La clase File

Crear Directorios

```
import java.io.*;
import java.text.Format;
import java.text.SimpleDateFormat;
import java.util.Date;

public class CrearDirectorios {
    private final static String ayuda = "Ayuda:CrearDirectorios path1 ...\n"
            + "Crea cada path\n" + "Ayuda:CrearDirectorios -d path1 ...\n"
            + "Elimina cada path\n" + "Ayuda:CrearDirectorios -r path1 path2\n"
            + "Remonbra de path1 a path2\n";

    private static void ayuda() {
        System.err.println(ayuda);
        System.exit(1);
    }

    private static void datosFile(File f) {

        Format formateador = new SimpleDateFormat("yyyy.MM.dd HH.mm.ss");

        System.out.println("Absolute path: " + f.getAbsolutePath()
                + "\n Can read: " + f.canRead() + "\n Can write: "
                + f.canWrite() + "\n getName: " + f.getName()
                + "\n getParent: " + f.getParent() + "\n getPath: "
                + f.getPath() + "\n length: " + f.length()
                + "\n lastModified: "
                + formateador.format(new Date(f.lastModified())));
        if (f.isFile())
            System.out.println("Es un fichero");
        else if (f.isDirectory())
            System.out.println("Es un directorio");
    }
```

```
Ayuda:CrearDirectorios path1 ...
Crea cada path
Ayuda:CrearDirectorios -d path1 ...
Elimina cada path
Ayuda:CrearDirectorios -r path1 path2
Remonbra de path1 a path2
```

```
    public static void main(String[] args) {
        if (args.length < 1)
            ayuda();
        if (args[0].equals("-r")) {
            if (args.length != 3)
                ayuda();
            File anterior = new File(args[1]);
            File nuevo = new File(args[2]);
            anterior.renameTo(nuevo);
            datosFile(anterior);
            datosFile(nuevo);
            return; // Salir del main
        }
        int count = 0;
        boolean del = false;
        if (args[0].equals("-d")) {
            count++;
            del = true;
        }
        for (; count < args.length; count++) {
            File f = new File(args[count]);
            if (f.exists()) {
                System.out.println(f + " existe");
                if (del) {
                    System.out.println("borrando..." + f);
                    f.delete();
                }
            } else { // No Exite
                if (!del) {
                    f.mkdirs();
                    System.out.println("creado " + f);
                }
            }
            datosFile(f);
        }
    }
}
```

```
creado c:\miDirectorio
Absolute path: c:\miDirectorio
 Can read: true
 Can write: true
 getName: miDirectorio
 getParent: c:\
 getPath: c:\miDirectorio
 length: 0
 lastModified: 2005.05.07 13.53.53
Es un directorio
```

Copiar Fichero

e1071. Copying One File to Another

This example uses file streams to copy the contents of one file to another file.
channels.

```
// Copies src file to dst file.
// If the dst file does not exist, it is created
void copy(File src, File dst) throws IOException {
    InputStream in = new FileInputStream(src);
    OutputStream out = new FileOutputStream(dst);

    // Transfer bytes from in to out
    byte[] buf = new byte[1024];
    int len;
    while ((len = in.read(buf)) > 0) {
        out.write(buf, 0, len);
    }
    in.close();
    out.close();
}
```

```java
public class CopiarFichero {

    public static void copiar(String fuente, String destino) throws IOException {
        // Si el fichero fuente y el destino son el mismo fichero ...
        if (fuente.compareTo(destino) == 0)
            throw new ECopiarFichero("No puede sobreescribirse un "
                    + "fichero sobre sí mismo");
        // Definiciones de variables, referencias y objetos
        File fichFuente = new File(fuente);
        File fichDestino = new File(destino);
        FileInputStream fFuente = null;
        FileOutputStream fDestino = null;
        byte[] buffer;
        int nbytes;
        try {
            // Asegurarse de que "fuente" es un fichero, existe
            // y se puede leer.
            if (!fichFuente.exists() || !fichFuente.isFile())
                throw new ECopiarFichero("No existe el fichero " + fuente);
            if (!fichFuente.canRead())
                throw new ECopiarFichero("El fichero " + fuente
                        + " no se puede leer");
            // Si "destino" existe, asegurarse de que es un fichero que
            // se puede escribir y preguntar si se quiere sobreescribir.
            if (fichDestino.exists()) // ¿existe el destino?
            {
                if (fichDestino.isFile()) // ¿es un fichero?
                {
                    if (!fichDestino.canWrite())
                        throw new ECopiarFichero("No se puede escribir en "
                                + "el fichero " + destino);
                    // Indicar que el fichero existe y preguntar si se desea
                    // sobreescribir.
                    System.out.print("El fichero " + destino + " existe. "
                            + "¿Desea sobreescribirlo? (s/n): ");
                    // Leer la respuesta
                    char resp = (char) System.in.read();
                    System.in.skip(System.in.available());
                    if (resp == 'n' || resp == 'N')
                        throw new ECopiarFichero("Copia cancelada");
                } else
                    throw new ECopiarFichero(destino + " no es un fichero");
```

```
        } else // si "destino" no existe verificar que el directorio
        // padre existe y no está protegido contra escritura
        {
            File dirPadre = directorioPadre(fichDestino);
            if (!dirPadre.exists())
                throw new ECopiarFichero("El directorio " + destino
                        + " no existe");
            if (!dirPadre.canWrite())
                throw new ECopiarFichero("No se puede escribir en el "
                        + "directorio " + destino);
        }

        // Para realizar la copia, abrir un flujo de entrada desde
        // el fichero fuente y otro de salida hacia el destino.
        fFuente = new FileInputStream(fichFuente);
        fDestino = new FileOutputStream(fichDestino);
        buffer = new byte[1024];

        // Copiar el fichero fuente en el destino
        while (true) {
            nbytes = fFuente.read(buffer);
            if (nbytes == -1)
                break; // se llegó al final del fichero
            fDestino.write(buffer, 0, nbytes);
        }
    }
    // Cerrar cualquier flujo que esté abierto
    finally {
        try {
            if (fFuente != null)
                fFuente.close();
            if (fDestino != null)
                fDestino.close();
        } catch (IOException e) {
            System.out.println("Error: " + e.toString());
        }
    }
}
// File.getParent devuelve null si el fichero se especifica sin
// un directorio. El método siguiente trata este caso.
private static File directorioPadre(File f) {
    String nombreDir = f.getParent();
    if (nombreDir == null)
        // El método getProperty con el parámetro "user.dir" devuelve
        // el directorio actual de trabajo.
        return new File(System.getProperty("user.dir"));
    else
        // Devolver el directorio padre del fichero
        return new File(nombreDir);
}
```

```
    public static void main(String[] args) {
        // main debe recibir dos parámetros: el fichero fuente y
        // el destino.
        if (args.length != 2)
            System.err.println("Sintaxis: java CopiarFichero "
                    + "<fichero fuente> <fichero destino>");
        else {
            try {
                copiar(args[0], args[1]); // realizar la copia
            } catch (IOException e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
    }
}

// Si se produce un error durante la copia, se lanzará
// el siguiente tipo de excepción:
class ECopiarFichero extends IOException {
    public ECopiarFichero(String mensaje) {
        super(mensaje);
    }
}
```

Grep

```
public class Grep {

    public static void BuscarEnFich(String nombrefich, String cadena) {
        // Definiciones de variables
        File fichFuente = new File(nombrefich);
        BufferedReader flujoE = null;

        try {
            // Asegurarse de que el fichero, existe y se puede leer
            if (!fichFuente.exists() || !fichFuente.isFile()) {
                System.err.println("No existe el fichero " + nombrefich);
                return;
            }
            if (!fichFuente.canRead()) {
                System.err.println("El fichero " + nombrefich
                        + " no se puede leer");
                return;
            }

            // Abrir un flujo de entrada desde el fichero fuente
            FileInputStream fis = new FileInputStream(fichFuente);
            InputStreamReader isr = new InputStreamReader(fis);
            flujoE = new BufferedReader(isr);

            // Buscar cadena en el fichero fuente
            String linea;
            int nroLinea = 0;
            while ((linea = flujoE.readLine()) != null) {
                // Si se alcanzó el final del fichero,
                // readLine devuelve null
                nroLinea++; // contador de líneas
                if (BuscarCadena(linea, cadena))
                    System.out.println(nombrefich + " " + nroLinea + " "
                            + linea);
            }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            // Cerrar el flujo
            try {
                if (flujoE != null)
                    flujoE.close();
            } catch (IOException e) {
                System.out.println("Error: " + e.toString());
            }
        }
    }
```

```
    public static boolean BuscarCadena(String cadena1, String cadena2) {
        // ¿cadena2 está contenida en cadena1?
        if (cadena1.indexOf(cadena2) > -1)
            return true; // sí
        else
            return false; // no
    }

    public static void main(String[] args) {
        // main debe recibir dos o más parámetros: la cadena a buscar
        // y los ficheros fuente. Por ejemplo:
        // java Grep catch Grep.java Leer.java

        if (args.length < 2)
            System.err.println("Sintaxis: java Grep " + "<cadena> "
                    + "<fichero 1> <fichero 2> ...");
        else {
            for (int i = 1; i < args.length; i++)
                // Buscar args[0] en args[i]
                BuscarEnFich(args[i], args[0]);
        }
    }
}
```

==========

```
c:\Prueba\grep.java 7   public static boolean BuscarCadena(String cadena1, String cadena2)
c:\Prueba\grep.java 16   public static void BuscarEnFich(String nombrefich, String cadena)
c:\Prueba\grep.java 43       String linea;
c:\Prueba\grep.java 68         System.out.println("Error: " + e.toString());
c:\Prueba\grep.java 73   public static void main(String[] args)
```

```java
public class PruebaFlujoSalidaArchivoBasico {
    public static void main(String[] args) {
        try {
            File  f = new File ("c:\\prueba\\ejemplo.dat");
            FileOutputStream fos = new FileOutputStream(f);
//          FileOutputStream fos = new FileOutputStream(f,true);
            for (int i = 256; i < 266; i++) {
                fos.write(i);
            }
            for (byte i = 0; i < 10; i++) {
                fos.write(i);
            }
            fos.close();
        } catch (FileNotFoundException e) {e.printStackTrace();}
         catch (IOException e) {
            e.printStackTrace();
        }
        try {
            File  f = new File ("c:\\prueba\\ejemplo.dat");
            FileInputStream fis = new FileInputStream(f);
            int tamaño = (int) f.length();
            System.out.println("\n Disponible: "+fis.available());
            for (int i = 0; i < tamaño; i++) {
                System.out.print(fis.read()+" ");
            }
            System.out.println("\n Disponible: "+fis.available());
            fis.close();
        } catch (FileNotFoundException e1) {e1.printStackTrace();}
         catch (IOException e1) {
            e1.printStackTrace();
        }
        try {
            File f = new File("c:\\prueba\\ejemplo.dat");
            FileInputStream fis = new FileInputStream(f);
            System.out.println("\n Disponible: " + fis.available());
            System.out.print(fis.read() + " ");
            System.out.println("\n Disponible: " + fis.available());
            while (fis.available()>0) {
                System.out.print(fis.read() + " ");
            }
            fis.close();
        } catch (FileNotFoundException e1) {e1.printStackTrace();}
         catch (IOException e1) {
            e1.printStackTrace();
        }
    }
}
```

```
 Disponible: 20
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
 Disponible: 0

 Disponible: 20
0
 Disponible: 19
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
```

```java
public class PruebaFlujoSalidaArchivo {

    public static void main(String[] args) {

        try {
            File  f = new File ("c:\\prueba\\ejemplo.dat");
//          FileOutputStream fos = new FileOutputStream(f);
            FileOutputStream fos = new FileOutputStream(f,true);

            byte vectorBytes[]= {10,20,30,40,50,60,70,80};
            fos.write(vectorBytes);
            fos.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        try {
            File  f = new File ("c:\\prueba\\ejemplo.dat");
            FileInputStream fis = new FileInputStream(f);

            int tamaño = (int) f.length();
            byte vectorBytes[] = new byte[tamaño];
            fis.read(vectorBytes);

            for (int i = 0; i < vectorBytes.length; i++) {
                System.out.print(vectorBytes[i]+" ");
            }
            fis.close();
        } catch (FileNotFoundException e1) {
            e1.printStackTrace();
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
}
```

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 10 20 30 40 50 60 70 80
```

```java
public class PruebaDataStreams {

    public static void main(String[] args) {
          try {
            File f = new File("c:\\prueba\\ejemplo2.dat");
              FileOutputStream fos = new FileOutputStream(f);
              DataOutputStream dos  = new DataOutputStream(fos);

              dos.writeInt(987654321);
              dos.writeLong(11111111L);
              dos.writeFloat(22222222F);
              dos.writeDouble(3333333D);
              dos.writeChar('A');
              dos.writeBoolean(true);

              dos.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

          try {
            File f = new File("c:\\prueba\\ejemplo2.dat");
              FileInputStream fis = new FileInputStream(f);
              DataInputStream dis = new DataInputStream(fis);

              System.out.print(dis.readBoolean()+" ");
              System.out.print(dis.readInt()+" ");
              System.out.print(dis.readLong()+" ");
              System.out.print(dis.readFloat()+" ");
              System.out.print(dis.readDouble()+" ");
              System.out.print(dis.readChar()+" ");
              System.out.print(dis.readBoolean()+" ");

              dis.close();
        } catch (FileNotFoundException e1) {
            e1.printStackTrace();
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
}
```

```
true -563564288 2844444491 -6.162996E-14 5.426398515261209E45 ? java.io.EOFException
        at java.io.DataInputStream.readBoolean(DataInputStream.java:222)
        at com.practicas.octava.ficheros.PruebaDataStreams.main(PruebaDataStreams.java:50)
```

Listar directorios y archivos

```
    public void listarDirectorios(File dir) {
        if (dir.isDirectory()) {
            System.out.println(dir.getName());
            String[] hijos = dir.list();
            for (int i = 0; i < hijos.length; i++) {
                listarDirectorios(new File(dir, hijos[i]));
            }
        }
    }

    public void listarArchivos(File dir) {
        if (dir.isDirectory()) {
            String[] hijos = dir.list();
            for (int i = 0; i < hijos.length; i++) {
                listarArchivos(new File(dir, hijos[i]));
            }
        } else {
            System.out.println(dir.getName());
        }
    }
}
```

Ejemplos de archivos de texto (javaalmanac)

e35. Reading Text from a File

```
try {
    BufferedReader in = new BufferedReader(new FileReader( "infilename"));
    String str;
    while ((str = in.readLine()) != null) {
        process(str);
    }
    in.close();
} catch (IOException e) {
}
```

e37. Writing to a File

If the file does not already exist, it is automatically created.

```
try {
    BufferedWriter out = new BufferedWriter(new FileWriter( "outfilename"));
    out.write( "aString");
    out.close();
} catch (IOException e) {
}
```

e38. Appending to a File

```
try {
    BufferedWriter out = new BufferedWriter(new FileWriter( "filename", true));
    out.write( "aString");
    out.close();
} catch (IOException e) {
}
```

```java
import java.io.Serializable;
import java.util.Date;

public class Movimiento implements Serializable {

    private String mConcepto;
    private Date mFecha;
    private double mImporte;

    public Movimiento() {
        mFecha = new Date();
    }

    public double getImporte() {
        return mImporte;
    }

    public String getConcepto() {
        return mConcepto;
    }

    public void setConcepto(String concepto) {
        mConcepto = concepto;
    }

    public Date getFecha() {
        return mFecha;
    }

    public void setFecha(Date fecha) {
        mFecha = fecha;
    }

    public void setImporte(double importe) {
        mImporte = importe;
    }
}
```

```java
import java.io.Serializable;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Iterator;

public class CuentaCorriente implements Serializable {

    private String mNumero;
    private String mTitular;
    private ArrayList mMovimientos;

    public CuentaCorriente(String numero, String titular) {
        this.mNumero = numero;
        this.mTitular = titular;
        this.mMovimientos = new ArrayList();
    }

    public void ingresar(String concepto, double x) throws Exception {
        if (x <= 0)
            throw new Exception("No se puede ingresar una cantidad negativa");
        Movimiento m = new Movimiento();
        m.setConcepto(concepto);
        m.setImporte(x);
        this.mMovimientos.add(m);
    }

    public void retirar(String concepto, double x) throws Exception {
        if (x <= 0)
            throw new Exception("No se puede retirar una cantidad negativa");
        if (getSaldo() < x)
            throw new Exception("Saldo insuficiente");
        Movimiento m = new Movimiento();
        m.setConcepto(concepto);
        m.setImporte(-x);
        this.mMovimientos.add(m);
```

```
    public double getSaldo() {
        double saldo = 0.0;
        for (Iterator iter = mMovimientos.iterator(); iter.hasNext();) {
            Movimiento m = (Movimiento) iter.next();
            saldo += m.getImporte();
        }
        return saldo;
    }

    public void listado() {

        DateFormat formatter = new SimpleDateFormat("dd/MM/yyyy ");
        System.out.println("Titular    \t\tNúmero Cuenta");
        System.out.println("-----------\t\t-------------");
        System.out.println(mTitular +"\t\t"+mNumero);
        System.out.println();

        System.out.println("Fecha\t\t\tDescripcion\t\t\tPrecio");
        System.out.println("-----\t\t\t-----------\t\t\t-----");

        for (Iterator iter = mMovimientos.iterator(); iter.hasNext();) {
            Movimiento m = (Movimiento) iter.next();
            String s = formatter.format(m.getFecha()) + "\t\t"
                    + m.getConcepto() + "\t\t\t" + m.getImporte();
            System.out.println(s);
        }
    }

    public void addMovimiento(Movimiento m) {
        mMovimientos.add(m);
    }
}
```

```java
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

class CuentaCorrienteIO {

    private String nombreArchivo;

    public CuentaCorrienteIO(String nombreArchivo) {
        this.nombreArchivo = nombreArchivo;
    }

    public void escribir(CuentaCorriente cuenta) throws IOException {

        File f = new File(nombreArchivo);
        FileOutputStream fos = new FileOutputStream(f);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(cuenta);
        oos.close();
    }

    public CuentaCorriente leer() throws IOException {
        CuentaCorriente cuenta=null;

        File f = new File(nombreArchivo);
        FileInputStream fis = new FileInputStream(f);
        ObjectInputStream ois = new ObjectInputStream(fis);
        try {
            cuenta = (CuentaCorriente) ois.readObject();
        } catch (ClassNotFoundException e) {
        }
        ois.close();
        return cuenta;
    }

    public void setNombreArchivo(String nombreArchivo) {
        this.nombreArchivo = nombreArchivo;
    }
    public String getNombreArchivo() {
        return nombreArchivo;
    }
}
```

```java
public class PruebaCuentaCorriente {

    public static void main(String[] args) throws Exception {

        CuentaCorriente cuenta = new CuentaCorriente ("111-111", "Jose Pérez");
        CuentaCorrienteIO cuentaIO = new CuentaCorrienteIO("c:\\prueba\\cuenta.data");

        Movimiento m1 = new Movimiento();
        m1.setConcepto("concepto 1");
        m1.setImporte(-12.12);
        cuenta.addMovimiento(m1);

        Movimiento m2 = new Movimiento();
        m2.setConcepto("concepto 2");
        m2.setImporte(12.12);
        m2.setFecha(new Date());
        cuenta.addMovimiento(m2);

        System.out.println("Saldo: "+cuenta.getSaldo());
        cuenta.ingresar("ahorrillos", 8.0);
        System.out.println("Saldo: "+cuenta.getSaldo());

        cuentaIO.escribir(cuenta);
        System.out.println("Listado===========");
        cuenta=cuentaIO.leer();
        cuenta.listado();
    }
}
```

```
Saldo: 0.0
Saldo: 8.0
Listado===========
Titular                 Número Cuenta
-----------             -------------
Jose Pérez              111-111

Fecha                   Descripcion                     Precio
-----                   -----------                     -----
16/05/2005              concepto 1                      -12.12
16/05/2005              concepto 2                      12.12
16/05/2005              ahorrillos                      8.0
```