La librería GNUJPDF

- **Librería:** *gnujpdf*
- **Archivo:** *gnujpdf.jar*
- **Desarrollador:** Eric Z. Beard
- **Paquete:** gnu.jpdf

Gnujpdf es una librería open-source muy sencilla con la que se pueden generar documentos pdf utilizando una clase que conocen muy bien todos los programadores Java: Graphics.

Con gnujpdf generar un documento pdf se convierte en algo tan sencillo como indicar el archivo pdf de salida, obtener un objeto Graphics para cada una de las páginas y dibujar sobre él. Finalmente, el archivo debe cerrarse y entonces el pdf se crea físicamente en el disco.

La principal limitación de esta librería es que el pdf no se va escribiendo en el disco conforme se va generando, sino que se genera entero en memoria y al cerrarlo es cuando se vuelca en el disco. Esto hace que en los documentos muy grandes se den problemas cuando se acaba la memoria reservada por el sistema operativo para el proceso de Java[1].

Gnujpdf permite realizar pdfs con lo básico: texto e imágenes. Para situaciones en las que se necesita más funcionalidad (campos rellenables, firma de documentos, etc) debe usarse una librería más avanzada, como **iText**.

[1] Una forma de ampliar la cantidad de memoria que necesita un la máquina virtual de Java es usar el parámetro **-Xmx** en la línea de comandos cuando se va a ejecutar el programa. Por ejemplo:

java -Xmx200m Programa

abriría el bytecode Programa.class y le reservaría 200MB de memoria RAM. Todos los IDE disponen de una opción para indicar este parámetro. Todo esto se utiliza cuando se prevé que un programa va a necesitar más de los 64MB que son asignados como máximo por defecto.

1) PDFJob

- **Librería:** *gnujpdf*
- **Archivo:** *gnujpdf.jar*
- **Desarrollador:** *Eric Z. Beard*
- **Paquete:** *gnu.jpdf*

Esta clase representa un documento pdf que va a ser generado por nuestro programa.

PDFJob
+ PDFJob (String f) throws IOException + PDFJob (OutputStream a) + Graphics getGraphics() + Graphics getGraphics (PageFormat p) + int getCurrentPageNumber() + Dimension getPageDimension() + void end()

- El primer constructor crea un documento pdf que se guardará en el fichero cuya ruta se pasa como parámetro. Si no hay permisos de escritura se lanzará una excepción.
- El segundo constructor crea un documento pdf que se escribirá sobre el canal genérico de salida pasado como parámetro.
- El primer método getGraphics crea una nueva página en el documento y nos devuelve un objeto Graphics[2] con el que podremos escribir y dibujar sobre ella. Para poder obtener una nueva página es necesario cerrar la página actual mediante el método **dispose** de la clase Graphics.
- El segundo método getGraphics funciona igual que el anterior, pero obtiene una página cuyo formato (orientación y dimensiones) se pasa como parámetro en forma de objeto PageFormat.
- El método getCurrentPageNumber devuelve el número de la página actual del documento PDF.
- El método getPageDimension devuelve el ancho y alto de la página actual en píxeles.
- El método end hace que finalice el documento PDF y se escriba físicamente.

[2] Casi todos los métodos de la clase Graphics están soportados. Aquellos que no lo están, no producen ningún efecto.

2) PageFormat

- **Librería:** *gnujpdf*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun microsystems*
- **Paquete:** *java.awt.print*

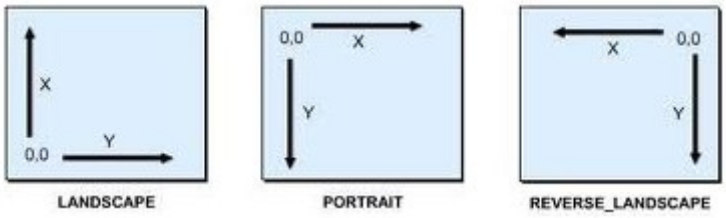
Esta clase representa un formato de página, que está formado por una orientación y unas dimensiones (ancho, alto) de la página. Es una clase propia de Java, que se utiliza en más ocasiones, no solo en la librería gnujpdf.

PageFormat
+ PageFormat() + void setOrientation (int tipo) + void setPaper (Paper p)

- El constructor crea un formato de página vertical por defecto.

- El método setOrientation permite definir la orientación de la página, de acuerdo con las siguientes constantes:

PageFormat.PORTRAIT	El origen está en la esquina superior izquierda
PageFormat.LANDSCAPE	El origen está en la esquina inferior izquierda
PageFormat.REVERSE_LANDSCAPE	El origen está en la esquina superior derecha



- El método setPaper permite indicar el tamaño (ancho, alto) de la página. Requiere un objeto de la clase Paper.

3) Paper

- **Librería:** *gnujpdf*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun microsystems*
- **Paquete:** *java.awt.print*

Esta clase representa una configuración de un papel, y sirve principalmente para indicar el alto, ancho y márgenes de una página. Por tanto, podemos usarlo no solo para decir cómo de grande debe ser una página, sino para indicar si es horizontal o vertical.

Paper
+ Paper() + void setSize (double w, double h) + void setImageableArea (double x, double y, double w, double h) + double getHeight() + double getWidth()

- El constructor crea una página cuyo formato es A4 vertical con márgenes de 1 pulgada.
- El método setSize permite definir el tamaño en píxeles del papel. A la hora de trabajar con documentos reales y píxeles debe atenderse a la **resolución del documento** (*medida en ppx, o píxeles por pulgada*). Dos resoluciones típicas son 72ppx (para archivos de ordenador) y 300ppx (para impresión en papel). Para dichas resoluciones, las dimensiones de los documentos más habituales son:

Tipo de papel	Dimensiones (72ppx)	Dimensiones (300ppx)
A4	595 x 842	2480 x 3508
A3	843 x 1191	3508 x 4961
A2	1191 x 1684	4961 x 7016
A5	420 x 595	1748 x 2480
A6	298 x 420	1240 x 1748

- El método setImageableSize sirve para definir la zona del papel donde se puede escribir (márgenes). Dicha zona es el rectángulo de vértice superior izquierdo en las coordenadas (x,y) y dimensiones (w,h).
- getWidth devuelve la anchura del papel.
- getHeight devuelve la altura del papel.