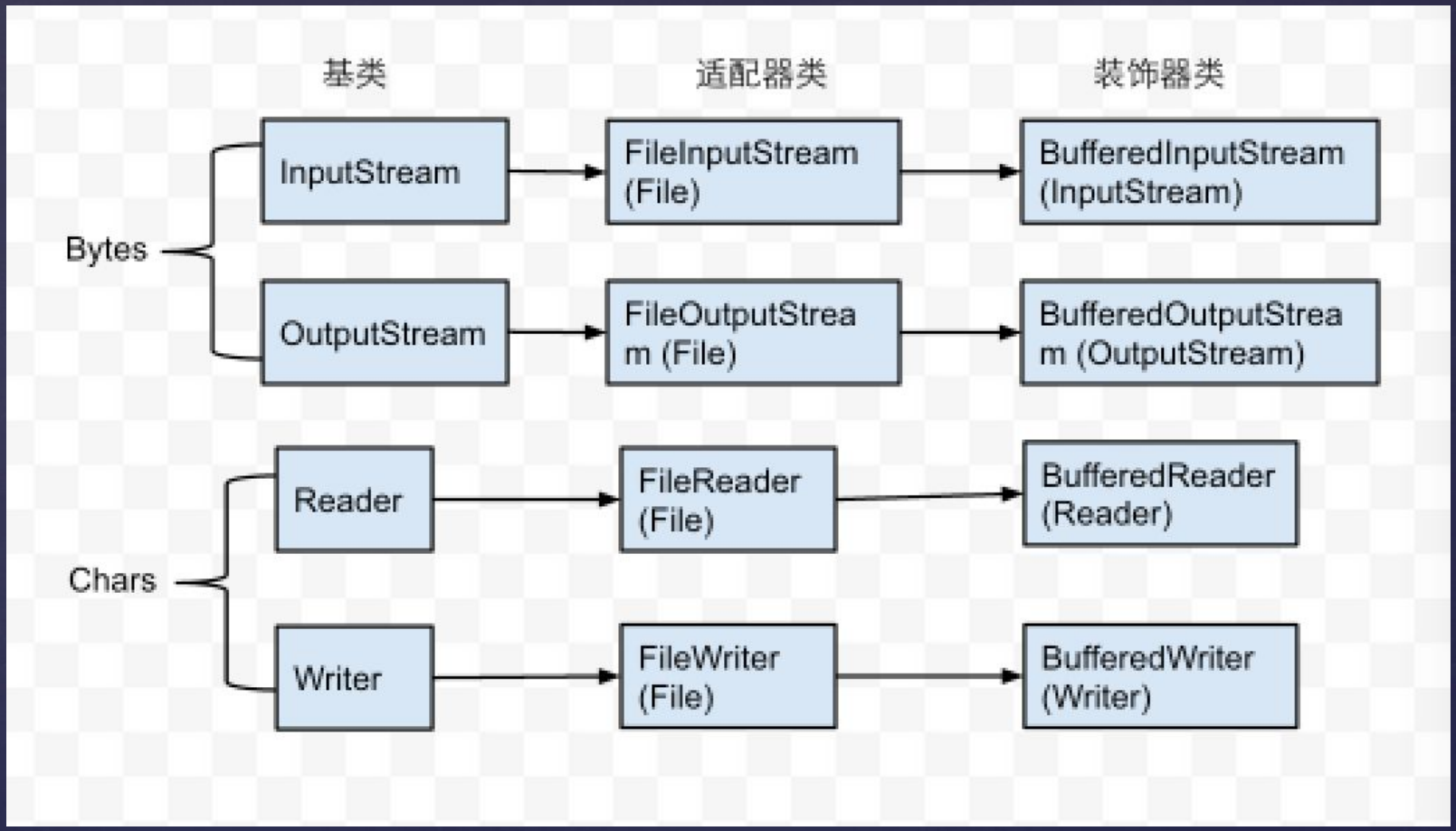
I/O & Encoding & Cache

什么是 I/O

- input & output
- 硬件
- 文件
- 网络
- …

- OIO / NIO

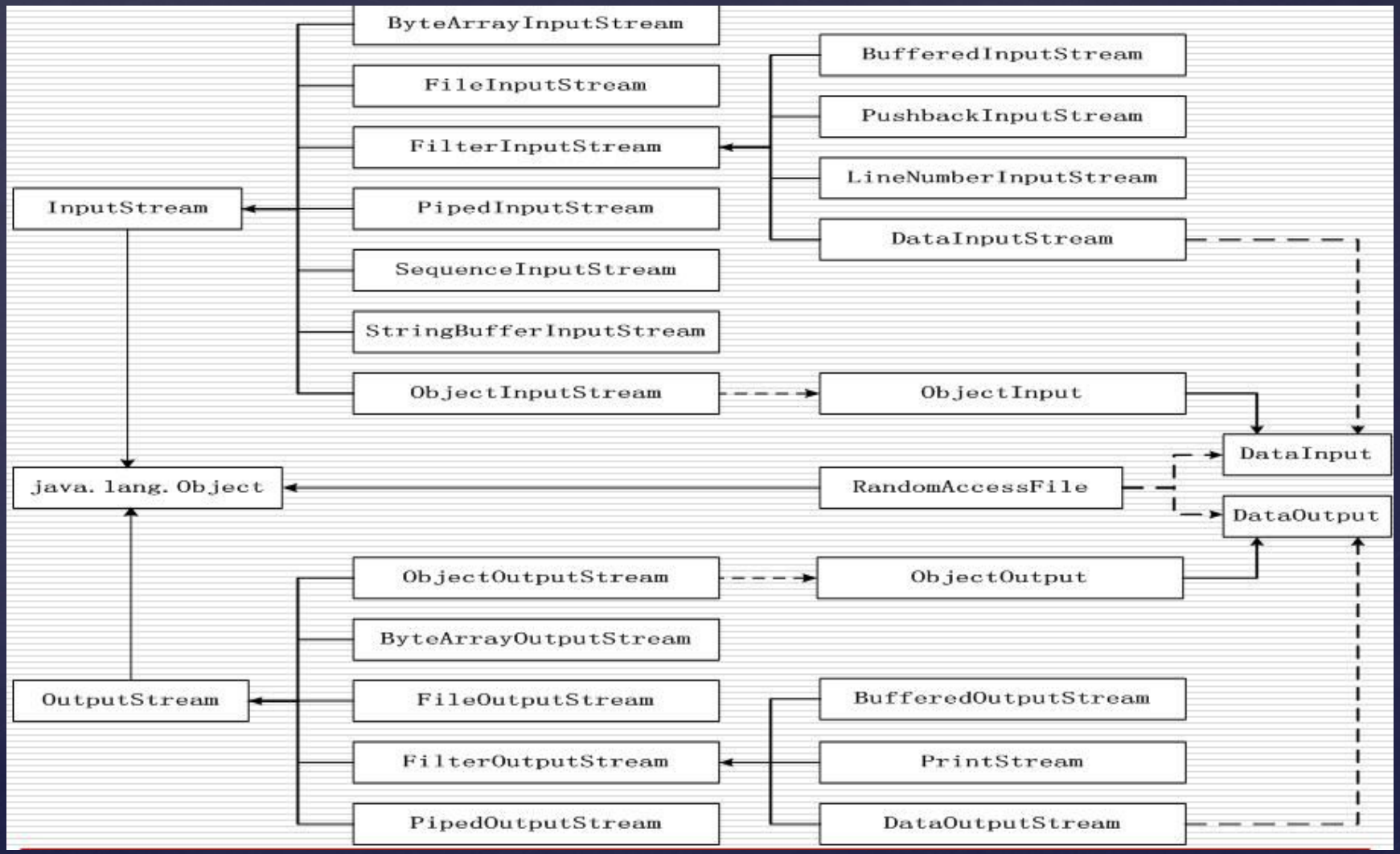
一个简化的结构图 (90%+使用率)



上代码

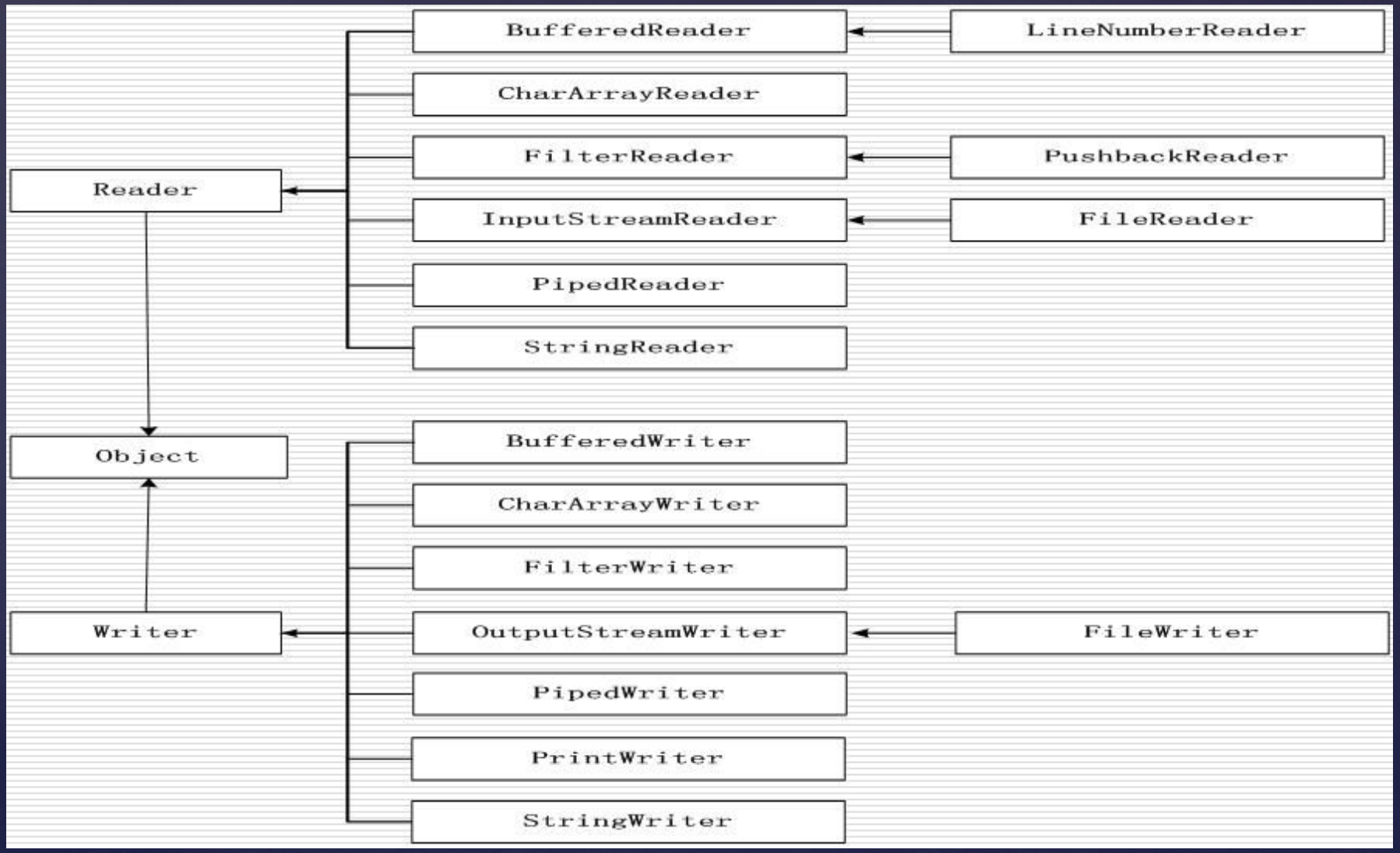
文件 IO 体系 – Input/Output Stream - Byte

基类　　装饰器/适配器



文件 IO 体系 – Input/Output Reader - Char

基类　　装饰器/适配器



基类

InputStream

OutputStream

Reader

Writer

原始流处理器接收Byte数组对象,String对象,FileDescriptor对象将其适配成InputStream,以供其他装饰器使用,他们都继承自InputStream包括如下几个:

ByteArrayInputStream: 接收Byte数组为流源,为多线程通信提供缓冲区操作功能

FileInputStream: 接收一个File作为流源,用于文件的读取

PipedInputStream: 接收一个PipedOutputStream,与PipedOutputStream配合作为管道使用

StringBufferInputStream: 接收一个String作为流的源(已弃用)

装饰器:

链接流处理器可以接收另一个流处理器(InputStream,包括链接流处理器和原始流处理器)作为源,并对其功能进行扩展,所以说他们是装饰器.

1) FilterInputStream继承自InputStream,是装饰器的父类, FilterInputStream内部也包含一个InputStream, 这个InputStream就是被装饰类--一个原始流处理器, 它包括如下几个子类:

- BufferedInputStream: 用来将数据读入内存缓冲区,并从此缓冲区提供数据
- DataInputStream: 提供基于多字节的读取方法,可以读取原始数据类型(Byte, Int, Long, Double等等)
- LineNumberInputStream: 提供具有行计数功能的流处理器
- PushbackInputStream: 提供已读取字节"推回"输入流的功能

2) ObjectInputStream: 可以将使用ObjectOutputStream写入的基本数据和对象进行反串行化

3) SequenceInputStream: 可以合并多个InputStream原始流,依次读取这些合并的原始流

NIO

优点

- 异步 – read/write
- 单线程对多 channel (文件) - selector

场景

- 高并发

Guava IO

Source / Sink:

- Reading: ByteSource CharSource
- Writing: ByteSink CharSink
- 无需关闭
- 便捷方法

- Files:

readLines() 实现 (上代码)

File Encoding

me 是什么意思 ?

数据在文件中是如何存储的?

- enca –L chinese file
- hexdump –C file

存储的编码/读取的编码
vim
write ++enc=gbk gbk_test2
e! ++enc=utf8

乱码: 读的编码 != 存储编码

Java encoding

Default Encoding : utf 16

上代码

问题 : 为什么网页会乱码

Cache

什么是缓存?
为了解决性能瓶颈 (计算, IO, 文件, 数据库, 网络) 的高速存储结构

缓存的分类:
单机, 分布式

缓存的结构:
上图

为什么不只用缓存?
内存与硬盘的关系

使用情景

读缓存
- 缓存命中
- 上代码
- 定时拉取 或 后台更新后立即更新到缓存 (通过数据库事务保证顺序)

写缓存
- 生产者消费者队列, 减少 IO 次数, 批处理
- 定时, 定量刷
- 上代码

Guava Cache

为什么使用

- 默认生成策略
- 插入回调方法
- 删除回调方法
- 容量失效策略
- 时间失效策略 (钩子清除策略)
- 权重失效策略
- 定时刷新策略 (同步)

- 上代码

分布式缓存

单机缓存存在的问题: 同步问题

分布式缓存: redis, memcached

作业

1. 使用 Guava 设计一个缓存, 从文件 cache.txt (编码 GBK) 中读入缓存数据, 文件第一列为 key, 第二列为 value
2. 缓存 value 每经过 10 s 尾部加一个 "#"
3. 缓存最多只能存储 10 个 key-value
4. 缓存失效时需要记录日志
5. 不存在的缓存返回 "No such value"
6. 将缓存的 key-value 对调后存入文件 result.txt 并使用 UTF-8 编码

要求:

7. 日志记录规范
8. 文件不能乱码
9. 全部使用课上提到的 Guava API 完成

题外话

关于 intern