

## Package java.io

### Classes

- class File
- final class FileDescriptor
- abstract class InputStream
  - class ByteArrayInputStream
  - class FileInputStream
  - class FilterInputStream
    - class BufferedInputStream
    - class DataInputStream (interface DataInput)
    - class LineNumberInputStream
    - class PushbackInputStream
  - class PipedInputStream
  - class SequenceInputStream
  - class StringBufferInputStream
  
- abstract class OutputStream
  - class ByteArrayOutputStream
  - class FileOutputStream
  - class FilterOutputStream
    - class BufferedOutputStream
    - class DataOutputStream (interface DataOutput)
    - class PrintStream
  - class PipedOutputStream
  
- class RandomAccessFile (interfaces DataInput,DataOutput)
  
- class StreamTokenizer

### Interfaces

- Interface DataInput
- Interface DataOutput
- Interface FilenameFilter

### Exceptions

- 2.27 Class EOFException
- 2.28 Class FileNotFoundException
- 2.29 Class IOException
- 2.30 Class InterruptedIOException
- 2.31 Class UTFDataFormatException

## Constructeurs

### fichiers d'entrée

```
FileInputStream fis=new FileInputStream("toto");  
DataInputStream d=new DataInputStream(fis);
```

### fichiers de sortie

```
File f= new File("toto");  
FileOutputStream fos=new FileOutputStream(f);  
DataOutputStream d=new DataOutputStream(fos);
```

## flots standards

```
public final class java.lang.System extends java.lang.Object{  
// Fields  
    public static PrintStream err;  
    public static InputStream in;  
    public static PrintStream out;  
// Methods  
public static void arraycopy(Object src, int  src_pos,  
                             Object dst, int  dst_pos, int  length);  
public static long currentTimeMillis();  
public static void exit(int  status);  
public static void gc();  
public static Properties getProperties();  
public static String getProperty(String  key);  
public static String getProperty(String  key, String  def);  
public static SecurityManager getSecurityManager();  
public static void load(String  filename);  
public static void loadLibrary(String  libname);  
public static void runFinalization();  
public static void setProperties(Properties props);  
public static void setSecurityManager(SecurityManager s);  
}
```

```
System.out.println(System.in.getClass().getName());
```

**résultat :** java.io.BufferedReader

## exemples d'utilisation des flots standards

```
import java.io.*;
public class lecture {
static public void main(String[] args)
    {int x=0;
    System.out.print("saisie :");
    System.out.flush();
    try
        {x=System.in.read();
        System.out.println(System.in.available()+
                            " car. sont en attente.");
        }
    catch (IOException e)
        System.out.println(e.getMessage());

    System.out.println("int lu :" + x);
    System.out.println("car. lu :" + (char) x);
    }
}
```

### 2 exemples d'exécution

| fin de saisie avec CTRL-D ou CTRL-Z   | fin de saisie avec CR   |
|---|---|
| <pre>&gt;java lecture saisie :ABCD3 car. sont en attente. int lu :65 car. lu :A</pre> | <pre>&gt;java lecture saisie :ABCD 4car. sont en attente. int lu :65 car. lu :A</pre> |

## exemples d'utilisation des flots standards

```
import java.io.*;
public class majuscule2 {
static public void main(String[] args)
    {int x=0;
    System.out.print("saisie :");
    System.out.flush();
    try{
    do
        {x=System.in.read;
        System.out.println(Character.toUpperCase((char) x));}
    while (System.in.available()>0);
    System.out.println(System.in.available()+
                        " car. sont en attente.");}
    catch (IOException e)
        System.out.println(e.getMessage());
    } //main
} // class
```

### 2 exemples d'exécution

| fin de saisie avec CTRL-D ou CTRL-Z                                       | fin de saisie avec CR  |
|---|--|
| <pre>java majuscule2 saisie :abcdeA B C D E 0 car. sont en attente.</pre> | <pre>java majuscule2 saisie :abcde A B C D E 0 car. sont en attente.</pre> |

## variables d'environnement

### programme

```
import java.io.*;
import java.util.Properties;
public class demoProperties {
    static public void main(String[] args)
        {Properties p=System.getProperties();
        p.list(System.out);}
}
```

### exemple d'exécution :

```
-- listing properties --
java.home=/unige/java/SUNWjws/JDK/bin/..
java.version=1.0.2ss:08/01/96-23:00
file.separator=/
line.separator=

java.vendor=Sun Microsystems Inc.
user.name=legrand
os.arch=sparc
os.name=Solaris
java.vendor.url=http://www.sun.com/
user.dir=/user/u3/legrand
java.class.path=./unige/java/SUNWjws/JDK/bin/./clas...
java.class.version=45.3
os.version=2.x
path.separator=:user.home=/user/u3/legrand
```

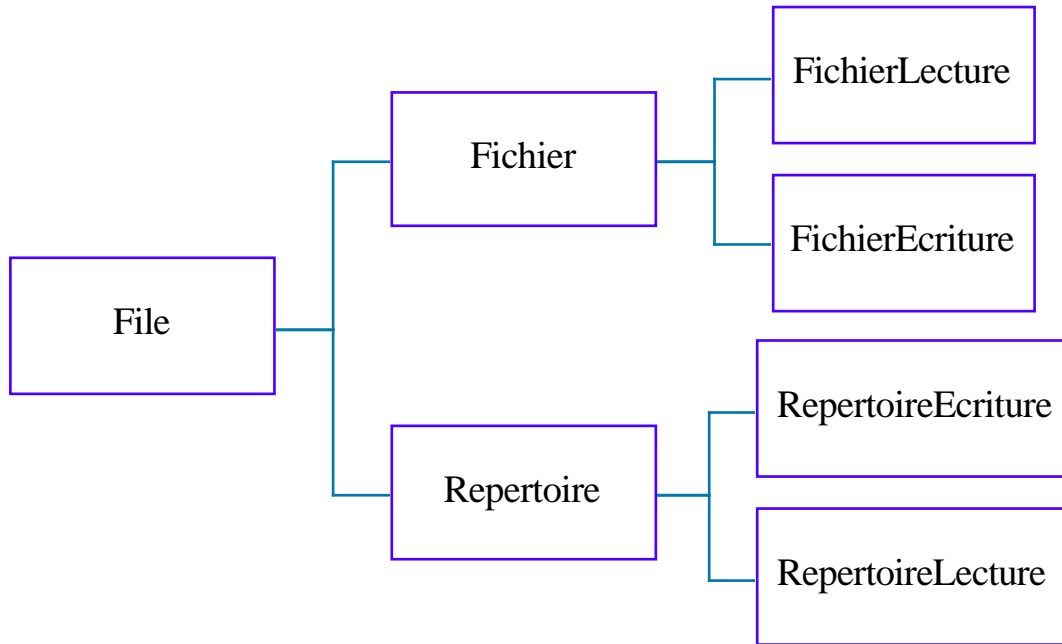
### accès au répertoire de travail

```
import java.io.*;
public class getWorkingDir {
    static public void main(String[] args)
        {System.out.println(System.getProperty("user.dir"));}
}
```

## classe File

```
public class java.io.File extends java.lang.Object {
// Fields
    public final static String pathSeparator;
    public final static char pathSeparatorChar;
    public final static String separator;
    public final static char separatorChar;
// Constructors
    public File(File dir, String name);
    public File(String path);
    public File(String path, String name);
// Methods
    public boolean canRead();
    public boolean canWrite();
    public boolean delete();
    public boolean equals(Object obj);
    public boolean exists();
    public String getAbsolutePath();
    public String getName();
    public String getParent();
    public String getPath();
    public int hashCode();
    public boolean isAbsolute();
    public boolean isDirectory();
    public boolean isFile();
    public long lastModified();
    public long length();
    public String[] list();
    public String[] list(FileNameFilter filter);
    public boolean mkdir();
    public boolean mkdirs();
    public boolean renameTo(File dest);
    public String toString();
}
```

## Des sous-classes de File utiles



```

import java.io.*;
public abstract class Fichier extends File{
    public Fichier(String s) throws NullPointerException
        {super(s);}
    public String RepertoireDe()
    {String NomComplet=this.getAbsolutePath();
    return NomComplet.substring(0,
        NomComplet.lastIndexOf(File.separator));
    }
}

```

```

import java.io.*;
public class FichierLecture extends Fichier{
    public FichierLecture(String s)
        throws IOException, NullPointerException
    {super(s);
    if (!this.exists() || !this.isFile())
        throw new IOException( this.getName() +
            ": fichier inexistant.");
    if (!this.canRead())
        throw new IOException( this.getName() +
            ": fichier non lisible.");
    }
}

```

## classe Repertoire

```

import java.io.*;
public abstract class Repertoire extends File {
    private static final int lim=20;

    public Repertoire(String s) throws IOException
    {super(s);}

    private void afficheUnFichier(String ref)
    {File unElement= new File(this,ref);
    System.out.print((unElement.canRead())? "L":" ");
    System.out.print((unElement.canWrite())? "E":" ");
    System.out.print(" "+ref);
    if (unElement.isDirectory())
        System.out.println("\t"+"repertoire");
    else if (unElement.canRead())
        System.out.println("\t"+unElement.length());
    else System.err.println(" ???");
    }

    public void afficheListe(String[] fichiers)
    {System.out.println(this.getName());
    if (fichiers.length==0)
        {System.out.println("aucun fichier");
        return;}
    File unElement;
    for(int i=0;i<fichiers.length;i++)
        {if ((i>0) && ((i % lim)==0))
            {System.out.println("<retour> pour continuer");
            try {System.in.read();}
            catch (IOException e){}}
        this.afficheUnFichier(fichiers[i]); };
    System.out.println(" "+fichiers.length+ " fichiers");
    }
}

```





## classe RepertoireLecture

```
import java.io.*;
public class RepertoireLecture extends Repertoire {
    public RepertoireLecture(String s) throws IOException
    {super(s);
    if (!this.exists() || !this.isDirectory())
        throw new IOException( this.getName() +
            ": repertoire inexistant.");
    if (!this.canRead())
        throw new IOException( this.getName() +
            ": repertoire non lisible.");
    }
}
```

## Définition d'un filtre

```
import java.io.*;
class FiltreSuffixe implements FilenameFilter{
    private String suffixe;
    public FiltreSuffixe(String suffixe)
        {this.suffixe=suffixe;}
    public boolean accept(File dir, String nom)
        {return nom.endsWith(suffixe);}
}
```

## Affichage du contenu d'un répertoire

```
import java.io.*;
public class AfficheRepertoire{
    public static void main(String args[])
    {Repertoire rep;
    String[] fichiers;
    try {
        rep= new RepertoireLecture(args[0]);
        switch (args.length)
        {case 1:
            fichiers=rep.list();
            break;
        case 2:
            FilenameFilter ff;
            ff= new FiltreSuffixe(args[1]);
            fichiers=rep.list(ff);
            break;
        default:
            System.out.println(
            "Usage: java AfficheRepertoire <nom> [filtre]");
            return;} // switch
        rep.afficheListe(fichiers);} // try
        catch (IOException e)
            {System.err.println(e.getMessage());}
    }
}
```



## copie de fichiers

```

import java.io.*;
public class CopieurFichier {

    public static void copie(String src,String dest)
    {FileInputStream s=null;
      FileOutputStream d=null;
      try {
        FichierLecture Source= new FichierLecture(src);
        FichierEcriture Destination=new FichierEcriture(dest);
        s=new FileInputStream(Source);
        d=new FileOutputStream(Destination);
        byte[] tampon=new byte[1024];
        int lu=0;
        do {lu=s.read(tampon);
          if (lu!=-1) d.write(tampon,0,lu);}
        while (lu!=-1);
      } // try
      catch (IOException e)
        {System.err.println(e.getMessage());}
      finally
        { if (s!=null)
          {try {s.close();} catch (IOException e){}}
          if (d!=null)
          {try {d.close();} catch (IOException e){}}
        } // finally
    } // copie

    public static void main(String args[])
    {if (args.length!=2)
      {System.err.println("Usage: java CopieurFichier <src>
<dest>");
        return;}
      copie(args[0],args[1]);
    }
}

```

## lecture filtrante d'un fichier

### définition du filtre

```
public class RechercheMot extends FilterInputStream {
    private String mot;
    private int lineNumber;
    DataInputStream fichier;

    private int getLineNumber() {return lineNumber;}

    public RechercheMot(DataInputStream fichier, String mot)
    {super(fichier);
     this.fichier=fichier;
     this.mot=mot;
     lineNumber=0; }

    public final String readLine() throws IOException
    {String line;
     do
     {line=fichier.readLine();
      lineNumber++;}
     while ((line!=null) && (line.indexOf(mot)==-1));
     return line;
    }
}
```

## lecture filtrante d'un fichier

### utilisation du filtre

```
import java.io.*;
public class Cherche {
    public static void main(String args[])
    {if (args.length!=2)
        {System.err.println(
            "Usage: java Cherche <mot> <fichier>");
        return;}
    try
    {FichierLecture f=new FichierLecture(args[1]);
    FileInputStream fis=new FileInputStream(f);
    DataInputStream d=new DataInputStream(fis);
    RechercheMotif g=new RechercheMotif(d,args[0]);
    String line;
    do
        {line=g.readLine();
        if (line!=null)
            System.out.println(g.getLineNumber()+
                ": "+line);
        }
    while (line!=null);
    g.close();
    }
    catch (IOException e)
        {System.err.println(e.getMessage());}
    } //main
} //class
```

**compteur de mots**

```
import java.io.*;
public class compteur {
    public static void main(String args[]){
        int nNombres=0;
        int nMots=0;
        FileInputStream fis=null;
        StreamTokenizer st=null;
        try
            {fis=new  FileInputStream(args[0]);
             st=new StreamTokenizer(fis);}
        catch (IOException e)
            {System.out.println(
                "usage: java compteur <filename>");
             try {fis.close();} catch (IOException e1){}
             System.exit(1);}
        // fin de ligne est un separateur d'unites syntaxiques
        st.eolIsSignificant(true);
        // '_' fait partie des mots
        st.wordChars((int) '_',(int) '_');
        // '.' ne fait partie d'aucune unité syntaxique
        st.ordinaryChar((int) '.');
        try
            while (st.nextToken()!=st.TT_EOF)
                {if (st.ttype== st.TT_WORD)
                    {System.out.println(st.sval); nMots++;}
                 if (st.ttype== st.TT_NUMBER)
                    {System.out.println(st.nval);nNombres++;}
                } // while
        catch (IOException e)
            {System.out.println("error when reading.");}
        finally
            {try {fis.close();} catch (IOException e1) {}}
        System.out.println("mots: "+ nMots);
        System.out.println("nombres: "+ nNombres);
    }
}
```