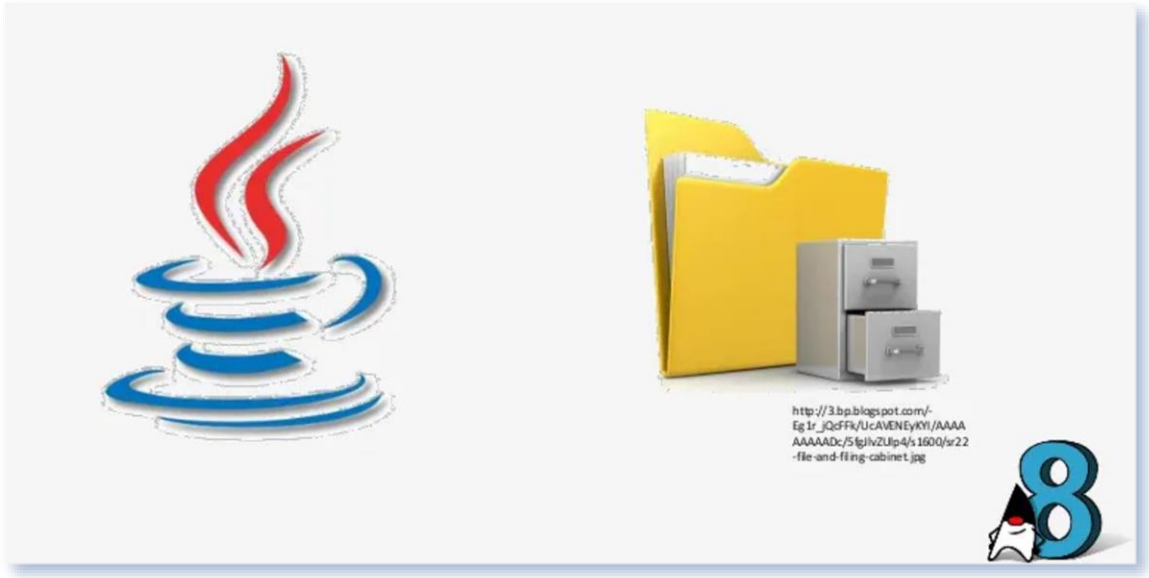


Unit 2

Persistence in files. Java Streams. (I)

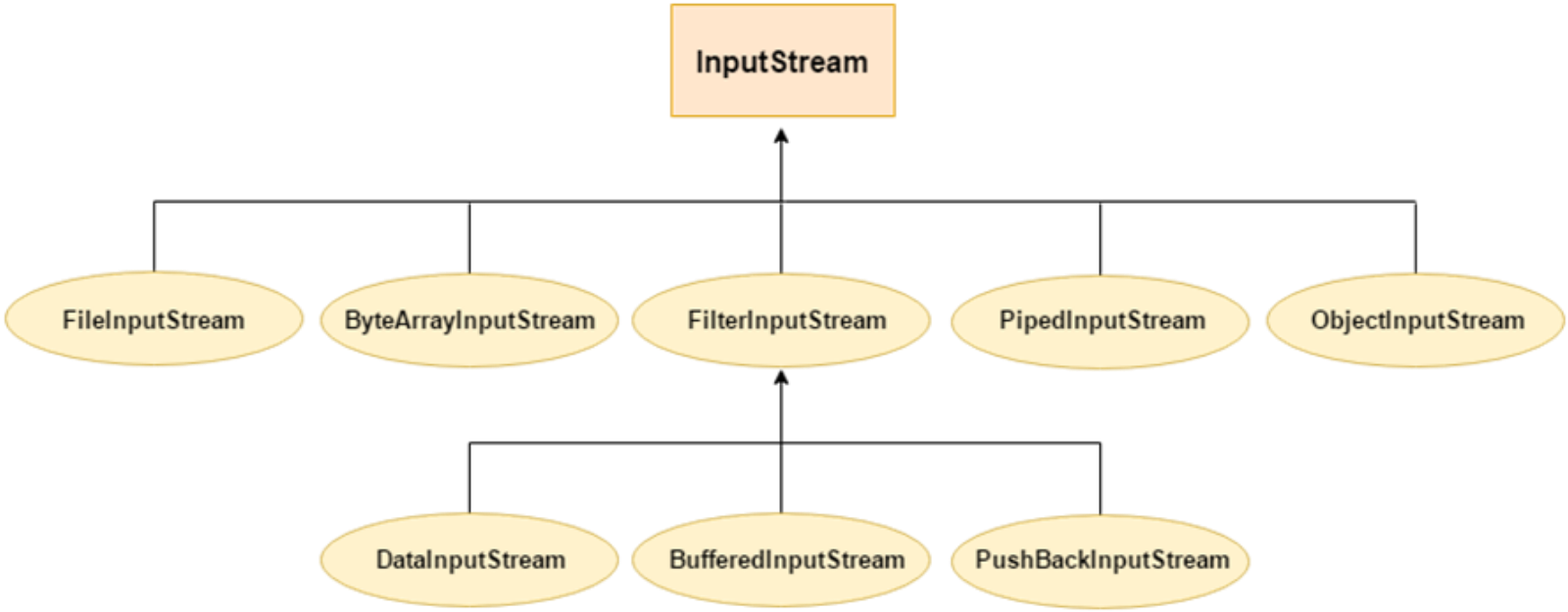
Data Persistence

- Characteristic of the state of a system that outlives (persists more than) the process that created it
- Achieved by storing the state as data in computer data storage
- Two aspects:
  - Dealing with the data itself: that's made via the Stream class.
  - Dealing with the file system: that's made via the File Class.

Streams

- Stream: sequence of data
- Standard streams:
  - Standard input: System.in
  - Standard output: System.out
  - Standard error: System.err
- Scanner class: utility to read from System.in

Binary read: InputStream



- `read( )`
- `read( byte[] )`
- `mark()`
- `reset()`

Jose Ramón García Sevilla

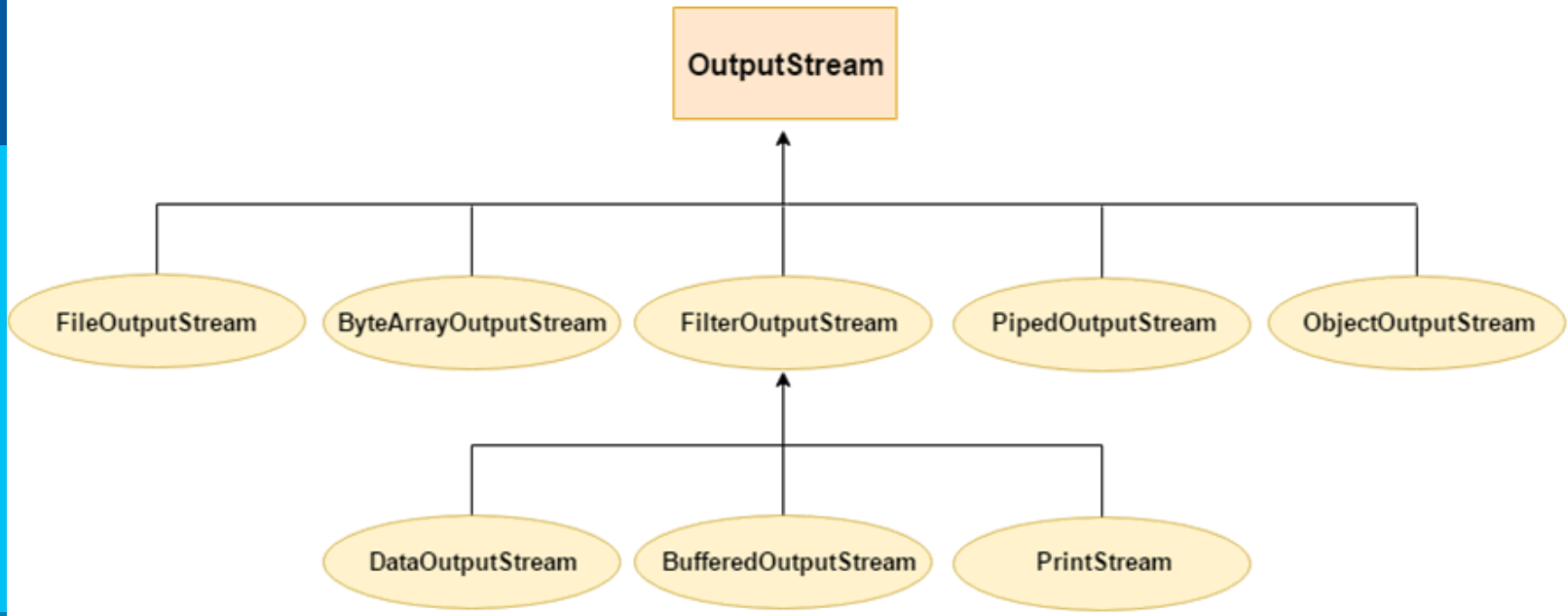
jrgarcia@iesmarenostrum.com

InputStream

Example:

```
InputStreaminputstream = new
FileInputStream("c:\\data\\input-text.txt");
byte[] data = new byte[1024];
int bytesRead = inputstream.read( data );
while( bytesRead != -1 ) {
    doSomethingWithData( data, bytesRead );
    bytesRead = inputstream.read(data);
}
inputstream.close();
```

Binary write: OutputStream



- `write( char )`
- `write( byte[] )`
- `flush()`
- `close()`

Jose Ramón García Sevilla

jrgarcia@iesmarenostrum.com

OutputStream

Example:

```
OutputStream output = null;
try{
     output = new FileOutputStream("c:\\data\\output-text.txt");
     while( hasMoreData() ) {
        int data = getMoreData();
        output.write( data );
     }
}
finally {
     if( output != null ) {
        output.close();
     }
}
```

Jose Ramón García Sevilla

jrgarcia@iesmarenostrum.com

FileInput(Output)Stream

- Used to deal with data stored in files

```
import java.io.*;
public class FileOutputStreamExample {
    public static void main( String args[] ) {
        try {
            FileInputStream fin = new FileInputStream("D:\\testin.txt");
            FileOutputStream fout = new FileOutputStream("D:\\testout.txt");
            int i=0;
            while( ( i = fin.read() )!=-1 )
                fout.write( i );
            fout.close();
            System.out.println("Copy successful...");
        }
        catch( Exception e ) {
            System.out.println( e );
        }
    }
```

Jose Ramón García Sevilla jrgarcia@iesmarenostrum.com

Character Streams: File Reader/Writer

- Characters: 16-bit unicode

Constructor	Description
FileReader(String file)	It gets filename in string. It opens the given file in read mode. If file doesn't exist, it throws FileNotFoundException.
FileReader(File file)	It gets filename in file instance. It opens the given file in read mode. If file doesn't exist, it throws FileNotFoundException.
int read()	It is used to return a character in ASCII form. It returns -1 at the end of file.
void close()	It is used to close the FileReader class.

Constructor	Description
FileWriter(String file)	Creates a new file. It gets file name in string.
FileWriter(File file)	Creates a new file. It gets file name in File object.
FileWriter(FileWriter(String fileName, boolean append)	Constructs a FileWriter object given a file name with a boolean indicating whether or not to append the data written.
void write(String text)	It is used to write the string into FileWriter.
void write(char c)	It is used to write the char into FileWriter.
void write(char[] c)	It is used to write char array into FileWriter.
void flush()	It is used to flushes the data of FileWriter.
void close()	It is used to close the FileWriter.

Buffered Reader/Writer

- Use a buffer for read/write operations
- Better performance
- Flushing should be made in case of error
- readLine supported

PrintWriter

- Best option to write formatted information
- It can be combined with other classes for improved functionality.

```
PrintWriter printWriter = null;
try {
    printWriter = new PrintWriter(new BufferedWriter(
            new FileWriter( "ejemplo.txt", true )));
    printWriter.println("Hello!");
    printWriter.println("and...");
    printWriter.println("see you soon!");
}
```

Jose Ramón García Sevilla

jrgarcia@iesmarenostrum.com

File class

- Use to manage the file system
- Use for operations like move files/directories, create files/directories, delete files…
- It’s an abstract representation of a pathname

Serialization

- Process of translating a data structure or object state into a format that can be stored.

- ObjectOutputStream class
  - writeObject
  - readObject

Jose Ramón García Sevilla

jrgarcia@iesmarenostrum.com

SAX

- Read XML node by node.
- Faster and require less memory; it has no state
- Objects: SAXParserFactory, SAXParser, DefaultHandler

SAX

```
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

class myXMLContactsHandler extends DefaultHandler {
    protected String currentTag;
    protected String tagContent;

    // Tag opening found
    //
    public void startElement(String uri, String localName,
            String qName, Attributes attributes) throws SAXException {

        currentTag = qName;
        if ( currentTag.equals("contact") ) {
            System.out.println( "ID: " + attributes.getValue("id"));
        }
    }
```

Jose Ramón García Sevilla jrgarcia@iesmarenostrum.com

SAX

```
    // Tag content, usually CDATA
    //
    public void characters( char ch[], int start, int length )
            throws SAXException {
        tagContent = new String( ch, start, length );
    }

    // Tag ending
    //
    public void endElement(String uri, String localName, String qName)
            throws SAXException {
        if ( !currentTag.isBlank() ) {
            if ( !currentTag.equals( "contact" ) ) {
                System.out.println("   "   +   currentTag   +   ":   "   +
tagContent);
                currentTag = "";
            }
        }
    }
}
```

Jose Ramón García Sevilla *jrgarcia@iesmarenostrum.com*

SAX

```
public class Main {
    public static void main( String[] args ) {
        try {
            SAXParser saxParser = SAXParserFactory.
                  newInstance().newSAXParser();
            saxParser.parse("contacts.xml", new
                  myXMLContactsHandler());
        } catch ( Exception e ) {
            e.printStackTrace();
        }
    }
}
```

Jose Ramón García Sevilla

jrgarcia@iesmarenostrum.com