Java I/O

Università di Modena e Reggio Emilia
Prof. Nicola Bicocchi (nicola.bicocchi@unimore.it)

ING

Java I/O

- Stream
- Buffer
- File
- StringTokenizer, StreamTokenizer
- Serialization

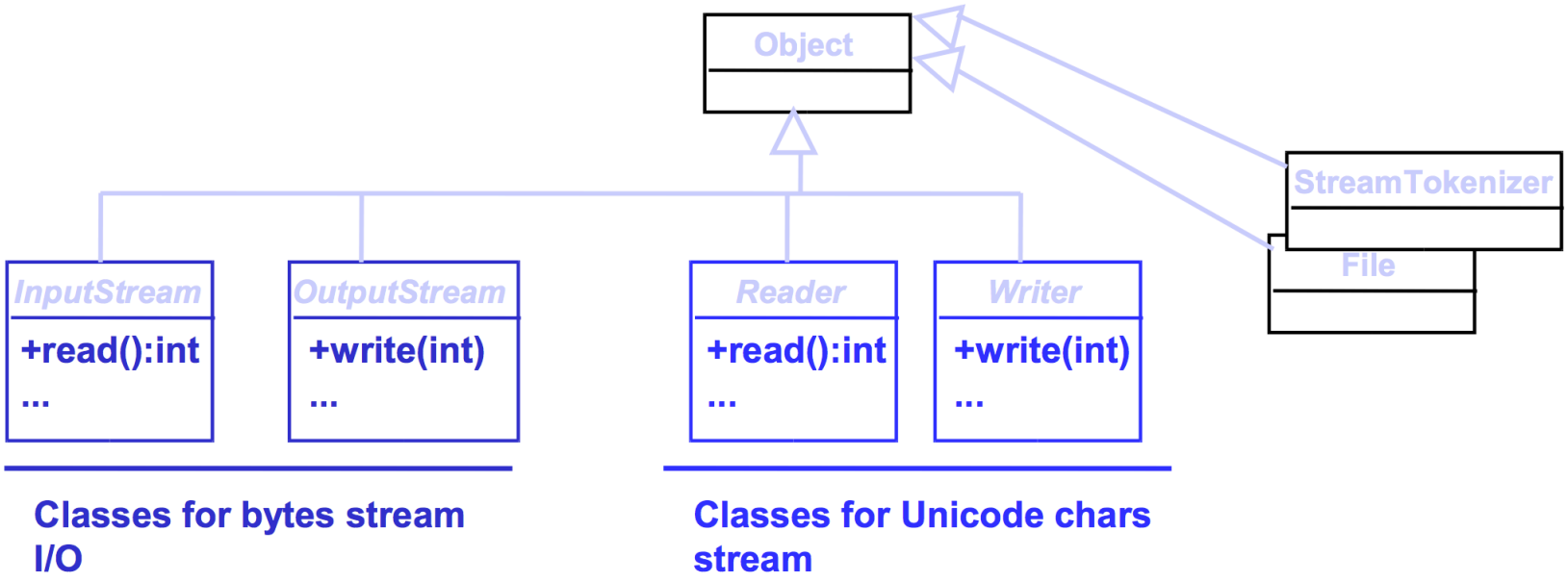
Stream

- All I/O operations rely on the abstraction of STREAM ("bytes flow")
- A stream can be:
  - A file on the disk
  - standard input, output, error
  - A network connection
  - A data-flow from/to whichever hardware device
- I/O operations work in the same way with ALL kinds of stream

Stream

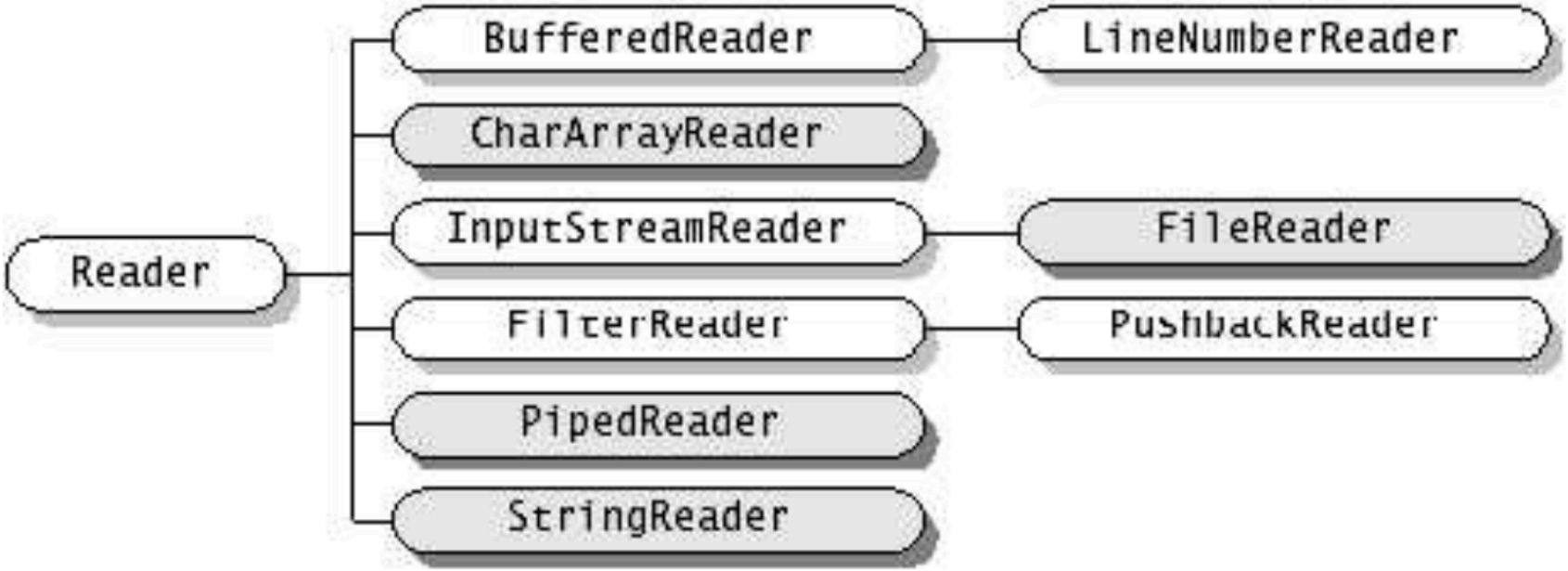
- Reader Writer
  - stream of chars (Unicode Chars 16 bit)
    - All characters
- InputStream OutputStream
  - stream of bytes (8 bit)
    - Binary data, sounds, images
- package java.io
- All related exceptions are subclasses of IOException

Base classes in java.io



ING

java.io.Reader

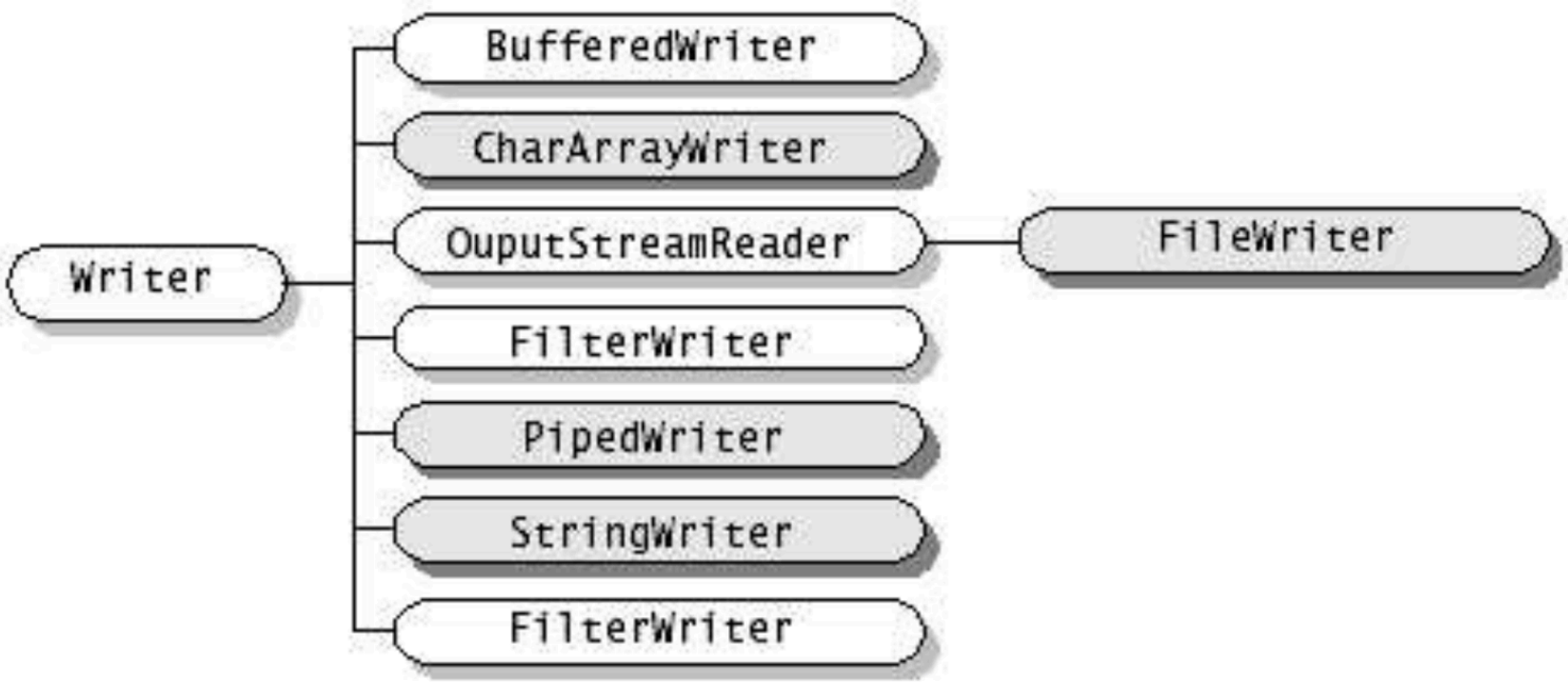


ING

java.io.Reader

- void close()
  - Close the stream.
- int read()
  - Read a single character: -1 when end of stream. Block until char is available, I/O error, end of stream.
- int read(char[] cbuf)
  - Read characters into an array.
- abstract int read(char[] cbuf, int off, int len)
  - Read characters into a portion of an array.
- boolean ready()
  - Tell whether this stream is ready to be read.
- void reset()
  - Reset the stream.
- long skip(long n)
  - Skip characters.

java.io.Writer

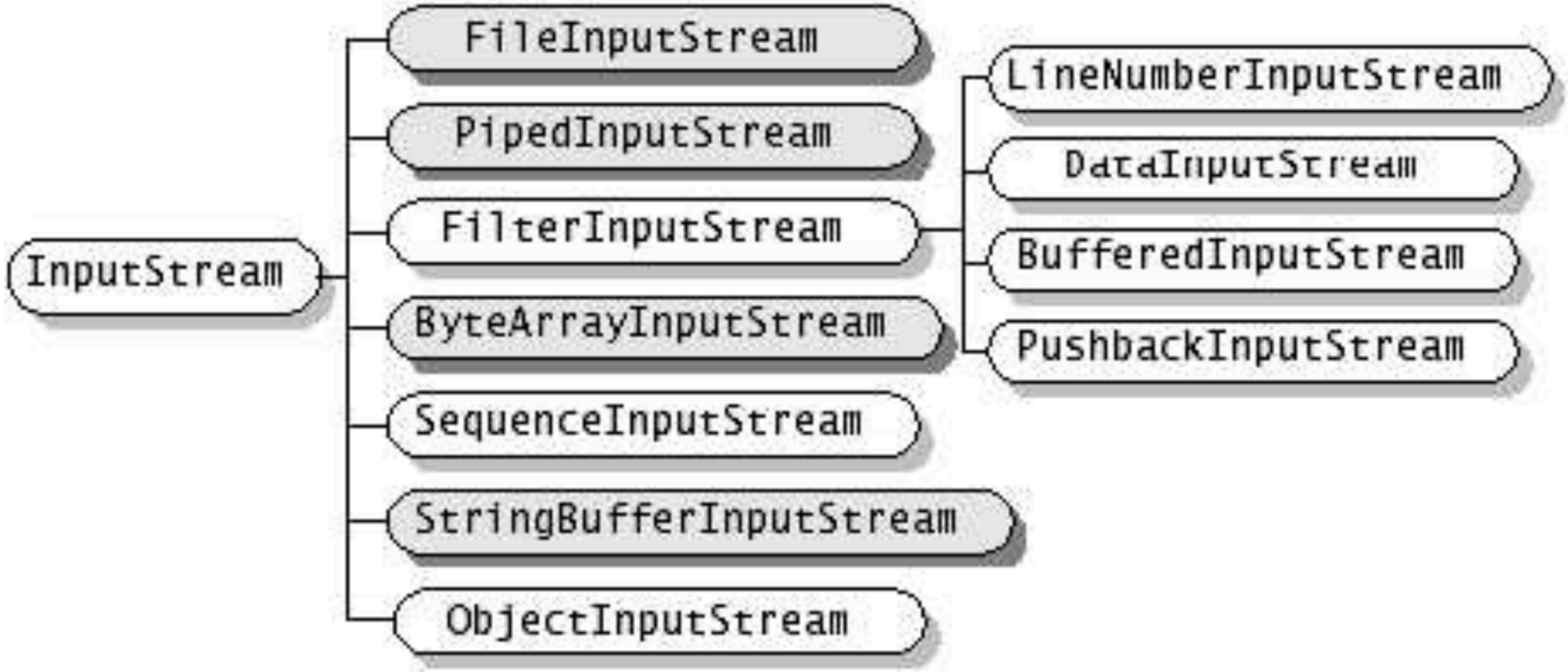


ING

java.io.Writer

- close()
  - Close the stream, flushing it first.
- abstract void flush()
  - Flush the stream.
- void write(char[] cbuf)
  - Write an array of characters.
- abstract void write(char[] cbuf, int off, int len)
  - Write a portion of an array of characters.
- void write(int c)
  - Write a single character.
- void write(String str)
  - Write a string.
- void write(String str, int off, int len)
  - Write a portion of a string.

java.io.InputStream

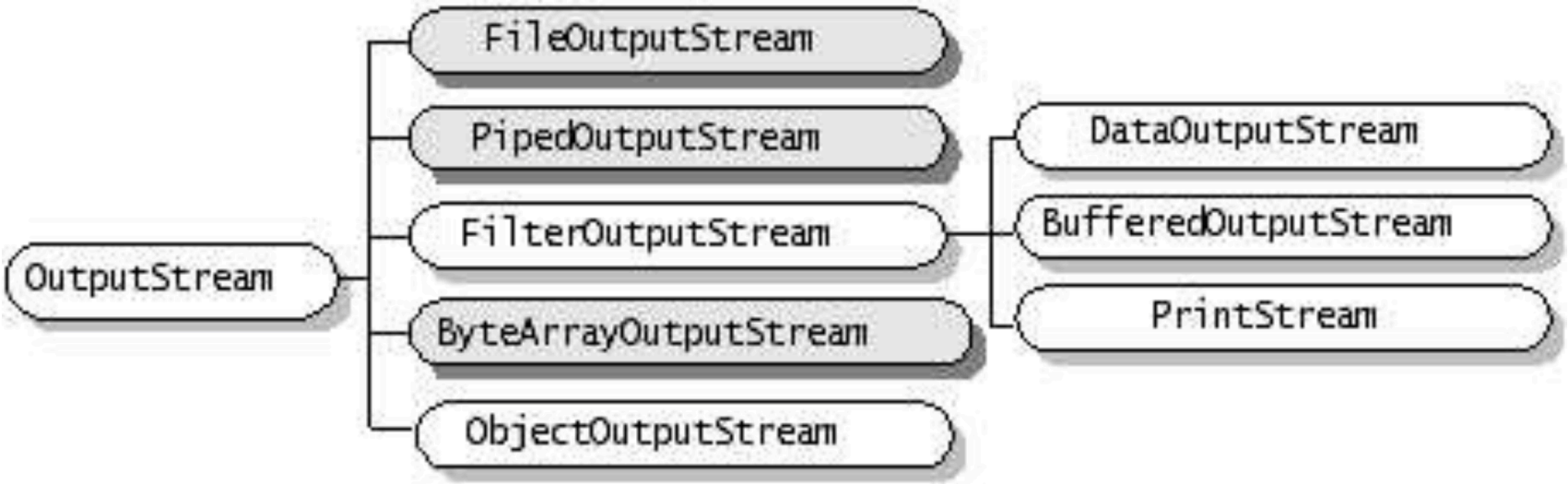


ING

java.io.InputStream

- int available()
  - Returns the number of bytes that can be read (or skipped over) from this input stream without blocking by the next caller of a method for this input stream.
- void close()
  - Closes this input stream and releases any system resources associated with the stream.
- abstract int read()
  - Reads the next byte of data from the input stream.
- int read(byte[] b)
  - Reads some number of bytes from the input stream and stores them into the buffer array b.
- int read(byte[] b, int off, int len)
  - Reads up to len bytes of data from the input stream into an array of bytes.
- void reset()
  - Repositions this stream to the position at the time the mark method was last called on this input stream.
- long skip(long n)
  - Skips over and discards n bytes of data from this input stream.

java.io.OutputStream



ING

java.io.OutputStream

- void close()
  - Closes this output stream and releases any system resources associated with this stream.
- void flush()
  - Flushes this output stream and forces any buffered output bytes to be written out.
- void write(byte[] b)
  - Writes b.length bytes from the specified byte array to this output stream.
- void write(byte[] b, int off, int len)
  - Writes len bytes from the specified byte array starting at offset off to this output stream.
- abstract void write(int b)
  - Writes the specified byte to this output stream.

System.in and System.out

- System defines default input and output streams

```
class System {
    static InputStream in;
    static PrintStream out;   // see after
…
}
```

Stream specializations

- Memory (source/destination)
- Pipe (source/destination)
- File (source/destination)

- Buffered (functionality)
- Printed (functionality)
- Interpreted (functionality)

Read/Write in memory

- CharArrayReader
- CharArrayWriter
  - R/W char or byte from/to array in memory
- ByteArrayInputStream
- ByteArrayOutputStream
  - R/W char or byte from/to array in memory
- StringReader
- StringWriter
  - R/W chars from/to String

Read/Write of pipes

- Pipes are used in inter-process communication
- PipedReader
- PipedWriter
  - R/W chars from pipe
- PipedInputStream
- PipedOutputStream
  - R/W bytes from pipe

Read/Write of files

- FileReader
- FileWriter
  - R/W char from file
- FileInputStream
- FileOutputStream
  - R/W byte from file
- File
  - handles filename and pathname

Buffered Streams

- BufferedInputStream
- BufferedOutputStream
- BufferedReader
- BufferedWriter
- Examples:
  - BufferedInputStream(InputStream i)
  - BufferedInputStream(InputStream i, int size)

Printed Streams

- PrintStream(OutputStream o)
  - print() println() for primitive types and String
  - Do not throw IOException, but it sets a bit, to be checked with method checkError()

- System defines out and err

```
class System {
    static PrintStream out, err;
    …
}
```

Interpreted Streams

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

File

- abstract pathname
  - directory, file, file separator
  - absolute, relative
- convert abstract pathname <--> string
- methods:
  - create() delete() exists() , mkdir(), getName() getAbsolutePath(), getPath(), getParent(), isFile(), isDirectory(), isHidden(), length(), listFiles(), renameTo()

Example: text file copy

```
import java.io.*;
public class Copy {
      public static void main(String[] args) throws IOException {
            File inputFile = new File("farrago.txt");
            File outputFile = new File("outagain.txt");
            FileReader in = new FileReader(inputFile);
            FileWriter out = new FileWriter(outputFile);
            int c;
            while ((c = in.read()) != -1)
                  out.write(c);
            in.close();
            out.close();
}}
```

Conversion byte <--> char

- InputStreamReader
  - byte --> char
- OutputStreamWriter
  - char --> byte

Tokenizers

- StringTokenizer
  - Works on String
  - set of delimiters (blank, “,”, \t, \n, \r, \f)
  - Blank is the default delimiter
  - Divides a String in tokens (separated by delimiters), returning the token
  - hasMoreTokens(), nextToken()
  - Does not distinguish identifiers, numbers, comments, quoted strings

Tokenizers

- StreamTokenizer
  - Works on Stream (Reader)
  - More sophisticated, recognizes identifiers, comments, quoted string, numbers
  - Use symbol table and flag
  - nextToken(), TT_EOF if at the end

Deep/Shallow Copy

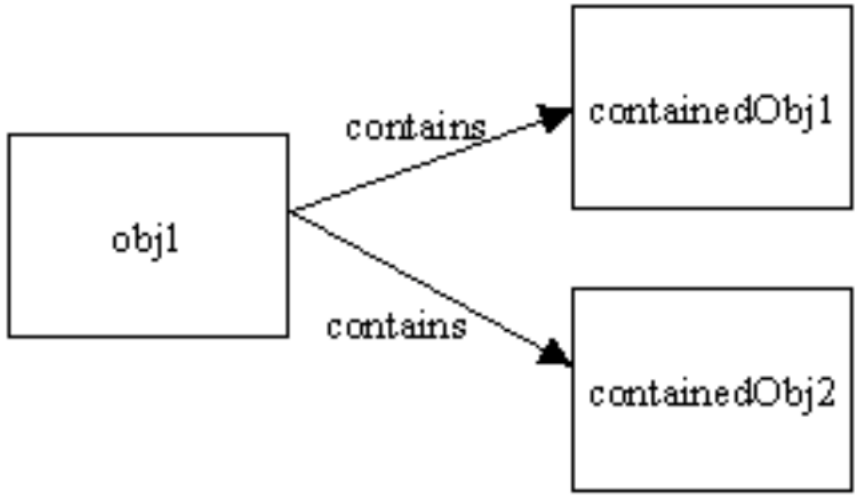


Figure 1. The original state of obj1

Deep/Shallow Copy

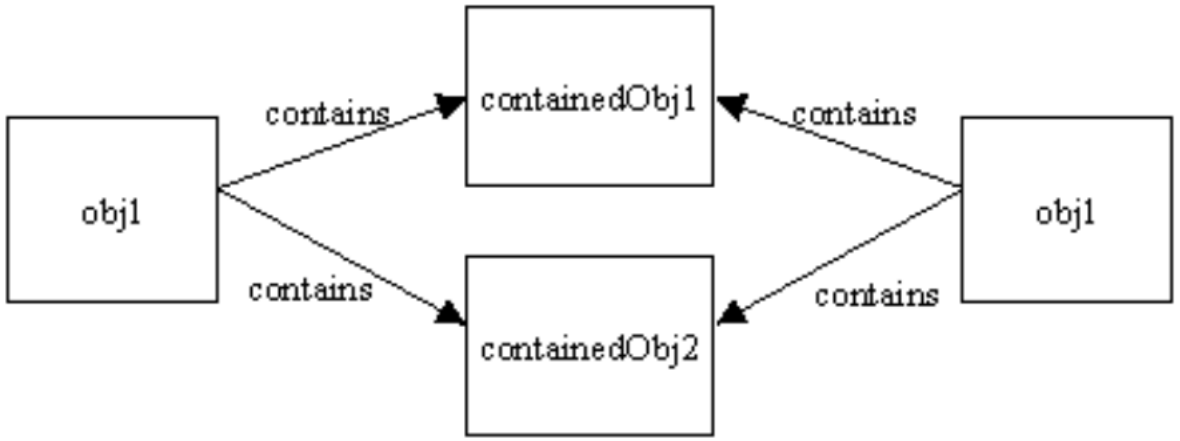


Figure 2. After a shallow copy of obj1

ING

Deep/Shallow Copy

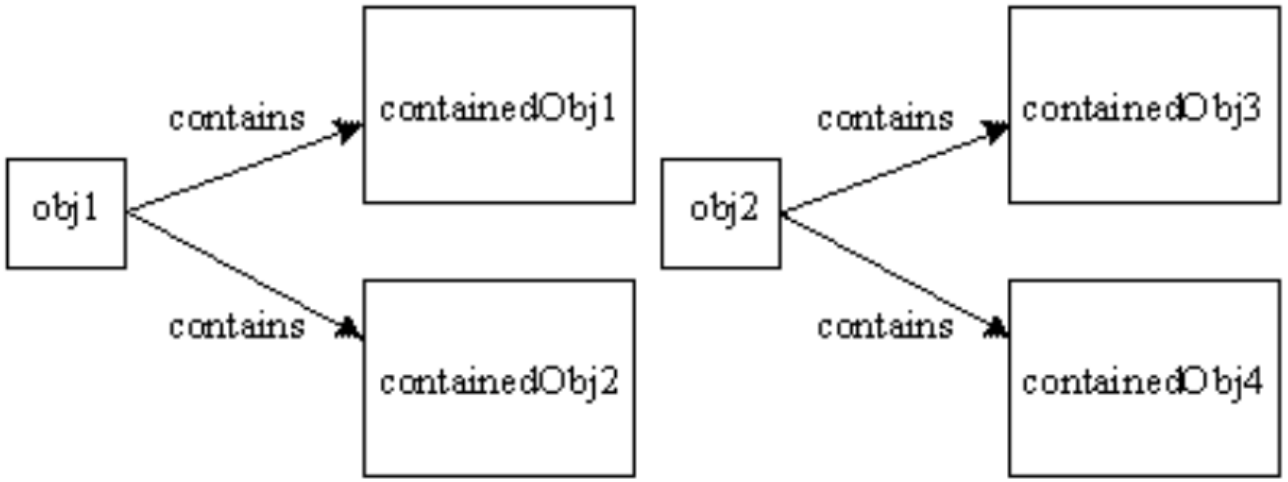


Figure 3. After a deep copy of obj1

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
  - This interface is empty. Just used to avoid serialization of objects, without permission of the class developer

Serialization, deep copy

- 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