Java I/O

Università di Modena e Reggio Emilia

Prof. Nicola Bicocchi (nicola.bicocchi@unimore.it)

ING

Stream

- Random access abstraction
  - java.io.RandomAccessFile
- Stream-oriented abstraction
  - I/O operations work in the same way with all kinds of streams. For example, a stream can be: standard input, output, error, files, data from/to memory or a pipe, a network connection

Stream

- Abstract classes Reader Writer
  - For reading and writing streams of chars (Unicode character 16 bit)
- Abstract classes InputStream OutputStream
  - For reading and writing streams of bytes (8 bit)
- Package java.io
- Exceptions are subclasses of java.io.IOException

java.io.Reader/InputStream

- int read()
  - Read a single char
- int read(char[] buffer)
  - Read chars into an array
- long skip(long n)
  - Skip n chars
- void close()
  - Close the stream

- int read()
  - Reads a single byte (packed into an int)
- int read(byte[] buffer)
  - Reads bytes into an array
- long skip(long n)
  - Skips n bytes
- void close()
  - Close the stream

java.io.Writer/OutputStream

- void write(int c)
  - Write a single char
- void write(char[] buffer)
  - Write an array of chars
- void write(String str)
  - Write a string
- void flush()
  - Flush the stream
- void close()
  - Close the stream, flushing it first

- void write(int b)
  - Write a single byte
- void write(byte[] buffer)
  - Write an array of bytes
- void flush()
  - Flush the stream
- void close()
  - Close the stream, flushing it first

java.io.PrintStream

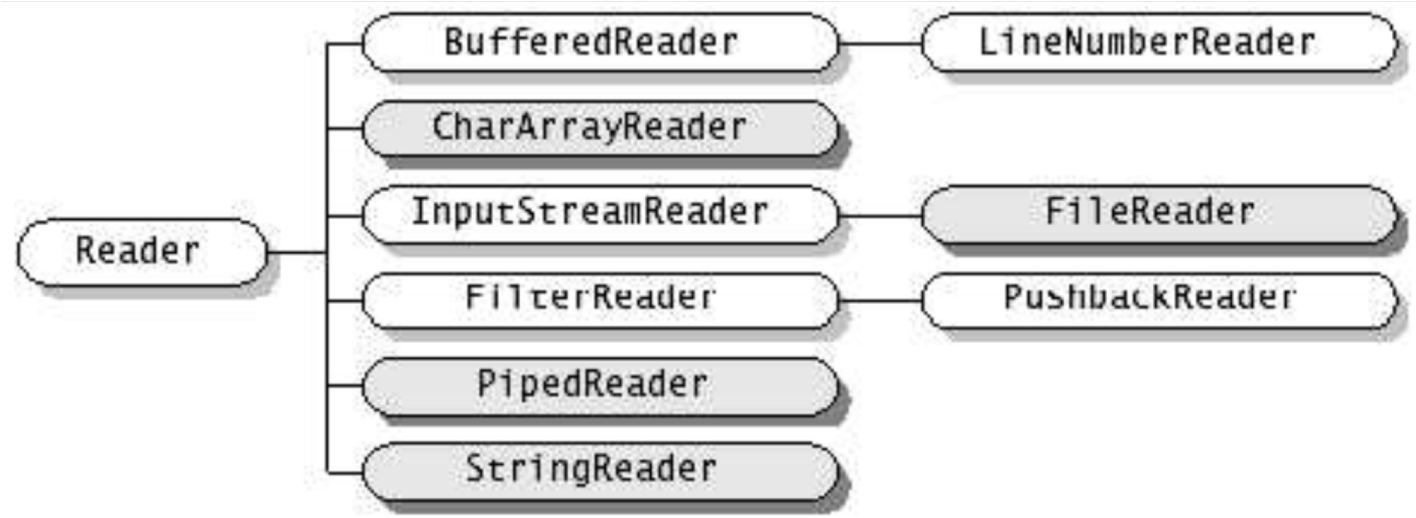
- A PrintStream adds functionality to a generic outputstream. Specifically the ability to print convenient representations of various data.
- Adds methods like print(), println(), printf(), format() for string formatting
- Unlike other output streams, a PrintStream never throws an IOException; instead, exceptional situations merely set an internal flag that can be tested via the checkError method.

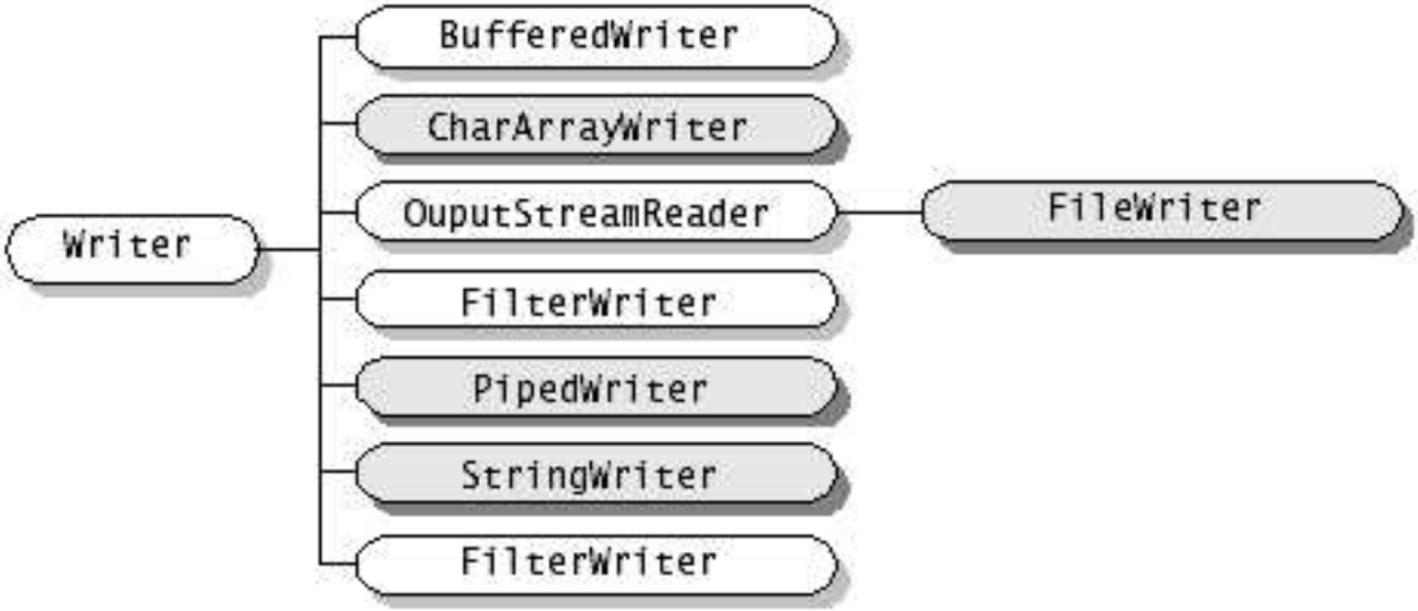
System.in and System.out

- System defines standard input, output and error as streams:

```
class System {
   static InputStream in;
   static PrintStream out;
    static PrintStream err;
   …
}
```

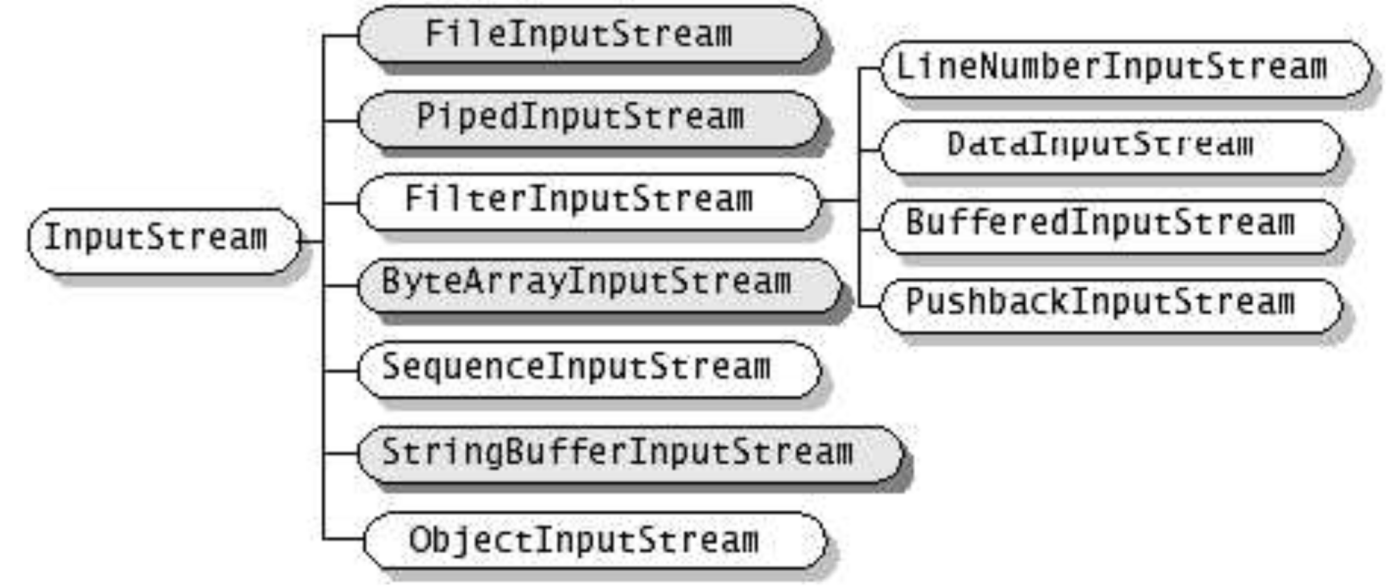
Reader and Writer

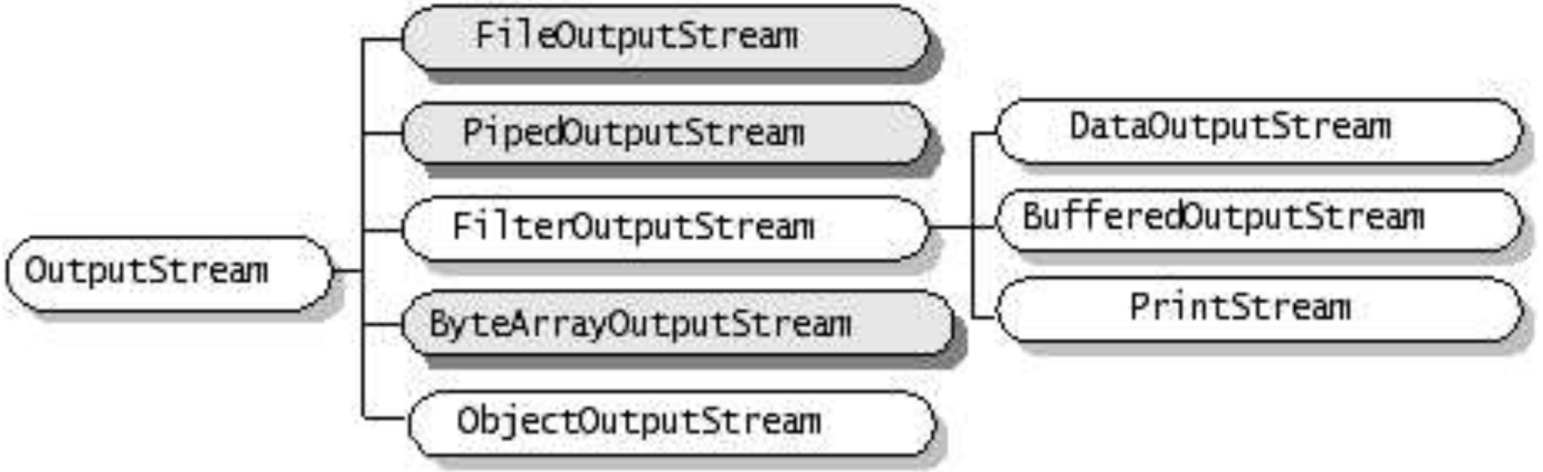




ING

InputStream and OutputStream





ING

Read/Write files

- java.io.FileReader
- java.io.FileWriter
  - R/W chars from files
- java.io.FileInputStream
- java.io.FileOutputStream
  - R/W bytes from files
- java.io.File
  - represents filename and pathname

Read/Write pipes

- java.io.PipedReader
- java.io.PipedWriter
  - R/W chars from pipes
- java.io.PipedInputStream
- java.io.PipedOutputStream
  - R/W bytes from pipes

Read/Write in memory

- java.io.CharArrayReader
- java.io.CharArrayWriter
  - R/W chars from/to arrays of chars
- java.io.StringReader
- java.io.StringWriter
  - R/W chars from/to Strings
- java.io.ByteArrayInputStream
- java.io.ByteArrayOutputStream
  - R/W bytes from/to arrays of bytes

Buffered Streams

- Buffered streams add buffering functionality. The manual alternative is to use read(int[] buf) or read(char[] buf).
  - BufferedInputStream
  - BufferedOutputStream
  - BufferedReader
  - BufferedWriter
  - Examples:
    - `BufferedInputStream(InputStream i)`
    - `BufferedOutputStream(OutputStream o)`
    - `BufferedReader(Reader r)`
    - `BufferedWriter(Writer w)`

Interpreted Streams

- Translates primitive types in standard format (UTF-8) on a stream
- DataInputStream(InputStream i)
  - readByte(), readChar(), readDouble(), readFloat(), readInt(), readLong(), readShort()
- DataOutputStream(OutputStream o)
  - writeByte(), writeChar(), writeDouble(), writeFloat(), writeInt(), writeLong(), writeShort()

ING

java.io.File

- An abstract representation of file and directory pathnames. Provides access to some file attributes (length, rights, etc.) and a mapping between File and String
- methods:
  - exists(), isFile(), isDirectory(), isHidden(), length(), canRead(), canWrite(), canExecute(), getPath()…
- File f = new File("/etc/passd").
  - Is this truly portable?

java.io.File

- Platform dependent
  - `File f = new File("tmp/abc.txt");`
- Platform independent
  - `File f = new File(“tmp” + File.separator + "abc.txt");`

@See Java System Properties

@See class File static attributes

java.nio.file.Files

- This class consists exclusively of static methods for manipulating files or directories (copy, move, delete, rename, set ownership, set attributes)
- In most cases, the methods defined here delegate to the associated file system provider to perform actual operations.

java.util.StringTokenizer

- StringTokenizer
  - Divides a String in tokens (using delimiters)
  - Blank is the default delimiter
  - Does not distinguish identifiers, numbers, comments, quoted strings
- StreamTokenizer
  - Divides a Stream in tokens
  - More sophisticated, recognizes identifiers, comments, quoted string, numbers
  - Use symbol table and flags

Deep and Shallow copies

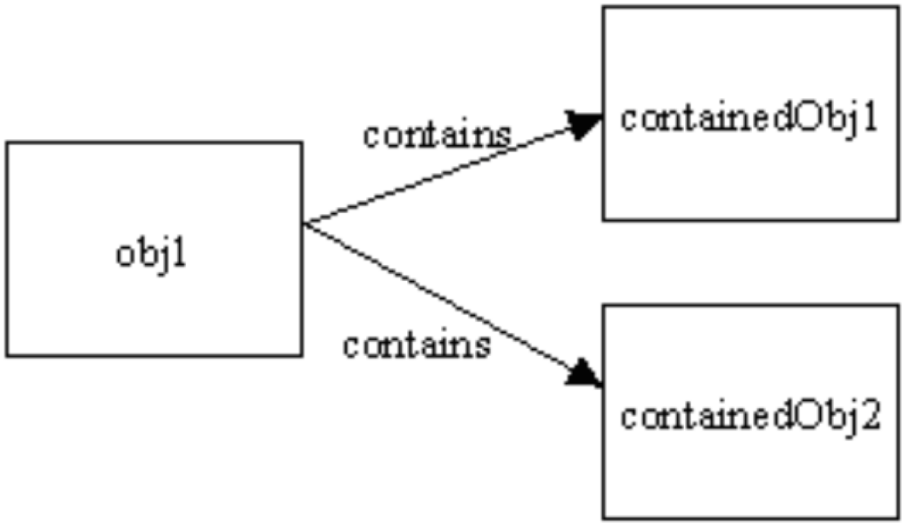


Figure 1. The original state of obj1

ING

Deep and Shallow copies

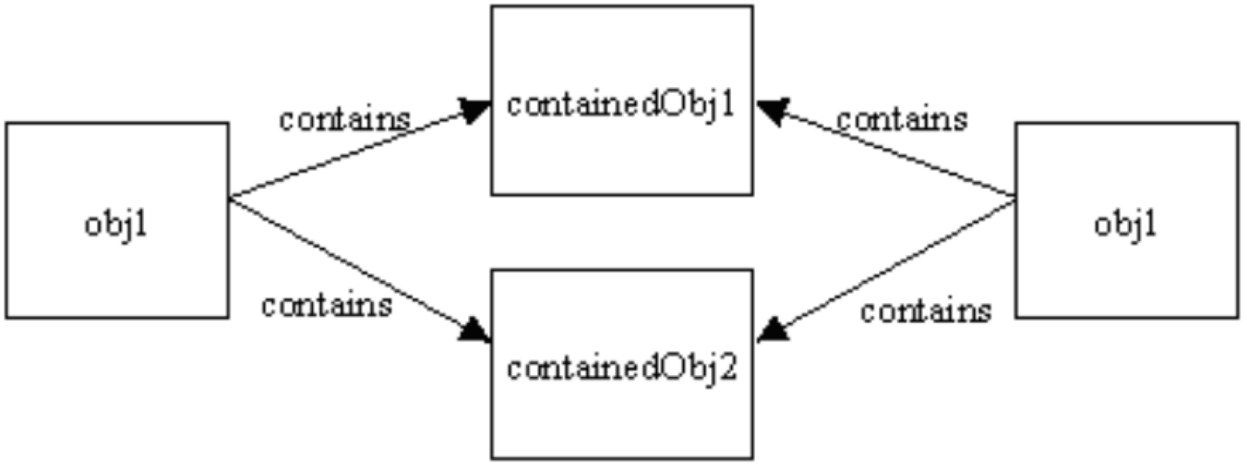


Figure 2. After a shallow copy of obj1

Deep and Shallow copies

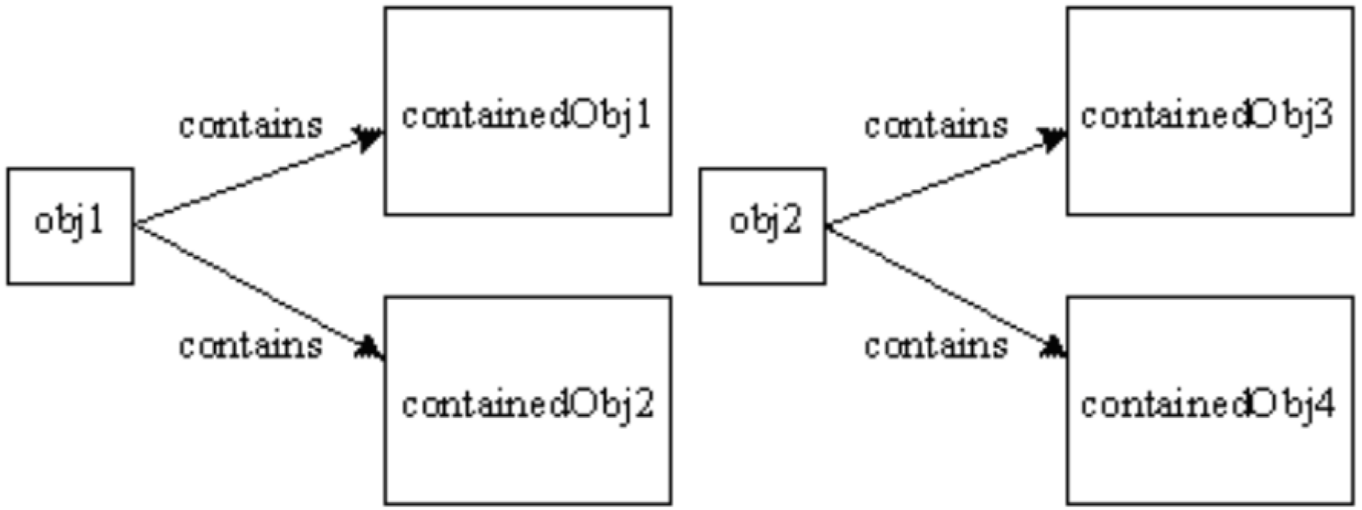


Figure 3. After a deep copy of obj1

ING

Serialization

- Read / write of an object imply:
  - read/write attributes (and optionally the type) of the object
  - Correctly separating different elements
  - When reading, create an object and set all attributes values
- These operations (serialization) are automated by
  - ObjectInputStream
  - ObjectOutputStream

Serialization

- Methods to read/write objects are:
  - void writeObject(Object)
  - Object readObject()
- ONLY objects implementing interface Serializable can be serialized. Serializable is an empty interface. It is used to avoid serialization of objects, without the permission of the class developer

Serialization (deep copy, no duplicates)

- An ObjectOutputStream saves automatically all objects referred by its attributes
  - objects serialized are numbered in the stream
  - references are saved like ordering numbers in the stream
- If I save 2 objects pointing to a third one, this is saved just once
  - Before saving an object, ObjectOutputStream checks if it has not been already saved
  - Otherwise it saves just the reference (as a number)

Serialization, type recovery

- When reading, an object is created
- … but which is its type?
- Down casting to the exact type is useful only to send specific messages
- A viable solution could be down casting to a common ancestor

ING