Ejemplo: Accesos a un recurso compartido sin mecanismos de sincronización

Ejemplo que vamos a ver

Vamos a implementar dos aplicaciones:

1. Una aplicación que accede a un fichero que contiene un valor entero. Lee ese valor, lo incrementa en 1 y escribe ese valor actualizado en el mismo fichero. Esquemáticamente:

```
valor = LeerValorFichero(fichero);
valor++;
EscribirValorFichero(fichero, valor);
```

2. Una aplicación que crea varios procesos de la anterior. De forma que todos esos procesos utilizarán el mismo fichero para realizar esas sencillas instrucciones.

Veremos cómo probar el ejemplo. ¿El resultado final, es el esperado? ¿Cómo vemos qué es lo que ha pasado?

Para la implementación de estos ejemplos, hemos utilizado NetBeans y proyectos estándar de consola Java.

Aplicación 1:
Modificar el valor de un fichero.

1. En este caso, vamos a recordar cómo implementar una **aplicación sencilla**, que:
   1. **Lee** un valor de un fichero.
   2. **Incrementa** ese valor en uno.
   3. **Escribe** ese valor incrementado en el fichero del que lo había leído.
2. Por supuesto, **gestionaremos las posibles excepciones**, que se pueden producir en los accesos a un fichero.
3. **Comprobaremos que la ejecución** de la aplicación cumple los resultados que esperamos de ella; y que por lo tanto está **correcta**.

El proyecto NetBeans implementado se llama: AccesoMultipleFichero. Y, el archivo con el código fuente que vamos a mostrar aquí es el fichero Main.java de ese proyecto.

Aplicación 1:
Modificar el valor de un fichero.

Comencemos con las partes básicas:

- **Leer de fichero**:

```
archivo = new File(nombreFichero);
//Preparamos el acceso al fichero
//Leemos de fichero
try{
   leer = new FileReader(archivo);
   br = new BufferedReader(leer);
   // Lectura del fichero
   String linea;
   linea = br.readLine();
   valor = Integer.parseInt(linea);
   System.out.println("Proceso"+ orden +
            ": Valor leído del fichero: " + valor);
}catch(Exception e){
   System.err.println("P"+orden+" Error al leer del fichero");
}finally{
   // En el finally cerramos el escribir, para asegurarnos
   // que se cierra tanto si todo va bien como si salta
   // una excepcion.
   try{
      if( null != leer ){
         leer.close();
      }
   }catch (Exception e2){
      System.err.println("P"+orden+" Error al cerrar el fichero");
      System.exit(1);  //Si hay error, finalizamos
   }
}
```

- **Incrementar el valor**:

```
//Incrementamos
valor ++;
```

Aplicación 1:
Modificar el valor de un fichero.

Para finalizar con las partes básicas:

- **Escribir en fichero**:

```
//Escribimos en fichero
try {
      escribir = new FileWriter(nombreFichero);
      pw = new PrintWriter(escribir);
      pw.println(String.valueOf(valor));
      System.out.println("Proceso"+ orden +
            ": Valor escrito en el fichero: " + valor);
}
catch(Exception e){
   System.err.println("P"+orden+" Error al escribir en el fichero");
}finally{
  try{
     // Nos aseguramos de que se cierra el fichero.
     if (null != escribir)
        escribir.close();
     } catch (Exception e2) {
        System.err.println("P"+orden+" Error al cerrar el fichero");
        System.exit(1);  //Si hay error, finalizamos
     }
}
```

Aplicación 1:
Modificar el valor de un fichero.

También hemos incluido en al comienzo, el código necesario para crear el fichero en el caso de que no exista:

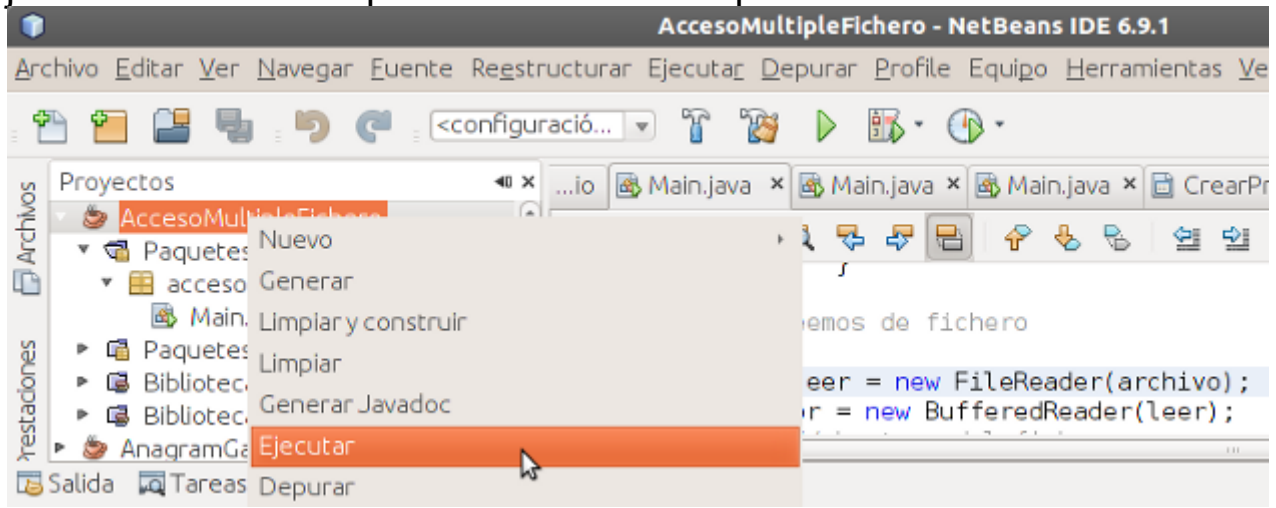
- **Crear el fichero si no existe**:

```
archivo = new File(nombreFichero);
//Preparamos el acceso al fichero
if (!archivo.exists()){
    //Si no existe el fichero
  try {
      archivo.createNewFile(); //Lo creamos
      escribir = new FileWriter(nombreFichero);
      pw = new PrintWriter(escribir);
      pw.println(String.valueOf(0)); //Escribimos el valor 0 en el fichero
      System.out.println("Proceso"+ orden + ": Creando el fichero.");
  }catch(Exception e){
      System.err.println("P"+orden+" Error al crear el fichero");
  }finally{
      try{
      // Nos aseguramos que se cierra el fichero.
          if (null != escribir)
              escribir.close();
         } catch (Exception e2) {
              System.err.println("Error al cerrar el fichero");
              System.exit(1); //Si hay error, finalizamos
          }
      }
}
```

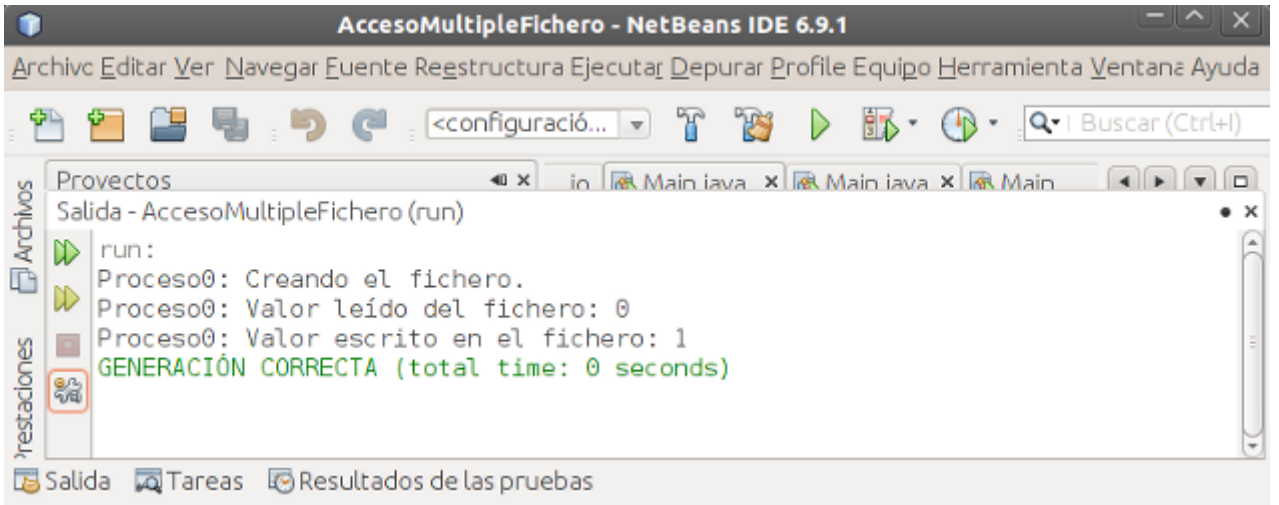
Probar nuestra aplicación de acceso a fichero

Probamos nuestro ejemplo:

1. Lanzar la ejecución de nuestro proceso AccesoMultipleFichero.



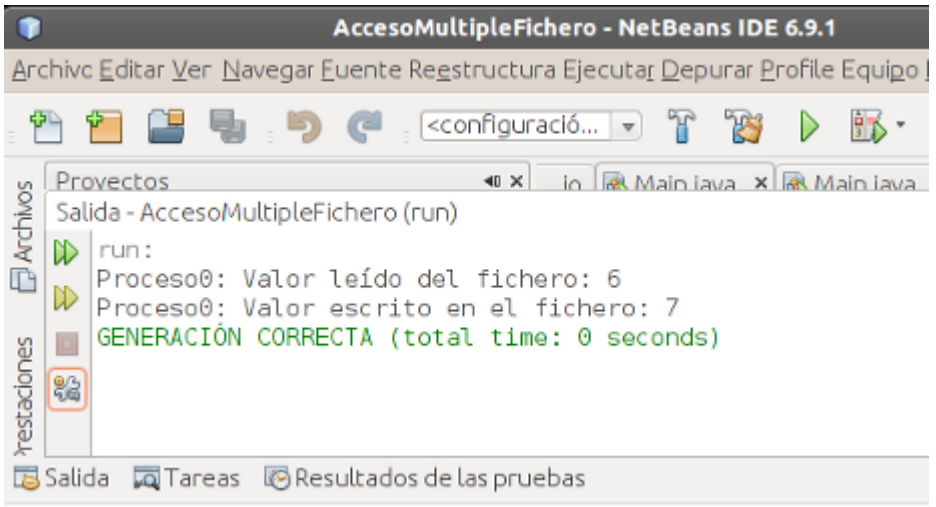
2. Comprobamos la salida de la ejecución.



Probar nuestra aplicación de acceso a fichero

Seguimos probando nuestro ejemplo:

3. Lanzamos **varias ejecuciones** de nuestro proceso AccesoMultipleFichero.



4. Y el **contenido del fichero**, es **el esperado**:

5. Con estas pruebas, podemos decir que **la aplicación funciona como nosotros esperamos**.

Aplicación 2:
Lanzar la ejecución de varias instancias de la aplicación 1.

Queremos probar, que, aunque la aplicación 1 es correcta, como hemos visto. Si se ejecuta con otros procesos que intenten acceder de forma concurrente a ese mismo fichero, el valor final que contiene el fichero, no es el que debería ser; y por lo tanto, **la implementación de la aplicación no es concurrentemente correcta**.

- En el ejemplo PSP01_CONT_R023_CreacionProcesos, ya vimos cómo crear varios procesos de la misma aplicación. En este caso, vamos a utilizar el mismo código, con la salvedad, de que el fichero .jar que queremos lanzar, se encontrará en el mismo directorio que el .jar de esta aplicación.

```
Process nuevoProceso; //Definimos una variable de tipo Process
try{
    for (int i = 0; i <=20; i++){
        nuevoProceso = Runtime.getRuntime().exec("java -jar "+
            "AccesoMultipleFichero.jar " + i + " nuevo.txt");
        //Creamos el nuevo proceso y le indicamos el número de orden y
        //el fichero que debe utilizar.
        System.out.println("Creado el proceso " + i);
        //Mostramos en consola que hemos creado otro proceso

    }
}catch (SecurityException ex){
    System.err.println("Ha ocurrido un error de Seguridad."+
            "No se ha podido crear el proceso por falta de permisos.");
}catch (Exception ex){
    System.err.println("Ha ocurrido un error, descripción: "+
            ex.toString());
}
```

Podemos apreciar, en el comando con el que lanzamos la ejecución del proceso, que hemos incluido paso de parámetros por la línea de comandos. Es una pequeña modificación que utilizaremos para poder conocer mejor, qué es lo que está sucediendo en nuestro sistema.

Aplicación 1:
Incluir código para tratar los argumentos recibidos en la línea de comandos.

Como hemos visto en el código anterior, la aplicación 2 pasará parámetros por línea de comandos a los procesos de la aplicación 1. Hemos decidido que los procesos reciban los siguientes parámetros:

1. El número de orden de creación de ese proceso.
2. El nombre o ruta del archivo que queremos que utilice.

Con estos dos parámetros, buscamos poder ver de forma más clara qué sucede durante la ejecución de los procesos, y poder cambiar el fichero al que acceden los ficheros, para hacer distintas pruebas.

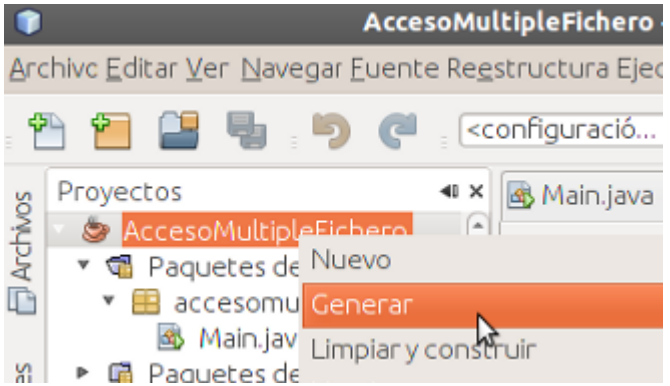
El código para tratar los argumentos recibidos por la línea de comandos, sería:

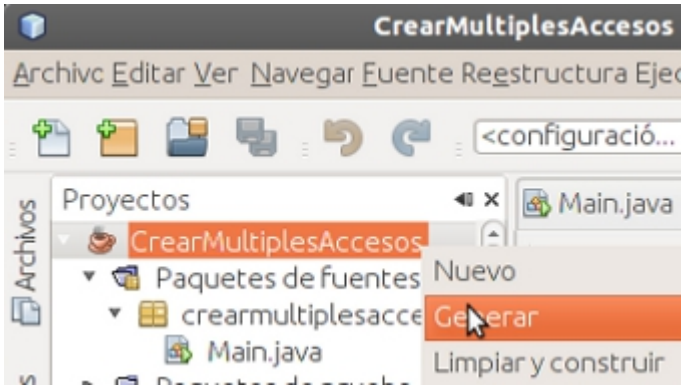
```
//Comprobamos si estamos recibiendo argumentos en la línea de comandos
if (args.length > 0){
    orden = Integer.parseInt(args[0]);
  }
//Identificamos el sistema operativo para poder acceder por su ruta al
//fichero de forma correcta.
String osName = System.getProperty("os.name");
if (osName.toUpperCase().contains("WIN")){ //Windows
  if (args.length > 1)
      nombreFichero = args[1].replace("\\", "\\\\");
      //Hemos recibido la ruta del fichero en la línea de comandos
  else{
      nombreFichero = "C:\\valor.txt";
      //Fichero que se utilizará por defecto
    }
}else{ //GNU/Linux
    if (args.length > 1)
      nombreFichero = args[1];
    //Hemos recibido la ruta del fichero en la línea de comandos
    else{
        nombreFichero = "/home/margye/valor.txt";
        //Fichero que se utilizará por defecto
  }
}
```

Probar la ejecución con varios procesos

Para probar nuestro ejemplo con los procesos que lo forman, haremos lo siguiente:

1. Generar los ficheros .jar de ambos proyectos.





2. Copiar los archivos .jar de ambos proyectos en la misma carpeta.
3. Lanzar la ejecución desde un terminal de comandos.

Lanzamos la ejecución desde línea de comandos, para que los ejecutables tomen como directorio de trabajo, el directorio en el que nos encontramos (creará el fichero al que van a acceder los procesos en nuestro directorio actual de trabajo).

En Windows:

```
C:\Users\usuario\AccesoMultiplesFichero>java -jar
 CrearMultiplesAccesos.jar
Creado el proceso 0
Creado el proceso 1
Creado el proceso 2
Creado el proceso 3
Creado el proceso 4
Creado el proceso 5
Creado el proceso 6
Creado el proceso 7
Creado el proceso 8
Creado el proceso 9
Creado el proceso 10
Creado el proceso 11
Creado el proceso 12
Creado el proceso 13
```

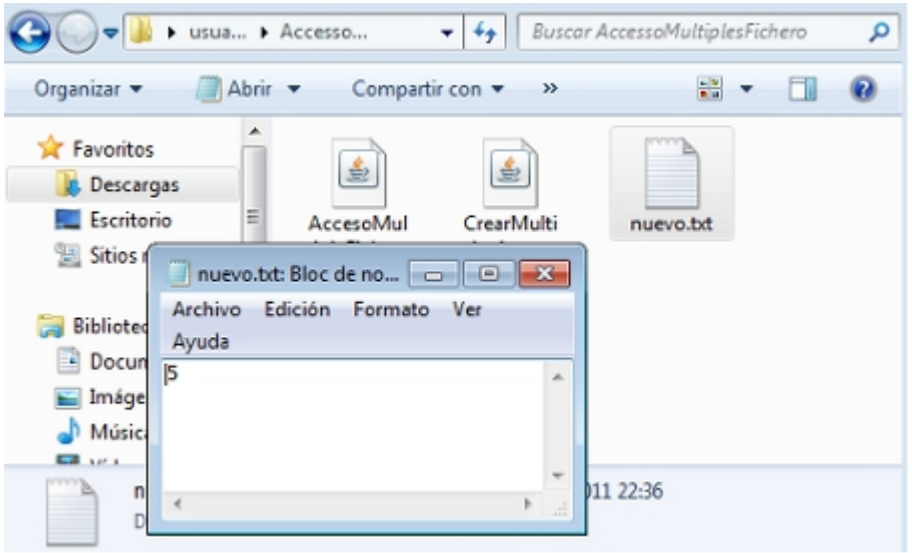
En GNU/Linux:

```
usuario@Pearl:~/AccesosFicherosSinSincro$ java -jar
 CrearMultiplesAccesos.jar
Creado el proceso 0
Creado el proceso 1
Creado el proceso 2
Creado el proceso 3
Creado el proceso 4
Creado el proceso 5
Creado el proceso 6
```

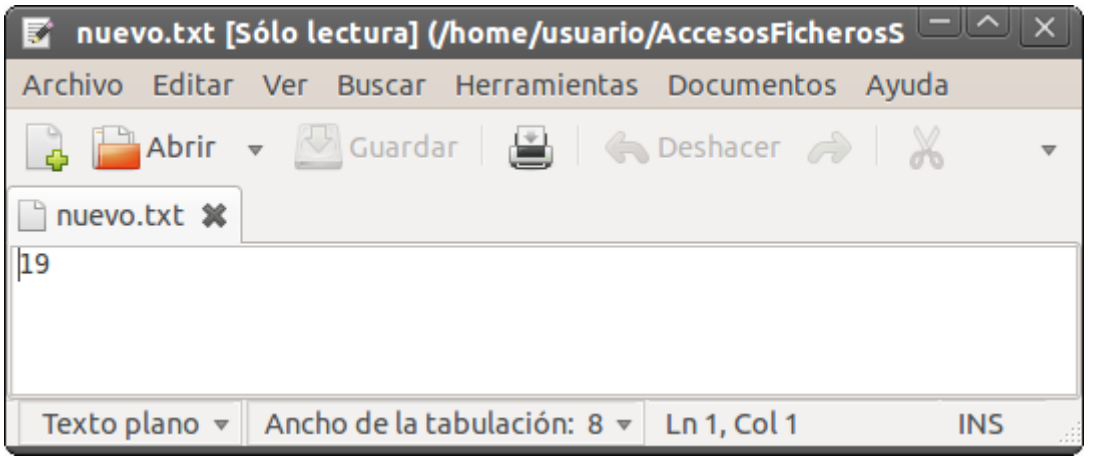
Probar la ejecución con varios procesos

Revisamos el contenido del fichero que han estado utilizando:

En Windows:



En GNU/Linux:



Pero..., ¿qué ha pasado? ¿Cómo es posible que tenga esos valores? Y, si lanzamos la ejecución varias veces... Los valores son distintos a los esperados, siempre, o casi siempre.

Si hemos creado 21 procesos y cada uno incrementa en uno el valor anterior, al final debería ser 21.

¿Cómo podemos ver lo que han hecho esos procesos?

Vamos a incluir una pequeña modificación en nuestras dos aplicaciones, para que todos los procesos escriban su salida en un fichero que después podamos revisar nosotros.

Aplicación 2:
Escribir la salida estándar en un fichero.

Vamos a hacer que la salida estándar de las aplicaciones estén redirigidas a un mismo fichero (javalog.txt en el directorio de ejecución).

- El código es el mismo, lo incluimos al comienzo de cada aplicación.
- Comenzamos con la aplicación que crea los procesos:

```
public static void main(String[] args) {
    Process nuevoProceso; //Definimos una variable de tipo Process
    try{
        PrintStream ps = new PrintStream(
                         new BufferedOutputStream(new FileOutputStream(
                         new File("javalog.txt"),true)), true);
        System.setOut(ps);
        System.setErr(ps);
        for (int i = 0; i <=20; i++){
            nuevoProceso = Runtime.getRuntime().exec("java -jar "+
                "AccesoMultipleFichero.jar " + i + " nuevo.txt");
            //Creamos el nuevo proceso y le indicamos el número de orden y
            //el fichero que debe utilizar.
            System.out.println("Creado el proceso " + i);
            //Mostramos en consola que hemos creado otro proceso
        }
    }catch (SecurityException ex){
        System.err.println("Ha ocurrido un error de Seguridad."+
                "No se ha podido crear el proceso por falta de permisos.");
    }catch (Exception ex){
        System.err.println("Ha ocurrido un error, descripción: "+
                ex.toString());
    }
}
```

- En el constructor del objeto FileOutputStream(), indicamos en su segundo argumento que queremos que se cree el fichero para añadir siempre al final del fichero.
- No habrá que modificar las escrituras en la salida estándar y la salida de error de la aplicación.

Aplicación 1: Escribir la salida estándar en un fichero.

Vamos a hacer que la salida estándar de las aplicaciones estén redirigidas a un mismo fichero (javalog.txt en el directorio de ejecución).

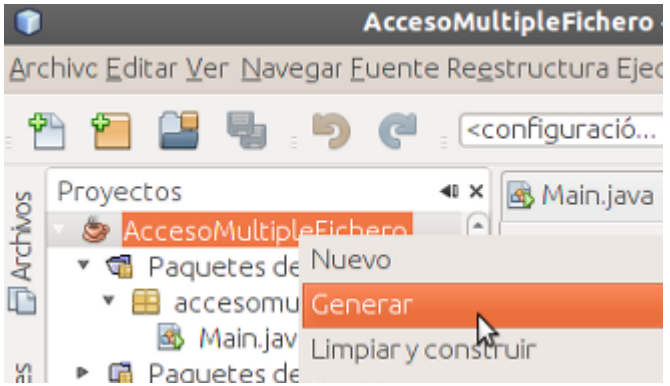
- El código es el mismo, lo incluimos al comienzo de cada aplicación.
- En el constructor del objeto FileOutputStream(), indicamos en su segundo argumento que queremos que se cree el fichero para añadir siempre al final del fichero.
- No habrá que modificar las escrituras en la salida estándar y la salida de error de la aplicación.

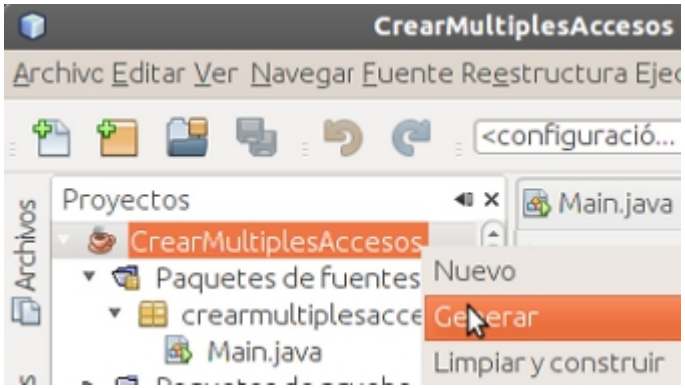
```
/**
 * @param args Argumentos de la línea de comando
 * El primer argumento pasado, será el número de orden de creación del proceso
 * El segundo argumento, será la ruta del fichero
 */
public static void main(String[] args) {
  int orden = 0;
  String nombreFichero = "";
  File archivo = null;
  FileReader leer = null;
  BufferedReader br = null;
  FileWriter escribir = null;
  PrintWriter pw = null;
  int valor = 0;
  //Comprobamos si estamos recibiendo argumentos en la línea de comandos
  if (args.length > 0)
      orden = Integer.parseInt(args[0]);
      //Número de orden de creación de este proceso
  try{
      PrintStream ps = new PrintStream(
                       new BufferedOutputStream(new FileOutputStream(
                       new File("javalog.txt"),true)), true);
        System.setOut(ps);
        System.setErr(ps);
  }catch(Exception e){
        System.err.println("P"+orden+" No he podido redirigir salidas.");
  }
  //Identificamos el sistema operativo para poder acceder por su ruta al
  //fichero de forma correcta.
  String osName = System.getProperty("os.name");
  if (osName.toUpperCase().contains("WIN")){ //Windows
    if (args.length > 1)
        nombreFichero = args[1].replace("\\", "\\\\");
        //Hemos recibido la ruta del fichero en la línea de comandos
    else{
```

Probar la ejecución con varios procesos escribiendo resultados en un fichero.

Para probar nuestro ejemplo con los procesos que lo forman, haremos lo siguiente:

1. Volvemos a generar los ficheros .jar de ambos proyectos.





2. Volvemos a copiar los archivos .jar de ambos proyectos en la misma carpeta.
3. Lanzamos la ejecución desde un terminal de comandos.

Lanzamos la ejecución desde línea de comandos, para que los ejecutables tomen como directorio de trabajo, el directorio en el que nos encontramos (se crearán los ficheros en el directorio actual).

En Windows:

```
C:\Users\usuario\AccessoMultiplesFichero>java -jar CrearMultiplesAccesos.jar

C:\Users\usuario\AccessoMultiplesFichero>dir
20/08/2011  12:42             4.900 AccesoMultipleFichero.jar
20/08/2011  10:24             2.529 CrearMultiplesAccesos.jar
20/08/2011  13:00             2.209 javalog.txt
20/08/2011  13:00                 4 nuevo.txt
               4 archivos          9.642 bytes
```

En GNU/Linux:

```
usuario@Pearl:~/AccesosFicherosSinSincro$ java -jar CrearMultiplesAccesos.jar
usuario@Pearl:~/AccesosFicherosSinSincro$ ls
AccesoMultipleFichero.jar  CrearMultiplesAccesos.jar  javalog.txt  nuevo.txt
usuario@Pearl:~/AccesosFicherosSinSincro$
```

Ver lo que ha estado pasando

Igual que nos pasaba antes, el valor final, no es el esperado. Revisemos el contenido del fichero javalog.txt:

En Windows:

```
Creado el proceso 17
Creado el proceso 18
Creado el proceso 19
Creado el proceso 20
Proceso0: Creando el fichero.
Proceso0: Valor leído del fichero: 0
Proceso0: Valor escrito en el fichero: 1
Proceso1: Valor leído del fichero: 1
Proceso1: Valor escrito en el fichero: 2
Proceso7: Valor leído del fichero: 2
Proceso7: Valor escrito en el fichero: 3
Proceso14: Valor leído del fichero: 3
Proceso14: Valor escrito en el fichero: 4
Proceso15: Valor leído del fichero: 3
Proceso15: Valor escrito en el fichero: 4
Proceso16: Valor leído del fichero: 4
Proceso16: Valor escrito en el fichero: 5
Proceso11: Valor leído del fichero: 5
Proceso11: Valor escrito en el fichero: 6
Proceso6: Valor leído del fichero: 6
Proceso6: Valor escrito en el fichero: 7
Proceso5: Valor leído del fichero: 7
Proceso5: Valor escrito en el fichero: 8
Proceso17: Valor leído del fichero: 8
Proceso18: Valor leído del fichero: 8
Proceso18: Valor escrito en el fichero: 9
Proceso13: Valor leído del fichero: 9
Proceso13: Valor escrito en el fichero: 10
Proceso10: Valor leído del fichero: 10
Proceso10: Valor escrito en el fichero: 11
Proceso9: Valor leído del fichero: 11
Proceso9: Valor escrito en el fichero: 12
Proceso17: Valor escrito en el fichero: 9
Proceso4: Valor leído del fichero: 8
Proceso4: Valor escrito en el fichero: 9
Proceso8: Valor leído del fichero: 9
Proceso8: Valor escrito en el fichero: 10
Proceso19: Valor leído del fichero: 10
Proceso20: Valor leído del fichero: 10
Proceso3: Valor leído del fichero: 10
Proceso2: Valor leído del fichero: 10
Proceso12: Valor leído del fichero: 10
Proceso19: Valor escrito en el fichero: 11
Proceso20: Valor escrito en el fichero: 11
Proceso3: Valor escrito en el fichero: 11
Proceso2: Valor escrito en el fichero: 11
Proceso12: Valor escrito en el fichero: 11
```

En GNU/Linux:

```
Creado el proceso 11
Proceso3: Valor escrito en el fichero: 3
Proceso2: Valor leído del fichero: 3
Proceso2: Valor escrito en el fichero: 4
Creado el proceso 12
Creado el proceso 13
Proceso4: Valor leído del fichero: 4Creado el proceso 14
Creado el proceso 15

Proceso4: Valor escrito en el fichero: 5
Creado el proceso 16
Proceso6: Valor leído del fichero: 5
Proceso6: Valor escrito en el fichero: 6
Proceso5: Valor leído del fichero: 6
Proceso5: Valor escrito en el fichero: 7
Creado el proceso 17
Proceso7: Valor leído del fichero: 7
Proceso7: Valor escrito en el fichero: 8
Proceso9: Valor leído del fichero: 8
Creado el proceso 18
Proceso9: Valor escrito en el fichero: 9
Creado el proceso 19
Creado el proceso 20
Proceso10: Valor leído del fichero: 9Proceso8: Valor leído
del fichero: 9
Proceso8: Valor escrito en el fichero: 10

Proceso10: Valor escrito en el fichero: 10
Proceso12: Valor leído del fichero: 10
Proceso12: Valor escrito en el fichero: 11
```

Ver lo que ha estado pasando

- Al fijarnos en las capturas anteriores, podemos darnos cuenta de cómo varios procesos leen el mismo valor del fichero en lugar de leer cada uno un valor distinto.
- Podemos ver claramente, cómo unos procesos ejecutan unas instrucciones entre las de otros, dependiendo de lo que decida el planificador de procesos del sistema operativo.
- Cada vez que hagamos una prueba de ejecución. El resultado que se produzca será indeterminado, dependiendo de la carga del sistema, el planificador, etc.

Nuestra aplicación no se puede utilizar en un entorno concurrente, con varios procesos accediendo al mismo recurso. **La aplicación no está implementada para tener en cuenta un entorno de ejecución concurrente. No es correcta.**

¿Cómo lo solucionamos?

```
Creado el proceso 17
Creado el proceso 18
Creado el proceso 19
Creado el proceso 20
Proceso0: Creando el fichero.
Proceso0: Valor leído del fichero: 0
Proceso0: Valor escrito en el fichero: 1
Proceso1: Valor leído del fichero: 1
Proceso1: Valor escrito en el fichero: 2
Proceso7: Valor leído del fichero: 2
Proceso7: Valor escrito en el fichero: 3
Proceso14: Valor leído del fichero: 3
Proceso14: Valor escrito en el fichero: 4
Proceso15: Valor leído del fichero: 3
Proceso15: Valor escrito en el fichero: 4
Proceso16: Valor leído del fichero: 4
Proceso16: Valor escrito en el fichero: 5
Proceso11: Valor leído del fichero: 5
Proceso11: Valor escrito en el fichero: 6
Proceso6: Valor leído del fichero: 6
Proceso6: Valor escrito en el fichero: 7
Proceso5: Valor leído del fichero: 7
Proceso5: Valor escrito en el fichero: 8
Proceso17: Valor leído del fichero: 8
Proceso18: Valor leído del fichero: 8
Proceso18: Valor escrito en el fichero: 9
Proceso13: Valor leído del fichero: 9
Proceso13: Valor escrito en el fichero: 10
Proceso10: Valor leído del fichero: 10
Proceso10: Valor escrito en el fichero: 11
Proceso9: Valor leído del fichero: 11
Proceso9: Valor escrito en el fichero: 12
Proceso17: Valor escrito en el fichero: 9
Proceso4: Valor leído del fichero: 8
Proceso4: Valor escrito en el fichero: 9
Proceso8: Valor leído del fichero: 9
Proceso8: Valor escrito en el fichero: 10
Proceso19: Valor leído del fichero: 10
Proceso20: Valor leído del fichero: 10
Proceso3: Valor leído del fichero: 10
Proceso2: Valor leído del fichero: 10
Proceso12: Valor leído del fichero: 10
Proceso19: Valor escrito en el fichero: 11
Proceso20: Valor escrito en el fichero: 11
Proceso3: Valor escrito en el fichero: 11
Proceso2: Valor escrito en el fichero: 11
Proceso12: Valor escrito en el fichero: 11
```

Ver lo que ha estado pasando

- Al fijarnos en las capturas anteriores, podemos darnos cuenta de cómo varios procesos leen el mismo valor del fichero en lugar de leer cada uno un valor distinto.
- Podemos ver claramente, cómo unos procesos ejecutan unas instrucciones entre las de otros, dependiendo de lo que decida el planificador de procesos del sistema operativo.
- Cada vez que hagamos una prueba de ejecución. El resultado que se produzca será indeterminado, dependiendo de la carga del sistema, el planificador, etc.

La aplicación no está implementada para tener en cuenta un entorno de ejecución concurrente. No es correcta.

¿Cómo lo solucionamos?

```
Creado el proceso 11
Proceso3: Valor escrito en el fichero: 3
Proceso2: Valor leído del fichero: 3
Proceso2: Valor escrito en el fichero: 4
Creado el proceso 12
Creado el proceso 13
Proceso4: Valor leído del fichero: 4Creado el proceso 14
Creado el proceso 15

Proceso4: Valor escrito en el fichero: 5
Creado el proceso 16
Proceso6: Valor leído del fichero: 5
Proceso6: Valor escrito en el fichero: 6
Proceso5: Valor leído del fichero: 6
Proceso5: Valor escrito en el fichero: 7
Creado el proceso 17
Proceso7: Valor leído del fichero: 7
Proceso7: Valor escrito en el fichero: 8
Proceso9: Valor leído del fichero: 8
Creado el proceso 18
Proceso9: Valor escrito en el fichero: 9
Creado el proceso 19
Creado el proceso 20
Proceso10: Valor leído del fichero: 9Proceso8: Valor leído
del fichero: 9
Proceso8: Valor escrito en el fichero: 10

Proceso10: Valor escrito en el fichero: 10
Proceso12: Valor leído del fichero: 10
Proceso12: Valor escrito en el fichero: 11
```

Recomendaciones para probar el ejemplo

Si no has ido construyendo paso a paso el ejemplo, puedes probar el ejemplo que te suministramos implementado en la plataforma.

Recuerda:

1. Generar cada uno de los proyectos de forma independiente.

2. Copiar los archivos .jar resultantes de la generación en el mismo directorio.

3. Lanzar desde el terminal de comandos el ejecutable CrearMultiplesAccesos.jar.
   - El comando será: java -jar CrearMultiplesAccesos.jar

4. Abrir los archivos nuevo.txt y javalog.txt que generan la ejecución de este ejemplo.

Credenciales

Imagen	Datos de licencia
Todas las capturas de pantalla de este presentación, tienen como datos de licencia: Autoría: Margarita I. Nieto Castillejo Licencia: Uso educativo-no comercial. Procedencia: Capturas de pantalla del IDE NetBeans 6.9.1; terminal de comandos, explorador de ficheros y Block de notas de Windows 7; consola de comandos, Nautilus y gedit de Ubuntu 10.10.	