁気や突起等の物理情報

人間による意味付けの有るもの.
音楽，絵画，図書等.

情報処理と入出力

- 計算機では直接に情報は入出力できない.
- 意味付けの無いデータとして扱うしかない.



- 情報をデータに変換する作業のかなりの部分は人間が明示的にプログラミングしないといけない・・・

例

- 音声
  - Linier PCM や MP3等，音声情報をデータとして記述する規格が存在する.
  - 専門じゃないので詳細はわかりません.
- 画像
  - 同様にjpegやgif等，画像情報をデータとして記述する規格.
- 図書，文書
  - 通常は文字列として扱う.
  - MSワードやPDF等も文書情報をデータとして記述する規格の例.

OSのファイルシステム

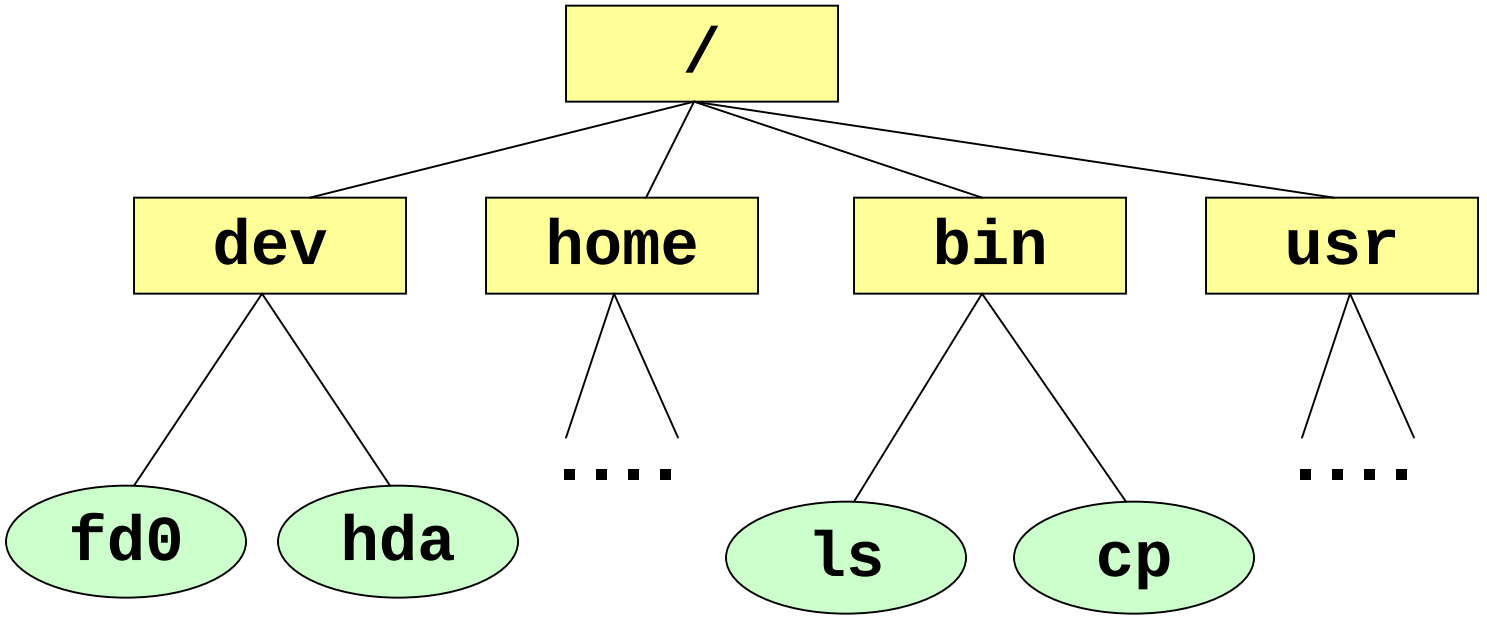
- (ご存知の通り，ほとんどの)OSは木構造のファイルシステムを持つ.
- 木の葉: ファイル
- 木の中間ノード: フォルダもしくはディレクトリ

- J2SEは一般的なOSのファイルシステムを扱うAPIを備えている.

OSのスライドから抜粋

UNIX系ファイルシステムの概要

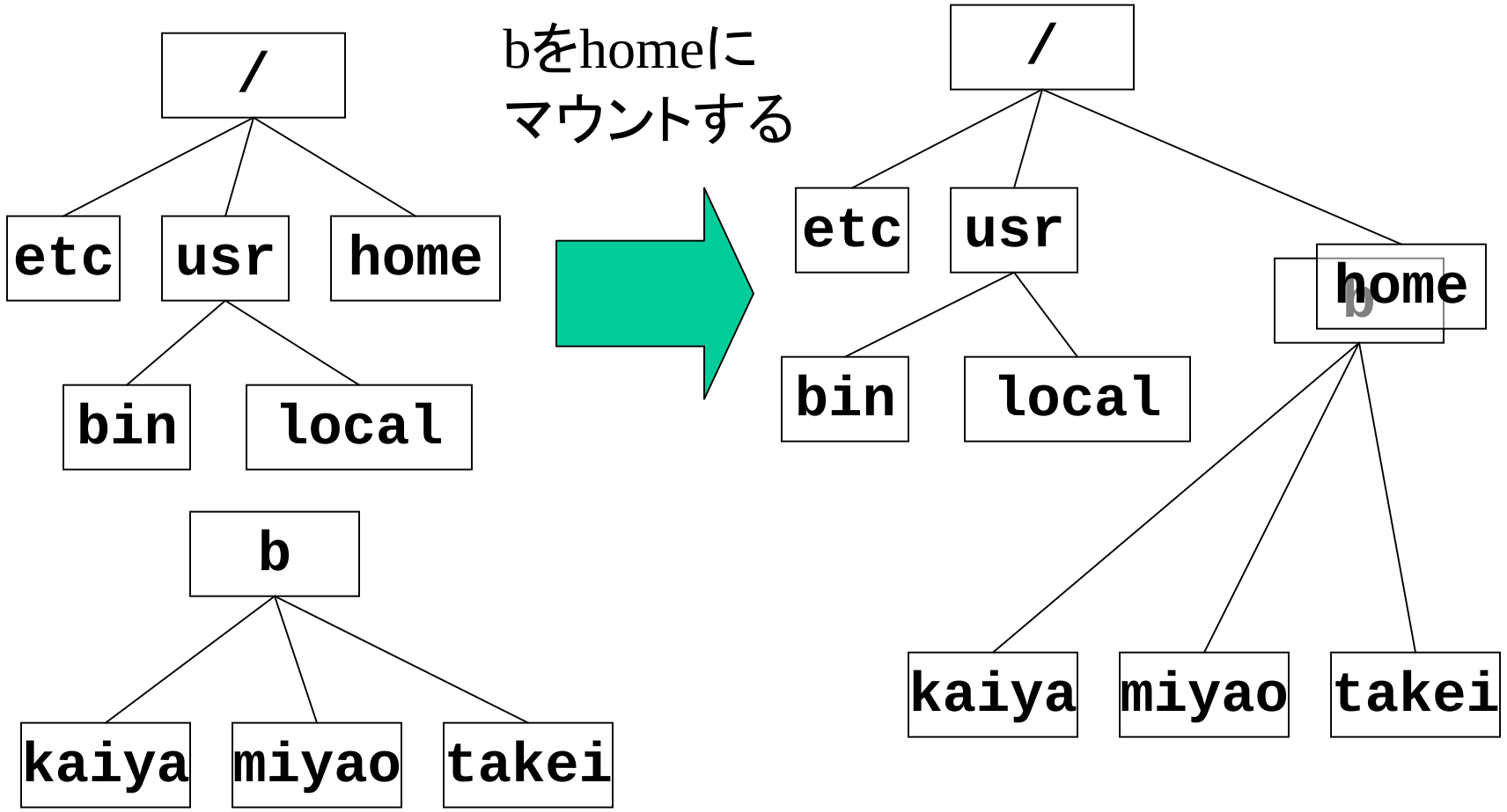
ご存知のとおり，UNIX系OS(その他も大抵そうだけど)は，階層的なファイルシステムを持っている．



文献5 p.15

OSのスライドから抜粋

mountの概念図

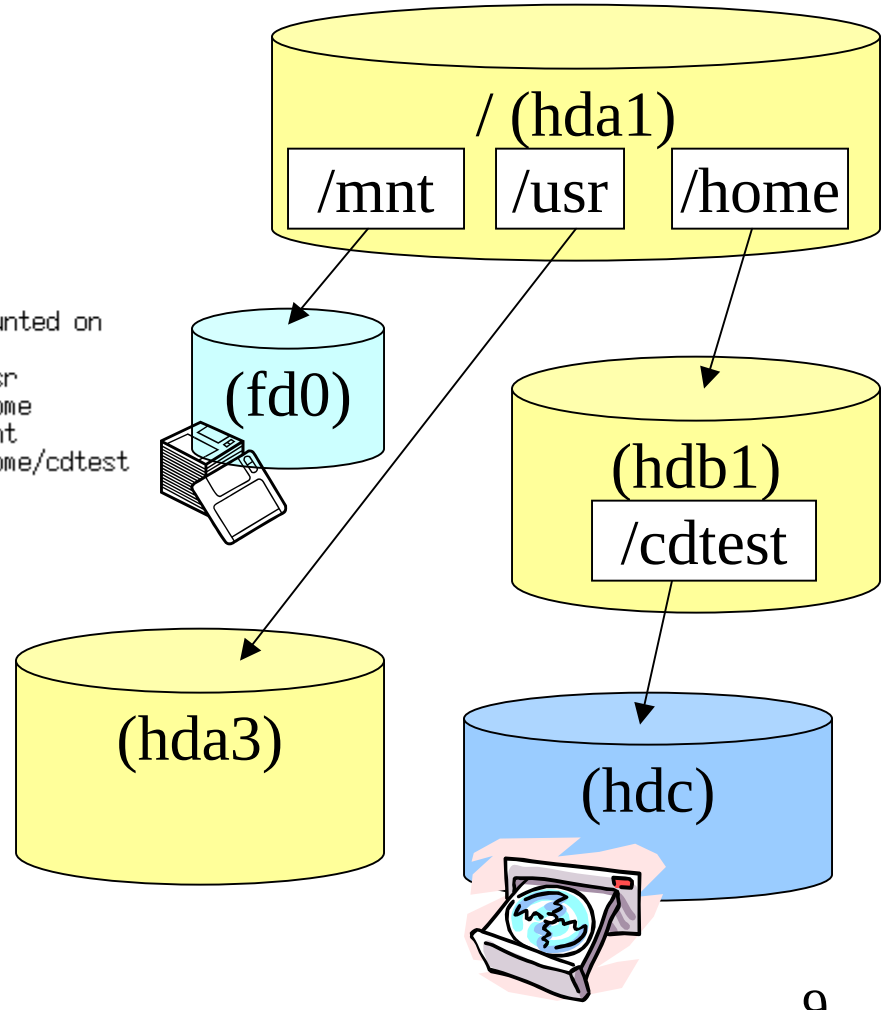


文献1 p.45 改

OSのスライドから抜粋

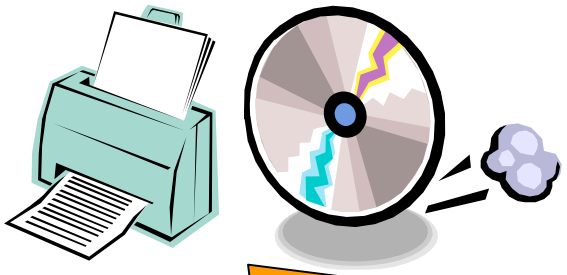
mountの実際

```
[kaiya@linux2001 ~]% !cat
cat /proc/mounts
/dev/root / ext2 rw 0 0
/proc /proc proc rw 0 0
usbdevfs /proc/bus/usb usbdevfs rw 0 0
/dev/hda3 /usr ext2 rw 0 0
none /dev/pts devpts rw 0 0
/dev/hdb1 /home ext2 rw 0 0
/dev/fd0 /mnt vfat ro 0 0
/dev/cdrom /home/cdtest iso9660 ro 0 0
[kaiya@linux2001 ~]% df
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hda1              2016016     85192   1828412   4% /
/dev/hda3              2016044   1466612    447020  77% /usr
/dev/hdb1             76920416    928080  72084928   1% /home
/dev/fd0                  1423       764       659  54% /mnt
/dev/hdc                  1896      1896         0 100% /home/cdtest
[kaiya@linux2001 ~]% []
```



外部データと内部データと情報

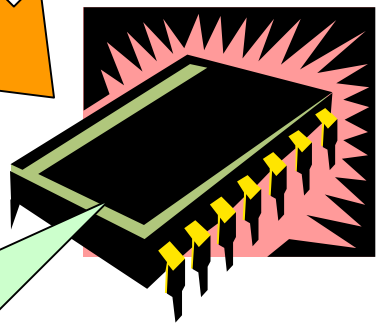
外部データ



情報

入出力

内部データ



```
int x, y, z;

class Foo extends Bar{
   String sakana="x";
};
```

内部データ(変数)の役割

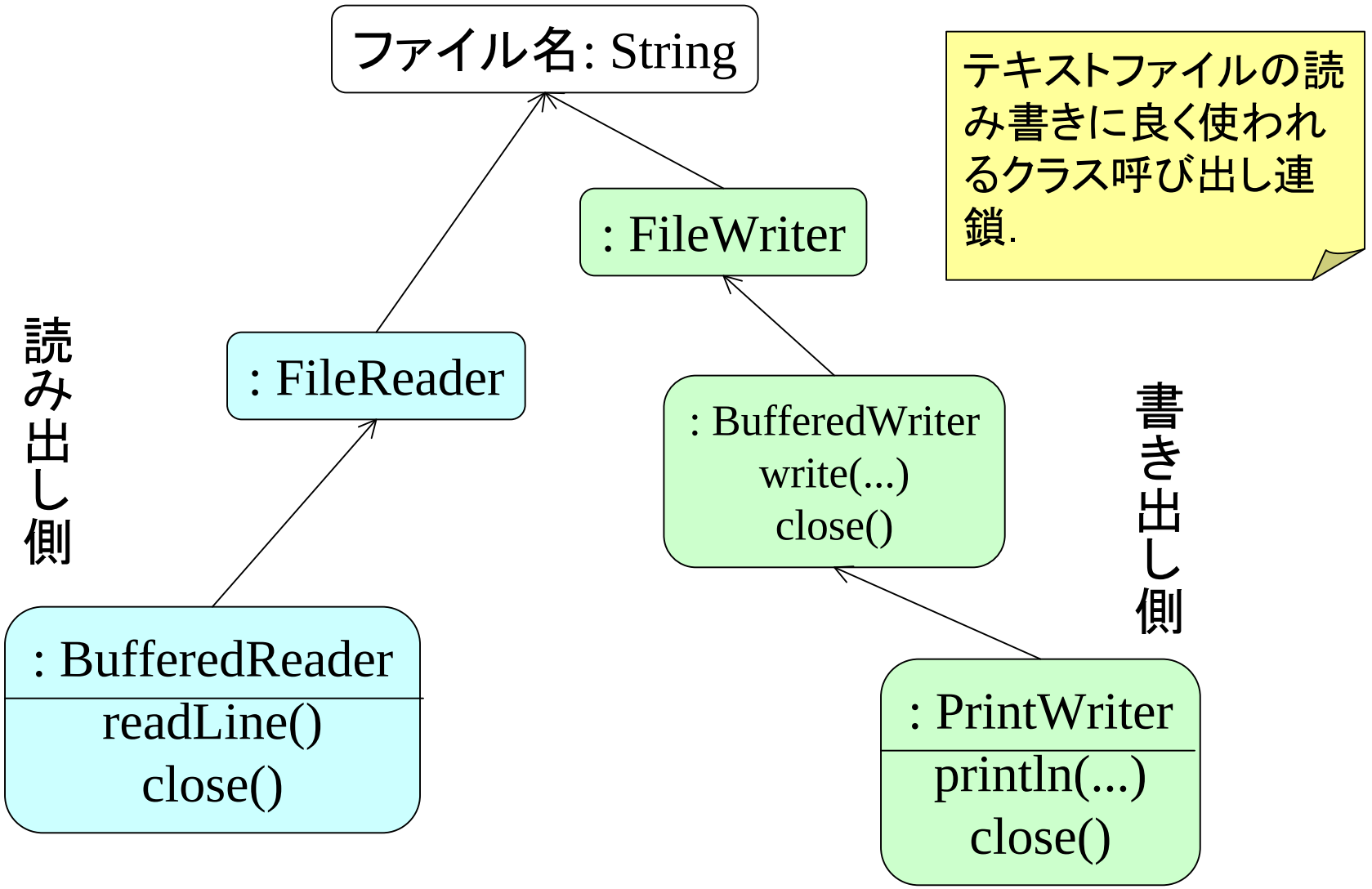
- コンピュータによる情報処理では直接に外部データを操作できない(わけじゃないけど，効率が悪い).
- 比較的早いメモリ上のデータ(変数)を操作して処理をするのが普通.
- 結局，入出力とは内部データと外部データを相互に変換する作業.
- この変換作業の手間を減らせればうれしい.

文字列: データ？ 情報？

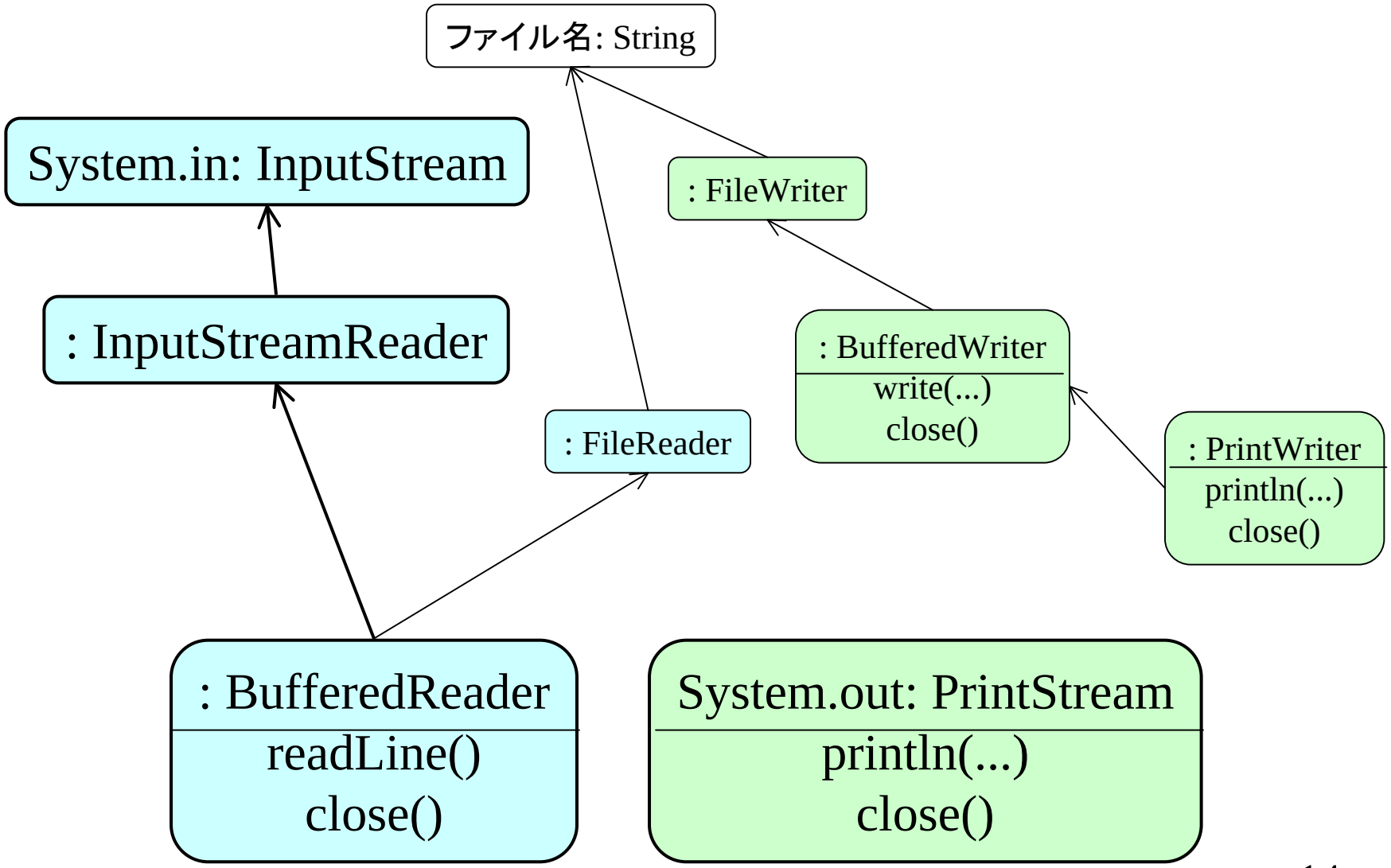
- JavaのStringは明らかにデータである.
- しかし，その字面を人間が解釈して(ある意味勝手に，しかし多くの人間間で共通理解を与えつつ)意味付けをすることができる.
- その意味で，文字列データは情報に近いものと考えられる.

- その意味でデータを(情報に近い形で保存するには)文字列が都合よい.

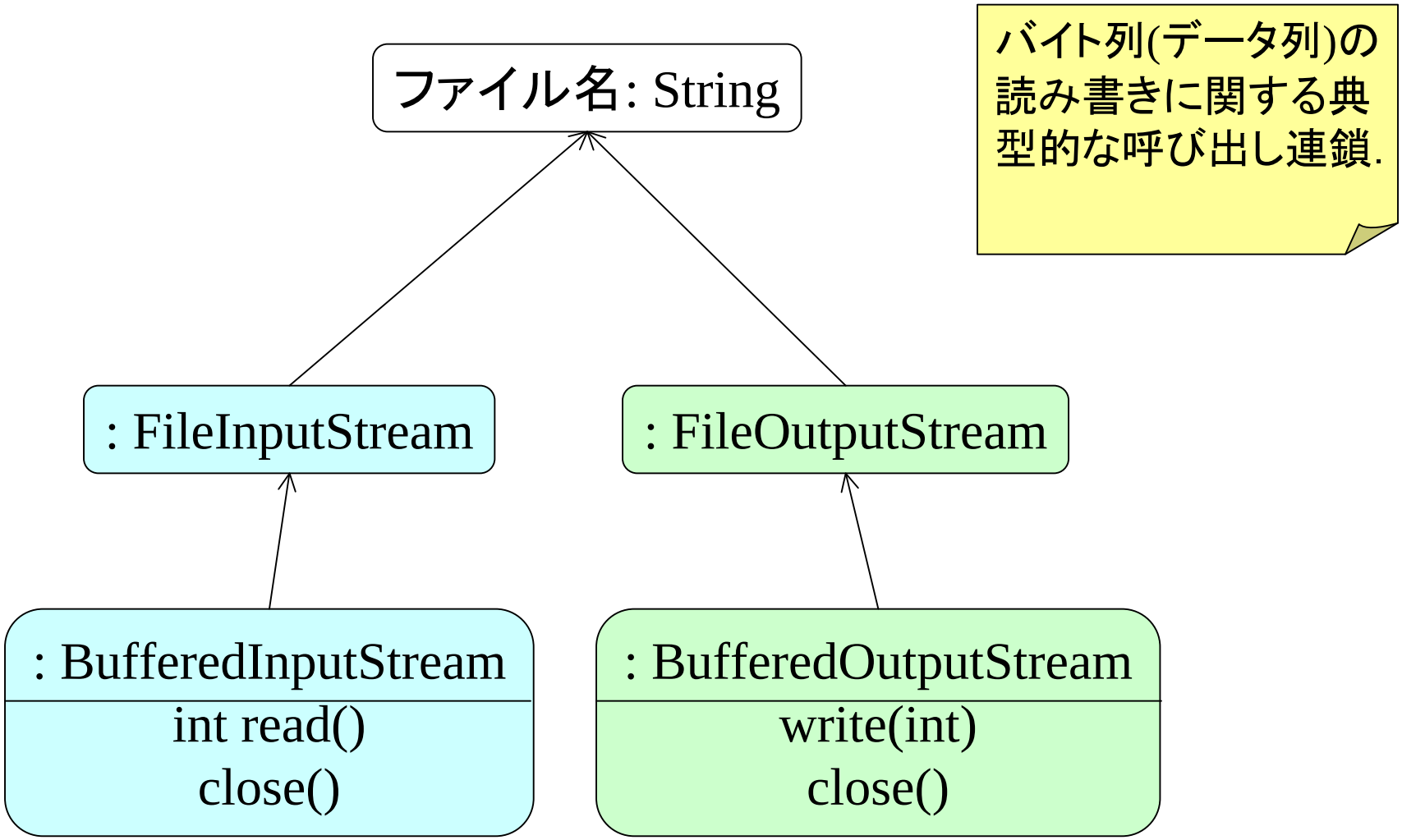
文字ストリーム


ファイル名: String
テキストファイルの読み書きに良く使われるクラス呼び出し連鎖.
: FileWriter
読み出し側
: FileReader
: BufferedWriter
write(...)
close()
書き出し側
: BufferedReader
readLine()
close()
: PrintWriter
println(...)
close()

Stdioからの読み書き



バイトストリーム


ファイル名: String
: FileInputStream
: FileOutputStream
: BufferedInputStream
int read()
close()
: BufferedOutputStream
write(int)
close()
バイト列(データ列)の読み書きに関する典型的な呼び出し連鎖.

入出力クラスの基本

- 基点はファイル名やストリーム.
- ストリーム: データの流れを示す抽象概念.
  - 詳細はOSの授業にてやります.
- いくつかのクラスを連鎖することで，扱う対象が単なるデータから情報に近くなっていく.
  - 単なるbit列から文字列になったりする.

XMLEncoder/Decoder

- Javaのオブジェクト(インスタンス)を手っ取り早く外部データ化するAPI.
- この外部データ化することを「永続化」と呼ぶ.
  - ファイルに保存とか，プログラム同士のデータ交換とか.

XML

- Extensible Markup Languageの略らしい.
  - 何故 EMLじゃないの？ 謎.
- 要はラベルのついた箇条書きの文字列データ.
- Cの構造体やJavaのメンバ属性等，構造化，グループ化されたデータを扱うのに便利.
- 流行・・・・
- XML文書を処理するためのAPIがどの言語でも豊富にある.

- あとはGoogleにでも聞いてください.

XMLEncode/Decodeの要件・利点

- 以下の要件を満たすクラスでないとならない.
  1. 無引数のpublicコンストラクタを持つ.
  2. 永続化したい属性にはsetter/getterがある.
- 利点
  - 永続化形式をプログラマが決めなくてもよい.
    - Cの演習でリスト構造のファイル保存形式を決めるような面倒はスキップできる.
  - あるオブジェクトを永続化すれば，参照されているオブジェクトも一緒に永続化される.
    - 無論，永続化の要件を満たさないとダメだけど.

Setter/Getter

- ある属性の値を設定(set)したり，獲得したり(get)したりするためのメソッド.
- アクセッサー(accesssors)とも呼ばれる.
- 属性の名前に従い，定められたメソッド名でないといけない.

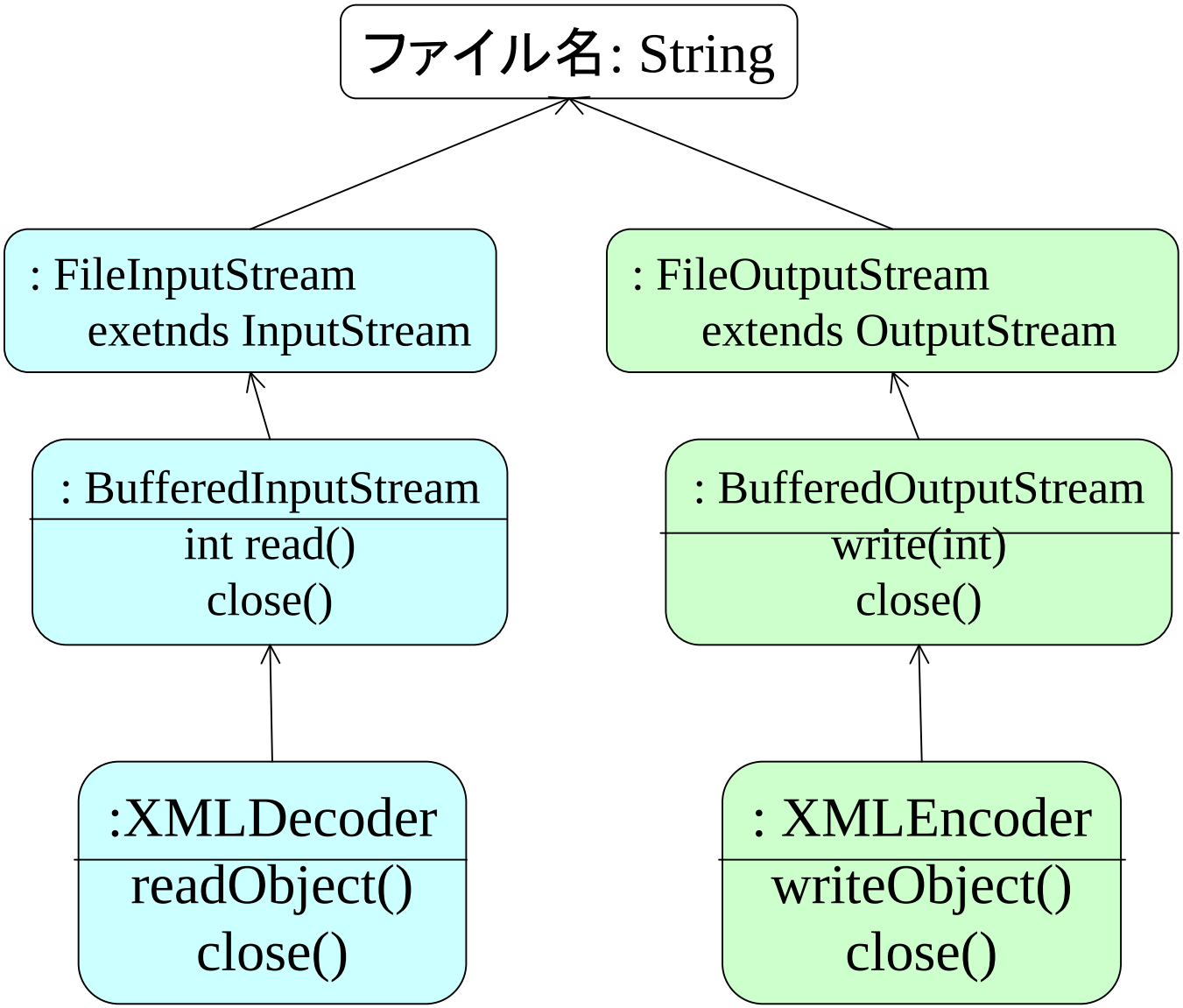
例

属性名 abc

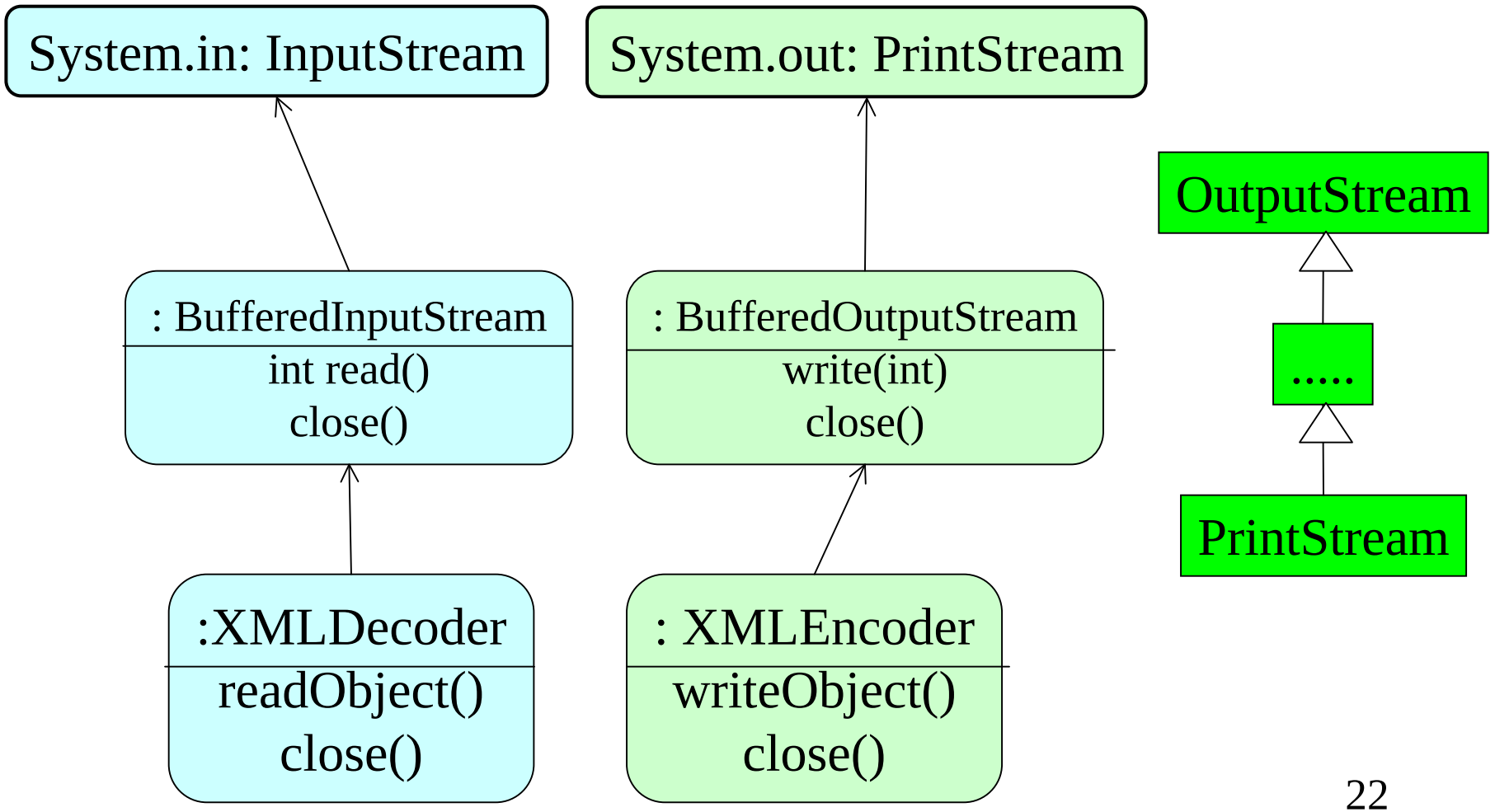
setter ⇒ setAbc

getter ⇒ getAbc

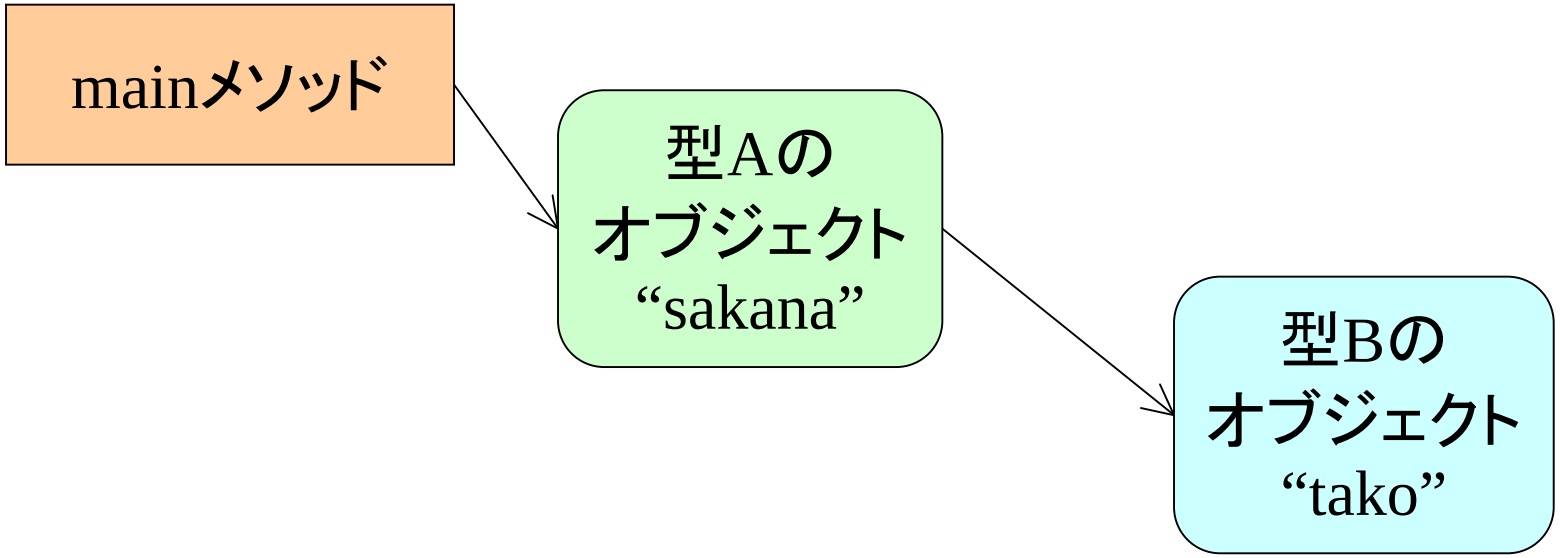
XMLEncoder/Decoderの連鎖



無論，標準IOからもできます



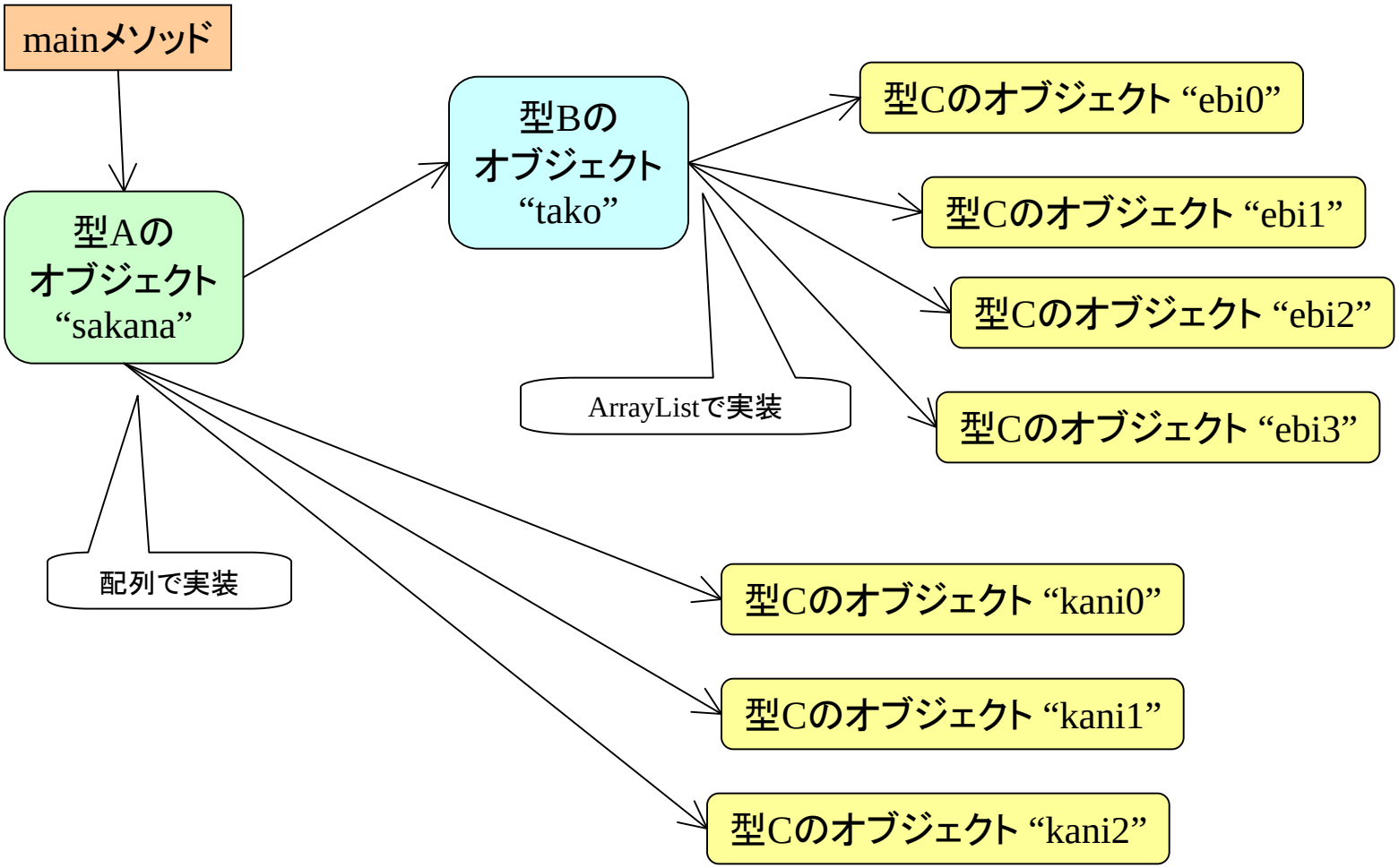
例題(XMLEncText1)の構造



Aのオブジェクトを保存すれば,
Bも自動的に保存される.

たたし，A, Bそれぞれ永続化の条件を満たす場合.

例題(XMLEncText2)の構造

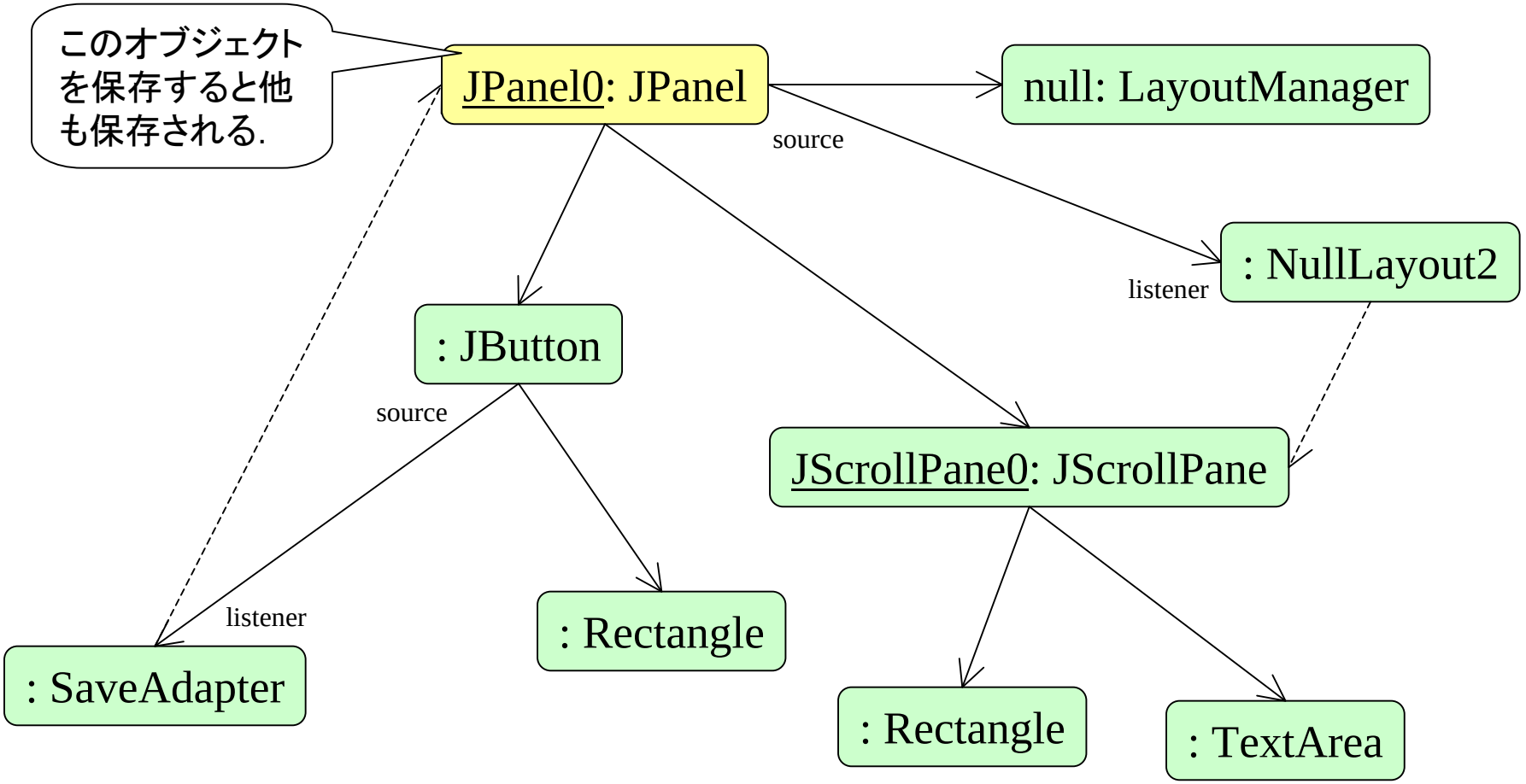


配列およびArrayListに対応している.

Swingの永続化

- 別に自分で保存形式をきめてもいい.
- しかし，XMLEncoder等を使うと凄く簡単に保存復元ができる.
- 特に，比較的トップレベルのJPanelから保存してしまえば，その上に乗っかっている部品もいもづるに保存される.
- 永続化条件を満たせば，Listener等も保存される.
- 例題(XMLEncDecSwing2)を参照.

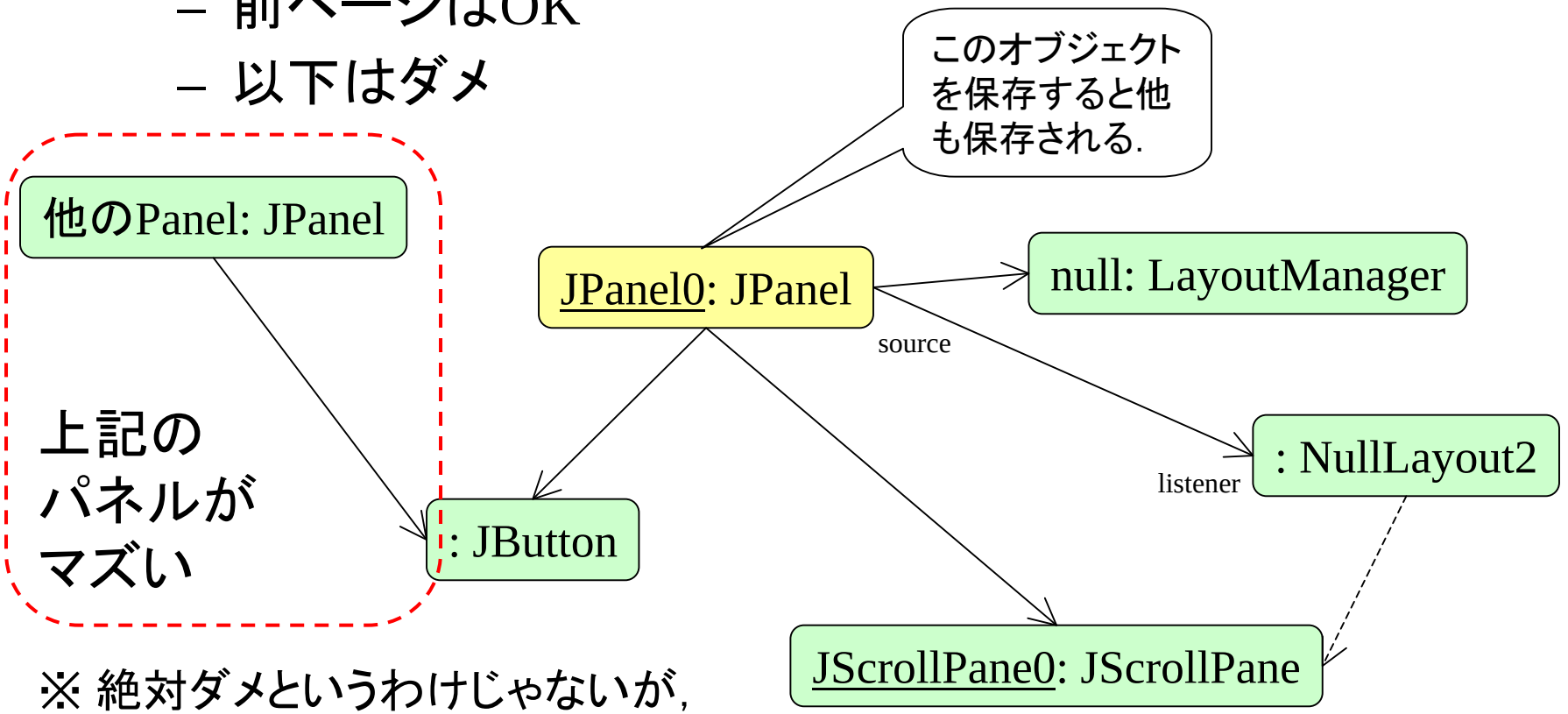
例題の永続化された構造



他のオブジェクトから参照されるオブジェクトには名前(id)がある.

永続化構造の条件

- 保存するオブジェクト以下の参照関係が閉じていること.
  - 前ページはOK
  - 以下はダメ

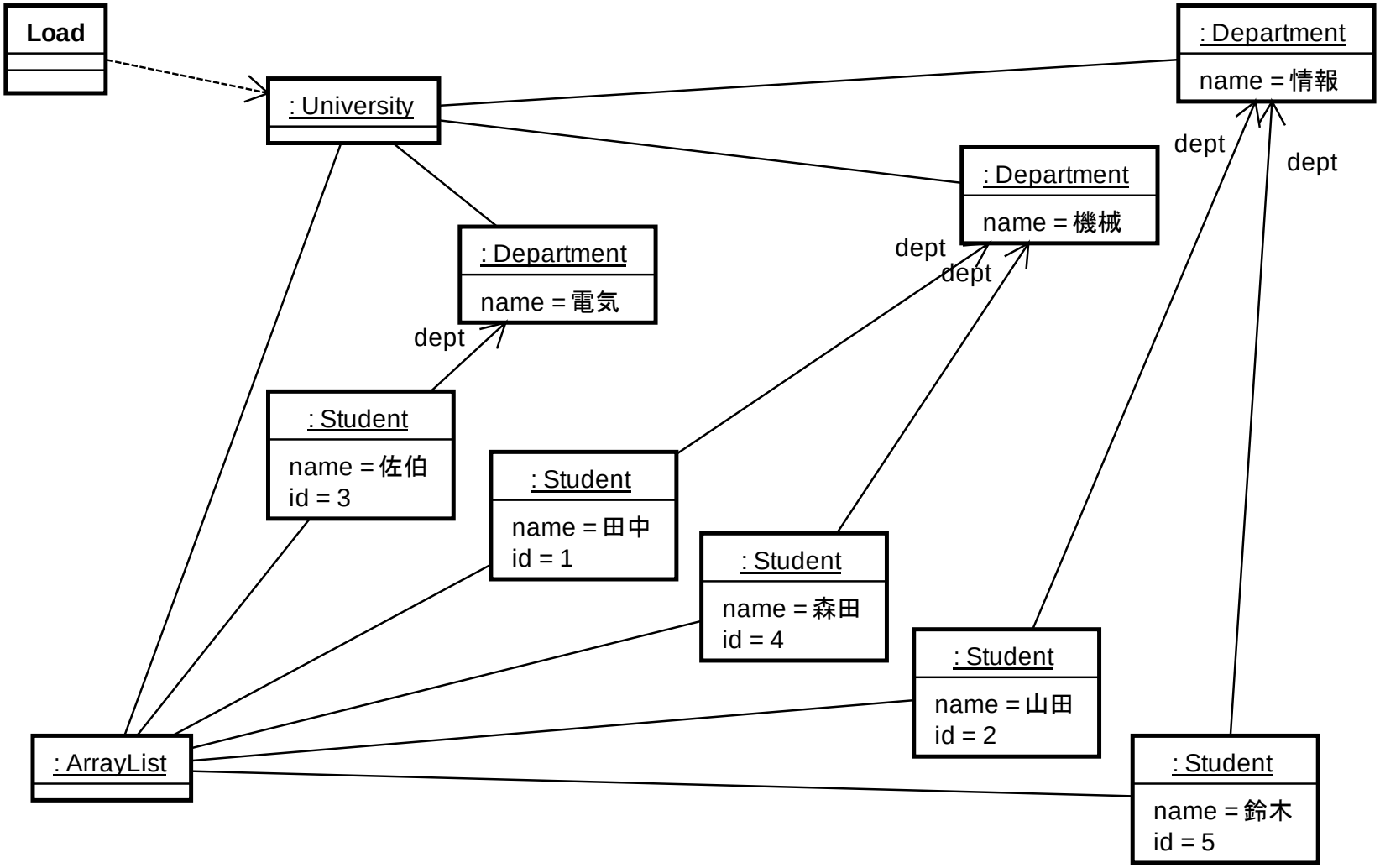


※ 絶対ダメというわけじゃないが, 復元のプログラムが複雑になる.

例題 XMLEncDecArrayCollection1

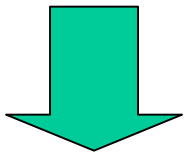
- Genericsに基づくCollectionを使ってみた.
- 大学には学部がいくつかあり，学生が幾人かいる.
- 学部は簡単に変化しないので，配列で実装.
- 学生は流動的なのでArrayListで実装.
- out.xml というファイルにデータを保存するように実装.

例題のメモリーイメージ



XMLEncoder/Decoder の利点

- インスタンスをプログラマがテキスト列等に展開・復元する手間が不要です.
- XML表現なので他の言語(perlやc等)でも作成したデータを容易に扱えます.
  - ただし，XML対応のAPIが整備されている場合.
- 複雑ではありますが，テキストファイルなので，ファイルの中身を目視確認できます.



- この授業の演習ではなるだけXMLEncoder/Decoderを使ってみてください.