DISEÑO DE INTERFACES DE USUARIO – CURSO 2018/19

PRÁCTICA 8 – COMPRIMIR ARCHIVOS EN FORMATO ZIP

Y llegó el momento de ver cómo podemos construir Interfaces de Usuario para aplicaciones que hacen tareas intensas en segundo plano. En estos casos, la Interfaz de Usuario no debe perder el control de la ejecución, debe mantener al Usuario informado en todo momento de cuál es el estado, y nivel de progreso, en el que se encuentra la tarea, así como poder detenerla si el Usuario así lo estima necesario.

Java Swing nos ofrece la clase *SwingWorker* para poner ejecutar nuestros procesos en segundo plano. De esta forma, la Interfaz de Usuario no se queda bloqueada hasta la finalización de un proceso y puede seguir actualizándose, informando al Usuario, o interactuando con él (por ejemplo, para recibir la petición de cancelación del proceso).

Otra clase que debemos conocer es *JProgressBar*, que permite mostrar al usuario una Barra de Progreso de la tarea.

La utilización conjunta de las dos anteriores clases, y de todos los conocimientos que ya hemos adquirido en las prácticas anteriores (en especial en la Práctica 5 con el uso de la clase *JFileChosser*), nos permitirá realizar la Interfaz de Usuario de una Aplicación que comprima a un archivo zip, todos los archivos que se encuentran en una carpeta determinada.

Para generar el archivo zip utilizaremos el paquete de clases *java.util.zip* que proporciona java, (se incorpora un ejemplo de utilización de dichas clases en el apartado de “Notas de Implementación”).

La funcionalidad que deberá tener la aplicación a desarrollar en esta práctica será la siguiente:

- Permitir al usuario seleccionar una **carpeta existente** (uso de la clase *JFileChosser*), la cual será la carpeta que contiene todos los archivos a comprimir.
- Permitir al usuario seleccionar una **carpeta existente de salida** (uso de la clase *JFileChosser*), la cual será la carpeta donde se genera el archivo **folder.zip**.
- Una vez seleccionadas la carpetas, y comprobada que nos son la misma carpeta (por razones obvias), la aplicación debe permitir mediante un botón (*JButton*) **iniciar la tarea de comprimir los archivos en segundo plano** (uso de la clase *SwingWorker*).
- Se debe mostrar el progreso de la tarea que está comprimiendo los archivos mediante una Barra de Progreso (uso de la clase *JProgressBar*).
- La aplicación debe permitir al usuario cancelar la generación del archivo zip mediante un botón de Cancelar (JButton), el cual sólo estará habilitado cuando se esté generando el archivo comprimido zip.

Notas de implementación

Ya hemos comentado anteriormente que Java permite comprimir archivos en formato zip mediante la utilización de las clases que se encuentran en el paquete *java.util.zip*. Se muestra a continuación un ejemplo de cómo se pueden comprimir todos los archivos que existan en una Lista (uso de la clase *List*):

```
import java.util.zip.*;
import java.util.List;

...

  List<String> files = new ArrayList<String>();

  files.add(“c:\\archivo_1.txt”);
  files.add(“c:\\archivo_2.txt”);
  files.add(“c:\\archivo_3.txt”);
...
  files.add(“c:\\archivo_n.txt”);

  try
  {
    // Objeto para referenciar a los archivos que queremos comprimir
    BufferedInputStream origin = null;

    // Objeto para referenciar el archivo zip de salida
    FileOutputStream dest = new FileOutputStream("c:\\folder.zip");
    ZipOutputStream out = new ZipOutputStream(new BufferedOutputStream(dest));

    // Buffer de transferencia para mandar datos a comprimir
    byte[] data = new byte[BUFFER_SIZE];

    Iterator i = files.iterator();
    while(i.hasNext())
    {
      String filename = (String)i.next();
      FileInputStream fi = new FileInputStream(filename);
      origin = new BufferedInputStream(fi, BUFFER_SIZE);

      ZipEntry entry = new ZipEntry( filename );
      out.putNextEntry( entry );

      // Leemos datos desde el archivo origen y los mandamos al archivo destino
      int count;
      while((count = origin.read(data, 0, BUFFER_SIZE)) != -1)
      {
        out.write(data, 0, count);
      }

      // Cerramos el archivo origen, ya enviado a comprimir
      origin.close();
    }

    // Cerramos el archivo zip
    out.close();
    }
  catch( Exception e )
  {
    e.printStackTrace();
  }

...
```

Documentación a entregar

La documentación que deberá entregar cada grupo es un fichero comprimido con el proyecto Netbeans de la aplicación desarrollada y una memoria explicando la realización de la misma. La memoria debe incluir una autoevaluación de la interfaz desarrollada, desde el punto de vista de los principios de Schneiderman y Plaisant (ver Tema 1).