LISTAR FICHEROS DE UNA CARPETA, FILTRANDO

```
public class Filtrar implements FilenameFilter {
     String extension;
     // Constructor
     Filtrar(String extension){
          this.extension = extension;
     }
     public boolean accept(File dir, String name){
          return name.endsWith(extension);
     }
     public static void main(String[] args) {
          try {
            // Obtendremos el listado de los archivos de ese directorio
               File fichero=new File("c:\\datos\\.");
               String[] listadeArchivos = fichero.list();
            // Filtraremos por los de extension .txt
               listadeArchivos = fichero.list(new Filtrar(".txt"));
            // Comprobamos el número de archivos en el listado
               int numarchivos = listadeArchivos.length ;
            // Si no hay ninguno lo avisamos por consola
               if (numarchivos < 1)
                    System.out.println("No hay archivos que listar");
            // Y si hay, escribimos su nombre por consola.
               else
               {
                    for(int conta = 0; conta < listadeArchivos.length;
                                        conta++)
                         System.out.println(listadeArchivos[conta]);
               }
          }
          catch (Exception ex) {
               System.out.println("Error al buscar en la ruta indicada");
          }
     }
}
```

SEPARADOR DE RUTAS

```
String substFileSeparator(String ruta){
 String separador = "\\";
          try{
      // Si estamos en Windows
               if ( File.separator.equals(separador) )
                    separador = "/" ;
          // Reemplaza todas las cadenas que coinciden con la expresión
          // regular dada oldSep por la cadena File.separator
               return ruta.replaceAll(separador, File.separator);
          }catch(Exception e){
     // Por si ocurre una java.util.regex.PatternSyntaxException
               return ruta.replaceAll(separador + separador, File.separator);
          }
        }
```

CREADOR DE FICHEROS

```
try {
          // Creamos el objeto que encapsula el fichero
          File fichero = new File("c:\\prufba\\miFichero.txt");
          // A partir del objeto File creamos el fichero físicamente
          if (fichero.createNewFile())
                    System.out.println("El fichero se ha creado correctamente");
          else
                    System.out.println("No ha podido ser creado el fichero");
} catch (Exception ioe) {
          ioe.getMessage();
}
```

CREADOR DE DIRECTORIOS

```
try {
// Declaración de variables
         String directorio = "C:\\prueba";
         String varios = "carpeta1/carpeta2/carpeta3";

// Crear un directorio
         boolean exito = (new File(directorio)).mkdir();
         if (exito)
            System.out.println("Directorio: " + directorio + " creado");
// Crear varios directorios
        exito = (new File(varios)).mkdirs();
        if (exito)
            System.out.println("Directorios: " + varios + " creados");
}catch (Exception e){
        System.err.println("Error: " + e.getMessage());
}
```

EJEMPLO:

Dado el siguiente código, rellena lo que consideres que falte y escribe una sola línea al final del código que obtenga el listado de archivos de la "carpetaActual" que tienen como extensión ".xml".

```
File carpetaActual = new File(".");
String extension = ".xml";

FilenameFilter filtro = new FilenameFilter() {
        @Override
        public boolean accept(File carpetaActual, String nombreArchivo) {
                                return nombreArchivo.endwith(.extension);
        }
};
```

String[ ] archivosxml = carpetaActual.list(filtro);

CONECTARSE A UNA BASE DE DATOS CON JDBC

```
// Establece la conexion
        Connection con = DriverManager.getConnection
                 ("jdbc:odbc:miNombreDeBD",
                  "miLogin",
                  "miPassword" );

        // Ejecuta la consulta
        Statement stmt = (Statement) con.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT nombre, edad FROM Empleados");

        // Procesa los resultados
        while (rs.next()) {
                String nombre = rs.getString("nombre");
                int edad = rs.getInt("edad");
                                                }
```

CREACIÓN DE SESIÓN E INICIO DE TRANSACCIONES

```
 public void altaFutbolista(int numFut, int codigoEquipo, String nombre, String apellidos,
                        String dir, Integer edad, Integer goles,
Integer salario) {
 Equipos codEquipo = obtenerEquipoPorCodigo(codigoEquipo);
 if (codEquipo != null) {
 // Crear la conexión a la base de datos con Hibernate
 SessionFactory sesion = HibernateUtil.getSessionFactory();
 Session session = sesion.openSession();
 Transaction tx = session.beginTransaction();
 // Nueva inserción
 Futbolistas futbolista = new Futbolistas(numFut, codEquipo, nombre,
apellidos, dir, edad, goles, salario);
 try {
//Añadir futbolista, si fuese eliminar sería “session.delete(objeto)”, si fuese modificar
sería “session.update(objeto)”
 session.save(futbolista);
 tx.commit();
 JOptionPane.showMessageDialog(null, "Futbolista dado de alta
correctamente.");
 } catch (Exception e) {
 // Deshacer los cambios en caso de error
 tx.rollback();
 JOptionPane.showMessageDialog(null, "Error al dar de alta el
futbolista\n" + e.getLocalizedMessage());
 } finally {
 // Cerrar la conexión de Hibernate con la base de datos
 session.close();
 }
 } else {
 JOptionPane.showMessageDialog(null, "El equipo con el código " + codigoEquipo
+ " no existe.");
 }
 }
```

MINI REPASO SQL/HQL:

1. Genera una consulta HQL simple con FROM:

- Consulta: `FROM Usuario`

Respuesta: Devuelve todos los registros de la entidad Usuario.

---

2. Agrega una cláusula WHERE a la consulta anterior:

- Consulta: `FROM Usuario WHERE edad > 25`

Respuesta: Devuelve los registros de usuarios cuya edad es mayor a 25.

---

3. Agrega una cláusula SELECT a la consulta con FROM:

- Consulta: `SELECT nombre FROM Usuario`

Respuesta: Devuelve solo los nombres de los usuarios.

---

4. Incluye un JOIN en la consulta HQL:

- Consulta: `FROM Usuario u JOIN u.direccion d`

Respuesta: Devuelve registros de usuarios y sus direcciones relacionadas.

---

5. Añade un GROUP BY a la consulta anterior:

- Consulta: `SELECT u.nombre, COUNT(d) FROM Usuario u JOIN u.direccion d GROUP BY u.nombre`

Respuesta: Devuelve el nombre de cada usuario y la cantidad de direcciones asociadas a cada uno.

---

6. Incorpora una cláusula ORDER BY en la consulta HQL:

- Consulta: `FROM Usuario u ORDER BY u.nombre ASC`

Respuesta: Devuelve registros de usuarios ordenados alfabéticamente por nombre de manera ascendente.