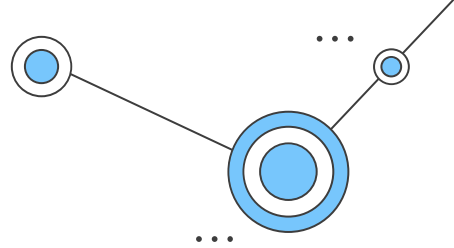


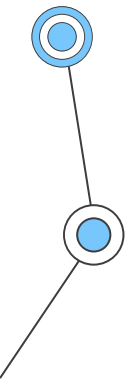
PDF desde JAVA

Utilizando la librería itext

Librería Itext



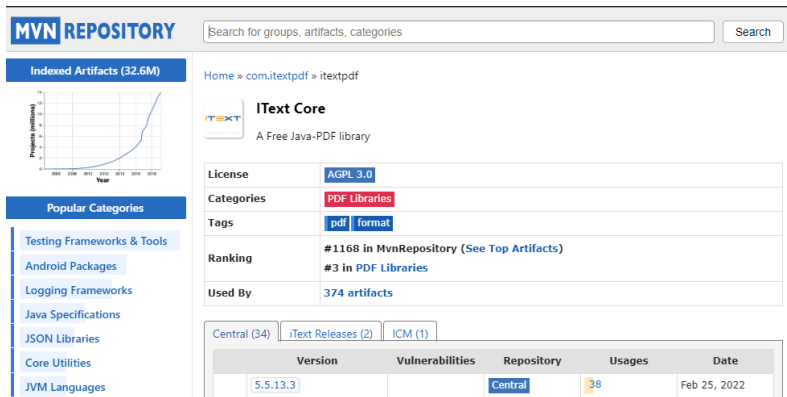
1. iText es una biblioteca de código abierto para crear y manipular documentos PDF en Java. Proporciona una API fácil de usar para crear documentos PDF desde cero o para modificar documentos PDF existentes. La biblioteca iText también admite funciones avanzadas, como la firma digital y la encriptación de documentos PDF.
2. La biblioteca iText se puede descargar desde su sitio web oficial o desde los repositorios de Maven.



Descargar la librería

Empezaremos descargando la última versión de esta librería, puedes descargarla desde este enlace:

<https://mvnrepository.com/artifact/com.itextpdf/itextpdf>



The screenshot shows the Maven Repository page for the IText Core library. The page includes a search bar, a sidebar with popular categories, and a main content area with details about the library.

Indexed Artifacts (32.6M)

Popular Categories

- Testing Frameworks & Tools
- Android Packages
- Logging Frameworks
- Java Specifications
- JSON Libraries
- Core Utilities
- JVM Languages

IText Core
A Free Java-PDF library

License: AGPL 3.0

Categories: PDF Libraries

Tags: pdf, format

Ranking: #1168 in MvnRepository (See Top Artifacts)
#3 in PDF Libraries

Used By: 374 artifacts

Central (34) | iText Releases (2) | ICM (1)

| Version | Vulnerabilities | Repository | Usages | Date |
|----------|-----------------|------------|--------|--------------|
| 5.5.13.3 | | Central | 38 | Feb 25, 2022 |



Para la descarga

Se elige la ultima versión disponible y se le da clic.

Se puede descargar de diferentes formas: .jar, .zip

En nuestro caso, se copiara el codigo de la dependencia de Maven



iText Core

A Free Java-PDF library

| | |
|------------|---|
| License | AGPL 3.0 |
| Categories | PDF Libraries |
| Tags | pdf format |
| Ranking | #1168 in MvnRepository (See Top Artifacts) #3 in PDF Libraries |
| Used By | 374 artifacts |

Central (34)

iText Releases (2)

ICM (1)

| Version | Vulnerabilities | Repository | Usages | Date |
|----------|-----------------|------------|--------|--------------|
| 5.5.13.3 | | Central | 38 | Feb 25, 2022 |

Maven

Gradle

Gradle (Short)

Gradle (Kotlin)

SBT

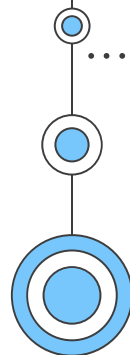
Ivy

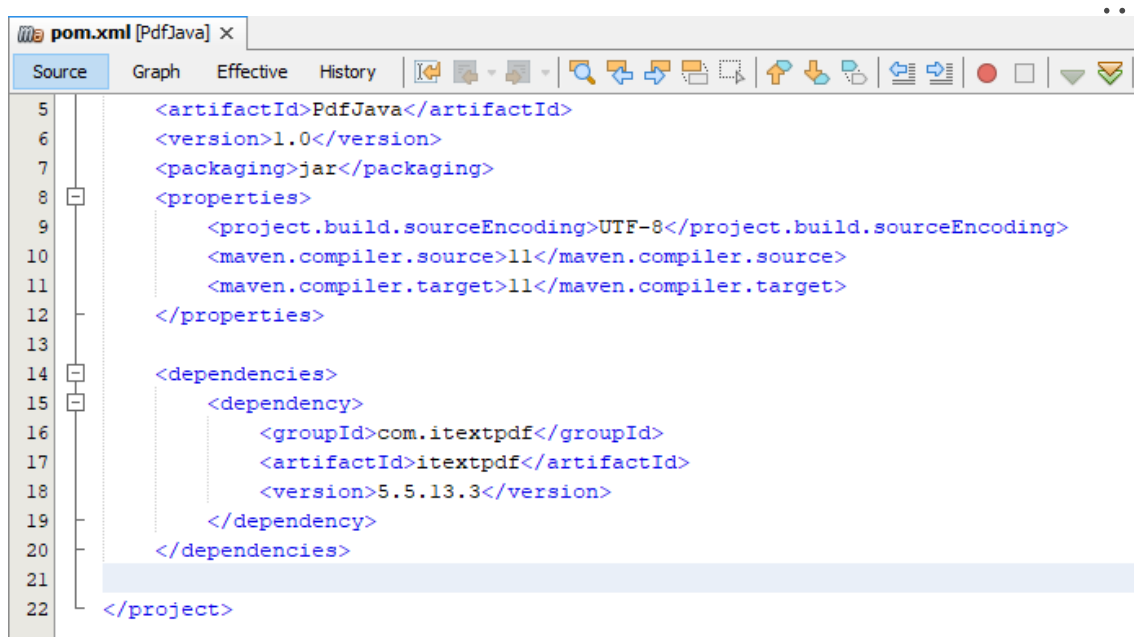
Grape

Leiningen

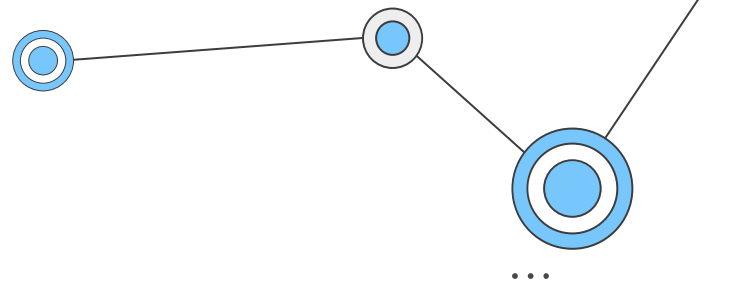
Builder

```
<!-- https://mvnrepository.com/artifact/com.itextpdf/itextpdf -->
<dependency>
  <groupId>com.itextpdf</groupId>
  <artifactId>itextpdf</artifactId>
  <version>5.5.13.3</version>
</dependency>
```





```
5 <artifactId>PdfJava</artifactId>
6 <version>1.0</version>
7 <packaging>jar</packaging>
8 <properties>
9   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
10  <maven.compiler.source>11</maven.compiler.source>
11  <maven.compiler.target>11</maven.compiler.target>
12 </properties>
13
14 <dependencies>
15   <dependency>
16     <groupId>com.itextpdf</groupId>
17     <artifactId>itextpdf</artifactId>
18     <version>5.5.13.3</version>
19   </dependency>
20 </dependencies>
21
22 </project>
```



Configuración del proyecto

1. Se crea el proyecto Java utilizando Maven.
2. Se agrega la dependencia iText al proyecto Maven, editando el archivo **pom.xml**
3. Al guardar el archivo, automáticamente se realiza la descarga de la Librería al Proyecto.



Desde la clase de Java se debe importar la librería itext

```
import com.itextpdf.text.*;  
import com.itextpdf.text.pdf.*;  
import java.io.FileOutputStream;
```

Se crea el documento

```
Document documento = new Document();
```

Se crea el OutputStream para el fichero donde queremos dejar el pdf.

```
PdfWriter.getInstance(document, new FileOutputStream("C:/Users/.../fichero.pdf"));
```

Se abre el documento.

```
documento.open();
```



Agregar un párrafo al documento

```
String texto = "¡Hola mundo!";  
Paragraph paragraph = new Paragraph(texto);  
document.add(paragraph);
```

Cerrar el documento

```
document.close();
```



```
public class GenerarPDF {  
    public static void main(String[] args) {  
        // Crear el documento PDF  
        Document document = new Document();  
        try {  
            // Escribir el documento en un archivo PDF  
            PdfWriter.getInstance(document, new FileOutputStream("C:/Users/.../archivo.pdf"));  
  
            // Abrir el documento  
            document.open();  
  
            // Agregar un párrafo al documento  
            String texto = "¡Hola mundo!";  
            Paragraph paragraph = new Paragraph(texto);  
            document.add(paragraph);  
  
        } catch (DocumentException e) {  
            e.printStackTrace();  
        } catch (Exception e) {  
            e.printStackTrace();  
        } finally {  
            // Cerrar el documento  
            document.close();  
        }  
    }  
}
```

Para Dar Formato

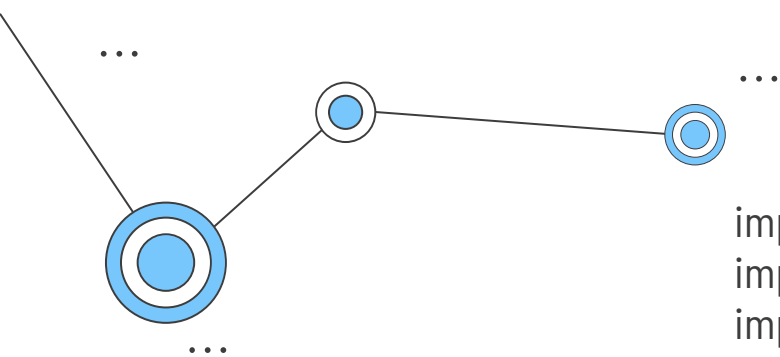
La clase **'Font'** se utiliza para crear un estilo de fuente personalizado para el título y los elementos de la lista.

Para el título, se crea un objeto **'Paragraph'** con el texto y el estilo de fuente especificado. La clase **'List'** se instancia y se agregan elementos de la lista utilizando la clase **'ListItem'**

El resultado final será un archivo PDF con un título centrado y una lista de elementos con viñetas.

```
// Agregar un título al documento
Font fontTitulo = new Font(Font.FontFamily.TIMES_ROMAN, size: 18, Font.BOLD);
Paragraph titulo = new Paragraph(string: "Titulo...", fontTitulo);
titulo.setAlignment(Element.ALIGN_CENTER);
document.add(titulo);

// Agregar una lista de elementos con viñetas al documento
List lista = new List(List.UNORDERED);
Font fontElementos = new Font(Font.FontFamily.HELVETICA, size: 12);
for (String ciudad : obtenerCiudades()) {
    ListItem item = new ListItem(ciudad, fontElementos);
    lista.add(item);
}
document.add(lista);
```

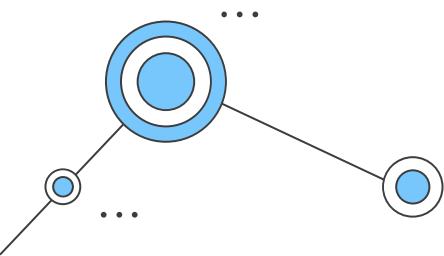



```
import com.itextpdf.text.Font;
import com.itextpdf.text.FontFactory;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.BaseColor;
```

```
documento.add(new Paragraph("Este es el primer párrafo, sin
formato"));
```

```
documento.add(new Paragraph("Este es el segundo y tiene una
fuente ",
```

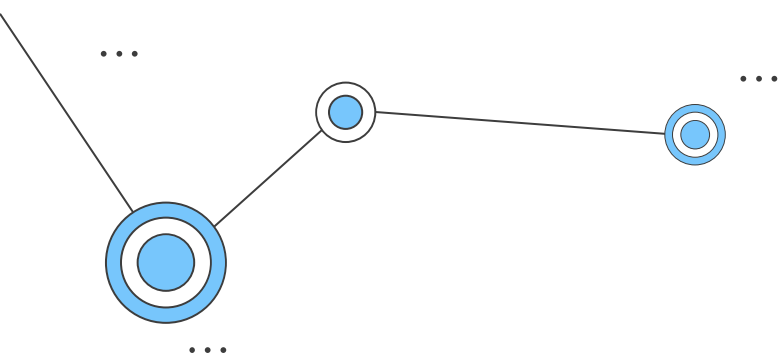
```
FontFactory.getFont("arial",           // fuente
                    22,                 // tamaño
                    Font.ITALIC,        // estilo
                    BaseColor.CYAN))); // color
```



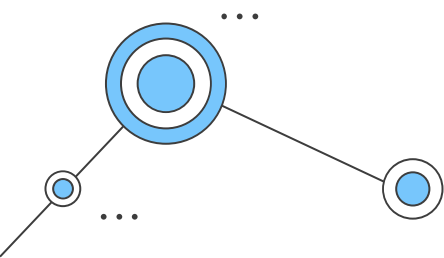
Añadir Párrafos

Añadir Imágenes

```
import com.itextpdf.text.Image;
...
try
{
    Image foto = Image.getInstance("image.jpg");
    foto.scaleToFit(100, 100);
    foto.setAlignment(Chunk.ALIGN_MIDDLE);
    documento.add(foto);
}
catch ( Exception e )
{
    e.printStackTrace();
}
```



Añadir tablas



```
import com.itextpdf.text.pdf.PdfPTable;
```

```
...
```

```
PdfPTable tabla = new PdfPTable(3);
```

```
for (int i = 0; i < 15; i++)  
{  
    tabla.addCell("celda " + i);  
}  
documento.add(tabla);
```

Se utiliza la clase `PdfPTable` de `iText`, que se instancia pasando el número de columnas y luego sólo hay que ir añadiendo celdas con el método **`addCell()`**. Hay varios métodos **`addCell()`** también admite distintos elementos dentro, como frases, imágenes, otras tablas, etc. En este ejemplo, se creará una tabla con 3 columnas y se le añaden 15 celdas, para que quede de 3 columnas por 5 filas.

ArrayList

```
try {  
    // Escribir el documento en un archivo PDF  
    PdfWriter.getInstance(document, new FileOutputStream( name: "C:/Users/.../archivo.pdf"));  
    // Abrir el documento  
    document.open();  
    |  
    // Crear un objeto Font  
    Font font = new Font(Font.FontFamily.TIMES_ROMAN, size: 14, Font.BOLD);  
  
    // Agregar los datos desde el ArrayList  
    ArrayList<String> datos = new ArrayList<>();  
    datos.add("Ciudad: Nueva York");  
    datos.add("País: Estados Unidos");  
    datos.add("Población: 8.336.817");  
    datos.add("Idiomas: Inglés");  
  
    for (String dato : datos) {  
        Paragraph paragraph = new Paragraph(dato, font);  
        document.add(paragraph);  
    }  
}  
} catch (DocumentException e) {  
    e.printStackTrace();  
}
```

Código de ejemplo

<https://github.com/Eliana-Janneth/Java-Librerialtext>

CREDITS: This presentation template was created by [Slidesgo](#), including icons by [Flaticon](#), infographics & images by [Freepik](#) and illustrations by [Stories](#)