

Una presentación sobre  
librerías de Java

# “Apache PDFbox”

Modificación y creación  
de archivos PDF

---

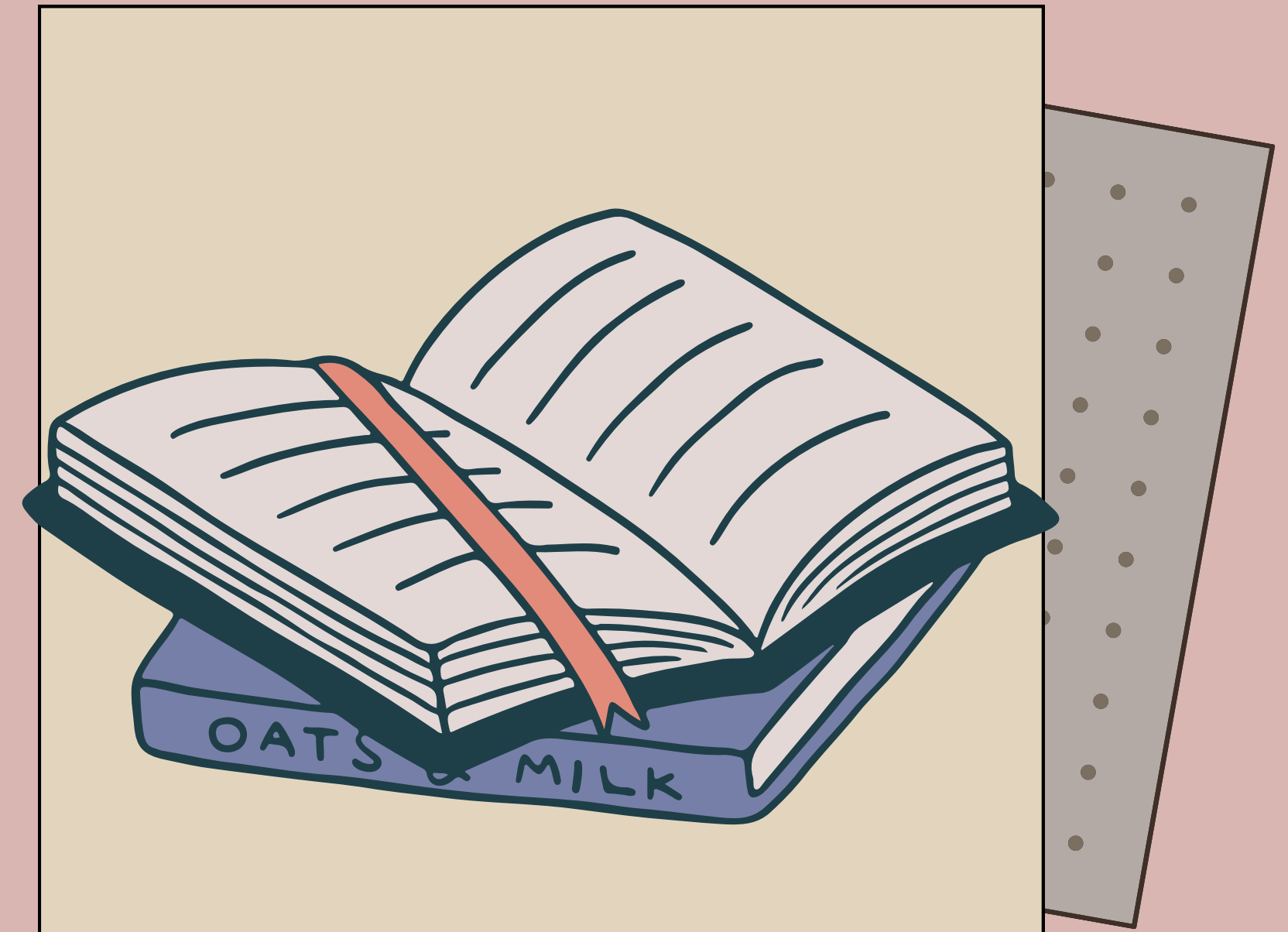
Patricio Vejar - José Rodríguez



# En PDFbox hay múltiples usos

La gracia de usar PDFbox es la facilidad que tiene de modificar, leer y crear archivos pdf de modo que uno podría fácilmente crear documentos con plantilla que llenar. Ejemplos serían formularios, boletas, manuales, etc.








Lo interesante es que como tal, este tipo de programas siempre tiene un uso en el mercado. Todas las compañías trabajan con archivos pdf's después de todo.



No son muchos los programas que permitan modificar libremente un archivo pdf y menos crear uno a base de una plantilla propia

PDFBOX

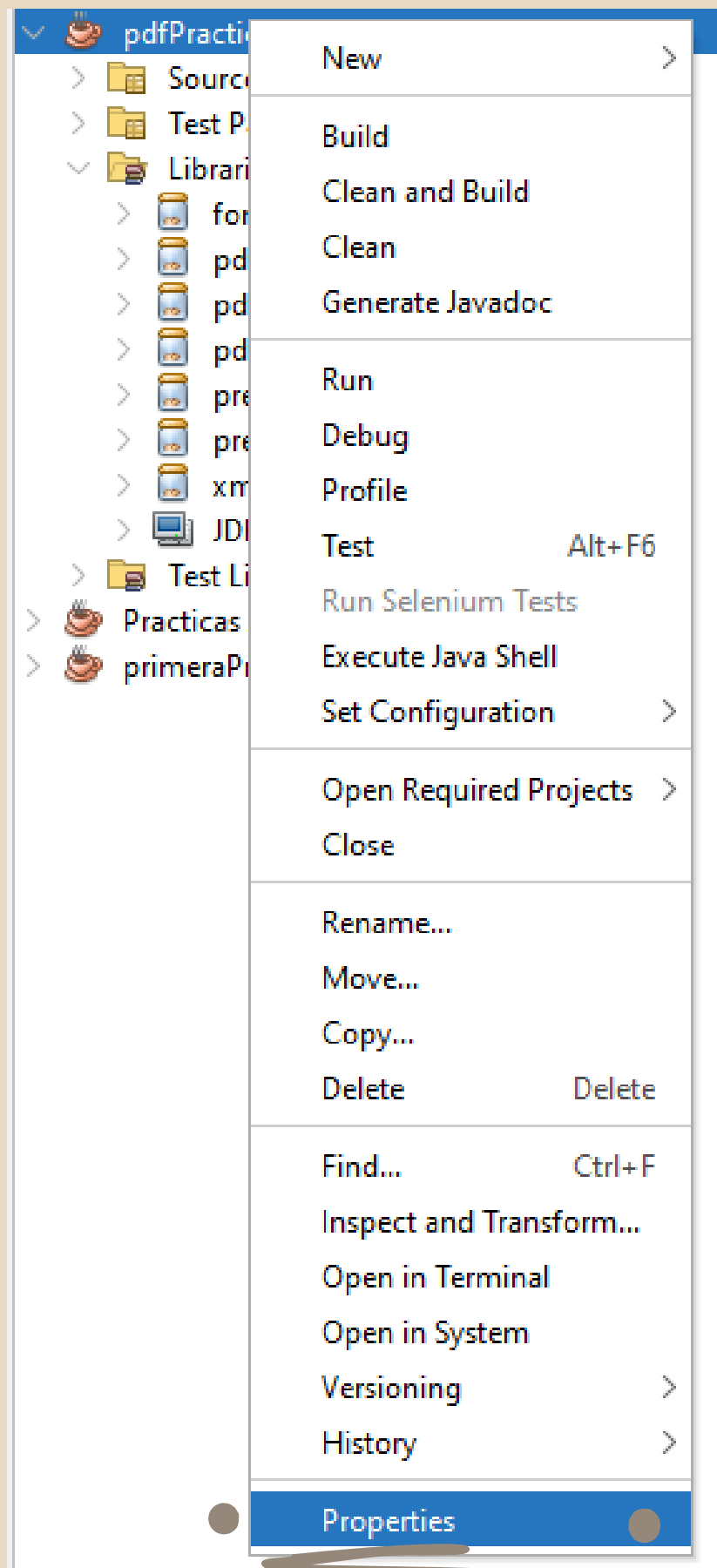
Para descargar la librería de pdfbox, requerimos los respectivos archivos java.  
Los se usan son principalmente:

-  fontbox-3.0.3
-  pdfbox-3.0.3
-  pdfbox-app-3.0.3
-  pdfbox-tools-3.0.3
-  preflight-3.0.3
-  preflight-app-3.0.3
-  xmpbox-3.0.3

Version	Description	Download Link	PGP Signature	SHA512 Checksum
PDFBox 3.0.3 feature	Command line tools			
	PDFBox standalone	pdfbox-app-3.0.3.jar	ASC	SHA512
	Debugger standalone	debugger-app-3.0.3.jar	ASC	SHA512
	Preflight standalone	preflight-app-3.0.3.jar	ASC	SHA512
	Libraries of each subproject			
	pdfbox	pdfbox-3.0.3.jar	ASC	SHA512
	pdfbox-io	pdfbox-io-3.0.3.jar	ASC	SHA512
	fontbox	fontbox-3.0.3.jar	ASC	SHA512
	preflight	preflight-3.0.3.jar	ASC	SHA512
	xmpbox	xmpbox-3.0.3.jar	ASC	SHA512
	pdfbox-tools	pdfbox-tools-3.0.3.jar	ASC	SHA512
	pdfbox-debugger	pdfbox-debugger-3.0.3.jar	ASC	SHA512

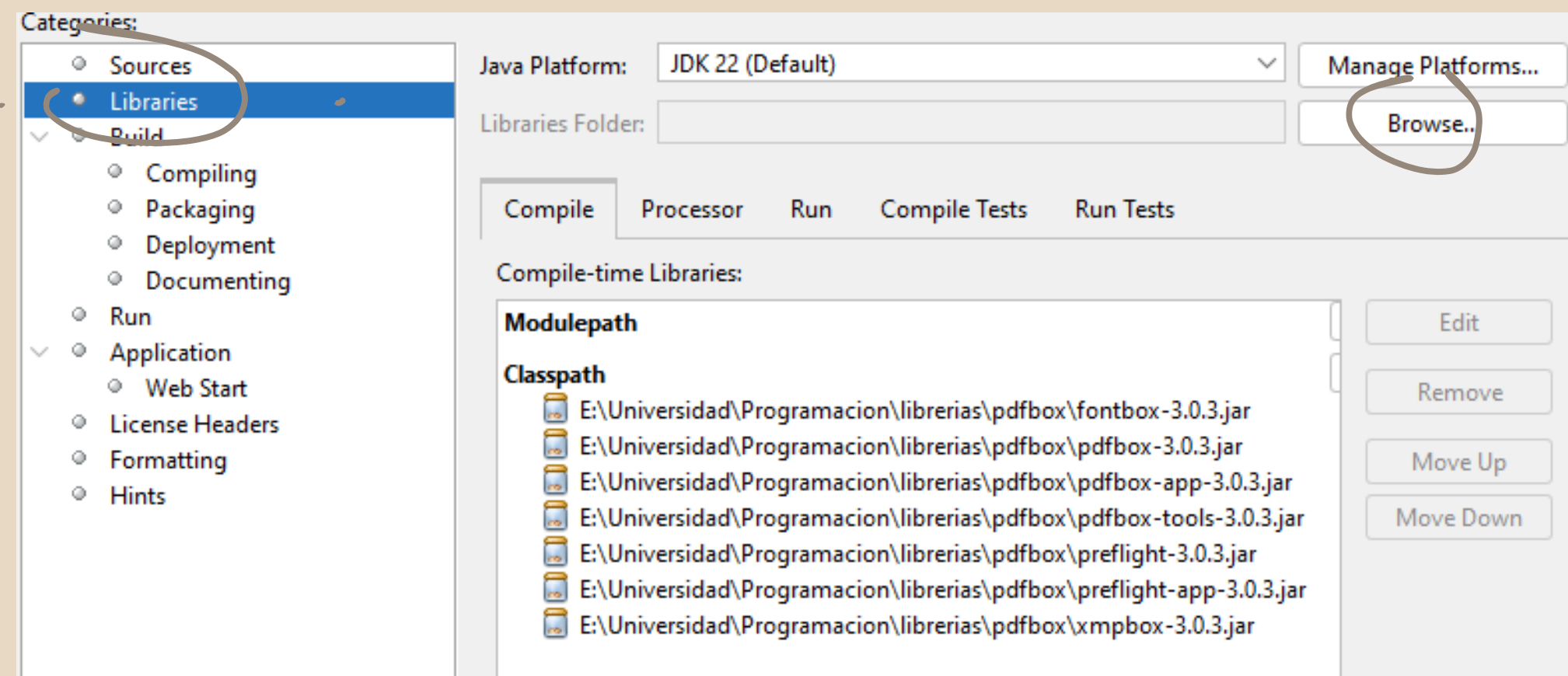
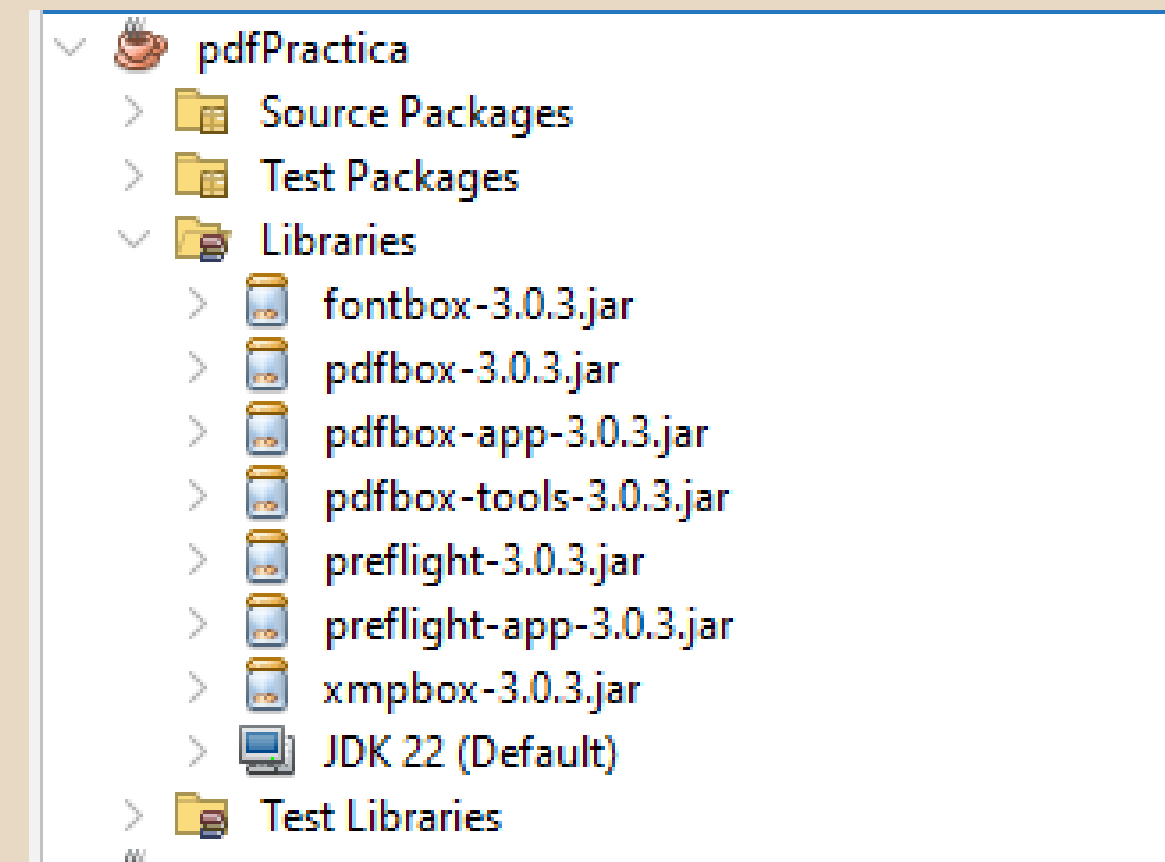
HTTP

<https://dlcdn.apache.org/pdfbox/3.0.3/pdfbox-app-3.0.3.jar>



Para poder integrarlos en un proyecto de NetBeans  
Solamente tienes que.  
Arrastrar los archivos a la subcarpeta 'Libraries'  
de tu proyecto  
O en las propiedades de tu proyecto importar los archivos

Si no usas NetBeans hay otros modos  
[aunque algunos puede que sean una aventura]



# Métodos y clases básicos

## PDDocument

La clase base  
Un objeto con varios  
métodos

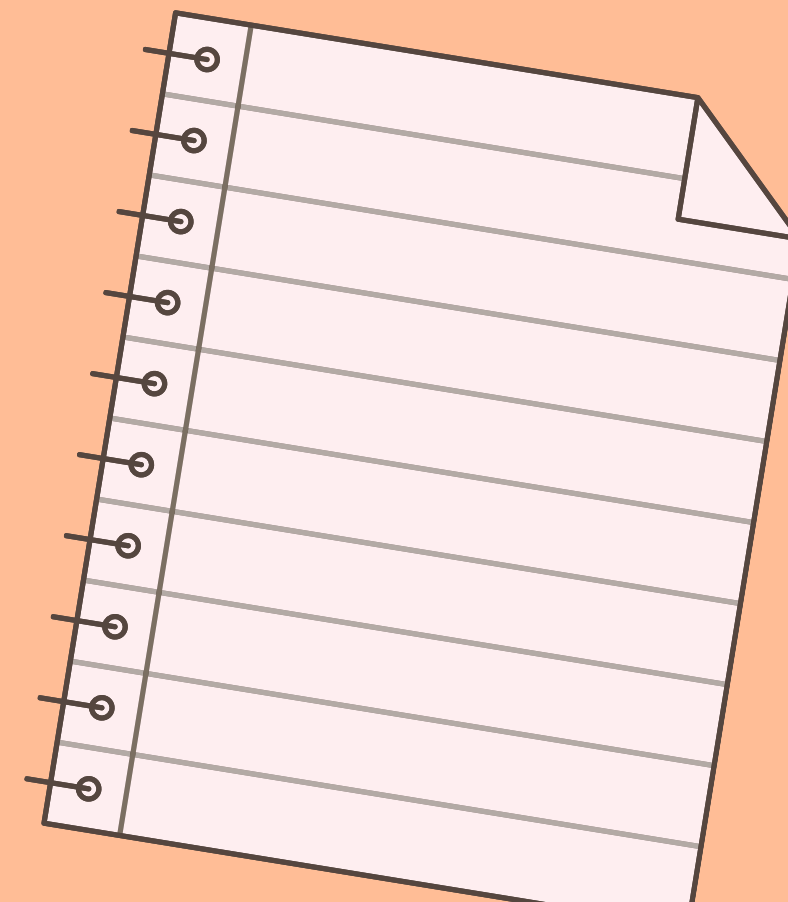
Pensad en el manejo  
de archivos de C, es  
el mismo principio,  
solamente que con  
mas herramientas a tu  
disposición



## PDFpage

Clase con la  
información que lleva  
una pagina de pdf

Puedes modificar  
según varias plantilla  
una serie de hojas



```
PDDocument doc = new PDDocument();  
PDFPage pagina = new PDFPage(); //si creamos un  
doc.addPage(pagina);  
//direccion donde guardarlo  
doc.save("E:/Universidad/Programacion/document  
  
//cerramos archivo  
doc.close();
```

## Acceder a pdf's existentes...

Uno puede modificar pdf ya existentes mientras tengas la dirección y no estén encriptados/conozcas su contraseña

Cuando accedes a un pdf de este modo. Puedes trabajar con el del mismo modo a que si lo creases

## Obejto File

Para cargar un archivo necesitan un objeto tipo File (librería integrada) con la dirección del archivo. Usando ese objeto y el método Loader (librería integrada) Puedes acceder al archivo pdf del mismo modo a que si lo hubieras creado.

```
File arch = new File("E:/Universidad/Programacion/documentos de prueba/doc.pdf");//llamamos el pdf que queremos cargar
PDDocument doc = Loader.loadPDF(arch); //lo cargamos con la libreria LOADER --importante

System.out.println("PDF cargado");

PDPage pagina = new PDPage();
//creamos y añadimos una pagina en BLANCO
//for(int i=0;i<3;i++){//creamos 3 paginas en blanco al FINAL del doc
//    doc.addPage(pagina);
//}
doc.close();
```

```
PDPPage hoja = doc.getPage(0);
PDPageContentStream flujo = new PDPageContentStream(doc, hoja);
flujo.beginText();

//logica de texto

flujo.endText();
flujo.close();
```

Fijaos que el flujo requiere el objeto documento y la pagina de la hoja.

La pagina de la hoja esta dado como un array de datos! Asi que la pagina 1 es referenciada como 0 en ".getPage"

# Textos

## Escritura y lectura

Para poder añadir textos y leerlos tenemos que crear un flujo de datos. Para ello creamos un objeto llamado "PDPageContentStream" Con este podemos añadir el texto al documento

ESTO SOBRE ESCRIBIRA SI NO ERES CUIDADOSO

Iniciamos el texto usando "flujo.beginText", ahora podemos escribir y modificar el texto del documento tranquilamente, si queremos parar de escribir (que se tiene que hacer antes de cerrar el archivo) usamos "flujo.endText" y "flujo.close"



```

flujo.beginText();

flujo.newLineAtOffset(25, 700);//""cordenada"" donde empieza a escribir
flujo.setLeading(14.5f); //espacio entre lineas..
//contentStream.setFont( font_type, font_size );
flujo.setFont(new PDFont(Standard14Fonts.FontName.HELVETICA_BOLD), 12);//fuente y
//bendito es stackOverflow -https://stackoverflow.com/questions/76985722/how-to-set-font-size-and-style-in-pdfbox

//ahora lo que queremos escribir..

String texto = "Mi nombre es Patricio y soy un estudiante de";
String texto2 = "Mi compañero con el que estoy trabajando es";
flujo.showText(texto);
flujo.newLine();//saltamos a la siguiente linea
flujo.showText(texto2);

flujo.endText();
flujo.close();

```

## Escribir Textos

Mientras estemos dentro de beginText podemos configurar el como se escribirá el texto. Desde la fuente, tamaño, espacio entre líneas, y coordenada donde empieza el párrafo

Después de configurarlo podemos escribir un String directamente usando ".showText()" Y para ir a la siguiente linea usamos ".newLine()"

Fijarse que todo esto son funciones de flujo, un objeto del tipo "PDPageContentStream"- Que creamos anteriormente



**Aviso No leas y escribas a la vez**

**Si haces esto existe la posibilidad de que se archivo se corrompa dependiendo de la lógica**

**Asique evítalo**

## Leer textos



Para poder leer textos , tenemos que cargar el archivo del que queremos leer y usamos el objeto "PDFTextStripper"

Este será nuestro lector. Del mismo modo que usamos un objeto para leer Strings de la consola. Se usa esto para el archivo. Este te dará todo el texto del documento. Todo. En una variable string asique usar a precaución

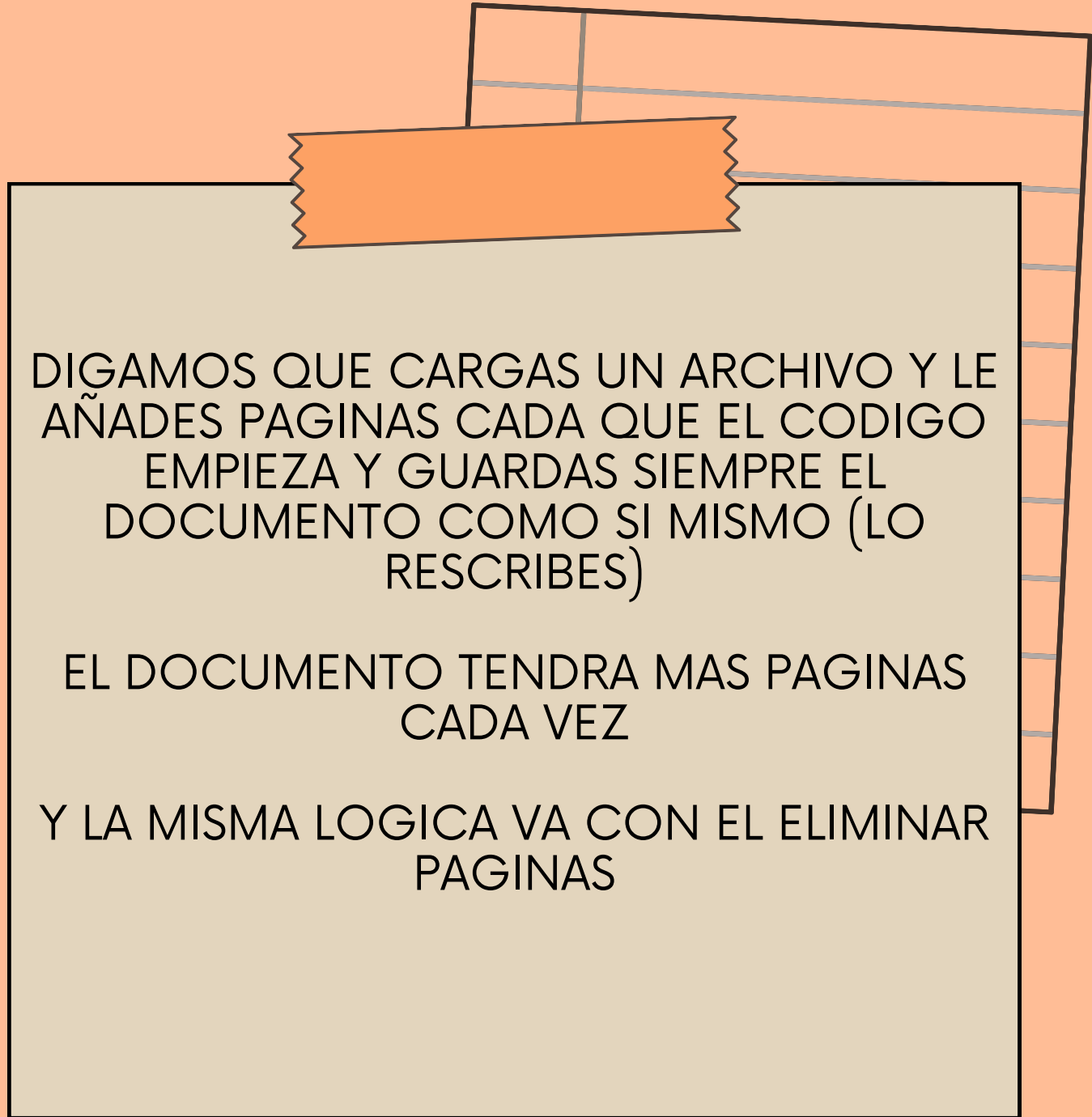
```
File arch2 = new File("E:/Universidad/Programacion/documentos de prueba/doc.pdf");
PDDocument doc2 = Loader.loadPDF(arch2); //lo cargamos con la libreria LOADER --imp
PDFTextStripper pdfLector = new PDFTextStripper();
String texto3 = pdfLector.getText(doc2);
System.out.println(texto3);

doc2.close();
```

# Paginas.

Puedes eliminar y agregar paginas como si nada.  
Claro al crearlas las paginas estarán en blanco.  
Y al eliminar requieres el índice de la hoja.  
pagina 1 = 0, pagina 2 = 1, pagina 3 = 2...  
Y claro puedes meterle lógica a todo si quieres que se cree de cierto modo

```
PDPPage pagina = new PDPPage();  
//creamos y añadimos una pagina en BLANCO  
for(int i=0;i<3;i++){//creamos 3 paginas e  
    doc.addPage(pagina);  
}  
  
doc.removePage(2);//elimina la pagina 3
```



DIGAMOS QUE CARGAS UN ARCHIVO Y LE  
AÑADES PAGINAS CADA QUE EL CODIGO  
EMPIEZA Y GUARDAS SIEMPRE EL  
DOCUMENTO COMO SI MISMO (LO  
RESCRIBES)

EL DOCUMENTO TENDRA MAS PAGINAS  
CADA VEZ

Y LA MISMA LOGICA VA CON EL ELIMINAR  
PAGINAS

# IMAGENES

```
//PDImageXObject pdImage = PDImageXObject.createFromFile("C:/logo.png", doc);
PDImageXObject imagen = PDImageXObject.createFromFile("E:/Universidad/Programacion/imagen_test.png", doc);
//para poner imagenes necesitamos un objeto de imagen, para esto usamos pdimage, requiere la direccion de c

PDPageContentStream flujo = new PDPageContentStream(doc, hoja);

flujo.drawImage(imagen, 0, 680); //tener en cuenta que esto ignora las letras. Osea que se pone por debajo de
//los datos dados, son las coordenadas donde va la imagen, la esquina inferior izquierda es 0,0.. x,y
//mas o menos el maximo de la hoja en altura es 680
flujo.close();
```

El añadir imágenes al pdf es relativamente fácil.  
Necesitas un objeto del tipo  
"PDImageXObject"

Tras crearlo con la dirección de la imagen, podemos implementarlo al iniciar el flujo de contenido

# Descripción e Encriptación

Al crear un documento de modo ""profesional necesitamos tener en cuenta que hay que dejar un registro de cuando fue creado ,por quien, quien es la editorial. Etc  
Y claro. A veces este tipo de documento es del tipo confidencial , asique como método de seguridad básico. Podemos implementar contraseñas

### Escribir una contraseña

Este archivo está protegido con contraseña. Escribe una contraseña para abrir el archivo.

Abrir archivoCancelar

## Descripción

Nombre de archivo:	doc.pdf
Tamaño de archivo:	22,4 KB
Título:	PDF de practica
Autor:	Patricio Vejar
Asunto:	practicas de pdf
Palabras clave:	Test, Pruebas, pdf
Fecha de creación:	23-09-2024, 10:30:06 a. m.
Modificado el:	03-07-2004, 10:30:06 a. m.
Creador:	Ejemplos pdf

# Descripción

El detallar los datos del documento es sumamente fácil. Gracias al objeto "PDDocumentInformation" que se obtiene a partir del documento. Llamando ciertos métodos, que les pasamos los datos en formato string y Calendar(librería integrada) Podemos definir la descripción.

Aquí puse un detallito para que tomara la fecha del dispositivo y la transformara al tipo de dato que toma ".setCreationDate(Calendar)"

```
PDDocumentInformation info = doc.getDocumentInformation();
```

```
//autor -piensa escritor  
info.setAuthor("Patricio Vejar");
```

```
//Nombre doc -piensa titulo  
info.setTitle("PDF de practica");
```

```
//Creador del doc -piensa editorial  
info.setCreator("Ejemplos pdf");
```

```
//sujeto -piensa tematica  
info.setSubject("practicas de pdf");
```

```
//tomamos la fecha actual del dispositivo -- LA FUNCION set()  
SimpleDateFormat sdf = new SimpleDateFormat("d-M-yyyy hh:mm:ss");  
String fechaActual = sdf.format(new Date());  
Date date = sdf.parse(fechaActual);  
Calendar calendar = Calendar.getInstance();  
calendar.setTime(date);
```

```
//escribimos la fecha de creacion del archivo..  
//info.setCreationDate(Calendar fecha);  
info.setCreationDate(calendar);
```

```
//escribimos la ultima fecha de modificacion del archivo.. Esto se llama  
calendar.set(2004,6,3);  
info.setModificationDate(calendar);
```

```
//palabras clave del documento -podria ser que lees el documento y to  
info.setKeywords("Test, Pruebas, pdf");
```

```
//guardamos el documento  
//si lo guardas igual a donde lo cargaste se va a reescribir, es posible  
doc.save("E:/Universidad/Programacion/documentos de prueba/doc.pdf");
```

```
//cerramos archivo  
doc.close();
```

```
AccessPermission acces = new AccessPermission(); //creamos el objeto de seguridad
StandardProtectionPolicy contra = new StandardProtectionPolicy("1234", "1234", acces);/
contra.setEncryptionKeyLength(128); //tamaño de la contraseña que el usuario y tu pueden
contra.setPermissions(acces); //este metodo acepta objetos del tipo: AccessPermission
doc.protect(contra); //registramos la contraseña y sus configuraciones al doc
```

**Esto se hace tras acceder al  
archivo y antes de guardarlo y  
cerrarlo naturalmente...**

## Encriptación

Mecanicamente lo mas facil..

Creamos un objeto

"StandardProtectionPolicy" (librería  
integrada) y otro objeto

"AccessPermission" (librería integrada)

El STP"" requiere de una contraseña,  
confirmación de contraseña y un objeto  
AP""

Definimos la cantidad de letras máximas  
de la posible contraseña y si se puede  
acceder. Y fin!

```
// Creando la instancia
Splitter splitter = new Splitter();

// Dividiendo el documento
//List<PDDocument> Paginas = Splitter.split(PDDocument );
List<PDDocument> Paginas = splitter.split(doc);

Iterator<PDDocument> iterador = Paginas.listIterator();

int i = 1;
while (iterador.hasNext()) {
    PDDocument pd = iterador.next();
    pd.save("D:/Personal Data/CLASES U/Progra/Java/PDFprac
}
System.out.println("División de PDF's exitosa.");
```

```
while (iterador.hasNext()) {
    PDDocument pd = iterador.next();
    pd.save("D:/Personal Data/CLASES U/Progra/Java/
}
System.out.println("División de PDF's exitosa.");

doc2.close();
```

# Splitter

Separar paginas del pdf tiene su truco, con el objeto 'Splitter', podemos separar cada pagina del documento , guardarlo en una lista y crear un nuevo archivo para la pagina seleccionada.

Cargamos el archivo deseado, creamos un objeto 'Splitter' y 'Iterator' (librería integrada) El objeto Iterator, creado a partir de la lista de paginas del Splitter. Nos ayudara a crear cada pagina del pdf por separado o si conocemos el índice, una pagina en concreto.



```
File archivo1 = new File("D:/Pe  
File archivo2 = new File("D:/Pe
```

```
// Instanciando PDFMergerUtility class  
PDFMergerUtility PDFmerge = new PDFMergerUtility();
```

```
// Configurando destino del archivo  
PDFmerge.setDestinationFileName("D:/Person
```

```
// Eligiendo los archivos a unir  
PDFmerge.addSource(archivo1);  
PDFmerge.addSource(archivo2);
```

```
// Uniendo archivos  
PDFmerge.mergeDocuments(null);  
System.out.println("Archivos Unidos con exito.");
```

```
return;  
/*
```

# Merge

Para poder unir distintos pdf's, usamos el objeto 'PDFmerge'- Cargando los archivos que quieres unir, y creando el archivo destino con el método '.setDestinationFileName'

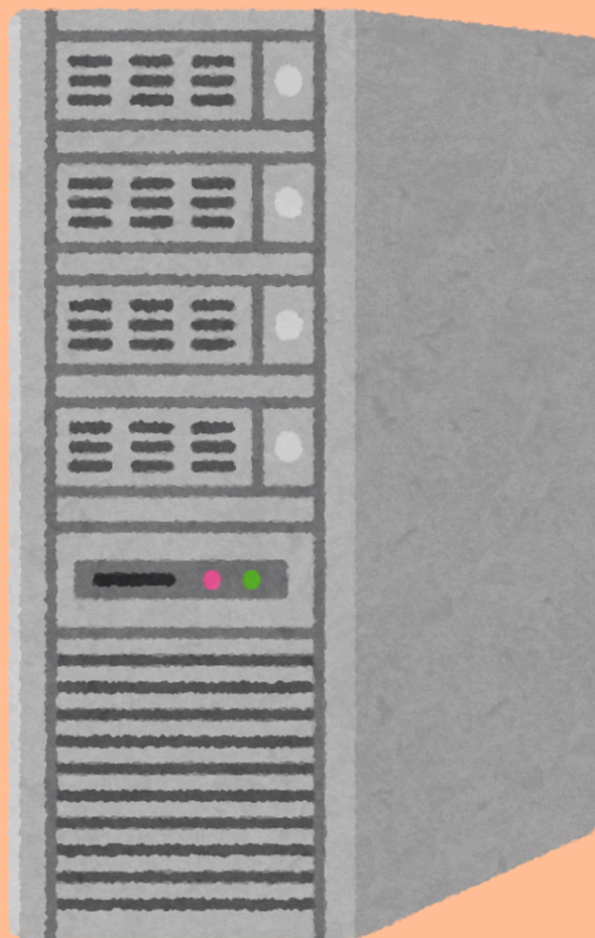
Solamente tenemos que darle los archivos deseados usando '.addSource' y terminamos el proceso con '.mergeDocuments'

De este modo los dos (o mas) documentos que unimos ahora están unidos en un documento aparte (dependiendo de la dirección y nombre del archivo naturalmente)

# Usos

Pocos son los programas que te dejan trabajar libremente con pdf's.

Los buenos de pdfbox es que puedes crear tu propio programa con las características que desees y mas..



La mayoría de compañías prefieren programas propios a comprar otros con solo parte de las cualidades que necesitan

Puedes configurar un pdf preestablecido para poder recrearlo con nueva información

Un ejemplo serian los hospitales, los programas que usan el personal, necesitan buena organizacion y que tenga todo los datos del paciente



# Gracias por escuchar

Si quiere acceder al código usado como ejemplo comentado y funcional, aquí está el [GitHub](#):

Referencias usadas:  
[TutorialsPoint](#)  
[JavaDoc](#) ([documentación pdfBox](#)).

Version PDFBox : 3.0.3

Presentado por José Rodríguez y Patricio Vejar

