



Les entrées/sorties en Java

input/output

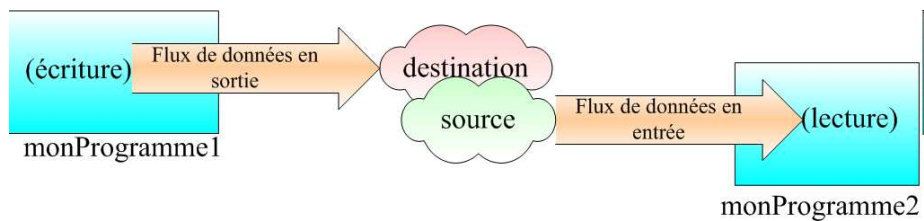


XH

1

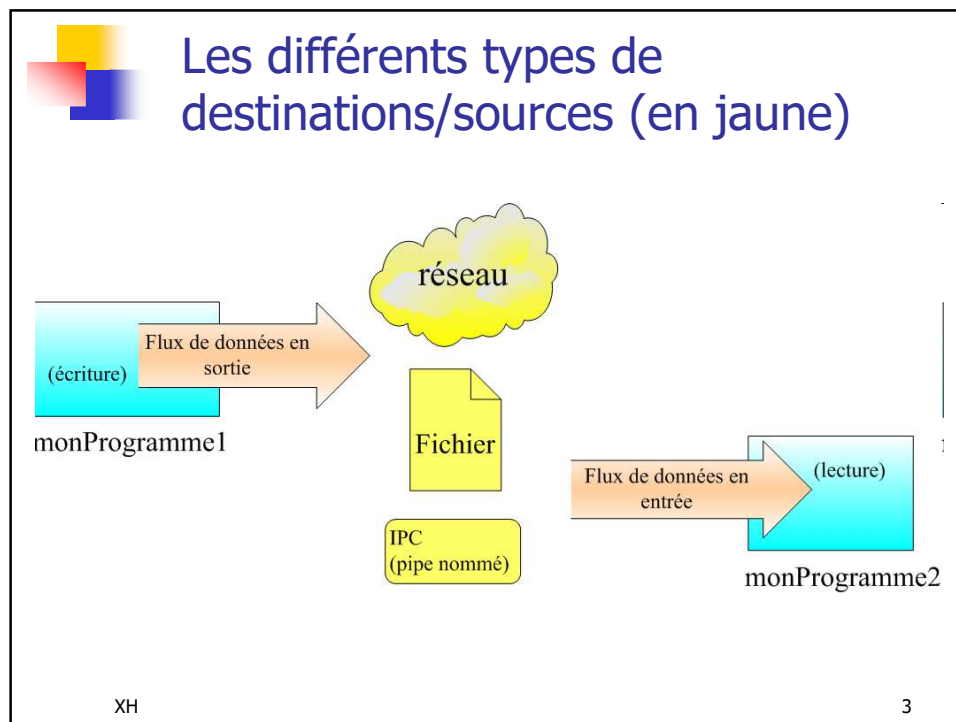


Concept des entrées/sorties en Java



XH


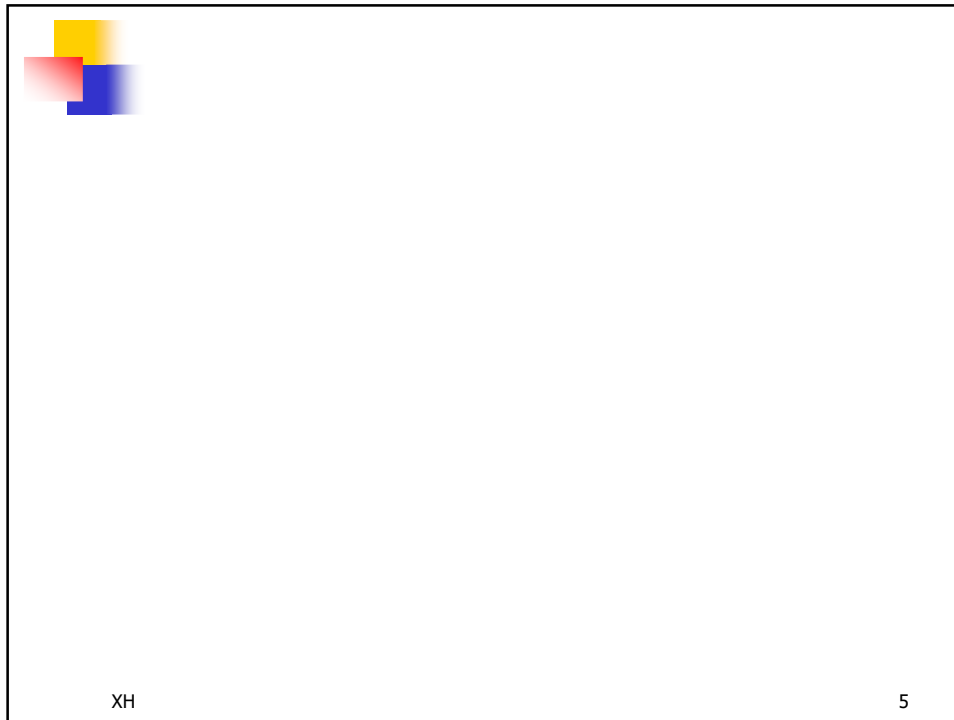
2



Package java.io

- Provides for system input and output through data streams, serialization and the file system
- java.io de JSE 6
 - 12 Interfaces
 - 51 Classes
 - 16 Exceptions
 - 1 Errors

XH 4



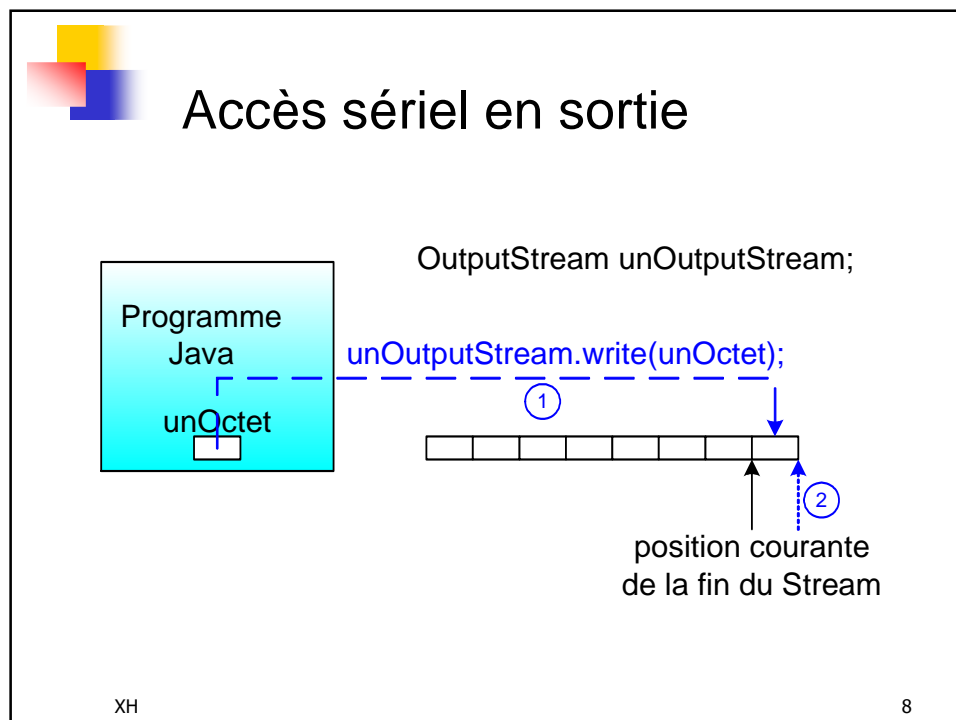
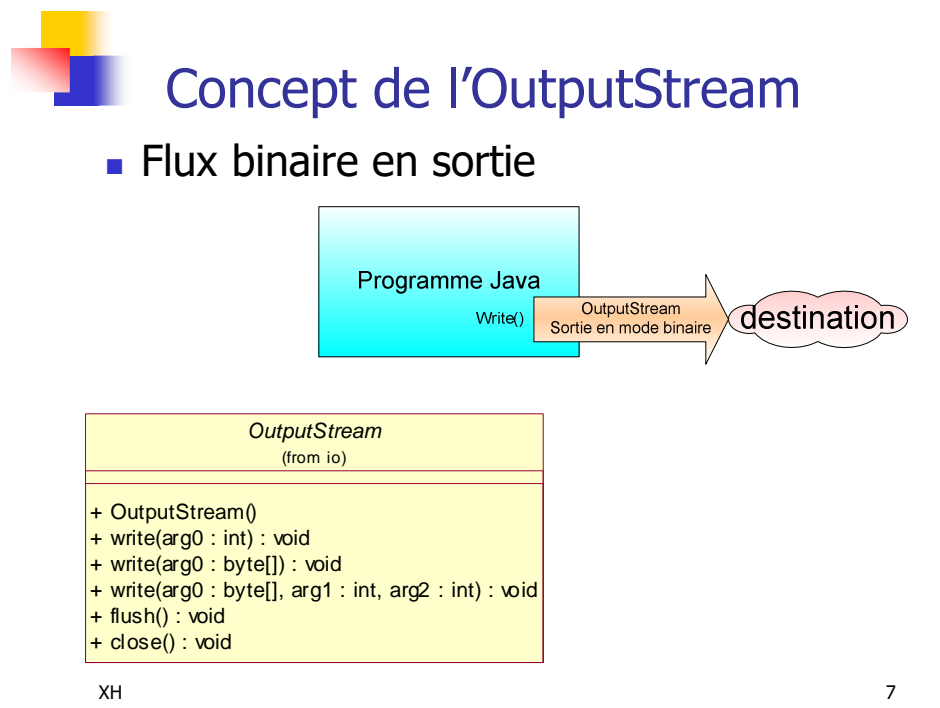
Entrées/sorties en mode **binaire**

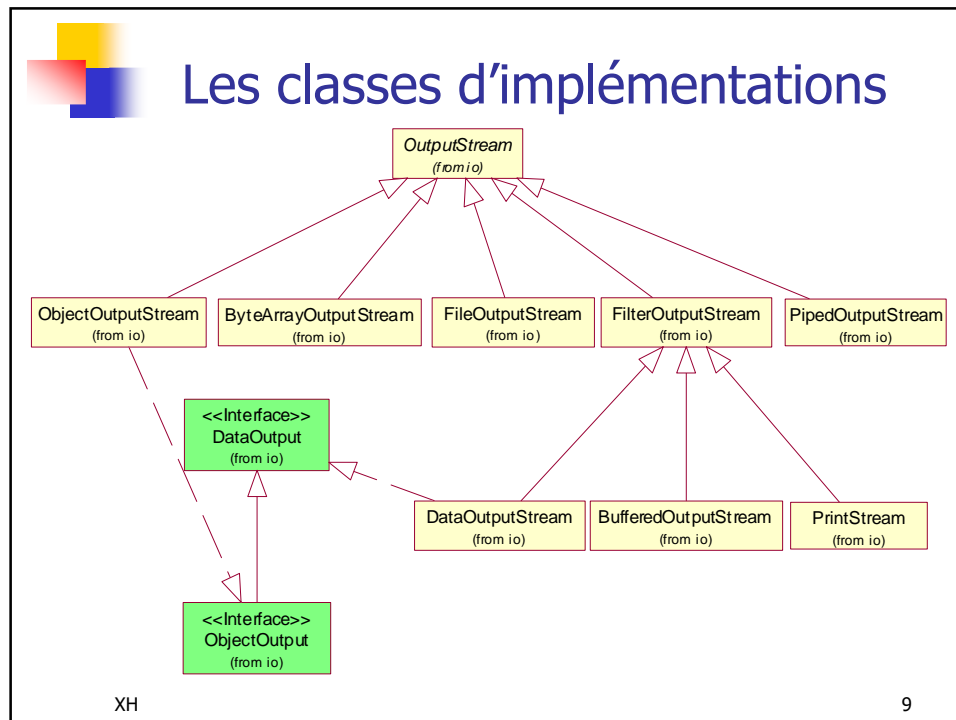
Flux de bytes - Flux d'octets

OutputStream
InputStream

XH

6





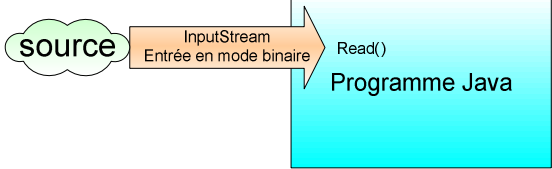
Explications des différentes implémentation

- Le concept des filtres
 - FilterOutputStream
- Filtre sur les types primitifs Java
 - DataOutputStream
- Cas Particulier de RandomAccessFile (accès aléatoire)

XH 10

Concept de l'InputStream

- Flux binaire en entrée



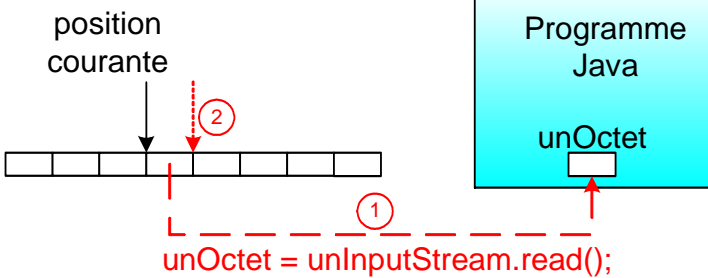
```

classDiagram
    class InputStream {
        +InputStream()
        +read() : int
        +read(arg0 : byte[]) : int
        +read(arg0 : byte[], arg1 : int, arg2 : int) : int
        +skip(arg0 : long) : long
        +available() : int
        +close() : void
        +mark(arg0 : int) : void
        +reset() : void
        +markSupported() : boolean
    }
  
```

XH 11

Accès sériel en entrée

```
InputStream unInputStream;
```



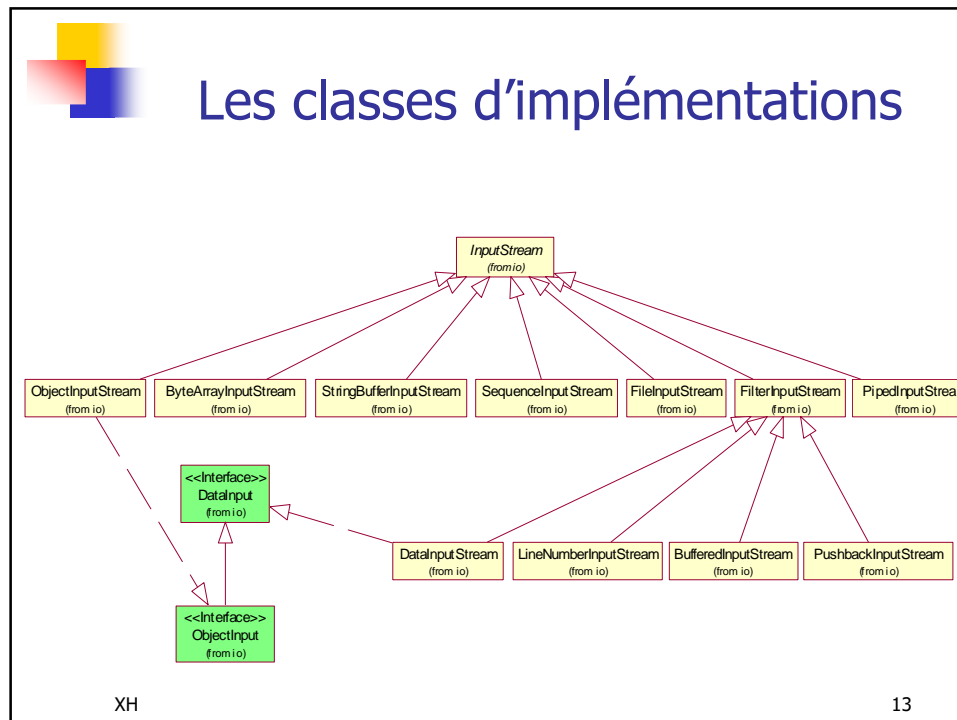
position courante

Programme Java

unOctet

unOctet = unInputStream.read();

XH 12



Explications des différentes implémentations

- Le concept des filtres
 - FilterInputStream
- Filtres sur les types primitifs Java
 - DataInputStream
- Cas Particulier de RandomAccessFile (accès aléatoire)

XH 14



Exemple de lecture d'un fichier texte pour comptage du nb d'octets

```
/* Lit un fichier et en compte le nombre d'octets */
import java.io.*;
public class CompterOctetsFichier{
public static void main (String args[]) {
    int unOctet;
    int compteurOctet = 0;
    FileInputStream unFichier;
    try{
        unFichier = new FileInputStream("C:/Fichier1.txt");
        while((unOctet = unFichier.read()) != -1)
            compteurOctet++;
        unFichier.close();
        System.out.println("Nombre d'octets du fichier : "
                           + compteurOctet);
    }
    catch (IOException e){
        System.err.println("Exception\n" + e.toString());
    }
}
}
```

XH

15



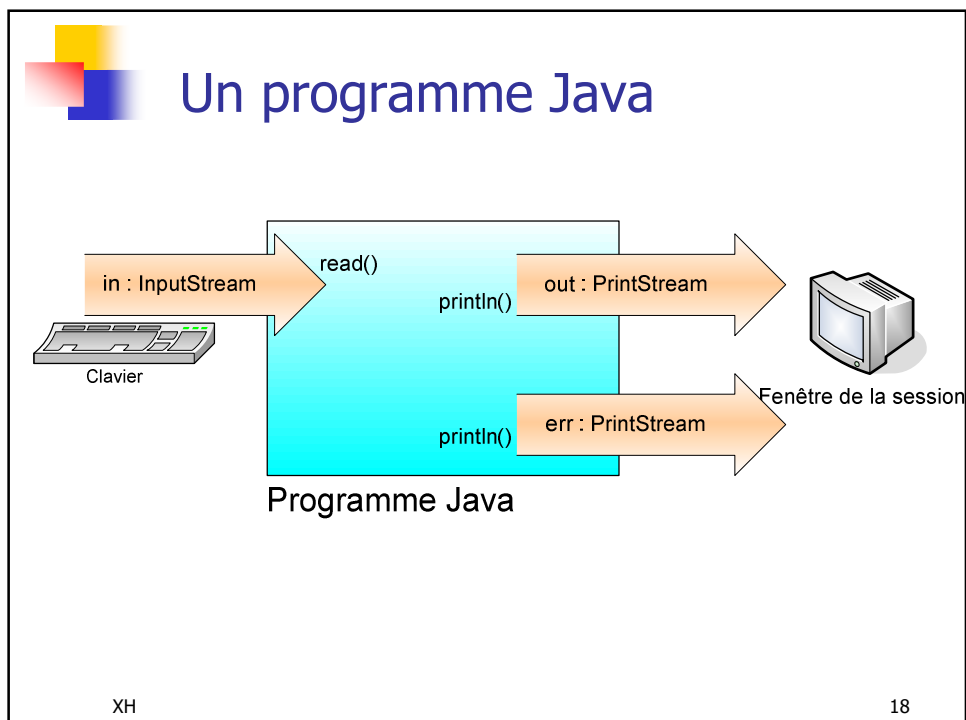
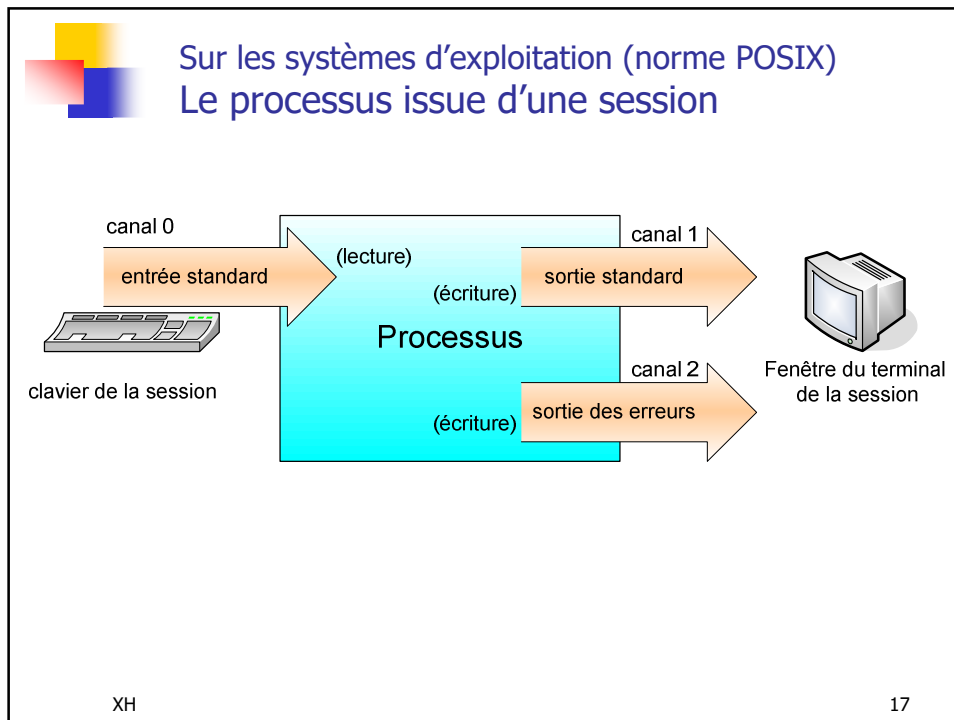
L'outil StreamTokenizer

- A ne pas confondre avec StringTokenizer
- dans java.io
- « découpeur de flux »
- Outil de décodage pour un flux binaire **ou** un flux de caractères
 - Il y a 2 constructeurs
 - A voir plus loin

StreamTokenizer (from io)
+ ttype : int + TT_EOF : int = -1 + TT_EOL : int = 10 + TT_NUMBER : int = -2 + TT_WORD : int = -3 + nval : double + sval : String
+ StreamTokenizer(arg0 : InputStream) + StreamTokenizer(arg0 : Reader) + resetSyntax() : void + wordChars(arg0 : int, arg1 : int) : void + whitespaceChars(arg0 : int, arg1 : int) : void + ordinaryChars(arg0 : int, arg1 : int) : void + ordinaryChar(arg0 : int) : void + commentChar(arg0 : int) : void + quoteChar(arg0 : int) : void + parseNumbers() : void + eolIsSignificant(arg0 : boolean) : void + slashStarComments(arg0 : boolean) : void + slashSlashComments(arg0 : boolean) : void + lowerCaseMode(arg0 : boolean) : void + nextToken() : int + pushBack() : void + lineno() : int + toString() : String

XH

16





Les attributs de la classe System

Class System

[java.lang.Object](#)
└ [java.lang.System](#)

public final class System
extends [Object](#)

The System class contains several useful class fields and methods. It cannot be instantiated.

Among the facilities provided by the System class are standard input, standard output, and error output streams; access to externally defined properties and environment variables; a means of loading files and libraries; and a utility method for quickly copying a portion of an array.

Since:
JDK1.0

Field Summary

static PrintStream	err	The "standard" error output stream.
static InputStream	in	The "standard" input stream.
static PrintStream	out	The "standard" output stream.

.9



Un pont + un filtre ???


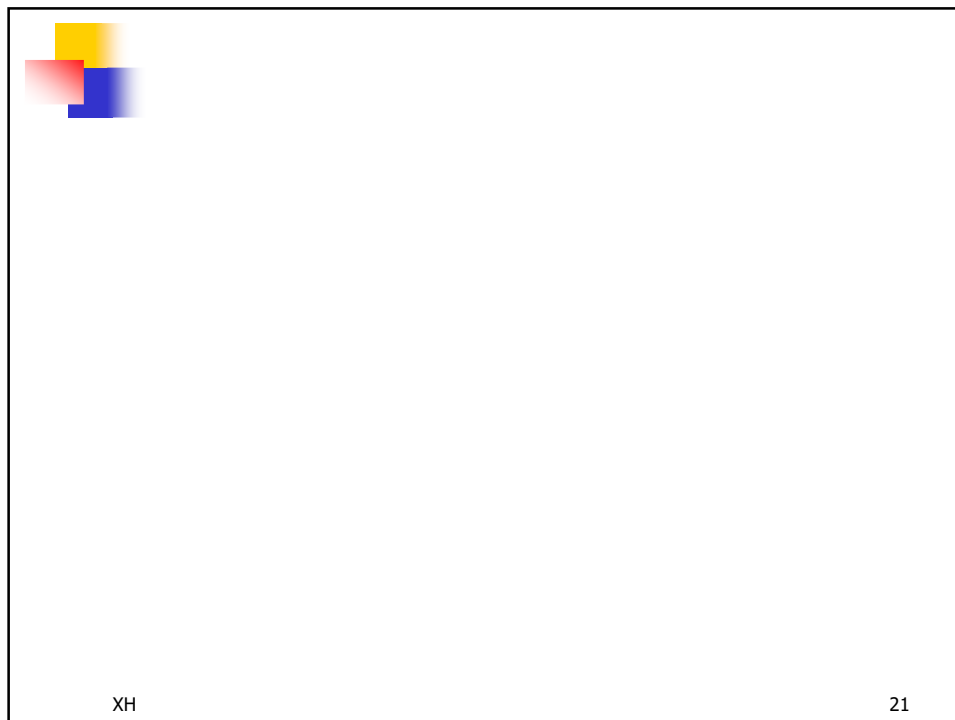
```
class SaisieClavier {
    public static void main (String[] argv) throws IOException,
        NumberFormatException {
        int somme = 0;
        String ligne;
        StringTokenizer st;
        BufferedReader entree = new BufferedReader(new
            InputStreamReader(System.in));

        ligne = entree.readLine();

        while(ligne.length() > 0)
        {
            st = new StringTokenizer(ligne);
            while(st.hasMoreTokens())
                somme += Integer.parseInt(st.nextToken());

            ligne = entree.readLine();
        }
        System.out.println("La somme vaut : "+somme);
    } }
XH
```

20



Entrées/sorties en mode **texte**

En mode caractères

Flux de caractères (char)

UTF-16

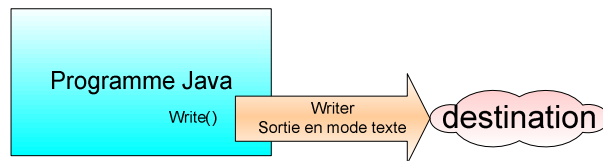
XH

22



Concept du flux Writer

■ Flux en sortie



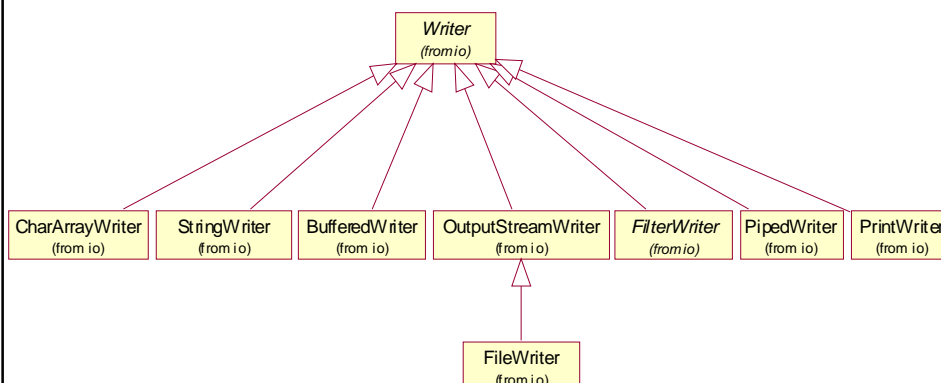
<i>Writer</i> (from io)
lock : Object
+ Writer()
Writer(arg0 : Object)
+ write(arg0 : int) : void
+ write(arg0 : char[]) : void
+ write(arg0 : char[], arg1 : int, arg2 : int) : void
+ write(arg0 : String) : void
+ write(arg0 : String, arg1 : int, arg2 : int) : void
+ flush() : void
+ close() : void

XH

23



Les classes implémentations de Writer



XH

24



Explications des différentes implémentations

- Entrée/sortie avec la mémoire
- Les pipelines
- Le concept des filtres
 - FilterWriter

XH

25



Les fichiers directement en flux de caractères

- FileWriter
 - Voir exemple projet Dexter

```
public static boolean saveStringToFile(String fileName, String
saveString) {
    boolean saved = false;
    BufferedWriter bw = null;

    try {
        bw = new BufferedWriter(
            new FileWriter(fileName));
        try {
            bw.write(saveString);
            saved = true;
        }
        finally {
            bw.close();
        }
    }
    catch (IOException ex) {
        ex.printStackTrace();
    }

    return saved;
}
```

XH

26



Encodage et jeu de caractères

- Voir autres slides donnés en début de formation

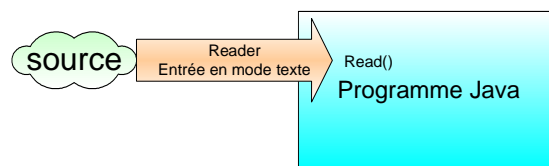
XH

27



Concept du flux Reader

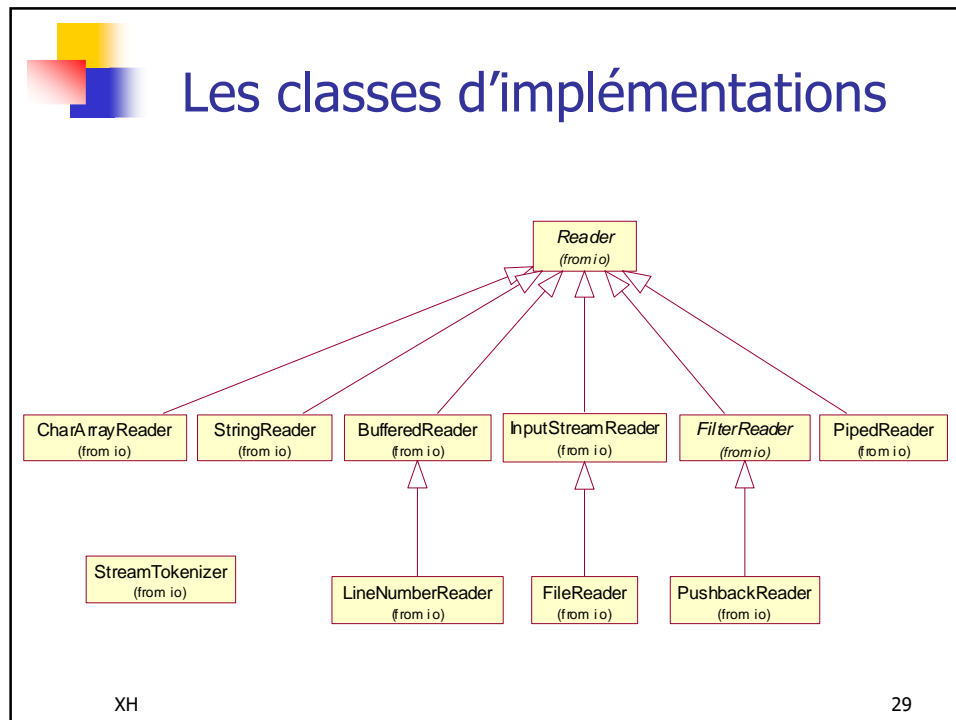
- Flux en entrée



<i>Reader</i> (from io)
lock : Object
+ Reader()
Reader(arg0 : Object)
+ read() : int
+ read(arg0 : char[]) : int
+ read(arg0 : char[], arg1 : int, arg2 : int) : int
+ skip(arg0 : long) : long
+ ready() : boolean
+ markSupported() : boolean
+ mark(arg0 : int) : void
+ reset() : void
+ close() : void

XH

28



Explications des différentes implémentations

- FilterReader : Le concept des filtres
- PushbackReader: Filtres sur ...

XH 30



Classe de conversion flux binaire → caractères : un pont (bridge)

- **InputStreamReader**
- Voir exemple de saisie clavier

XH

31



Lecture de fichier directement en flux de caractères

- **FileReader**
- Voir exemple projet Dexter

```
public static String getStringFromFile(String fileName) {
    BufferedReader br = null;
    StringBuilder sb = new StringBuilder();

    try {
        br = new BufferedReader(
            new FileReader(fileName));

        try {
            String s;

            while ((s = br.readLine()) != null) {
                // add linefeed (\n) back since stripped by readline()
                sb.append(s + "\n");
            }
        } finally {
            br.close();
        }
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return sb.toString();
}
```

XH

32



Lecture et écriture tamponnées

- `BufferedReader`
- `BufferedWriter`



XH

33



Ecriture de types primitifs dans un fichier éditable

- `PrintWriter`

XH

34



Encore l'outil StreamTokenizer

- Ne pas confondre avec StringTokenizer
- « découpeur de flux »
- Outil de décodage pour un flux binaire ou un flux de caractères
- Il y a 2 constructeurs
- Voir les méthodes ci-contre

```

StreamTokenizer
(from io)

+ ttype : int
+ TT_EOF : int = -1
+ TT_EOL : int = 10
+ TT_NUMBER : int = -2
+ TT_WORD : int = -3
+ nval : double
+ sval : String

+ StreamTokenizer(arg0 : InputStream)
+ StreamTokenizer(arg0 : Reader)
+ resetSyntax() : void
+ wordChars(arg0 : int, arg1 : int) : void
+ whitespaceChars(arg0 : int, arg1 : int) : void
+ ordinaryChars(arg0 : int, arg1 : int) : void
+ ordinaryChar(arg0 : int) : void
+ commentChar(arg0 : int) : void
+ quoteChar(arg0 : int) : void
+ parseNumbers() : void
+ eollsSignificant(arg0 : boolean) : void
+ slashStarComments(arg0 : boolean) : void
+ slashSlashComments(arg0 : boolean) : void
+ lowerCaseMode(arg0 : boolean) : void
+ nextToken() : int
+ pushBack() : void
+ lineno() : int
+ toString() : String
  
```

XH

35



TP: Lecture d'une entrée clavier

- Réaliser un programme qui affiche le texte qui est entré au clavier

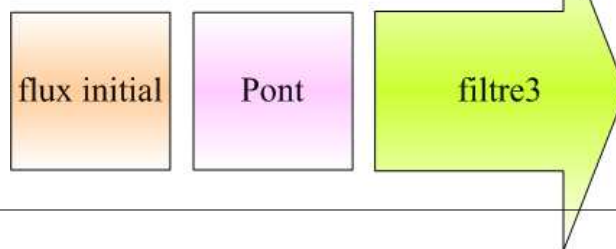
XH

36



Solution

```
BufferedReader entree = new BufferedReader(new  
InputStreamReader(System.in));  
ligne = entree.readLine();  
System.out.println("Voici l'entrée clavier : <" + ligne + ">");  
  
//int entreeNumerique = Integer.parseInt(entree.readLine());
```



XH

37



Les exceptions durant les entrées-sorties

XH

38



Javadoc de la classe IOException

java.io

Class IOException

```

java.lang.Object
├── java.lang.Throwable
│   └── java.lang.Exception
│       └── java.io.IOException
    
```

All Implemented Interfaces:
[Serializable](#)

Direct Known Subclasses:
[ChangedCharSetException](#), [CharacterCodingException](#), [CharConversionException](#), [ClosedChannelException](#), [EOFException](#), [FileLockInterruptedException](#), [FileNotFoundException](#), [FileException](#), [HttpRetryException](#), [IOException](#), [InterruptedIOException](#), [InvalidPropertiesFormatException](#), [JMXProviderException](#), [JMXServerErrorException](#), [MalformedURLException](#), [ObjectStreamException](#), [ProtocolException](#), [RemoteException](#), [SaslException](#), [SocketException](#), [SSLException](#), [SyncFailedException](#), [UnknownHostException](#), [UnknownServiceException](#), [UnsupportedDataTypeException](#), [UnsupportedEncodingException](#), [UTFDataFormatException](#), [ZipException](#)

```

public class IOException
    extends Exception
    
```

Signals that an I/O exception of some sort has occurred. This class is the general class of exceptions produced by failed or interrupted I/O operations.

Since:
 JDK1.0

See Also:
[InputStream](#), [OutputStream](#), [Serialized Form](#)

39

