Java Programming Language SE – 6
Module 10: I/O Fundamentals

Team Emertxe

ORACLE®
Certified Professional
Java SE 6 Programmer

ΣMERTXE

Objectives

- Write a program that uses command-line arguments and system properties

- Examine the Properties class

- Construct node and processing streams, and use them appropriately

- Serialize and deserialize objects

- Distinguish readers and writers from streams, and select appropriately between them

Oracle Certified Java Programmer
OCJP

Command-Line Arguments

- Any Java technology application can use command-line arguments.

- These string arguments are placed on the command line to launch the Java interpreter after the class name:

  java TestArgs arg1 arg2 "another arg"

- Each command-line argument is placed in the args array that is passed to the static main method:

  public static void main(String[] args)
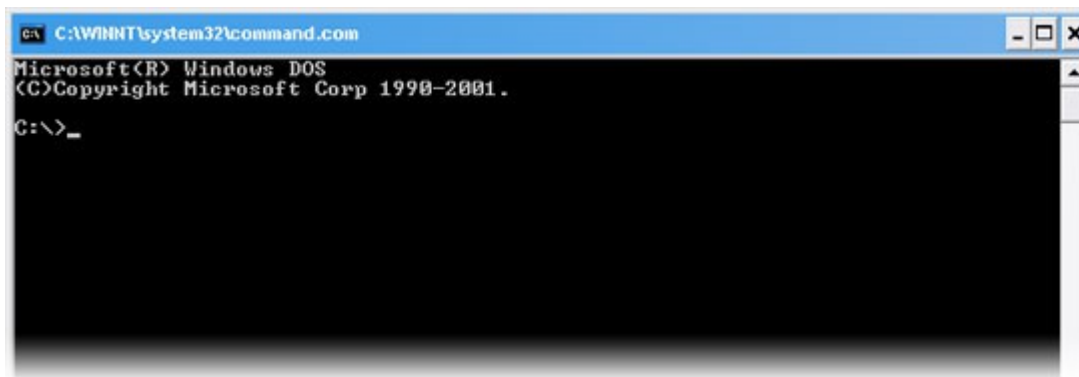
Command-Line Arguments

```
public class TestArgs {

public static void main(String[] args) {

for ( int i = 0; i < args.length; i++ ) {

System.out.println("args[" + i + "] is '" + args[i] + "'");

}}}
```

- Example execution:

java TestArgs arg0 arg1 "another arg"

args[0] is 'arg0'

args[1] is 'arg1'

args[2] is 'another arg'

ΣMERTXE

System Properties

- System properties are a feature that replaces the concept of environment variables (which are platform-specific).

- The System.getProperties method returns a Properties object.

- The getProperty method returns a String representing the value of the named property.

- Use the -D option on the command line to include a new property.

The Properties Class

- The Properties class implements a mapping of names to values (a String-to-String map).

- The propertyNames method returns an Enumeration of all property names.

- The getProperty method returns a String representing the value of the named property.

- You can also read and write a properties collection into a file using load and store.

The Properties Class

```java
import java.util.Properties;

import java.util.Enumeration;

public class TestProperties {

public static void main(String[] args) {

Properties props = System.getProperties();

props.list(System.out);

}

}
```

The Properties Class

The following is an example test run of this program:

java -DmyProp=theValue TestProperties

- The following is the (partial) output:

java.runtime.name=Java(TM) SE Runtime Environment

sun.boot.library.path=C:\jse\jdk1.6.0\jre\bin

java.vm.version=1.6.0-b105

java.vm.vendor=Sun Microsystems Inc.

java.vm.name=Java HotSpot(TM) Client VM

file.encoding.pkg=sun.io

user.country=US

myProp=theValue

ΣMERTXE

I/O Stream Fundamentals

- A stream is a flow of data from a source or to a sink.

- A source stream initiates the flow of data, also called an input stream.

- A sink stream terminates the flow of data, also called an output stream.

- Sources and sinks are both node streams.

- Types of node streams are files, memory, and pipes between threads or processes.

Fundamental Stream Classes

Stream	Byte Streams	Character Streams
Source streams	InputStream	Reader
Sink streams	OutputStream	Writer

ΣMERTXE

Data Within Streams

- Java technology supports two types of streams: character and byte.

- Input and output of character data is handled by readers and writers.

- Input and output of byte data is handled by input streams and output streams:

- Normally, the term stream refers to a byte stream.

- The terms reader and writer refer to character streams.

ΣMERTXE

The InputStream Methods

- The three basic read methods are:

  - int read()

  - int read(byte[] buffer)

  - int read(byte[] buffer, int offset, int length)

- Other methods include:

  - void close()

  - int available()

  - long skip(long n)

  - boolean markSupported()

  - void mark(int readlimit)

  - void reset()

The OutputStream Methods

- The three basic write methods are:

    - void write(int c)

    - void write(byte[] buffer)

    - void write(byte[] buffer, int offset, int length)

- Other methods include:

    - void close()

    - void flush()

The Reader Methods

- The three basic read methods are:
  - int read()
  - int read(char[] cbuf)
  - int read(char[] cbuf, int offset, int length)
- Other methods include:
  - void close()
  - boolean ready()
  - long skip(long n)
  - boolean markSupported()
  - void mark(int readAheadLimit)
  - void reset()

ΣMERTXE

The Writer Methods

- The basic write methods are:

  - voidwrite(int c)

  - Void write(char[] cbuf)

  - Void write(char[]cbuf, int offset, int length)

  - Void write(String string)

  - Void write(String string, int offset, int length)

- Other methods include:

  - void close()

  - void flush()

Node Streams

Type	Character Streams	Byte Streams
File	FileReader FileWriter	FileInputStream FileOutputStream
Memory: array	CharArrayReader CharArrayWriter	ByteArrayInputStream ByteArrayOutputStream
Memory: string	StringReader StringWriter	N/A
Pipe	PipedReader PipedWriter	PipedInputStream PipedOutputStream

ΣMERTXE

A Simple Example

This program performs a copy file operation using a manual

buffer: *java TestNodeStreams file1 file2*

```java
public static void main(String[] args) {

try {

FileReader input = new FileReader(args[0]);

try {

FileWriter output = new FileWriter(args[1]);

try {

char[] buffer = new char[128];

int charsRead;

charsRead = input.read(buffer);

while ( charsRead != -1 ) {

// next slide
```

ƩMERTXE

A Simple Examples

```
output.write(buffer, 0, charsRead);
charsRead = input.read(buffer);
}
} finally {
output.close();}
} finally {
input.close();}
} catch (IOException e) {
e.printStackTrace();
}}
```

ΣMERTXE

Buffered Streams

This program performs a copy file operation using a built-in buffer: java TestBufferedStreams file1 file2

```java
public static void main(String[] args) {

try {

FileReader input = new FileReader(args[0]);

BufferedReader bufInput = new BufferedReader(input);

try {

FileWriter output = new FileWriter(args[1]);

BufferedWriter bufOutput= new BufferedWriter(output);

try {

String line;

line = bufInput.readLine();

while ( line != null ) {
```
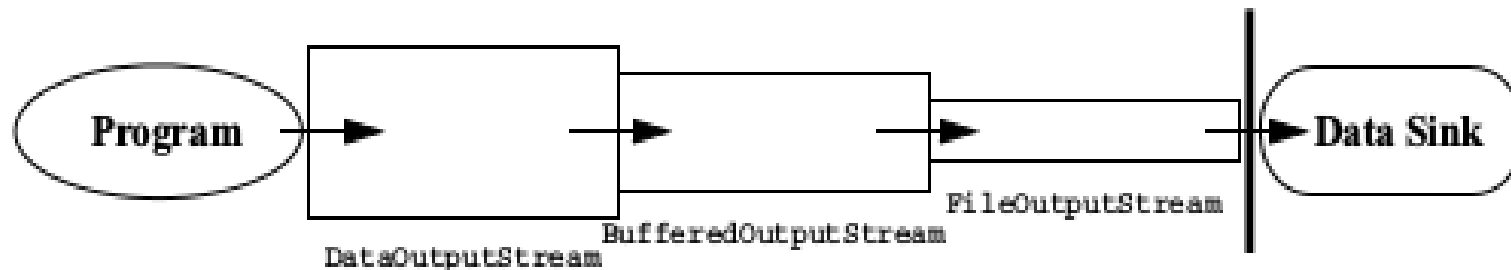
Buffered Streams

```
bufOutput.write(line, 0, line.length());
bufOutput.newLine();
line = bufInput.readLine();
}
} finally {
bufOutput.close();
}
} finally {
bufInput.close();
}
} catch (IOException e) {
e.printStackTrace();
}}
```

I/O Stream Chaining

Input Stream Chain

Data Source → FileInputStream → BufferedInputStream → DataInputStream → Program

Output Stream Chain

Program → DataOutputStream → BufferedOutputStream → FileOutputStream → Data Sink

EMERTXE
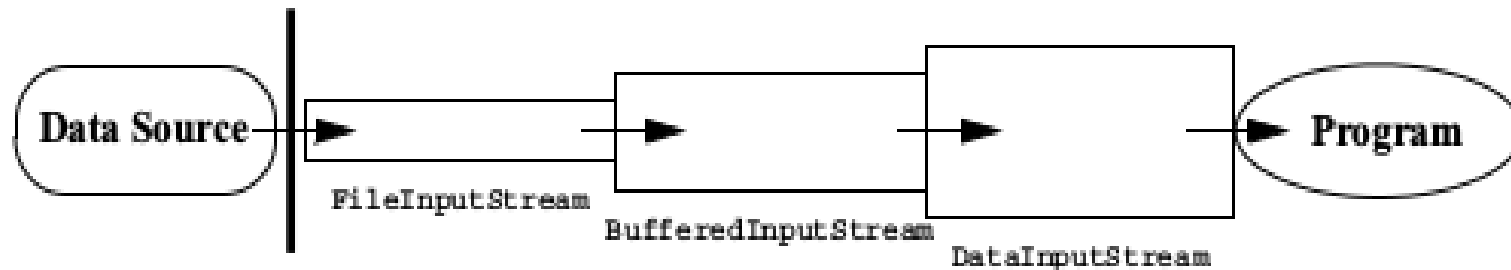
Processing Streams

Type	Character Streams	Byte Streams
Buffering	`BufferedReader` `BufferedWriter`	`BufferedInputStream` `BufferedOutputStream`
Filtering	*`FilterReader`* *`FilterWriter`*	*`FilterInputStream`* *`FilterOutputStream`*
Converting between bytes and character	`InputStreamReader` `OutputStreamWriter`	
Performing object serialization		`ObjectInputStream` `ObjectOutputStream`

Processing Streams

Type	Character Streams	Byte Streams
Performing data conversion		`DataInputStream` `DataOutputStream`
Counting	`LineNumberReader`	`LineNumberInputStream`
Peeking ahead	`PushbackReader`	`PushbackInputStream`
Printing	`PrintWriter`	`PrintStream`

The InputStream
Class Hierarchy
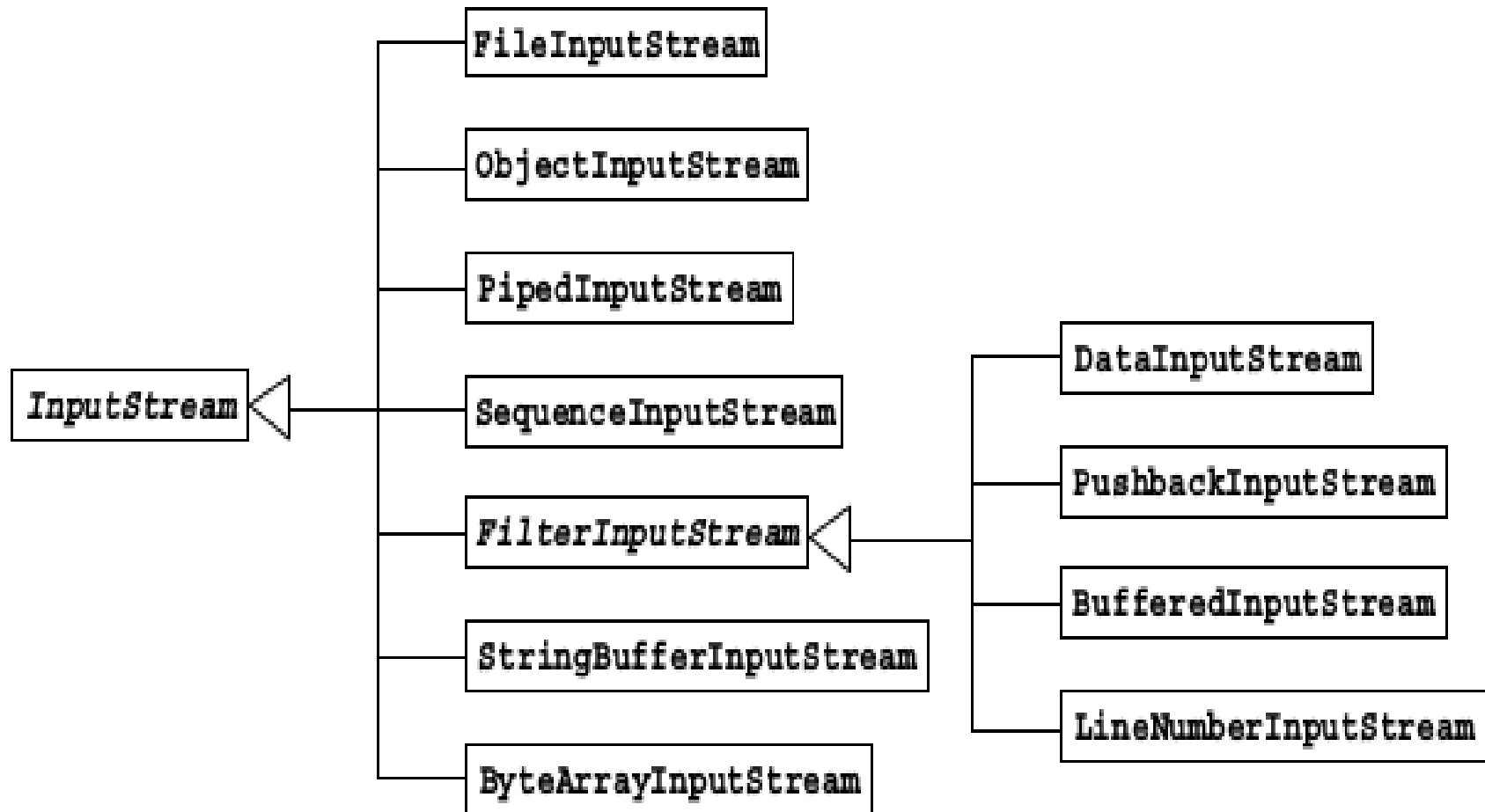
The OutputStream Class Hierarchy
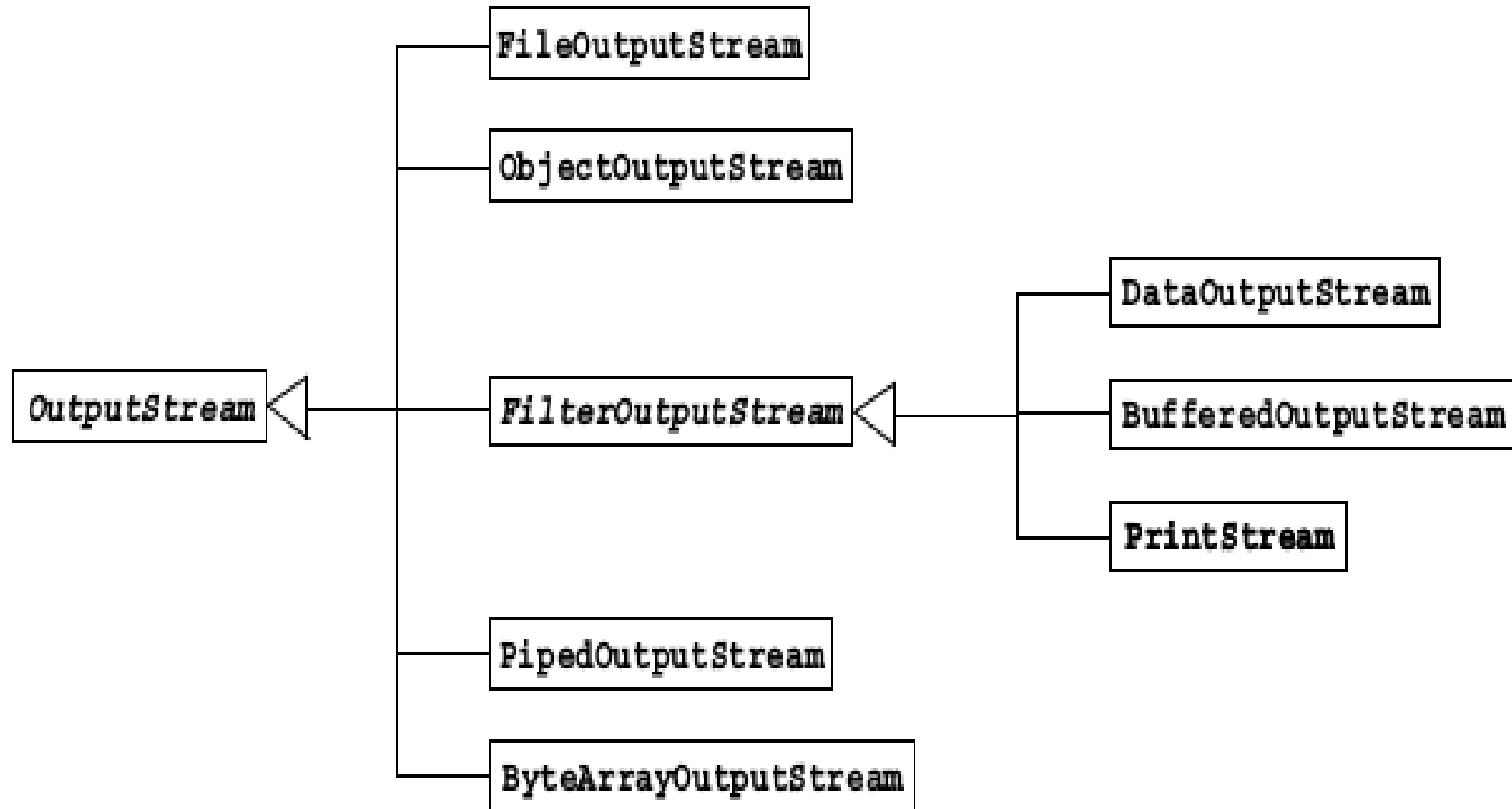
The ObjectInputStream and The ObjectOutputStream Classes

- The Java API provides a standard mechanism (called object serialization) that completely automates the process of writing and reading objects from streams.

- When writing an object, the object input stream writes the class name, followed by a description of the data members of the class, in the order they appear in the stream, followed by the values for all the fields on that object.

ΣMERTXE

The ObjectInputStream and The ObjectOutputStream Classes

- When reading an object, the object output stream reads the name of the class and the description of the class to match against the class in memory, and it reads the values from the stream to populate a newly allocation instance of that class.

- Persistent storage of objects can be accomplished if files (or other persistent storage) are used as streams.

ΣMERTXE

# Input Chaining Combinations: A Review
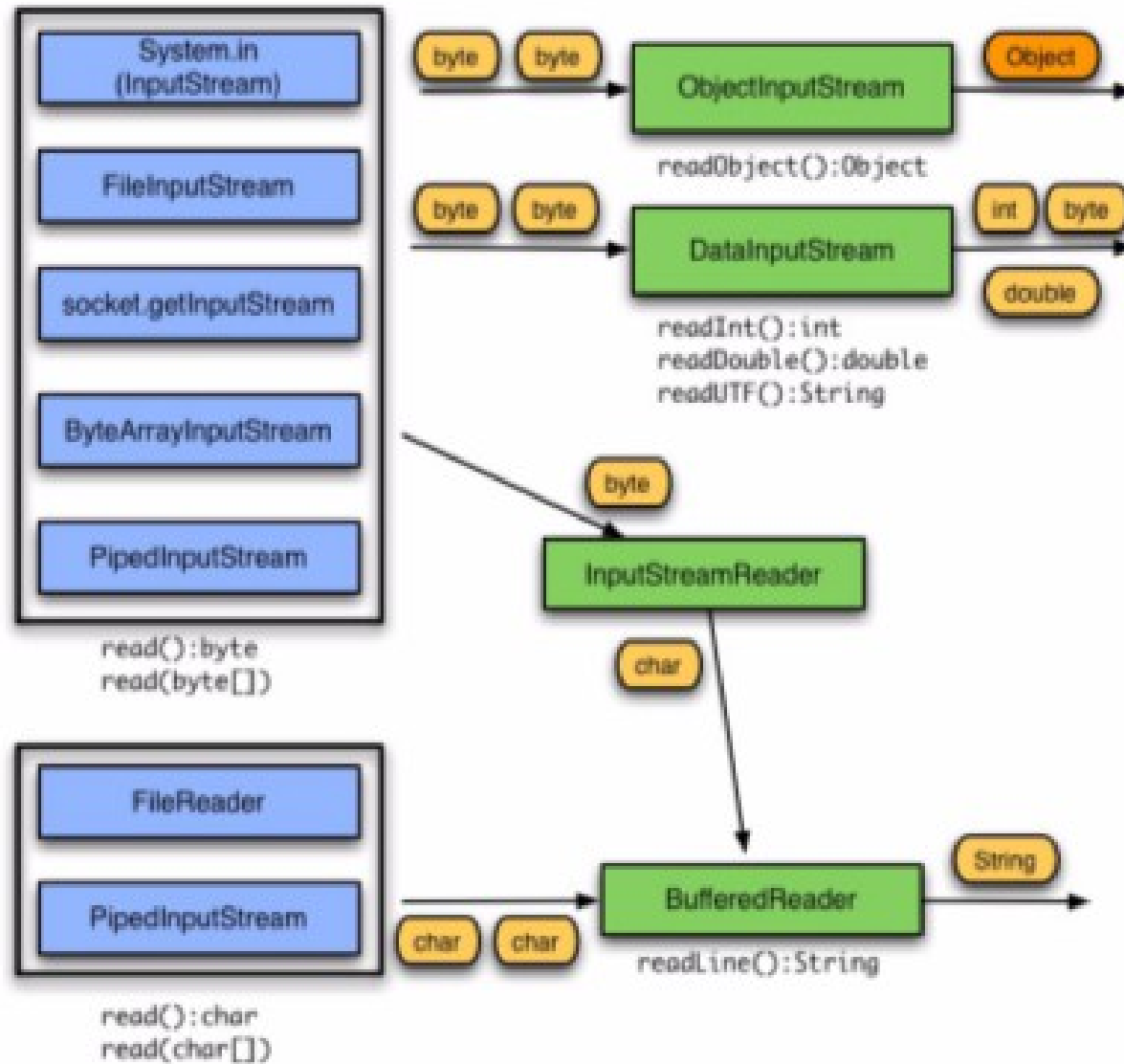
Output Chaining Combinations: A Review

EMERTXE
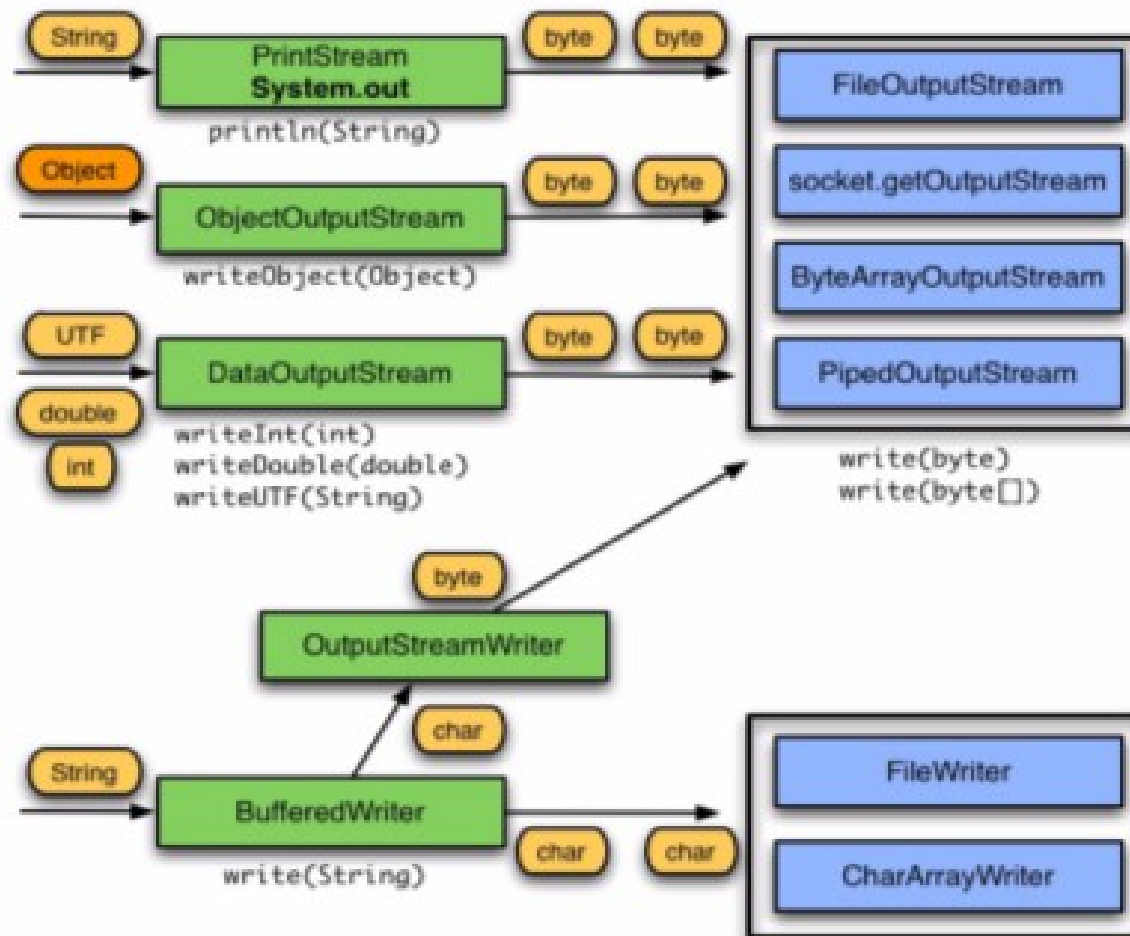
Serialization

- Serialization is a mechanism for saving the objects as a sequence of bytes and rebuilding them later when needed.

- When an object is serialized, only the fields of the object are preserved

- When a field references an object, the fields of the referenced object are also serialized

- Some object classes are not serializable because their fields represent transient operating system-specific information.

The SerializeDate Class

```java
public class SerializeDate implements Serializable{
SerializeDate() {
Date d = new Date ();
try {
FileOutputStream f =
new FileOutputStream ("date.ser");
ObjectOutputStream s =
new ObjectOutputStream (f);
s.writeObject (d);
s.close ();
} catch (IOException e) {
e.printStackTrace ();
}
```
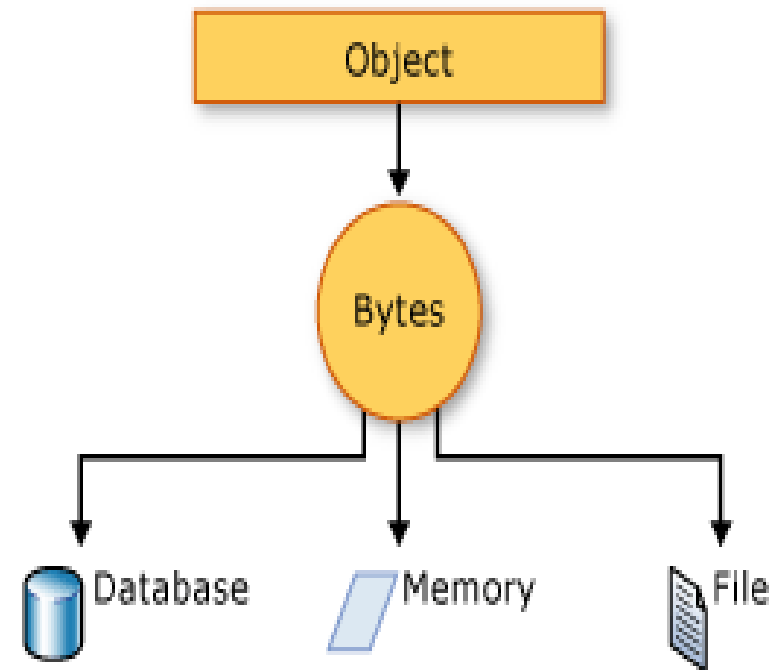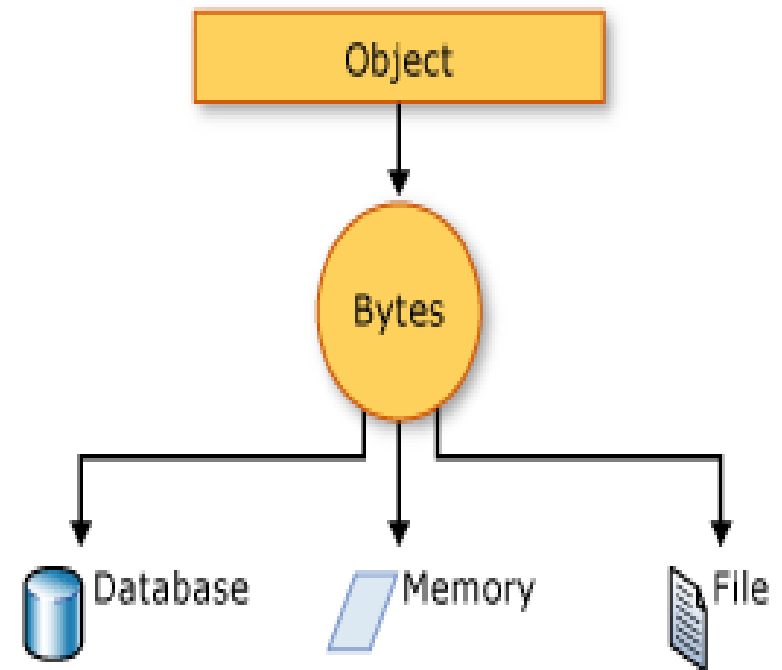
The SerializeDate Class

```java
public static void main (String args[]) {

new SerializeDate();

}}
```

The DeSerializeDate Class

```java
public class DeSerializeDate {
DeSerializeDate () {
Date d = null;
try {
FileInputStream f =
new FileInputStream ("date.ser");
ObjectInputStream s =
new ObjectInputStream (f);
d = (Date) s.readObject ();
s.close ();
} catch (Exception e) {
e.printStackTrace ();
}
```

The SerializeDate Class

```java
public class SerializeDate implements Serializable{
SerializeDate() {
Date d = new Date ();
try {
FileOutputStream f =
new FileOutputStream ("date.ser");
ObjectOutputStream s =
new ObjectOutputStream (f);
s.writeObject (d);
s.close ();
} catch (IOException e) {
e.printStackTrace ();
}
```
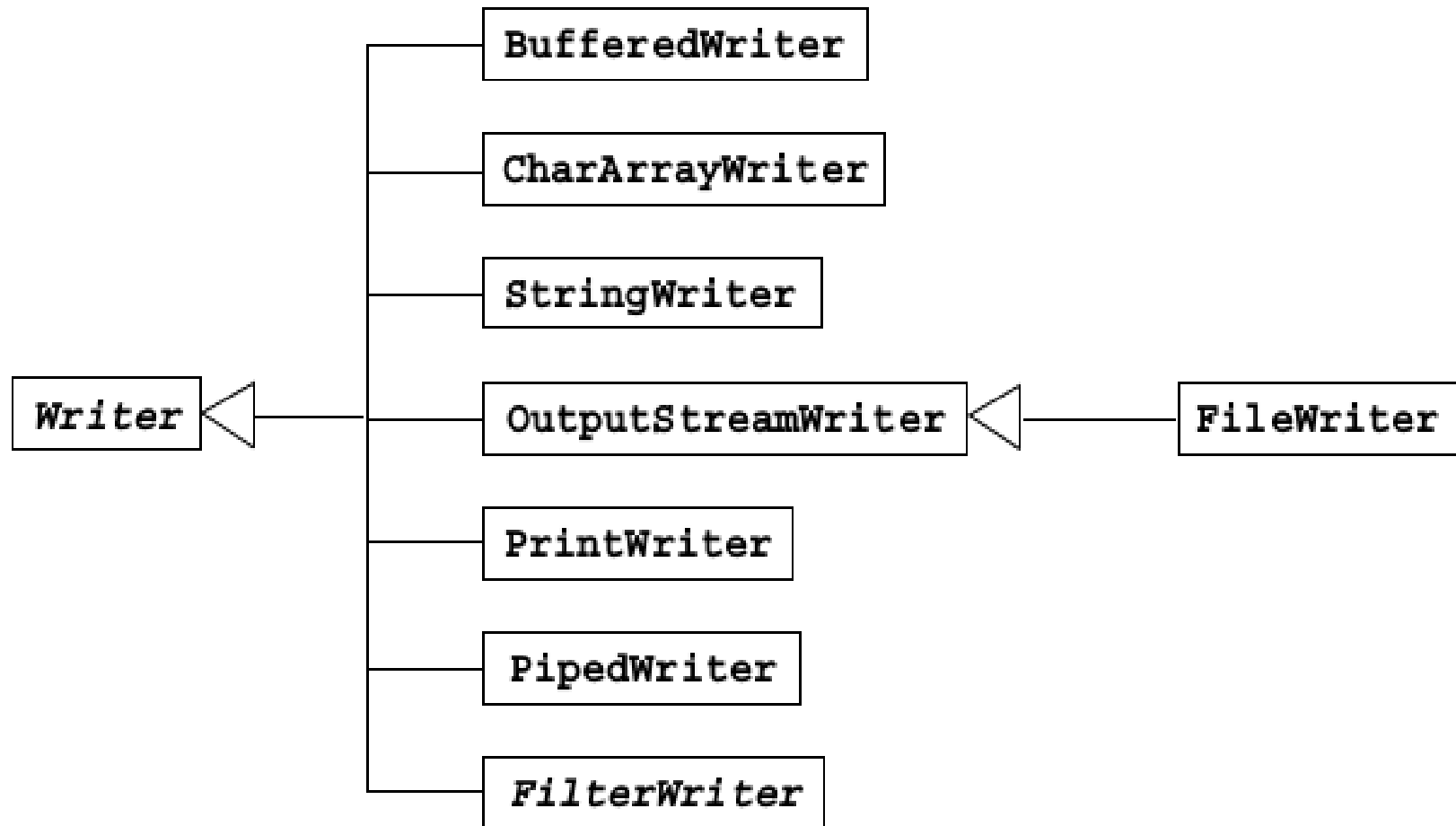
The DeSerializeDate Class

```java
System.out.println(

"Deserialized Date object from date.ser");

System.out.println("Date: "+d);

}

public static void main (String args[]) {

new DeSerializeDate();

}}
```

The Reader Class Hierarchy
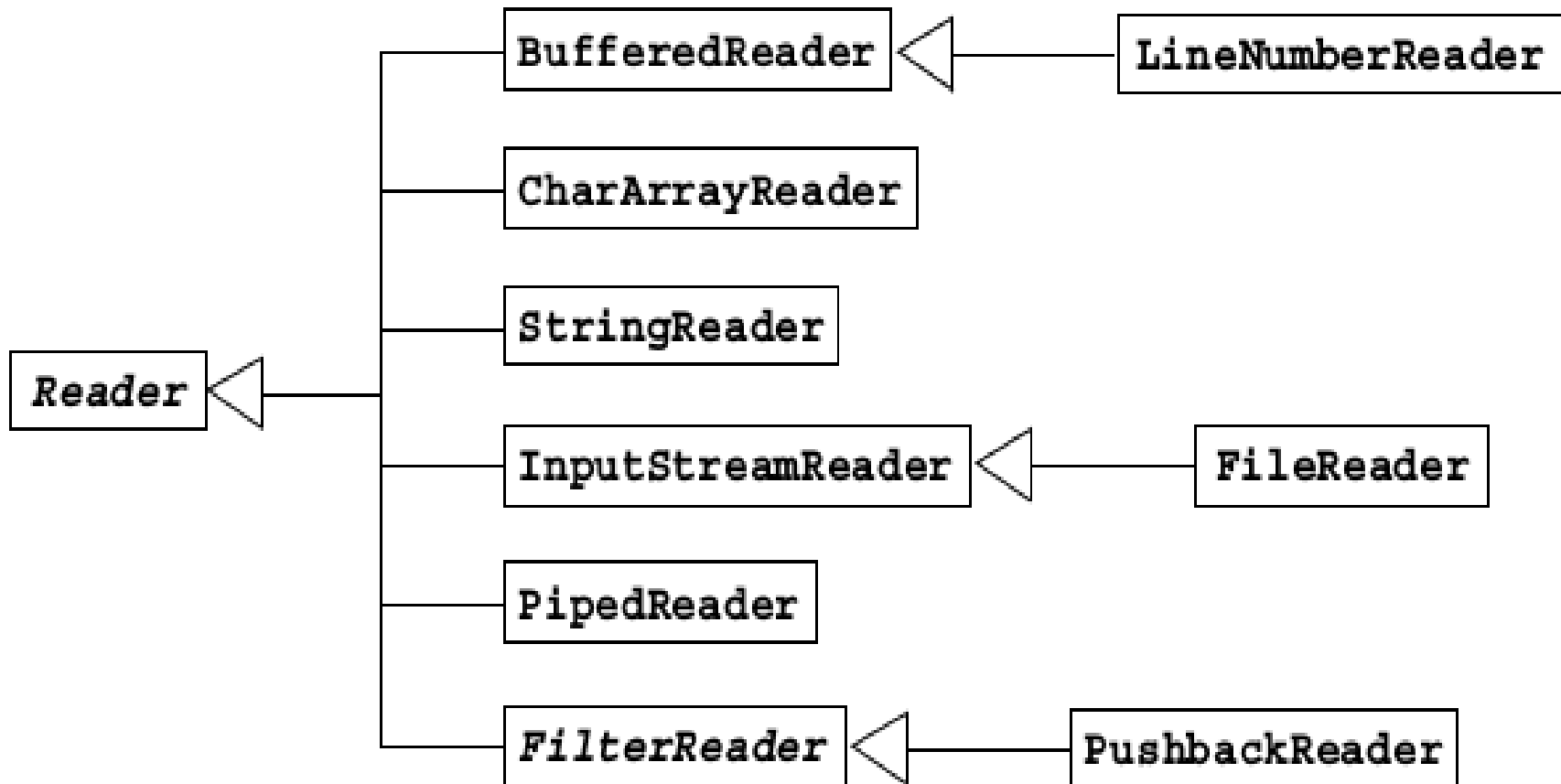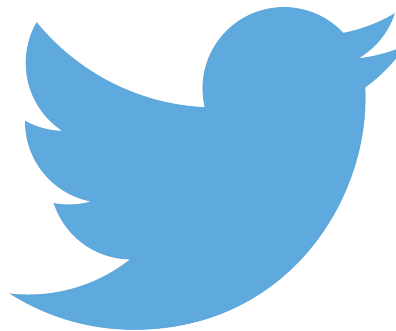
The Writer Class Hierarchy

Stay connected

About us: Emertxe is India's one of the top IT finishing schools & self learning kits provider. Our primary focus is on Embedded with diversification focus on Java, Oracle and Android areas

Emertxe Information Technologies,
No-1, 9th Cross, 5th Main,
Jayamahal Extension,
Bangalore, Karnataka 560046
T: +91 80 6562 9666
E: training@emertxe.com

https://www.facebook.com/Emertxe

https://twitter.com/EmertxeTweet

https://www.slideshare.net/EmertxeSlides

slideshare
Present Yourself

ƩMERTXE

Thank You