XINCA Server 4.2
ユーザーズガイド

2003 年 12 月 1 日改定

MIST Management Infomation System Technology Co.,Ltd.
エム・アイ・エス・テクノロジー株式会社
〒105-0004 東京都港区新橋5丁目14番3号
ユタカビル8F
TEL 03-3434-8861
FAX 03-3434-8863
URL http://www.mist.co.jp/

警告

ユーザは、最初に米国特許 No.4,558,302 及び、その海外特許に基づく米国ユニシス社に対するライセンス契約に同意しない限り、いかなる目的においても、LZW 機能を提供する本ソフトウェアの使用は許可されません。ライセンスに関する情報は下記に連絡して下さい、

Unisys　Corporation
Welch　Licensing　Department　E8－114
Township　line　&　Union　Meeting　Roads
P.O.Box, 500
Blue　Bell, PA　19424

本製品は米国特許 4,558,302 のライセンス付与された LZW アルゴリズムを使用しています。

はじめに

本書の目的

弊社の XINCA テクノロジーは PDF に関係する一連な技術及び商品を指します。この中に数多くの INTERNET 方面の先進技術が含まれています。例えば XML、XSLT と JAVA 等の技術です。帳票を生成する全てのソリューションが提供されています。この中に XINCA SERVER はコア部分です。

本書は弊社商品の XINCA SERVER バージョン 4.0 のマニュアルです。バージョン 3.0 と比べて XINCA SERVER バージョン 4.0 は、多数の新機能が追加され、性能も向上しております。

新しいマニュアルは内容と構成上、大幅に変更されています。第 1 部の(XINCA CORE CLASS)はバージョン 3.0 を継承していますが、他の部分は一新されています。旧バージョンからのアップディトされる場合には、ご注意ください。

本書には XINCA クラスの詳細解説と、解り易いサンプルが記載されています。本書は初めて XINCA を使用するユーザーに対し、入門書として使われ、既に XINCA を使用されているユーザーに対しては、XINCA SERVER 参考マニュアルとして使われます。

本書の対象ユーザー

XINCA は 100% PURE JAVA で開発されたソフトパッケージです。その為、ある程度、JAVA による開発経験を有していることが望ましいとしています。

レイアウト

本書は下記の幾つか部分が含まれています。

前書き：本書の紹介

第 1 章　概略：PDF の基本概念と XINCA プログラムモデルを紹介します。

第 1 部　XINCA コアクラス

第 2 章　～　第 1 0 章：XINAC Server コア部分を紹介します。

第2部 XINCA の拡張クラス

第11章 ～ 第17章：XINCA Server の拡張された部分を紹介します。これらのクラスは第 1 部のクラスに基づいて、XINCA を使用し易くために設計されました。例えば、PDFTable クラスは帳票を生成する為に、一種類、簡単なインタフェースを提供しています。PDFTextArea クラスはテキストをレイアウトする為に、設計されました。

第3部 XINCA パッケージとクラスに関するレファレンス

付録 A～E

参考資料

本書は初歩的な JAVA プログラミング知識を有る方を対象として説明されています。JAVA 及び XINCA のプログラミングに関する説明が必要であれば、XINCA CD-ROM 中の XINCA SAMPLES を参照して下さい。

PDF に関する仕様を勉強したければ、Adobe Systems Incorporated 社より出版されている PDF 資料をご覧ください。

例えば、<<Portable Document Format Reference Manual Version 1.3>>。この資料は Adobe 社のホームページ (www.adobe.com) よりダウンロードできます。

問い合わせ

本書の内容において、万一不備な点や誤り、記載漏れなど、お気付きの点が御座いましたら、弊社までご連絡下さい。

メールアドレス： info@mist.co.jp

ホームページアドレス: http://www.mist.co.jp/

第１章 概要

1.1 PDF ファイルの特徴

PDF (Portable Document Format) は Adobe 社によって開発された移植可能なドキュメントフォーマットです。以下の特徴が有ります。

プラットフォームに依存しない

PDF ファイルは生成時のアプリケーション、ハードウェアと OS や、ディスプレー装置あるいはプリンタ出力時の出力装置に依存しません。

移植可能

PDF ファイルは 8 ビット、1バイトによって生成されています。PDF ファイルの設計目標は全てのプラットフォーム間に移植できることです。PDF ファイルは 8 ビット、1バイト表示方式よりプラットフォーム間の相違をなくす事が出来ます。

PDF ファイルは 7 ビットの ASCII 形式でも表現できます。あるネット環境では 7 ビットの ASCII コードしか転送できません。この様な時にはこの形式用いられます。

圧縮

PDF ファイルは数多くの圧縮メソッドをサポートしていますので、ファイルサイズを圧縮する事が出来ます。この理由、PDF ファイルの保存及び転送する時、スピードを向上が計れます。PDF ファイルは以下の幾つかの圧縮メソッドをサポートします。

●JPEG：カラー及びモノクローム画像に適応されます。
●CCITT Group3、 CCITT Group4 と run-length 圧縮：単色画像に適応されます。
●LZW、Flate 圧縮：テキスト、グラフィック及び各種画像に適応されます。

以上の圧縮メソッドは全部 2 進法データを生成し、これらのデータを更に ASCII Base-85 Encoding 計算方法で 7 ビットの ASCII コードに置き換えます。

フォント管理

PDF ファイルの出力効果一貫性を保証するために、フォント管理が必要です。PDF では主に以下のフォントメソッドが提供されます。

●Type1、TrueType 及び CID-keyed3種フォント書式をサポートします。
●14 種の標準フォントを提供されています。
●14 種の標準フォント以外のフォントに対し、PDF ファイルの中に一つの font descriptor が含まれる必要があります。font descriptor は、フォントに関する説明メッセージです。

1.2 PDF ファイルの生成及び使用

PDF ファイルはアプリケーション(例えば Acrobat PDFWrite, MIST XINCA Server)によって直接生成され、あるいは他の書式(例えば PostScript)を置き換えることより生成できます。生成される PDF ファイルは PDF Viewer アプリケーション(例えば Acrobat Reader)によって表示とプリントできます。図1－1を参照して下さい。

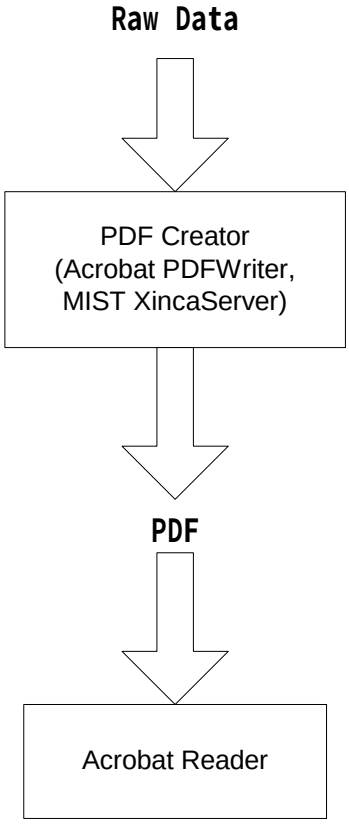


図1-1　PDF生成と表示過程

1.3 PDF ファイルの構造

PDF ファイルは各種オブジェクトにより構成されます。全てのオブジェクトはリレーションより1つのオブジェクトツリーから構成されています。ツリーのルートオブジェクトは CATALOG オブジェクトになります。多くのオブジェクトは PDF のデータタイプの一種類で、DICTIONARY と呼ばれます。例えば、PDF ファイルのページ毎に1つのページオブジェクトによって表示されます。ページオブジェクトの DICTIONARY ENTRY にページの属性、内容及びリファレンスが含まれています。図 1-2 は PDF ファイルのオブジェクト構造図です。

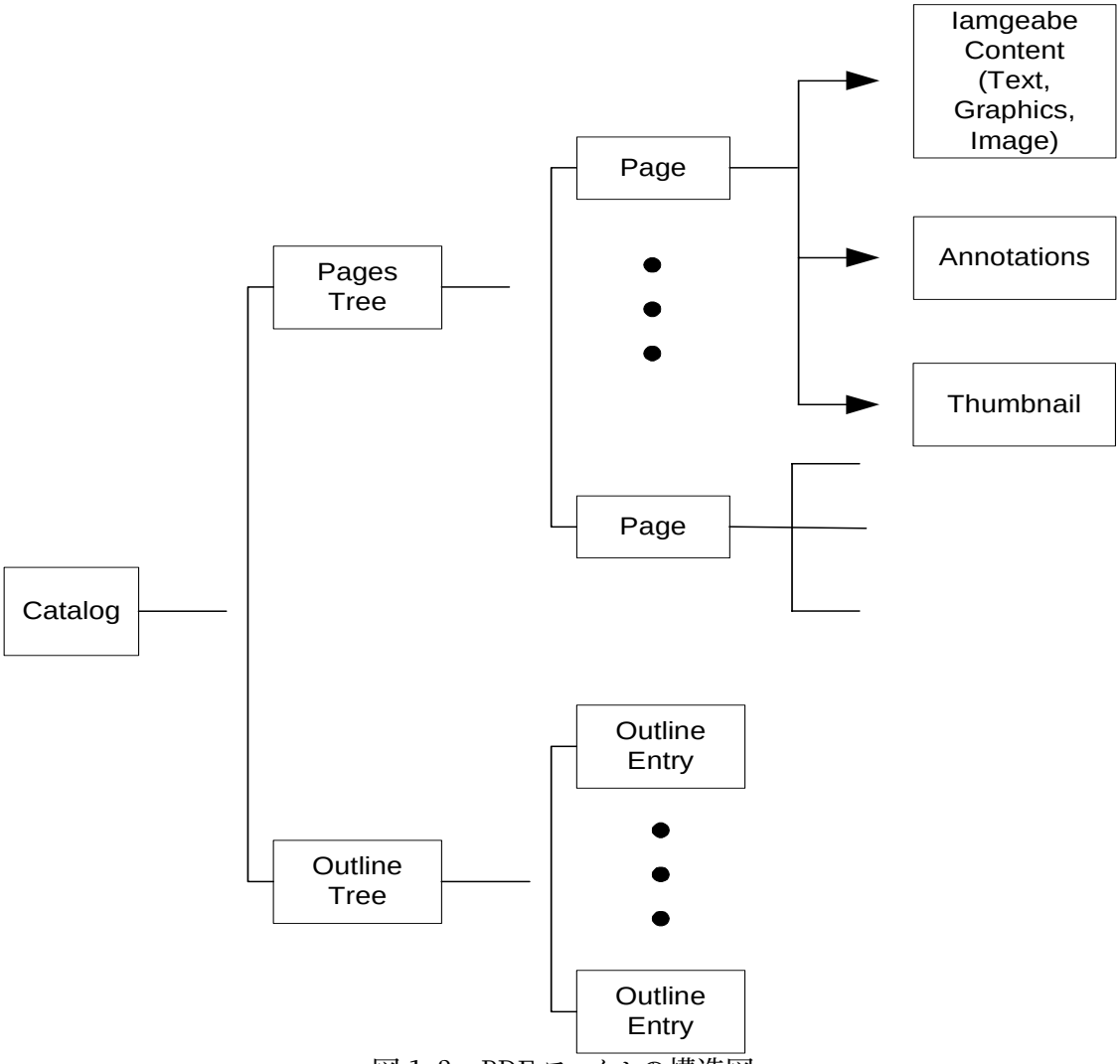


図 1-2　PDF ファイルの構造図

1.4 XINCA のクラス構造

JAVA はオブジェクト指向のコンピュータプログラム言語です。PDF はドキュメント説明言語の一種類です。この2つの言語には幾つかの共通な特徴があります。

●プラットフォームに依存しない
●オブジェクト指向

その為に、XINCA は両者の長所を取り入れて設計されています。XINCA は JAVA で実現された PDF 生成ツールです。JAVA と PDF の長所を十分に活用することができます。その上、XINCA のクラスは完全に PDF のオブジェクト構造によって設計され、ネーミング方式も含まれます。これによって、ユーザーは XINCA を理解し、PDF を理解するのに便利です。図 1-3 は XINCA のクラス構造図です。

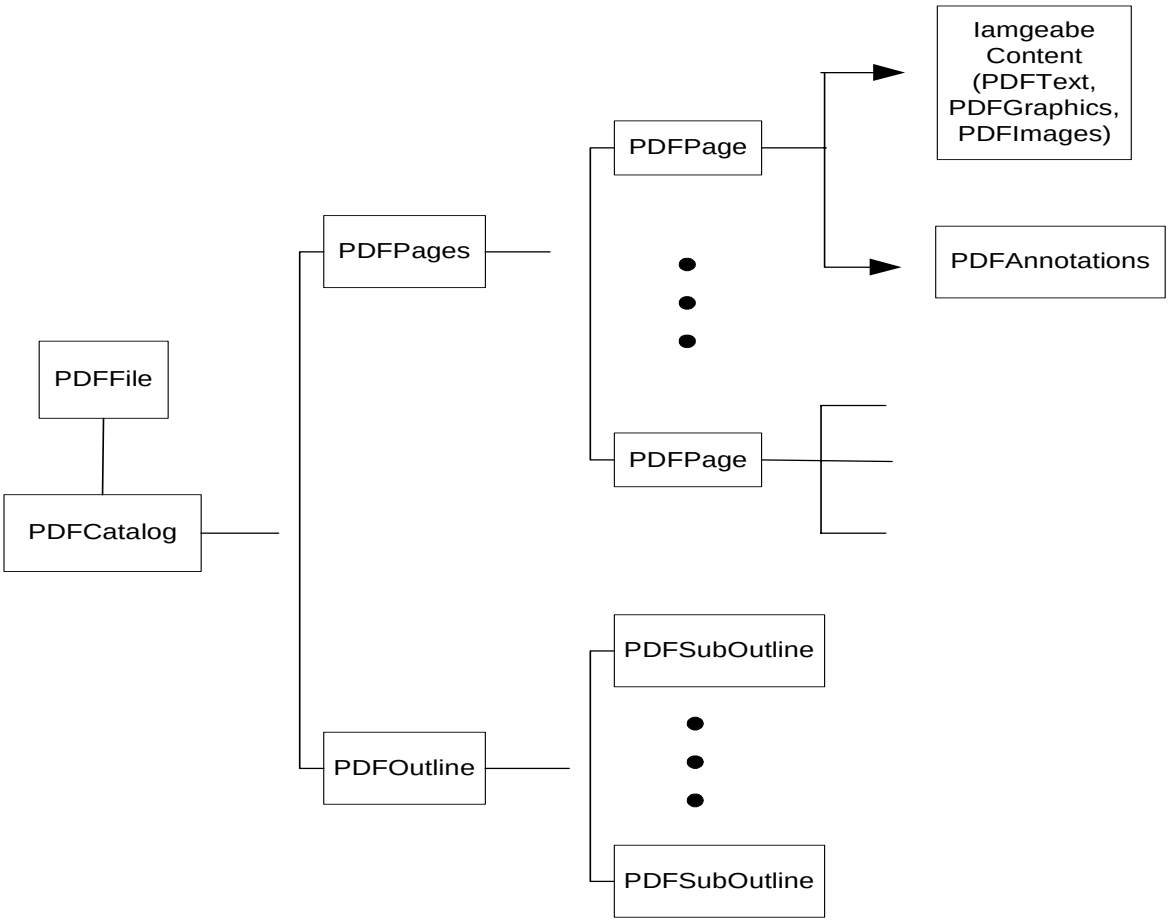


図 1-3　XINCA のクラス構造図

1.5 XINCA で PDF ファイルを生成するモデル

XINCA のクラスは PDF のオブジェクト構造によって設計されており、XINCA のプログラミングモデルも PDF のオブジェクト構造と似ています。PDF ファイルを生成するには、以下の4つのステップに分けられます。

●クラスのコンストラクタ(java の new センテンス)を設定します。
●コンスタラクタの属性を設定します。(ページの内容：例えば、テキスト PDFText、　グラフィックス PDFGraphics)
●コンストラクタとコンストラクタをリレーション構造にします。(クラスの APPEND メソッドより1つのコンスタラクタを別のコンストラクタに追加します)
●PDF ファイルを生成します。

図 1-4 はこの過程を詳しく説明しています。

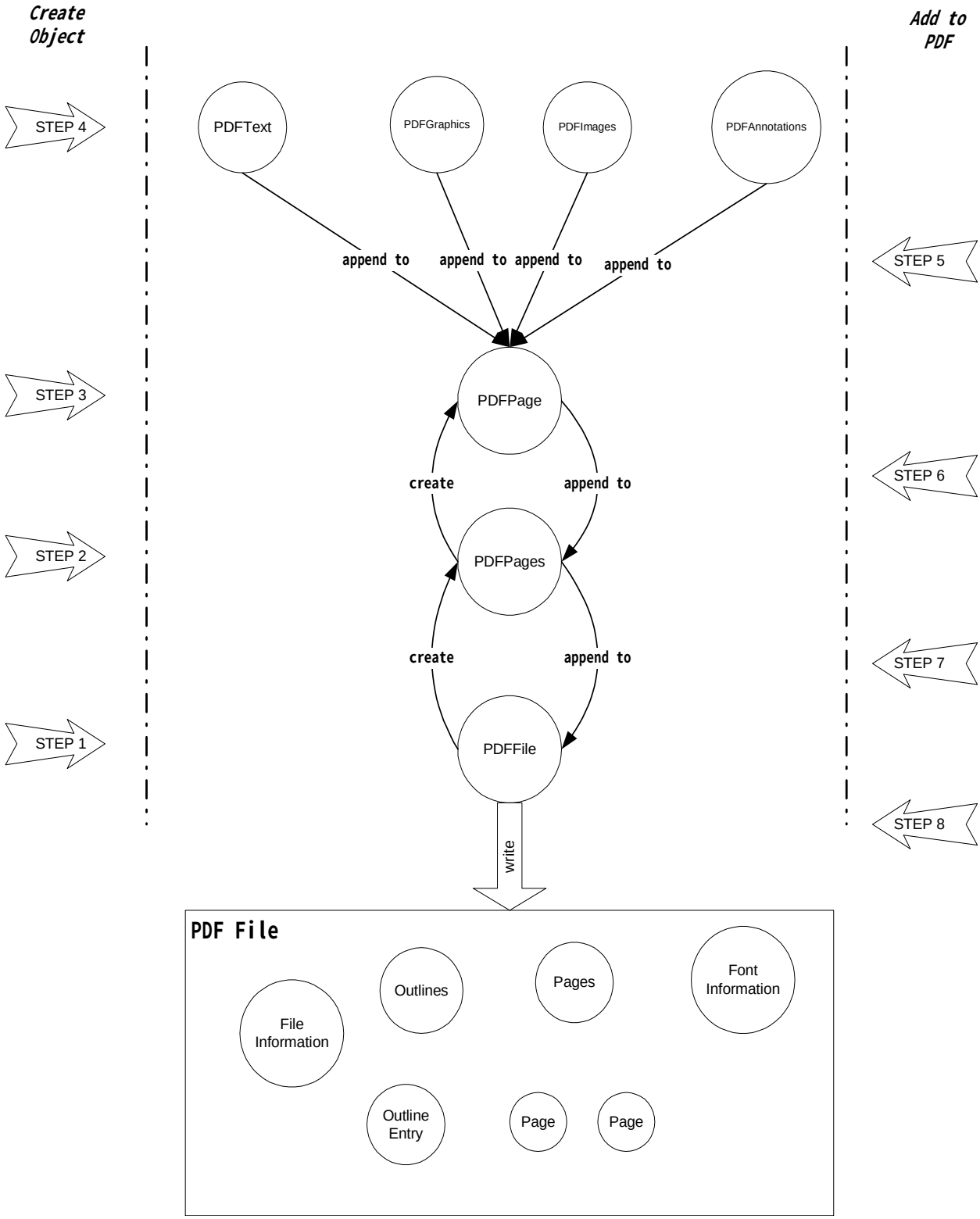


図 1-4　XINCA プログラミングモデル

1.6 シンプルなXINCAプログラムサンプル

図1-5はシンプルなXINCAサンプルで、図1-6は生成されるPDFファイルです。

```
import java.awt.*;

import xinca.*;                                    <- import XINCA package

public class simple
{

    public static void main(String argv[ ])
    {
        new simple();
    }

    public simple()
    {
        PDFFile  pdffile = new PDFFile("simple.pdf");
        PDFPages pgs     = new PDFPages(pdffile);

        createPage(pgs);                           <- create the page content

        pdffile.initialFile(pgs);
        pdffile.writeFile();
    }                                              <- main flowchat

    void createPage(PDFPages pgs)
    {
        PDFPage  page    = new PDFPage(pgs);       <- create page object

        PDFText text = new PDFText();
        text.setBasefont("gothic", Font.ITALIC);
        text.setCharSize(26);
        text.setXY(140, 700);
        text.setText("This was created by PDFText");
        page.append(text);                         <- append to page

        text.setBasefont("gothic");
        text.setUnderLine(new Color(175, 175, 175));
        text.setColor(Color.blue);
        text.setCharSize(20);
        text.setXY(100, 650);
        text.setText("日本語文字です！");
        page.append(text);                         <- create TEXT content

        PDFGraphics graphic = new PDFGraphics();
        graphic.setStrokeColor(0, 125, 0);
        graphic.strokeLine(100, 500, 400, 600);
        page.append(graphic);
        graphic.setStrokeColor(125, 125, 125);
        graphic.strokeRectangle(100, 500,  300, 100);
        page.append(graphic);                      <- append to page

        graphic.setFillColor(Color.red);
        graphic.drawCircle(200, 300,  80);
        page.append(graphic);

        PDFImages img = new PDFImages("../images/zebralogo.jpg");
        img.setXY(350, 300);
        page.append(img);                          <- append to page

        pgs.append(page);                          <- append to pages tree
    }                                              <- create GRAPHIC content
}
```

図1-5 シンプルのXINCAサンプル

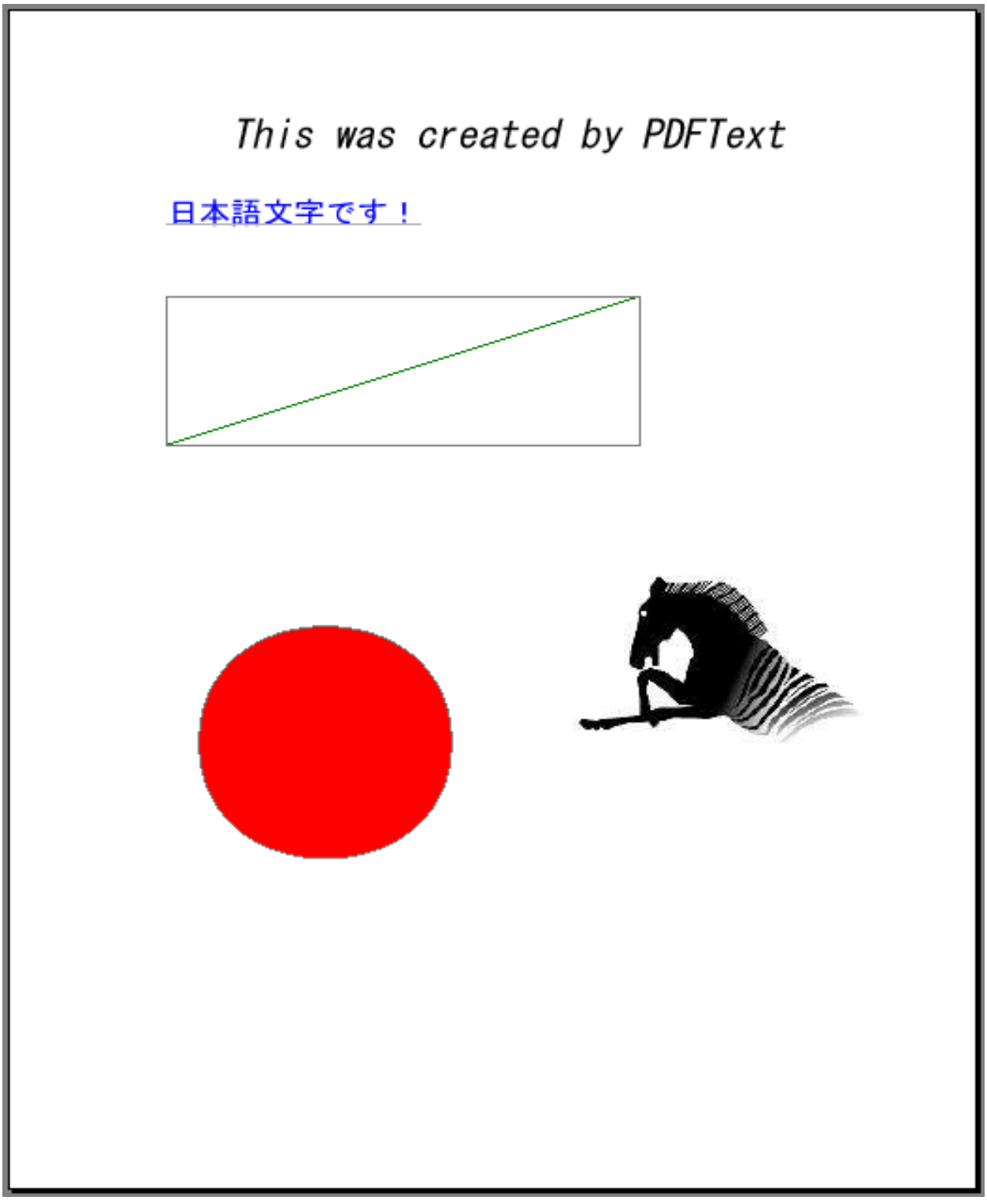


図1-6 生成されるPDFファイル

第１部 XINCA コアクラス

本部は第2章～第10章を含み、XINCA SERVER の9個のクラスを紹介します。本部で紹介する内容はPDF ファイルのオブジェクトと対応します。例えば、PDFPages は Pages オブジェクトと対応します。これらの9個のクラスは XINCA SERVER の基本的なクラスです。

xinca.PDFFile
xinca.PDFPages
xinca.PDFPage
xinca.PDFText
xinca.PDFGraphics
xinca.PDFImages
xinca.PDFAnnotation
xinca.PDFTextGrid
xinca.PDFMovie

第２章 PDFFile

2.1 概要

XINCAを使用するには、最初に生成されるＰＤＦファイルを宣言する必要があります。このクラスのコンストラクタオブジェクトに他のクラスオブジェクトを追加することができます。

2.2 PDFFileのコンストラクタ

new PDFFile (String　filename)

◆機能◆　PDFFile オブジェクトの宣言です。パラメータはＰＤＦのファイル名です。

new PDFFile (PrintWriter pw)

◆機能◆　PDFFile オブジェクトの宣言です。パラメータは Java の Writer オブジェクトです。

new PDFFile (Writer w)

◆機能◆　PDFFile オブジェクトの宣言です。パラメータは Java の Writer オブジェクトです。

以上2つのメソッドは通常には、JSP あるいは SERVLET のプログラムに使われます。HttpServletResponse から取得された PrintWriter を PDFFile に渡し、直接的に生成される PDF をクライアント側のマシンに送れます。

new PDFFile (OutputStream out)

◆機能◆　PDFFile オブジェクトの宣言です。パラメータは Java の OutputStream オブジェクトです。

new PDFFile ()

◆機能◆　PDFFile オブジェクトの宣言です。生成される PDF ファイルは標準出力設備(stdout)に出力する事が出来ます。例えば、ディスプレイです。

PERL あるいは C 等の言語の CGI アプリケーションによって XINCA を使用し、PDF ファイルを作る時には、このメソッドを使用すると直接的に生成される PDF をクライアント側のマシンに送れます。

2.3 PDFFile のメソッド

initialFile (PDFPages pages)

◆機能◆　initial ファイルオブジェクトです。このメソッドを使用する前に、PDFPage オブジェクトを作成しなければなりません。

wiriteFile ()

◆機能◆　すべてのＰＤＦオブジェクトをファイルに書き込みます。

boolean PDFFile. isOK ()

◆機能◆　PDFFile 出力オブジェクトが使用可能かをチェックします。例えば、指定されるファイルは書き込みかどうかをチェックします。他のアプリケーションが使用中であれば、このファイルはリードオンリーで書き込むことはできません。

書き込みが行われると“true”を返し、それ以外は“false”を返します。

setTitle(String title)

◆機能◆　PDF ファイルのタイトルを設定する。

setAuthor(String author)

◆機能◆　PDF ファイルの作成者を設定する。

setPageSize(String size)

◆機能◆　ページサイズを設定します。用紙の向きは縦方向となります。
指定できるパラメータは”a3”(A3),”a4”(A4),”b3”(B3),”b4”(B4)”,b5”(B5)です。

setPageSize(String size, boolean horizontal)

◆機能◆　ページサイズを設定します。
関数の size は上記のと同じパラメータを指定します。パラメータ horizontal の値は“true”であれば、用紙は横になり、“false”であれば、用紙は縦になります。

setPageSize(int width, int height)

◆機能◆　ページサイズを設定します。width はページの幅で、height はページの高さです。

setPageMode (int mode)

◆機能◆　PDF のディスプレイモデルを設定します。Mode で指定される値は次のテーブルに記入されている値を指定します。各々は異なるディスプレイモデルに対応しています。

PDFCatalog.PAGEMODE_USENONE	Open document with neither outline nor thumbnails visible.
PDFCatalog.PAGEMODE_USEOUTLINES	Open document with outline visible. This is the default value.
PDFCatalog.PAGEMODE_USETHUMBS	Open document with thumbnails visible.
PDFCatalog.PAGEMODE_FULLSCREEN	Open document in full-screen mode.

2.4 サンプル

```
PDFFile    pdffile = new PDFFile("simple.pdf");
                              . . . . . .
pdffile.initialFile(pgs);      // add all pages to PDFFile
pdffile.writeFile();            // write and close the PDFFile
```

第３章 PDFPages

3.1 概要

PDFPages オブジェクトを生成してから、PDFPage オブジェクトを追加します。PDFPages オブジェクトにまた PDFPages オブジェクトを追加します。

3.2　PDFPages のコンストラクタ

new PDFPages (PDFFile pdffile)

◆機能◆　PDFPages オブジェクトのインスタンスを宣言します。パラメータは PDFFile タイプのインスタンス です。

3.3　PDFPages のメソッド

append (PDFPage page)

◆機能◆　　選択されたページをカレントページツリーに追加します。このメソッドを使用する前に、PDFPage オブジェクトのインスタンスを作成しなければなりません。

append (PDFPages pages)

◆機能◆　パラメータとして選択されたページツリーをカレントページツリーに追加します。このメソッドを使用する前に、パラメータとしての PDFPages オブジェクトインスタンスを作成しなければなりません。

setLabelName (String name)

◆機能◆　PDF ページツリーのしおりを設定します。このメソッドが指定されなければ、ページツリーの名はデフォルト値に設定されることになります。つまり“Pages n”となります。“n”はページの番号です。

3.4 サンプル

```
. . . . . .
    PDFFile    pdffile = new PDFFile("simple.pdf");
    // add a pages object to this PDF file
    PDFPages pgs         = new PDFPages(pdffile);
. . . . . .
```

第４章 PDFPage

4.1 概要

PDFPage オブジェクトはページの内容を格納します。

4.2 PDFPage のコンストラクタ

new PDFPage (PDFPages pdfpages)

◆機能◆　PDFPage オブジェクトのインスタンスを宣言します。パラメータは PDFPages タイプのインスタンスです。

4.3 PDFPage の圧縮メソッド

setCompressMethod (int firstMethod, int secondMethod)

◆機能◆　生成されるページの圧縮メソッドを設定します。パラメータは2つあり、一番目の firstMethod のパラメータは PDFPage. LZW 及び PDFPage. FLATE のどちらかを設定し、二番目の secondMethod のパラメータは PDFPage. ASCII85及び PDFPage.ASCIIHEX のどちらかを設定します。

このメソッドを使用しなければ、デフォルトの圧縮メソッドを使用されます。
setCompressMethod(PDFPage.LZW, PDFPage.ASCII85)

nonCompress ()

◆機能◆　この機能を使用するとページの圧縮メソッドが行いません。

4.4 PDFPage のページサイズメソッド

setPageSize(String size)

◆機能◆　ページサイズを設定します。用紙の向きは縦になります。パラメータは”a3”(A3),”a4”(A4),”b3”(B3),”b4”(B4),”b5”(B5)の中から一つを指定できます。


このメソッドを使用しなければ、デフォルトのページサイズ A4 が使用されます。

setPageSize(String size, boolean horizontal)

◆機能◆　ページサイズを設定します。一番目の size に指定するパラメータは上記で指定された size と同じです。二番目のパラメータは用紙の向きを設定します。“true”を指定すると用紙は横になり、“false”であれば用紙は縦になります。

setPageSize(int width, int height)

◆機能◆　ページサイズを設定します。一番目のパラメータ width は用紙の幅さを設定し、2番目のパラメータ height は用紙の高さを設定します。

PDFPoint PDFPage.getPageSize ()

◆機能◆　ページサイズを取得します。PDFPoint.x は用紙の幅さです。PDFPoint.y は用紙の高さです。

setPageSize(double w, double h)

◆機能◆　PDF のページサイズを設定する、サイズの単位はセンチメートルです。

setPageSize(double w, double h , String unit)

◆機能◆　PDF のページサイズを設定する、unit はサイズの単位を指定する。

xinca.util.Ruler にサポートした単位を参照する。

4.5 PDFPage 別のメソッド

append (Object object)

◆機能◆　生成されたオブジェクトを PDFPage のインスタンスに追加します。パラメータのタイプは PDFText、PDFTextGrid、PDFGraphics、PDFMovie、PDFImages及び PDFAnnotation の中でどちらでも使用できます。このメソッドを使用する前に、Object タイプのインスタンスを作成しなければなりません。

setLabelName (String name)

◆機能◆　ページのしおり名を設定します。このメソッドが指定されなければ、ページはデフォルトのしおり名“pagen”が使用されます。“n”はページの番号です。

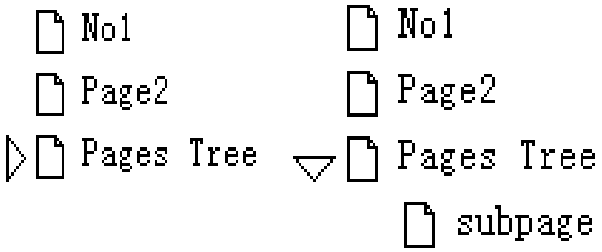


図 4-1 ページのしおり名

4.6 サンプル

```
// create a page from a pages object
PDFPage    page       = new PDFPage(pgs);

PDFText text = new PDFText();
. . . . . .
page.append(text);              // append this PDFText to PDFPage
pgs.append(page);               // append this PDFPage to PDFPages
```

．．．．．．．

第５章 PDFText

5.1 概要

テキストを格納するオブジェクトです。

5.2 PDFText のコンストラクタ

new PDFText ()

◆機能◆　PDFText オブジェクトのインスタンスを宣言します。

宣言されたオブジェクトのインスタンスはデフォルト状態で行間距離が 10、リンク無し、下線無し、テキスト色が黒に設定されています。

new PDFText(String orgencode,String dstencode)

◆機能◆　このパラメータ付きのコンストラクタメソッドを使用して、テキストオブジェクトインスタンスのエンコード変換を設定します 。一番目のパラメータ orgencode はテキストオブジェクト現在のエンコードが指定され、二番目のパラメータ dstencode に目標エンコードを指定します。変換されるエンコードは JDK の標準に合わせます。
例えば、”EUCJIS”は Unix 上の日本語文字のコード体系です。”SJIS”は Windows 上の日本語文字のコード体系です。

5.3 文字と位置を設定するメソッド

setText (String text)

◆機能◆　PDFText オブジェクトに表示される文字を設定します。文字列をパラメータとして使用します。このメソッドを使用して PDFText オブジェクトに追加します。文字の中に“￥n”があれば、次の文字を変行します。つまり、次の文字の座標 X は文字列の始点座標 X と同じ、座標 Y は始点座標 Y＋行間距離になります。

setXY（int x，int y）

◆機能◆　文字の座標を設定します。

setXY（double x，double y）

◆機能◆　文字の座標を設定します。

5.4 文字属性を設定するメソッド

setBasefont（Font font）

◆機能◆　文字のフォント名を設定します。XINCA がサポートするフォント名は8種類の英語フォント、12種類の日本語フォント，5種類の中国語フォントと1種類の韓国語フォントです。XINCA がサポートしているフォント名リストは付録A に収録しされていますので参照して下さい。

setBasefont（String fontname，int fontmode）

◆機能◆　文字のフォント名とフォントタイプを設定します。このメソッドには2つのパラメータがあります。一番目のパラメータ fontname にはフォント名を指定し、二番目のパラメータ fontmode にはフォントタイプを指定します。フォントタイプは複数の設定が必要であれば、JAVA の設定方法と同じように設定できます。例えば、Fontmode＝Font.BOLD、Font．ITALIC あるいは Font．BOLD＋Font．ITALIC です。

フォント名は全角と半角を区別して下さい。フォント属性の設定し直しが必要です。例えば、Font．PLAIN に戻す場合、fontmode を Font．PLAIN に設定して下さい。そうしないと、直前のフォント属性設定値が有効となります。

setColor（Color color）

◆機能◆　文字の色を設定します。デフォルト色は黒です。 色は color クラスを使用して設定し、パラメータとして設定します。

setColor (int rvalue,int gvalue,int bvalue)

◆機能◆　文字の色を設定します。デフォルト色は黒です。パラメータは3つが有ります。各々は赤、緑と青の色数値です。

setCharSize (int size)

◆機能◆　文字のフォントサイズを設定します。

setCharSize (double size)

◆機能◆　　文字のフォントサイズを設定します。

setCharSpace (int value)

◆機能◆　文字の連続しているキャラクタの距離を指定します。デフォルト値は“0”です。

setCharSpace (double value)

◆機能◆　文字の連続しているキャラクタの距離を指定します。デフォルト値は“0”です。

setWordSpace (int value)

◆機能◆　単語間の距離を指定します。デフォルトのスペース値は“0”です。

setWordSpace (double value)

◆機能◆　単語間の距離を指定します。デフォルトのスペース値は“0”です。

setHorizontalScale (int value)

◆機能◆　文字の横向き拡大縮小の比率を設定します。デフォルト値は 100 で、元の状態の横向きの幅を保持します。

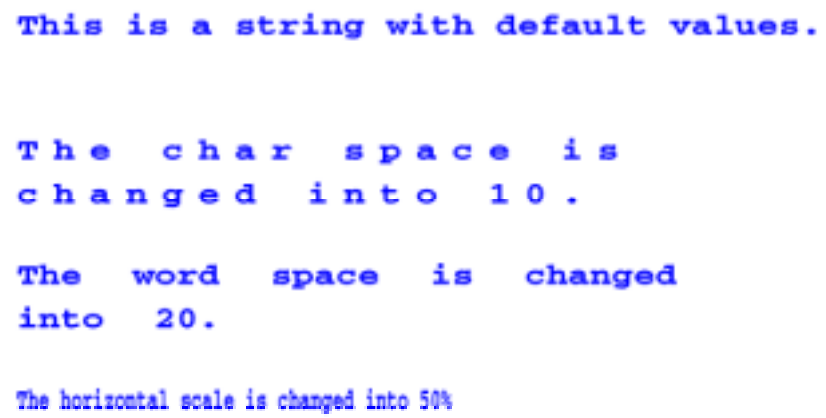


図 5-1 文字間の距離、スペース距離、横向き拡大縮小の比率

5.5 PDFText の文字の幅を取得するメソッド

int PDFText. getStringWidth()
double PDFText. GetStringFloatWidth()

◆機能◆　PDFText の属性により、PDFText に含まれている文字の幅を計算します。

int PDFText. getStringWidth(String s)
double PDFText. GetStringFloatWidth(String s)

◆機能◆　PDFTextの属性により、文字Sの幅を計算します。

5.6 PDFText の別の属性を設定する

setLineWidth (int width)

◆機能◆ 行間距離を設定します。デフォルト値は“10”です。

setLineWidth (double width)

◆機能◆　行間距離を設定します。デフォルト値は“10”です。

setRendering (int renderingMode)

◆機能◆　文字の Rendering モードを設定します。Rendering モードの値の範囲は“0”～“7”です。図5－2を参照して下さい。

図 5-2 テキストモード

setUnderLine ( )、
setUnderLine (Color color)、
setUnderLine (int width)
setUnderLine (double width)
setUnderLine (Color color, int width)
setUnderLine (Color color, double width)

◆機能◆　文字の下線を設定します。パラメータの指定がなければ、下線の色は文字と同色で、太さは“1”に設定されます。下線の幅は文字の幅より決定されます。文字の幅を変えると、下線の幅も自動的に変更されます。

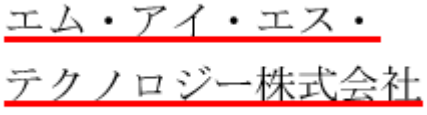


図 5-3　下線文字

setcirRectangle ( )
setcirRectangle (Color color)
setcirRectangle (int width,int height)
setcirRectangle (Color color,int windth, int height)
setcirRectangle (Color color, int width, int height)

◆機能◆　文字の外郭線を設定します。このメソッドを使用する時は、文字の座標を設定する必要があります。外郭線の色を設定しなければ、Text.setColor (Color color) で指定色が使用されます。

図5-4　外郭線のある文字

linkWith (Object)

◆機能◆　文字オブジェクトを現在使用中のプログラム以外の PDF オブジェクトとリンクさせます。パラメータ Object はオブジェクト名です。

linkWith (String pathname, int pagenum)
linkWith (String pathname)

◆機能◆　文字オブジェクトを現在プログラム中の他のＰＤＦファイルとリンクさせます。パラメータはリンクされるＰＤＦファイルのリンク先名 (pathname) とページ番号 (Pagenum) です。

接頭辞には￥￥を使用して下さい。例えば、c:\\root\\xinca です。ページ番号のデフォルト値は“1”です。

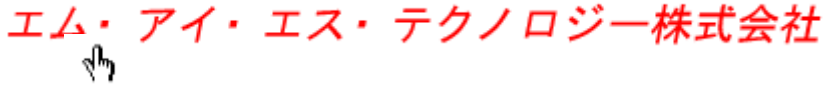


図 5-4 Web と繋がっているリンク文字

linkWith (String website)

◆機能◆　文字オブジェクトを1つのWeb アドレス(URL)とリンクさせます。使用者が文字をクリックすると、サーバーが自動的に起動して、Web アドレス(URL)とリンクします。

URL アドレス名はパラメータとして使われます。例えば、www.mist.co.jpとリンクさせる時は、PDFText.linkWith(“http://www.mist.co.jp”)のように記入して下さい。

setRotation (int angle)

◆機能◆　文字オブジェクトの回転モードを設定します。

図 5-5 回転モード30度の文字

5.7 サンプル

```
PDFText text = new PDFText();
text.setBasefont("gothic", Font.ITALIC);    // set font and style
text.setCharSize(26);                       // set char size
text.setXY(140, 700);                       // set the begining position
text.setText("This was created by PDFText");
page.append(text);                          // append this PDFText to
```

第６章 PDFGraphics

6.1 概要

PDF ファイルにグラフィックスを画く時、このクラスを使用します。

6.2 PDFGraphics のコンストラクタ

new PDFGraphics ( )

◆機能◆ PDFGraphics オブジェクトのインスタンスを宣言します。グラフィックスを追加するには、このメソッドを使用する必要があります。このメソッドで宣言されたインスタンスオブジェクトはデフォルトの属性値を持ち、ラインカラーと外郭線は黒色、ライン capstyle は“0”、太さも“0”となります。

6.3 PDFGraphics の属性を設定するメソッド

setStrokeColor (Color color)

◆機能◆　外枠の色を設定します。色を指定しないと、デフォルトの黒色になります。

setStrokeColor (int rvalue, int gvalue, int bvalue)

◆機能◆　外枠の色を設定します。各パラメータには RGB の赤、緑及び青の成分を指定します。

setFillColor (Color color)

◆機能◆　描画色を設定します。設定しなければ、デフォルトの黒色が設定されます。

setFillColor（int rvalue,int gvalue,int bvalue)

◆機能◆　描画色を設定します。各パラメータは RGB の赤、緑及び青の成分を表します。

setDashLine（int unit)

◆機能◆　点線タイプは白と黒の幅が等しく設定されます。パラメータ値には幅を指定します。

setDashLine（int black, int white)

◆機能◆ 点線のタイプは各指定された幅の単位に設定され、パラメータは白と黒の幅を指定します。

実際に点線を描くには PDFGraphics.drawDashLine()メソッドを使用します。

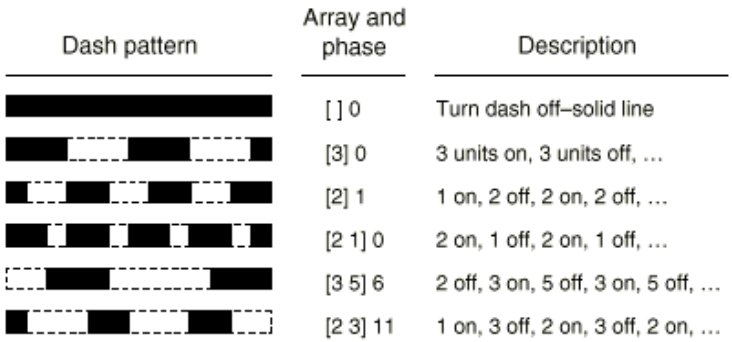


図 6-1 点線の設定

closeDashLine ()

◆機能◆ 指定されていた点線状態をキャンセルします。

点線を描き終わると、このメソッドを使用して点線状態をキャンセルします。点線の指定が継承されて、次に描く時も点線になります。

setCapStyle (int capstyle)

◆機能◆　線のスタイルを設定します。指定可能なパラメータ値は“0”から“2”までの範囲で指定できます。各々の値に対応するスタイルに関しては図6－2を参照して下さい。

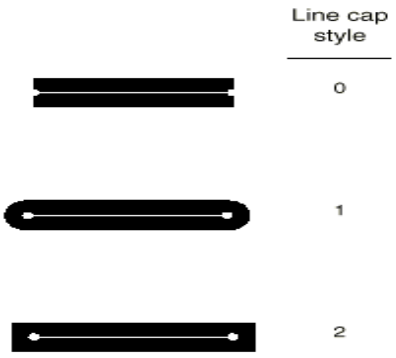


図 6-2 線のスタイル

setWidth (int width)

◆機能◆　グラフィックスの太さを設定します。デフォルトでは“0”が設定されています。このメソッドを使用して指定する必要があります。

setWidth (double width)

◆機能◆　グラフィックスの太さを設定します。上記の setWidth(int width)メソッド、あるいは、どちらか一方を使用しなければなりません。

setLineJoin (int mode)

◆機能◆　接合線のタイプを設定します。パラメータ値は3種類あり、各ＰＤＦGraphics.Miter、ＰＤＦGraphics. Round 及びＰＤＦGraphics.Bevel です。図6－3接合線のタイプを参照して下さい。

図 6-3 接合線のタイプ

setArrow(int arrorwMode,int voneMode,int arrowAngle,int arrowHeight)

◆機能◆　矢印の付いている実線を設定します。パラメータは4つ有ります。各パラメータの機能について説明します。一番目のパラメータ arrowMode の値は矢印の方向を設定し、ＰＤＦGraphics．ARROWNONE、ＰＤＦGraphics．ARROWLEFT、ＰＤＦGraphics．ARROWRIGHT 及びＰＤＦGraphics．ARROWBOTH の1つが使用されます。二番目のパラメータ VoneMode は矢印の形式を指定するために PDFGraphics. NoBone と PDFGraphics.isBone の1つが使用されます。三番目のパラメータ arrowAngle と四番目のパラメータ arrowHeight により、矢印の内角と高さが設定されます。これに関して図6－4を参照して下さい。

矢印の作成は PDFGraphics.strokeLine () あるいは PDFGraphics.drawDashLine () メソッドを使用します。

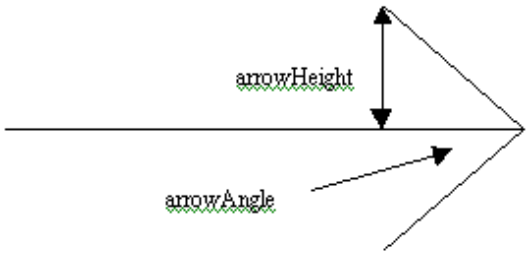


図 6-4 矢印の内角と高さの設定

setFillMode (int mode)

◆機能◆　長方形内部の柄を設定します。パラメータ mode の値は“1”～“6”が指定てきます。各パラメータは FillMode に横直線、縦直線、横縦直線等の6種類の柄を設定します。setFillColor () メソッドと同時使用する事もできます。

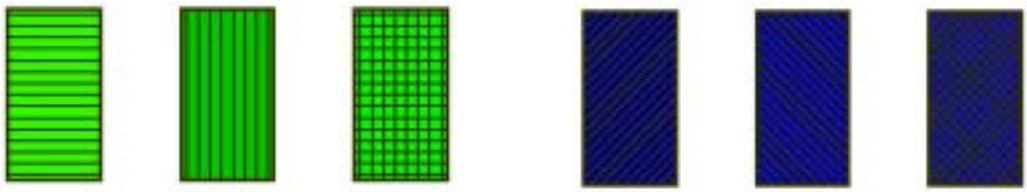


図 6-5 6種類の柄タイプ

6.4 PDFGraphics ラインを画くメソッド

strokeLine（int x0,int y0,int x１,int y１）

◆機能◆　外枠色で実線を作成します。このメソッドを使用する前に、矢印のプロパティ値を設定すると、図 6-6 の様な両方向の赤い矢印が描くことができます。

図 6-6 色の付けた矢印を持つ実線

strokeLine（double x0, doubley0, double x１, doubley１）

◆機能◆　外枠色で実線を作成します。

drawDashLine（int x0, int y0, int x１, int y１）

◆機能◆　点線あるいは矢印の付いている点線を作成します。パラメータには点線の座標を指定します。このメソッドをご使用する前に setDashLine（）を使用して、点線のタイプを設定する必要があります。さらに、setArrow（）メソッドを使用して、矢印が付いている点線を画くことができます。

図 6-7 点線

図 6-8 矢印が付いている点線

drawDashLine（doublex0, doubley0, doublex１, doubley１）

◆機能◆　点線あるいは矢印の付いている点線を作成します。

resetLineStyle（）

◆機能◆　線のスタイルを初期化します。ＰＤＦGraphics.setDashLine（）を使用した後、このメソッドを利用して線のスタイルをリセットします。リセットされるまで、点線の指定が有効となります。

6.5 PDFGraphics の長方形メソッド

strokeRectangle(int x, int y, int width,int Height)

◆機能◆　描画色で長方形を作成します。パラメータは4つ有り、前の2つは長方形左上角の座標を設定し、三番目パラメータのワイズと四番目パラメータのハイズを使って長方形を画きます。図6－9を参照して下さい。

strokeRectangle(double x,double y,double width, double height)

◆機能◆　描画色で長方形を作成します。パラメータの機能はstrokeRectangle(int x, int y, int width, int Height)と同じです。

fillRectangle(int x,int y,int width,int Height)

◆機能◆　長方形を作成して、外枠と同じ色で内部を塗りつぶします。図6－9を参照して下さい。

fillRectangle(double x,double y,double width, double Height)

◆機能◆　長方形を作成し、外枠と同じ色で内部を塗りつぶします。

drawRectangle(int x,int y,int width,int Height)

◆機能◆　塗りつぶす長方形を作成します。図6－9を参照して下さい。

drawRectangle(double x, double y,double width, double Height)

◆機能◆　塗りつぶす長方形を作成します。

図 6-9 作成された長方形

図 6-9 3つ長方形は異なったメソッドで作成されています。左側から使用されたメソッドは strokeRectangle ()、fillRectangle () と drawRectangle () です。

6.6 PDFGraphics のベジェ曲線メソッド

setCurrentPoint (int x, int y)

◆機能◆　カレント座標点を指定します。ベジェ曲線を作成する前に、必ず曲線の始点を指定します。

strokeBezier (double x 0,double y 0, double x1,double y1,double x2,double y2,double x3,double y3)

◆機能◆　第一種類のベジェ曲線を作成します。図 6-10 を参照して下さい。

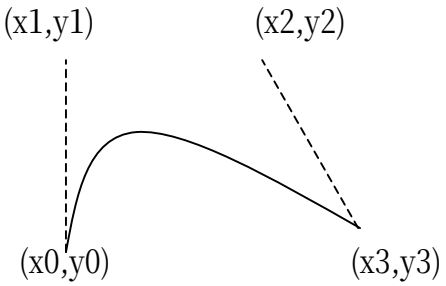


図 6-10 第一種類ベジェ曲線

StrokeBezier (double x0,double y0,double x1,double y1, double x2,double y2, int　style)

◆機能◆　第二種類のベジェ曲線を作成します。全部で2種類が有り、最後のパラメータstyle の値は“1”であれば、図 6-11 の様な曲線になり、“2”を指定すれば、図 6-12 の様な曲線になります。

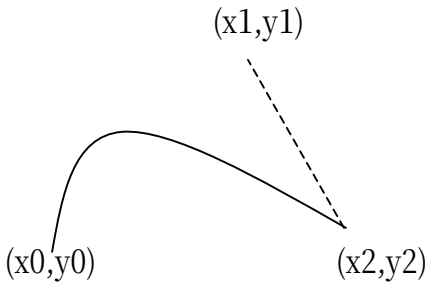


図 6-11 style=”1”のベジェ曲線

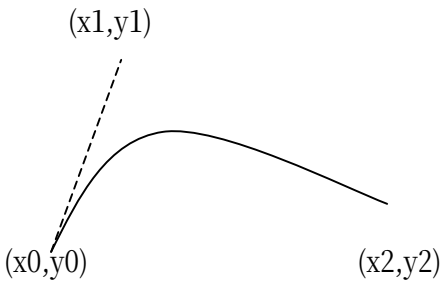


図 6-12 style=”2”のベジェ曲線

strokeCloseBezier （double x0,double y0,double x1,double y1,double x2,double y2,double x3,double y3)

◆機能◆　第一種類のベジェ曲線図を作成します。

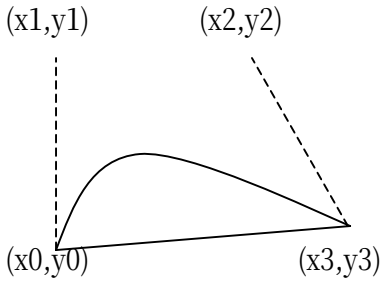


図 6-13 第一種類ベジェ曲線図

StrokeCloseBezier (double x0, double y0,double x1,double y1,double x2,double y2, int　style)

◆機能◆　第二種類のベジェ曲線図を作成します。全部で 2 種類が有り、最後のパラメータ style は“1”であれば、図 6-14 になり、“2”であれば、図 6-15 の様に画かれます。

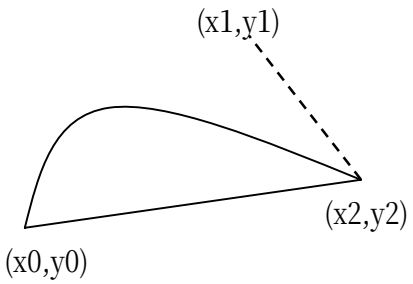


図 6-14 style=”1”のベジェ曲線図

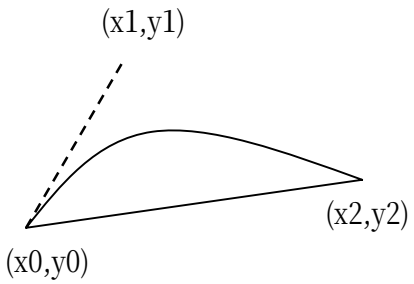


図 6-15 style=”2”のベジェ曲線図

FillBezier(double x0,double y0,double x1,double y1,double x2,double y2,double x3,double y3)

◆機能◆ 第一種類のベジェ曲線を画いて、内部を同じ色で塗りつぶします。

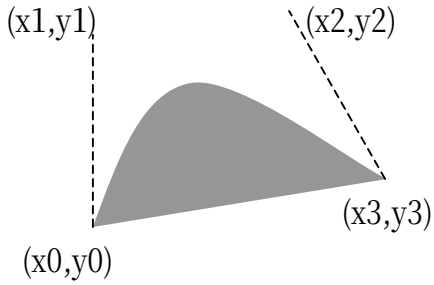


図 6-16 第一種類ベジェ塗りつぶし曲線図形

fillBezier (double x0,double y0,double x1,double y1, double x2,double y2, int style)

◆機能◆ 第二種類のベジェ曲線で塗りつぶし曲線図形を作成します。全部で 2 種類が有り、最後のパラメータstyle が“1”であれば、図 6-17 の様になり、“2”を指定すると、図 6-18 の様になります。

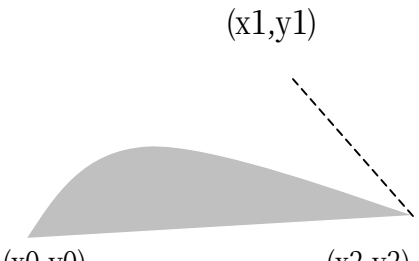


図 6-17 style=”1”の塗りつぶしベジェ曲線図形

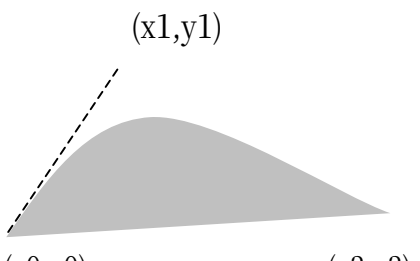


図 6-18 style=”2”の塗りつぶしベジェ曲線図形

drawBezier (double x0,double y0,double x1,double y1,double x2,double y2,double x3,double y3)

◆機能◆ 第一種類の塗りつぶしベジェ曲線図を作成します。図 6-19 を参照して下さい。

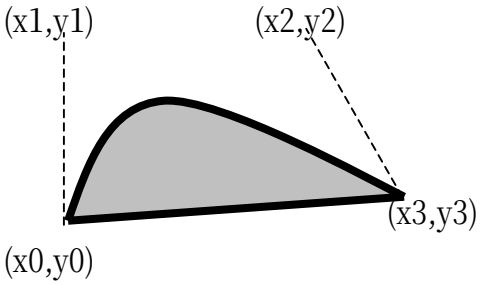


図 6-19 第一種類塗りつぶしベジェ曲線図

drawBezier (double x0,double y0,double x1,double y1,double x2,double y2, int　style)

◆機能◆ 第二種類の塗りつぶしベジェ曲線を作成します。全部で2種類が有り、最後のパラメータ style の値に“1”を指定すれば、図6－20の様になり、“2”を指定すれば図6－21の様になります。

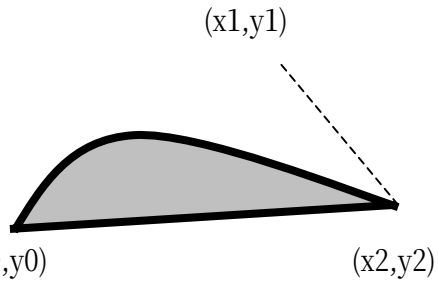


図 6-20 style=”1”の塗りつぶしベジェ曲線

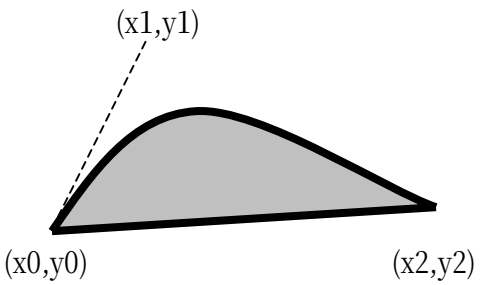


図 6-21 style=”2”の塗りつぶしベジェ曲線

6.7 PDFGraphics の円形種類メソッド

strokeCircle (int x0, int y0,int r)
strokeCircle (double x0, double y0, double r)

◆機能◆　枠線と同じ色で円形を作成します。このメソッドを使用する前に、枠線の色設定メソッド setStrokeColor ( ) を使用する必要があります。パラメータは3つあります。一番目と二番目のパラメータにより、円心の座標を設定し、三番目のパラメータで半径の値を指定します。図6－22を参照して下さい。

fillCircle (int x0,int y0,int r)
fillCircle (double x0, double y0, double r)

◆機能◆　円形を画いて、線と同じ色で内部を塗りつぶします。このメソッドを使用する前に、描画色の設定 setFillColor（ ）メソッドを使用する必要があります。図6－22を参照して下さい。

drawCircle (int x0,int y0,int r)
drawCircle (double x0, double y0, double r)

◆機能◆　塗りつぶした円形を作成します。外枠の色は setStrokeColor（）メソッドによって指定され、塗りつぶし色は setFillColor（）メソッドによって決められます。このメソッドの使用する前に外枠の色と塗りつぶしの色を設定する必要があります。図6－22を参照してください。

図 6-22 作成された円形

図6-22の3つ円形は異なったメソッドで作成されています。左側から、使用したメソッドは drawCircle（）、strokeCircle（）と fillCircle（）です。

6.8 PDFGraphics の多角形メソッド

strokePolygon (int[] x , int[] y, int number)
strokePolygon (double [] x , double [] y, int number)

◆機能◆　枠線の色で多角形を作成します。3つあるパラメータの一番目と二番目のパラメータは整数タイプのアレイで、多角形の頂点座標を保持します。三番目のパラメータ number は頂点の個数が指定され、座標のアレイ個数と合わなければなりません。このメソッドを使用する前に setStrokeColor()メソッドに

よって枠線の色を設定する必要があります。図6－23を参照して下さい。

fillPolygon (int[] x , int[] y , int number)
fillPolygon (double [] x , double [] y , int number)

◆機能◆　多角形を作成して、枠線と同じ色で内部を塗りつぶします。パラメータの機能は前記の内容と同じです。このメソッドを使用する前にsetFillColor( )メソッドによって塗りつぶし色を設定する必要があります。図6－23を参照して下さい。

drawPolygon (int[] x,int[] y,int number)
drawPolygon (double [] x, double [] y,int number)

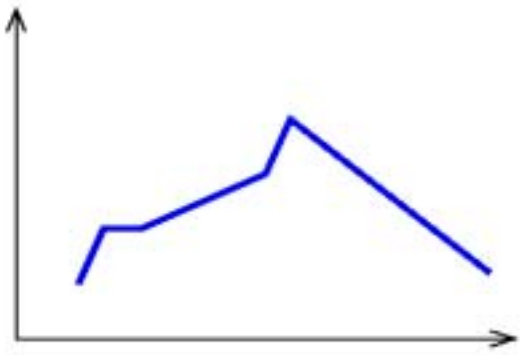
◆機能◆　塗りつぶし多角形を作成します。このメソッドを使用する前に、枠線色は setStrokeColor () メソッドによって設定されむ、塗りつぶし色はsetFillColor()メソッドによって設定しなければなりません。

図 6-23 n個の頂点を持つ多角形

図 6-23 3つ多角形は個々に異なるメソッドで作成されています。左側から使用されているメソッドは drawPolygon ()、strokePolygon () と fillPolygon () です。

drawSegment(int[ ]x,int[ ] y,int number)
drawSegment(double [ ]x, double [ ] y,int number)

◆機能◆　枠線と同色でn個の頂点を持つ線の組み合せを作成します。このメソッドを使用する前に setStrokeColor() メソッドによって、色を設定する必要があります。矢印と一緒に使用すると、図 6-24 の様な、統計グラフと似ている図形を作ることができます。



6.9 PDFGraphics の扇形メソッド

strokeScallop(int x0,int y0,int r,int startangle, int arcangle)
strokeScallop(double x0, double y0, double r,int startangle, int arcangle)

◆機能◆　枠線と同じ色で扇形を作成します。4つのパラメータが有り、一番目と二番目のパラメータは扇形の円中心座標を、三番目のパラメータ startangle は扇形の開始点の角度、四番目のパラメータ arcangle は開始点から左回りの角度を設定します。このメソッドを使用する前に setStrokeColor () メソッドによって枠線の色を設定する必要があります。

fillScallop (int x0,int y0,int r,int startangle, int arcangle)
fillScallop (double x0, double y0,int r , double startangle, int arcangle)

◆機能◆　扇形を作成して内部を同じ色で塗りつぶします。パラメータの機能は上とまったく同じです。このメソッドを使用する前に setFillColor () メソッドによって描画色を設定する必要があります。

drawScallop(int x0,int y0,int r,int startangle, int arcangle)
drawScallop(double x0, double y0, double r,int startangle, int arcangle)

◆機能◆　塗りつぶし扇形を作成します。パラメータの意味は上とまったく同じです。このメソッドを使用する前に setStrokeColor () メソッドによって枠線の色と setFillColor () メソッドによって塗りつぶし色を設定しなければなりません。図6－25は4つの扇形から円形を構成しています。

図 6-25　4つの扇形によって作成された円形

drawArc(int x,int y,int radius,int startAngle, int arcAngle)
drawArc(double x,int y, double radius,int startAngle, int arcAngle)

◆機能◆　円弧を作成します。5つのパラメータが有り、一番目と二番目のパラメータは円の中心座標、三番目のパラメータは半径、四番目のパラメータ startAngle は円弧の開始点の角度、五番目のパラメータ arcAngle は円弧の終了点角度を設定します。

6.10 PDFGraphics の楕円メソッド

strkoeEllips (int x0,int y0,double a,double b)
strkoeEllips (double x0, double y0,double a,double b)

◆機能◆　枠線の色で楕円を作成します。4つのパラメータが有り、一番目と二番目のパラメータは楕円の円中心座標値、三番目のパラメータは楕円のx半径、四番目のパラメータは楕円のy半径を設定します。このメソッドを使用する前に setStrokeColor () メソッドによって、枠線の色を設定する必要があります。パラメータの意味は図6－26を参照して下さい。

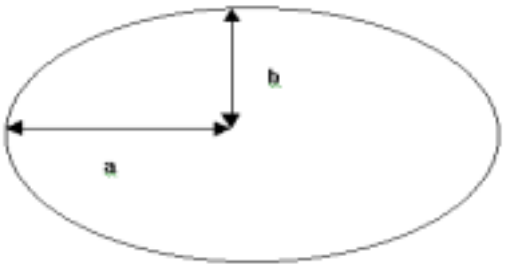


図 6-26 楕円のx半径とy半径

fillEllips (int x0,int y0,double a,double b)
fillEllips (double x0, double y0,double a,double b)

◆機能◆　描画色で楕円の内部を塗りつぶします。パラメータの意味は上のと同じです。

drawEllips(int x0,int y0,double a,double b)
drawEllips(double　x0, double　y0,double a,double b)

◆機能◆　塗りつぶし楕円を作成します。パラメータの意味は上のと同じです。

6.11 サンプル

1．Draw a Line

```
PDFGraphics graphic = new PDFGraphics();
graphic.setStrokeColor(0, 125, 0);                // set line color
graphic.strokeLine(100, 500, 100, 600);        // draw a line
page.append(graphic);
```


図 6-27 サンプル１の結果

2．Draw a Dash Line

```
PDFGraphics graphic = new PDFGraphics();
graphic. setDashLine(6, 2);
graphic.drawDashLine(100, 500, 100, 600);        // draw a dash line
page.append(graphic);
```


図 6-２8 サンプル２の結果

3．Draw a Rectangle

```
PDFGraphics graphic = new PDFGraphics();
graphic. strokeRectangle (100, 500, 50, 30);       // draw a dash rectangle
page.append(graphic);
```

図 6-29 サンプル３の結果

4．Draw a Dash Rectangle

```
PDFGraphics graphic = new PDFGraphics();
graphic. setDashLine(3);
graphic. strokeRectangle (100, 500, 50, 30);       // draw a dash rectangle
page.append(graphic);
```

図 6-30 サンプル４の結果

第７章　PDFImages

7.1 概要

ＰＤＦファイルに画像を挿入します。GIF、JPEGとTIFFファイルを直接ＰＤＦファイルに挿入することが出来ます。

7.2 PDFImages のコンストラクタ

new PDFImages (String pathname)

◆機能◆　PDFImages のインスタンスを宣言します。パラメータは画像ファイルのパス名とファイル名です。Xinca ソフトパッケージで処理できる画像ファイルはGIF、JPEG及びTIFです。このメソッドを使用して、ＰＤＦファイルに画像を入れます。デフォルト状況に挿入される画像の回転及び拡大縮小を行わなく、元の大きさが使用されます。色は DeviceRGB です。画像に、圧縮メソッド LZW と ASCII85を使うことになります。

new PDFImages (String pathname, boolean useLZW)

◆機能◆　PDFImages のインスタンスを宣言します。2つのパラメータが有り、一番目のパラメータ pathname は画像ファイルのパス名とファイル名です。Xinca ソフトパッケージが処理できる画像ファイルはGIF、JPEG及びTIFです。生成される PDF ファイルのサイズを縮める為に、XINCA で画像ファイルを圧縮します。デフォルト状況には、GIF と TIFF ファイルであれば LZW と ASCII85 圧縮算法、JPEG ファイルであれば、JPEG と ASCII85 圧縮算法を使います。JPEG 圧縮算法を使ったら、画像の効果に影響が有る為、JPEG 圧縮算法を使いたくなければ、二番目のパラメータ useLZW によって JPEG の画像ファイルの圧縮算法を変えられます。useLZW が“true”であれば、LZW と ASCII85 圧縮算法が採用されます。

7.3 PDFImages のメソッド

setSize (int width,int height)

◆機能◆　画像の大きさを設定します。このメソッドは使用しなくても構いません。デフォルト状況では、元の画像サイズが使用されます。

PDFPoint PDFImages. getSize ()

◆機能◆　画像の大きさを獲得します。

setXY (int x,int y)

◆機能◆　画像をＰＤＦ上の左下端座標に設定します。

setOffset (int x,int y)

◆機能◆　x座標とy座標に画像のオフセットを設定します。

図 7-1　x座標とy座標に Offset を付いている画像

setRotate (int angle)

◆機能◆　画像の回転モードを設定します。パラメータ angle は回転モード値です。

図7-2 回転された画像

LinkWith (Object)

◆機能◆　画像を他のオブジェクトとリンクさせます。パラメータには作成済みオブジェクトを指定します。例えば、ＰＤＦTextあるいはＰＤＦImagesなどを指定することが可能です。

LinkWith (String fileName, int pageNumber)

◆機能◆　画像は他のＰＤＦファイルの指定されたページとリンクさせます。

LinkWith (String　pathName)

◆機能◆　画像はWebとリンクさせます。パラメータ値はWebのドメインあるいはIPアドレスです。 例えば、http://www.mist.co.jp

getImageDPI ()

◆機能◆　毎インチのドット数、特別にTIFFで有効する。

setMaskColor ()

◆機能◆　白い色をマスクする。

マスクの意味はマスク色を非表示です。

setMaskColor (int red, int green, int blue)

◆機能◆　指定して色をマスクする。

マスクした色は赤、緑、青の三つ色で表示する、パラメターred、green とblue はマスク色を表示する。。

setMaskColor (int redFrom, int greenFrom, int blueFrom, int redTo, int greenTo, int blueTo)

◆機能◆　指定範囲の色をマスクする。

マスクした色は（redFrom、greenFrom、blueFrom）から（redTo、greenTo、blueTo）まで。

7.4 サンプル

```
PDFImages img = new PDFImages("../images/zebralogo.jpg");
img.setXY(350, 300);
page.append(img);
```

第８章 PDFAnnotation

8.1 概要

生成される PDF ファイルの指定場所に注釈行を追加します。

8.2 PDFAnnotation のコンストラクタ

new ＰＤＦAnnotation（）

◆機能◆　PDFAnnotation のインスタンスを宣言します。

8.3 PDFAnnotation のメソッド

setContent（String　content）

◆機能◆　注釈行を設定します。文字列の長さが注釈枠の幅を超えると、自動的に改行されます。図８－１を参照して下さい。

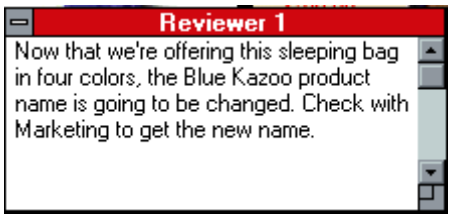


図 8-1 注釈枠

setXY（int ｘ１，int ｙ１，int ｘ２，int ｙ２）

◆機能◆　注釈枠の位置と大きさを設定します。パラメータｘ１、ｙ１は注釈枠の座標位置です。パラメータｘ２、ｙ２は注釈枠が開かれた後の右下端の座標位置です。

setUnicode（）

◆機能◆ 注釈枠の中の文字コードを Unicode に変更します。

setOpen（）

◆機能◆ 注釈枠をデフォルト状況で開かれる様に設定します。この設定を行うことにより、使用者はマウスで注釈枠の所をクリックする必要はなくなります。

8.4 サンプル

```
PDFAnnotation a = new PDFAnnotation();
a.setContent(“PDFAnnotation Test”);
a.setXY(10, 60, 700, 360);
a.setUnicode();
a.setOpen();
page.append(a);
```

第９章 PDFTextGrid

9.1 概要

PDF ファイルにテキストボックスを作ります。

9.2 PDFTextGrid のコンストラクタ

new PDFTextGrid ( )

◆機能◆　テキストボックスを作成します。テキストボックスを作成する前に、PDF Text を作成しなければなりません。作成する時には、setXY () メソッドよりテキストの座標位置を設定します。

9.3 PDFTextGrid のメソッド

setXY (int x, int y)

◆機能◆　テキストボックスの座標位置を設定します。パラメータxyには、はテキストボックスの左上端のxとyの座標です。

setSize (int width,int height)

◆機能◆　テキストボックスの幅と高さを設定します。

setAlignMode (int mode)

◆機能◆ テキストボック中での文字の位置を設定します。パラメータ mode の値範囲は以下のように記載されています。

PDFTextGrid. TopLeftAligned 、　PDFTextGrid. TopMiddleAligned、
PDFTextGrid. TopRightAligned、　PDFTextGrid. LeftAligned、

ＰＤＦTextGrid．MiddleAligned、　ＰＤＦTextGrid．RightAligned
ＰＤＦTextGrid．Bottom LeftAligned、ＰＤＦTextGrid．BottomMiddleAligned
ＰＤＦTextGrid．BottomRightAligned

setLineSpacing（int width）

◆機能◆　いくつのｐｄｆText をテキストボックスに書き込時に、テキストオブジェクト 間の行間を設定します。

appendText（PDFText text）

◆機能◆　作成されたＰＤＦText オブジェクトをテキストボックスに書き込みます。

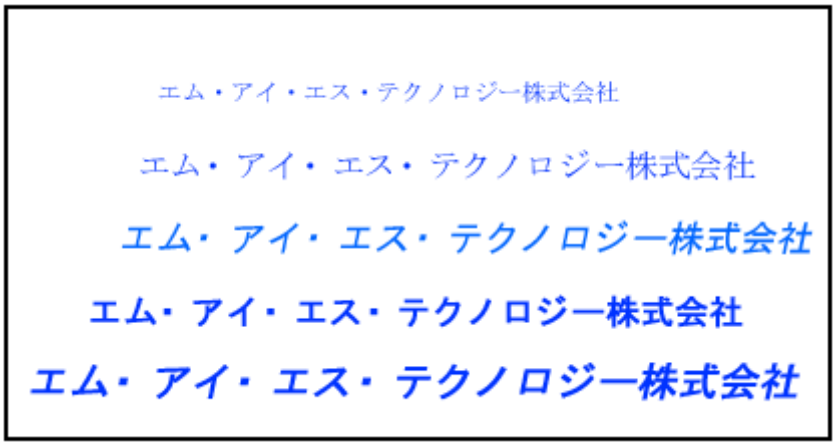


図 9-1 テキストボックス及びその中に書き込まれた文字の設定

9.4 サンプル

```
PDFTextGrid tg = new PDFTextGrid();
tg.setXY(200,200);
tg.setSize(250, 100);
tg.setAlignMode(PDFTextGrid.TopMiddleAligned);
text.setBasefont("MS-Mincho");
text.setText("this is TextGrid sample");
tg.appendText(text);
page.append(tg);
```

第 10 章 PDFMovie

10.1 概要

PDF ファイルに動画オブジェクトを追加します。

10.2 PDFMovie のコンストラクタ

new PDFMovie (String pathnam)

◆機能◆　動画オブジェクトのインスタンスを宣言します。パラメータ pathname は動画ファイルのフル名です。例えば、new ＰＤＦMovie（“ｃ：\\ｘｐ\\pdf\\sample\\Kayak.mov”）

10.3 PDFMovie のメソッド

setXY (int x, int y)

◆機能◆　動画表示エリアの座標値を設定します。

setBoundingBox (int width, int height)

◆機能◆　動画表示エリアの幅と高さを設定します。

図 10-1　ＰＤＦファイルから表示された動画

10.4 サンプル

```
PDFMovie movie = new PDFMovie(“test.mov”);
Movie.setXY(100, 100);
Page.append(movie);
```

第 2 部 XINCA の拡張クラス

本部分は第11章～第17章を含め、XINCA SERVER の拡張クラスに対して説明いたします。これらのクラスはXINCAのコアクラス（第1部に解説した）に基づいて、XINCA をもっと使いやすい為に、設計されました。例えば、PDFTableクラスは帳票生成する為の一種類で、もっと便利なインタフェースとして提供されます。PDFTextArea クラスはテキストレイアウトを行うことを目的として設計されています。

拡張クラスの殆どは XINCA SERVER4．0で新たに追加された機能です。あるクラスは旧いバージョンでも提供されていましたが機能として貧弱なため、新しいバージョンでは、機能強化と使い易いさを大幅に改善いたしました。

以下のリストは 4.0 において、追加、強化されたクラスです。

xinca.PDFEncodeJA
xinca.tools.PDFCheckbox
xinca.tools. PDFRoundRect
xinca.tools. PDFTextArea
xinca.tools. PDFTextV
xinca.tools. PDFTextAreaV
xinca.tools. PDFTable

第 11 章 PDFEncodeJA

11.1 概要

PDFEncodeJAクラスの２つの設計要因

1.　JDK では、ある日本語テキスト(例えば__、①と㈱等)をサポートしません。この訳は JDBC により、これらのテキストに含まれるデートを読み込む際に、エラーテキストとなります。テキストファイルから読み込む時も、同様な問題が発生します。
2. PDF ファイルは OS に依存しないという特徴が有りますが日本語には SJIS と EUC と言われる 2 つのエンコード方式があるので日本語を含む PDF ファイルは実は OS に深く関わっています。あるいは、OS の日本語文字のコード体系と関係があると言えます。例えば、図 11-1 の応用環境で日本語文字を直接 PDF ファイルに書き込むと、運用環境の日本語文字は EUC コード体系のため、生成される PDF ファイル中の日本語文字も EUC コード体系になります。しかし、PDF ファイルを表示する環境の日本語文字は SJIS コード体系ですから、この様な場合には日本語文字を正しく表示することができません。その為、PDF ファイルを生成する時には日本語文字を SJIS コード体系に変換する必要があります。JDK は一部の日本語文字をサポートしないため、java.lang.String.getBytes()メソッドを使用しても、正しいコードデータを得ることができません。

PDFEncodeJA クラスを使用することで、以上の問題を解決することができます。図 11-2 は通常の状況でデータベースを読み込むためのプロセスです。図 11-3 は PDFEncodeJA を利用してデータベースを読み込むためのプロセスです。図 11-4 は PDFEncodeJA より byte array タイプの日本語文字のコード体系を java.langString タイプに変換するロジック図です。図 11-5 は PDFEncodeJA により、日本語に含まれる java.langString タイプを byte array タイプに変換するロジック図です。

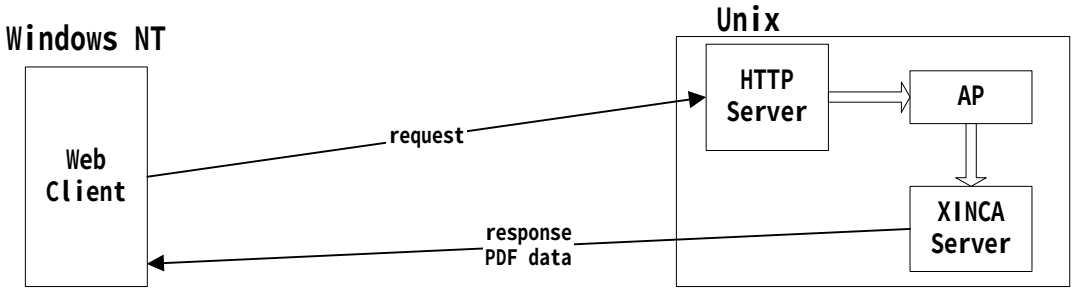


図11-1 応用サンプル

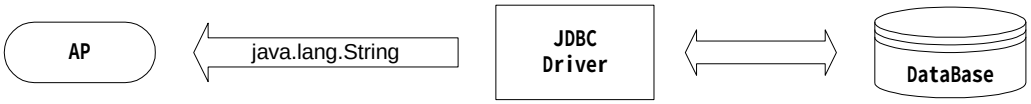


図11-2　通常データベースを読み込むプロセス

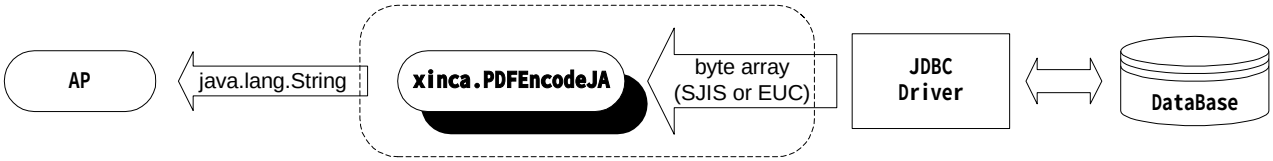


図11-3　PDFEncodeJAを使用する時データベースを読み込むプロセス

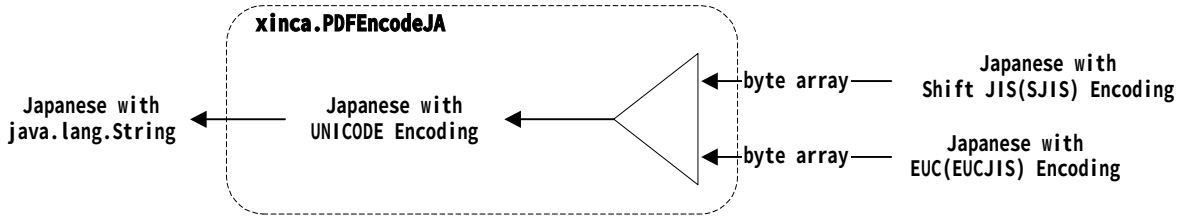


図11-4　PDFEncodeJAコード体系の変換方式1

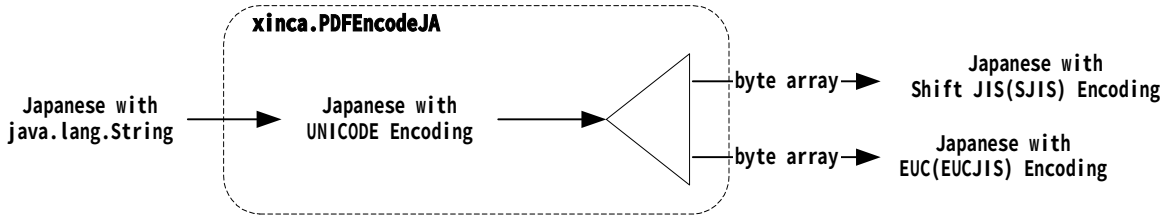


図11-5　PDFEncodeJAコード体系の変換方式2

11.2 PDFEncodeJA のメソッド

public static String getStringSJIS(byte[] bytes))

◆機能◆ byte array タイプで表示される日本語文字の SJIS コード体系を java.langString タイプに変換します。パラメータは SJIS 文字コード体系のデータです。戻り値は変換後の java.langString オブジェクトです。

public static String getStringEUCJIS(byte[] bytes)

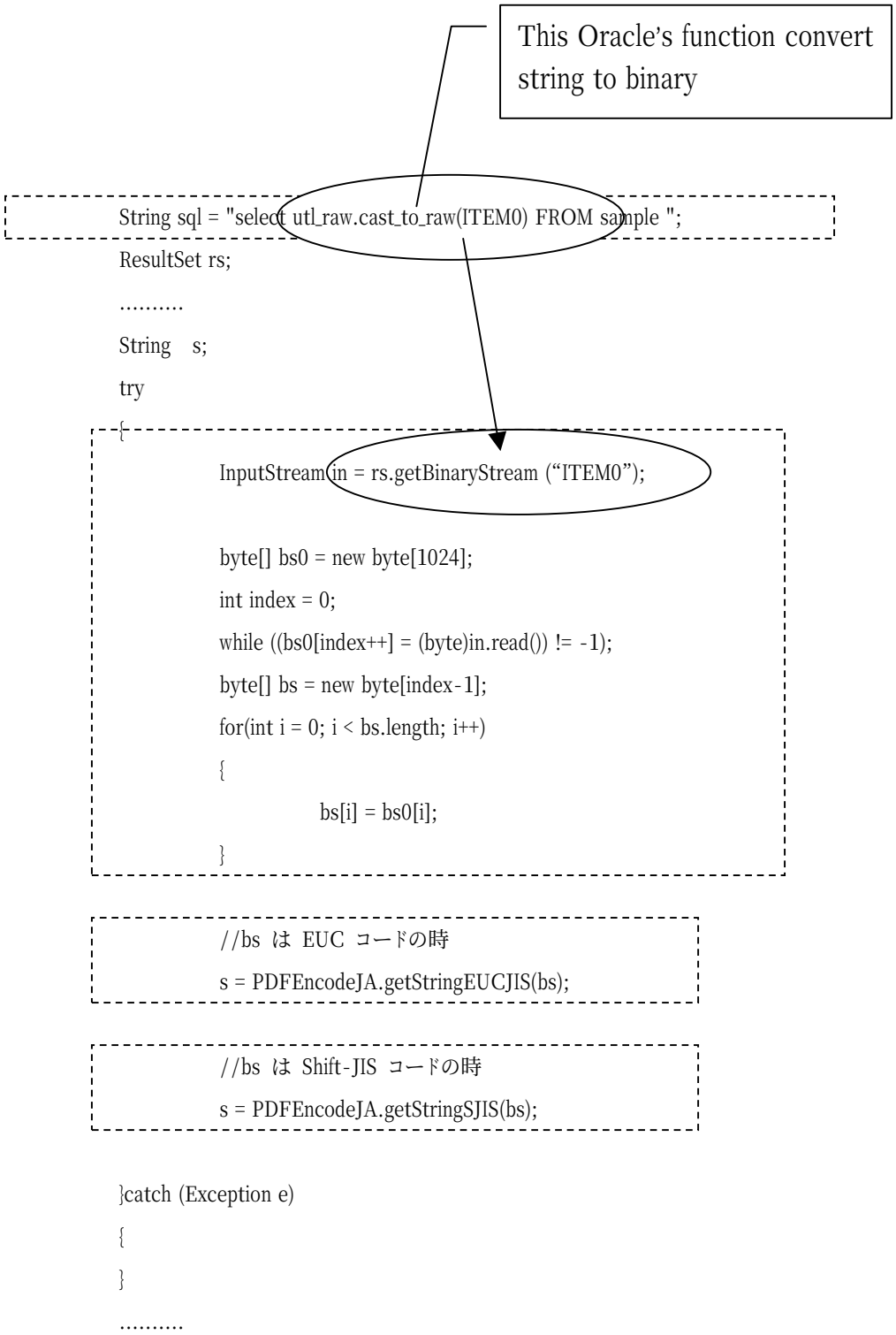
◆機能◆ byte array タイプで表示される日本語文字の EUC コード体系を java.langString タイプに変換します。パラメータは EUC 文字コード体系のデータです。戻り値は変換後の java.langString オブジェクトです。

public static byte[] getStringBytesSJIS(String s)

◆機能◆ 日本語ストリングの SJIS コード体系のデータを取得します。パラメータ s は 日本語文字を含む java.langString オブジェクトです。戻り値は入力されるパラメータと対応する SJIS コード体系のデータです。

getStringBytesEUCJIS () メソッドはありません。ご注意下さい。

11.3 サンプル


This Oracle's function convert string to binary
String sql = "select utl_raw.cast_to_raw(ITEM0) FROM sample ";
ResultSet rs;
.....
String　s;
try
{
InputStream in = rs.getBinaryStream ("ITEM0");
byte[] bs0 = new byte[1024];
int index = 0;
while ((bs0[index++] = (byte)in.read()) != -1);
byte[] bs = new byte[index-1];
for(int i = 0; i < bs.length; i++)
{
bs[i] = bs0[i];
}
//bs は EUC コードの時
s = PDFEncodeJA.getStringEUCJIS(bs);
//bs は Shift-JIS コードの時
s = PDFEncodeJA.getStringSJIS(bs);
}catch (Exception e)
{
}
.....

第 12 章 PDFCheckbox

12.1 概要

PDF ファイルにチェックボックスを作ります。

12.2 PDFCheckbox のコンストラクタ

PDFCheckbox(String title)
PDFCheckbox(PDFText title)

PDFCheckbox(String title, boolean checkflag)
PDFCheckbox(PDFText title, boolean checkflag)

PDFCheckbox(int x0,　　int y0,　　PDFText title)
PDFCheckbox(double x0, double y0, PDFText title)

PDFCheckbox(int x0,　　int y0,　　PDFText title, boolean checkflag)
PDFCheckbox(double x0, double y0, PDFText title, boolean checkflag)

PDFCheckbox(int x0,　　int y0,　　String title)
PDFCheckbox(double x0, double y0, String title)

PDFCheckbox(int x0,　　int y0,　　String title, boolean checkflag)
PDFCheckbox(double x0, double y0, String title, boolean checkflag)

PDFCheckbox(int x0,　　int y0)
PDFCheckbox(double x0, double y0)

以上のメソッドの1つを使用して PDFCheckbox のインスタンスを宣言します。パラメータ x0 と y0 は PDFCheckbox オブジェクトの左下角座標を指定します。データタイプは整数あるいは倍精度です。title は PDFCheckbox オブジェクトのテキスト内容です。checkflag は PDFCheckbox オブジェクトの状態です。PDFCheckbox には2つの定数があります。

PDFCheckbox.CHECKED は、チェック状態の CHECKBOX です。
PDFCheckbox.UNCHECK は、未チェック状態の CHECKBOX です。

12.3 PDFCheckbox のメソッド

setXY(int x0,　int y0)
setXY(double x0, double y0)

◆機能◆ PDFCheckbox オブジェクトの左下角座標を設定します。パラメータ x0 と y0 は PDFCheckbox を指定します。データタイプは整数あるいは倍精度です。

setBoxRightSpace(int w)
setBoxRightSpace(double w)

◆機能◆ ボックス枠とテキスト間のスペースを設定します。パラメータ w はピクセルを単位の幅を指定します。データタイプは整数あるいは倍精度です。

setBoxLeftSpace(int w)
setBoxLeftSpace(double w)

◆機能◆ BOX 枠左側のスペースを設定します。パラメータ w はピクセルを単位としての幅です。データタイプは整数あるいは倍精度です。

setTitle(String title)

◆機能◆ PDFCheckbox オブジェクトのタイトルを設定します。パラメータ title は PDFCheckbox のタイトルテキストです。データタイプは java.lang.String です。

setChecked()

◆機能◆ PDFCheckbox オブジェクトの状態を“CHECKED”に設定します。

setUncheck()

◆機能◆ PDFCheckbox オブジェクトを“UNCHECK”に設定します。

setBoxSize(int size)

◆機能◆ ボックスのサイズを設定します。パラメータ size はボックスの大きさをコントロールします。

setBoxColor(Color c)

◆機能◆ ボックス枠とタイトルテキストの色を設定します。パラメータ c は java.awt.Color タイプの色値です。

PDFPoint getSize()

◆機能◆ PDFCheckbox オブジェクトのサイズを取得します。ボックス枠、タイトルテキストとボックス枠前のスペース(設定される場合だけ)が含まれます。戻り値は

xinca.tools.PDFPoint.x　PDFCheckbox オブジェクトの幅
xinca.tools.PDFPoint.y　PDFCheckbox オブジェクトの高さ

となります。

12.4 サンプル

1. PDFCheckbox checkbox = new PDFCheckbox("Xinca CheckBox Sample1");
   checkbox.setXY(100, 100);
   page.append(checkbox);

図 12-1 サンプル1

2. checkbox = new PDFCheckbox("Xinca CheckBox Sample2",
   PDFCheckbox.CHECKED);
   checkbox.setXY(100, 200);
   checkbox.setBoxSize(12);
   page.append(checkbox);

図 12-2 サンプル2

第 13 章 PDFRoundRect

13.1 概要

角丸四角形を作成します。

NEW PDFRoundRect ()

◆機能◆ PDFRoundRect オブジェクトのインスタンスを宣言します。

13.3 PDFRoundRect のメソッド

setXYWHR(int x0, int y0, int rectWidth, int rectHeight, int round)

◆機能◆ PDFRoundRect オブジェクトの位置、サイズと角丸の半径を設定します。パラメータは5つが有ります。
x0、 y0 は左下角の座標値
rectWidth はボックスの幅
rectHeight はボックスの高さ
round は角丸の半径

setXY(int x0,int y0)

◆機能◆ PDFRoundRect オブジェクトの位置を設定します。パラメータ x0、y0 は左下角座標の値です。

setRectWidth(int rectWidth)

◆機能◆ PDFRoundRect オブジェクトの幅を設定します。パラメータ rectWidth はボックス幅の値です。

setRectHeight(int　rectHeight)

◆機能◆ PDFRoundRect オブジェクトの高さを設定します。パラメータ rectHeight はボックスの高さです。

setRound(int　round)

◆機能◆ PDFRoundRect オブジェクトの角丸外形を設定します。パラメータ round は角丸の半径です。

SetLineWidth(double Linewidth)
SetLineWidth(int Linewidth)

◆機能◆ PDFRoundRech オブジェクトのラインワイズを設定します。パラメータ Linewidth は四角形の枠のワイズです。パラメータのデータタイプは整数あるいは倍精度です。

setLineColor(Color　c)

◆機能◆ PDFRoundRect オブジェクトのラインの色を設定します。パラメータ c は java.awt.Color クラスで表示される色の値です。

setDashLine(int　black, int　white)

◆機能◆ PDFRoundRect オブジェクトの線のスタイルを点線に設定し、点線の属性も設定します。作成された点線は黒、白の様にワイズより表示されます。図13-3を参照してください。2つのパラメータがあります。一番目のパラメータblackは黒線のワイズ、二番目のパラメータwhiteは白線のワイズを表示します。

13.4 サンプル

1．
```
PDFRoundRect rect = new PDFRoundRect();
rect.setXYWHR(100, 500, 150, 50, 10);
rect.setLineWidth(3);
page.append(rect);
```

図 13-1 サンプル 1

2．
```
PDFRoundRect rect = new PDFRoundRect();
rect.setXYWHR(100, 500, 150, 50, 10);
rect.setXY(100, 300);
rect.setLineColor(Color.green);
rect.setLineWidth(1);
page.append(rect);
```

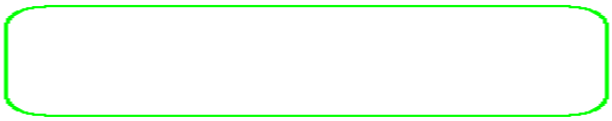


図 13-2 サンプル 2

3．
```
PDFRoundRect rect = new PDFRoundRect();
rect.setXYWHR(100, 500, 150, 50, 10);
rect.setDashLine(3, 2);
rect.setLineWidth(3);
page.append(rect);
```

図 13-3 サンプル 3

第 14 章 PDFTextArea

14.1 概要

PDFTextArea クラスは xinca.tools パッケージ中の1つのテキストレイアウトツールです。PDFTextArea オブジェクト中のテキストに対して、自動改行とレイアウト等の処理を行われます。ユーザーはテキストのスタイル、表示領域の幅、テキスト内容とスタートされる位置を指定すると、他の指定は要らなくて全部 PDFTextArea クラスより自動処理できます。主な機能を以下のように紹介させて頂きます。

テキストの属性

・　1つの PDFTextArea オブジェクト中に多数スタイル（フォント名、フォントサイズと色等）のテキストを含まれます。これらの属性は PDFTextArea オブジェクトに追加される PDFText オブジェクトの属性によって得られます。この後、追加される java.lang.String オブジェクトはすべてこのスタイルを採用されることになります。

段落(paragraph)・

PDFTextArea オブジェクト中のテキストを段落定義する方法は2つが有ります。1 つ目は PDFTextArea オブジェクトに1つの PDFText オブジェクトを追加します。2つ目はテキストに改行マーク(JAVA の場合は"\n")を追加します。

改行・

指定される範囲を超えるテキストに対して自動改行を行われます。その時、1バイトのアルファバイト英語単語だけじゃなくて、2バイトの日本語漢字でも処理できます。1バイトの英語に対して、単語を単位として処理され、2バイトの日本語に対して1つの漢字を単位として処理されます。

余白(margin)

PDFTextArea オブジェクトの上下左右の余白を設定されます。図 14-7 を参

照して下さい。

アレンジ方式

PDFTextArea は PDFAligned に定義される以下の3つ揃え方式をサポートされます。PDFAligned.LEFT_ALIGNED、PDFAligned.MIDDLE_ALIGNED と PDFAligned.RIGHT_ALIGNED です。図 14-3～図 14-5 を参照して下さい。

①1つ段落のテキストは同じのフォントスタイルを採用しなければなりません。即ち、フォントの属性を変えると新しい段落を始める事を表明します。
②PDFTextArea オブジェクトに追加される Java.lang.String オブジェクトに PDFText オブジェクトのフォント属性を使用する為、とりあえず PDFText オブジェクトを追加しなければなりません。

図14-1はPDFTextAreaクラスを使用して生成されるPDFファイルです。図14-2は対応するJAVAのソースコードです。

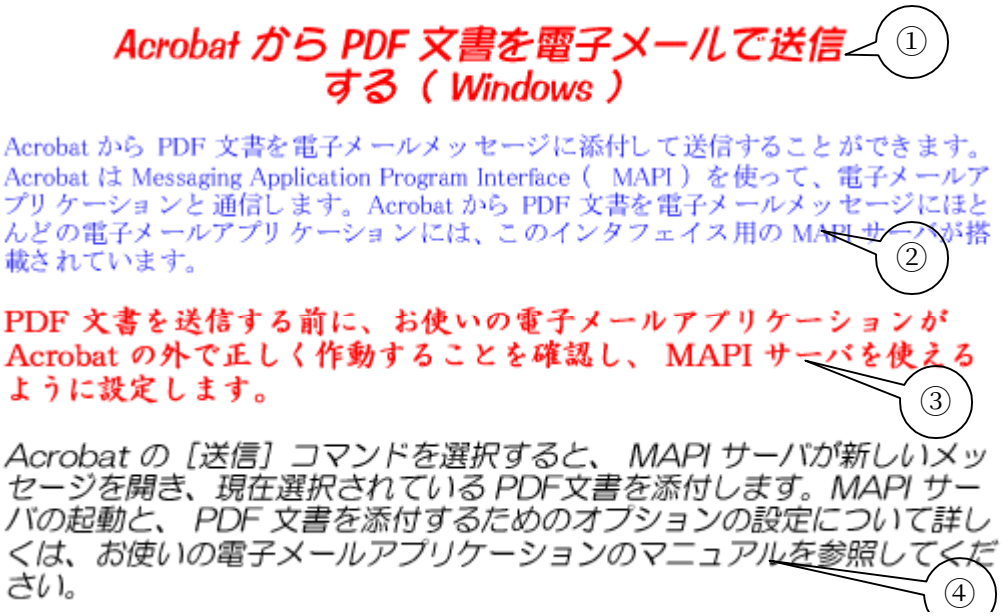


図 14-1 PDFTextArea サンプル

```
PDFTextArea textarea;
PDFText        text;
int            posY = 700;

textarea = new PDFTextextarea();
textarea.setAlignMode(PDFAligned.MIDDLE_ALIGNED);

text.setBasefont("ＤＦＰＰＯＰ体", Font.ITALIC);
text.setColor(Color.red);
text.setCharSize(15.5);
text.setText("\nAcrobat から PDF 文書を電子メールで送信する（ Windows ）\n");
textarea.append(text);

textarea.setSize(300);
p = textarea.getSize();
textarea.setXY(130, posY - p.y);
page.append(textarea);
posY -= p.y;

textarea = new PDFTextextarea();

text.setBasefont("ＭＳＰ明朝", Font.PLAIN);
text.setColor(Color.blue);
text.setCharSize(10.4);
text.setText("Acrobat から PDF 文書を電子メールメッセージに");
textarea.append(text);
textarea.append("添付して送信することができます。");
textarea.append("Acrobat は Messaging Application Program Interface （ MAPI ）を使って、");
textarea.append("電子メールアプリケーションと通信します。");
textarea.append("Acrobat から PDF 文書を電子メールメッセージに");
textarea.append("ほとんどの電子メールアプリケーションには、このインタフェイス用の ");
textarea.append("MAPI サーバが搭載されています。\n");

text.setBasefont("HG 正楷書体-PRO", Font.BOLD);
text.setColor(Color.red);
text.setCharSize(12.4);
text.setText("");
textarea.append(text);
textarea.append("PDF 文書を送信する前に、お使いの電子メール");
textarea.append("アプリケーションが Acrobat の外で正し");
textarea.append("く作動することを確認し、 MAPI サーバを使えるように設定します。 \n");

text.setBasefont("HG 丸ｺﾞｼｯｸM-PRO", Font.ITALIC);
text.setColor(Color.black);
text.setCharSize(12);
text.setText("");
textarea.append(text);
textarea.append("Acrobat の［送信］コマンドを選択すると、 ");
textarea.append("MAPI サーバが新しいメッセージを開き、");
textarea.append("現在選択されている PDF 文書を添付します。");
textarea.append("MAPI サーバの起動と、 PDF 文書を添付するための");
textarea.append("オプションの設定について詳しくは、");
textarea.append("お使いの電子メールアプリケーションのマニュアルを参照してください。");
textarea.setSize(400);
p = textarea.getSize();
textarea.setXY(100, posY - p.y);
page.append(textarea);
posY -= p.y;
```


図 14-2 サンプルのソース

14.2 PDFTextArea のコンストラクタ

new PDFTextArea()

◆機能◆ PDFTextArea オブジェクトのインスタンスを宣言します。

14.3 PDFTextArea のメソッド

setMargin(int left, int right, int top, int bottom)

◆機能◆ テキスト表示領域の上下左右余白を設定します。図 14-7 を参照して下さい。パラメータは4つが有ります。パラメータ left は左側の余白、right は右側の余白、top は上の余白と bottom は下側の余白を体表されます。

append(PDFText text)

◆機能◆ 1つ PDFText オブジェクトを PDFTextArea に追加します。パラメータ text は追加なされる PDFText オブジェクトです。パラメータはテキストの属性とテキスト内容を含まれます。

append(String s)

◆機能◆ テキストを PDFTextArea オブジェクトに追加します。当テキストの表示スタイルはこの前に追加された PDFText オブジェクトのフォント属性より決められます。パラメータ s は追加されるテキストの内容です。

setSize(int width)

◆機能◆ テキスト表示領域の幅を設定します。この時、テキスト表示領域の高さはこの幅値とテキスト内容及びテキスト属性によって自動計算されます。パラメータ width は幅です。

setSize(int width, int height)

◆機能◆ テキスト表示領域の幅と高さを設定します。この時、テキストの実際高さは設定値より大きい場合には、超える部分を表示されなくなります。パラメータ width は幅の値、height は高さの値です。

setAlignMode(int mode)

◆機能◆ テキストの揃え方式を設定します。図 14-3～図 14-5 を参照して下さい。パラメータ mode は揃え方式で、次の 3 つの PDFAligned で定義される揃え方が使用できます。

PDFAligned.LEFT_ALIGNED	左揃え
PDFAligned.MIDDLE_ALIGNED	中央揃え
PDFAligned.RIGHT_ALIGNED	右揃え

setXY(int x, int y)

◆機能◆ PDFTextArea オブジェクトの左下角位置を設定します。パラメータ x、y は左下角の座標値です。

PDFPoint getSize()

◆機能◆ PDFTextArea オブジェクトの大きさを取得します。戻り値は
PDFPoint.x　PDFTextArea オブジェクトの幅
PDFPoint.y　PDFTextArea オブジェクトの高さです。

setLineSpacing(int height)

◆機能◆ 行送りを設定します。パラメータ height は行送り値です。

setLineSpacing(double height)

◆機能◆ 行送りを設定します。パラメータ height は行送り値です。

setLeadingSpaceChar(int n)

◆機能◆ 行ごとに右のインデントを設定します。設定は段落の 2 行目から適

応されます。図 14-6 を参照して下さい。パラメータ n は1バイトを単位として設定されたインデントです。

setColor(Color c)

◆機能◆ フォント色を設定します。パラメータ c は色値、java.lang.Color オブジェクトです。

setBorder(Color c)

◆機能◆ PDFTextArea オブジェクトに枠を画きます。パラメータは枠色です。デフォルトには枠を画きません。パラメータ c は枠の色、java.lang.Color オブジェクトです。

setBorder(Color c, boolean dashborder)

◆機能◆ PDFTextArea オブジェクトに枠を画きます。パラメータ *C* 枠の色です。パラメータ *dashborder* の値は“true”であれば、枠線は点線です。デフォルトには枠を画きません。パラメータ c は枠の色、java.lang.Color オブジェクトです。

14.4 コーディングのステップ

1．xinca.tools をインポートします。

```
import xinca.tools.*;
```

2．PDFTextArea対象をコンストラクタします。

```
PDFTextArea textarea = new PDFTextArea();
```

3．テキストの揃え方式を設定します。

```
textarea.setAlignMode(PDFAligned.MIDDLE_ALIGNED);
```

4．PDFTextオブジェクトを作って、フォントスタイルを設定します。

```
PDFText text = new PDFText();
```

．．．．．．

5．このPDFTextオブジェクトをPDFTextAreaに追加します。

```
textarea.append(text);
```

6．他のテキスト内容を追加します。

```
textarea.append("test");
```

7．テキスト表示領域の幅を設定します。

```
textarea.setSize(width);
```

8．テキスト表示領域左下角座標の位置を設定します。

```
textarea.setXY(x, y);
```

9．このPDFTextAreaオブジェクトをPDFPageに追加します。

```
page.append(textarea);
```

14.5 サンプル

１．サンプル１

```
01        PDFText text = new PDFText()
02        PDFTextArea textarea = new PDFTextextarea();
03        text.setBasefont("HG 正楷書体-PRO", Font.ITALIC);
04        text.setColor(Color.black);
05        text.setCharSize(fontsize);
06        text.setText("PDF は、ファイルサイズが小さく、");
07        textarea.append(text);
08        textarea.append("どのプラットフォームでも使え、”);
          .......
09        textarea.append("または CD で配布することができます。\n”);
10        textarea.setSize(500);
11        textarea.setBorder(Color.green);
12        textarea.setXY(50, 300);
13        page.append(textarea);
```

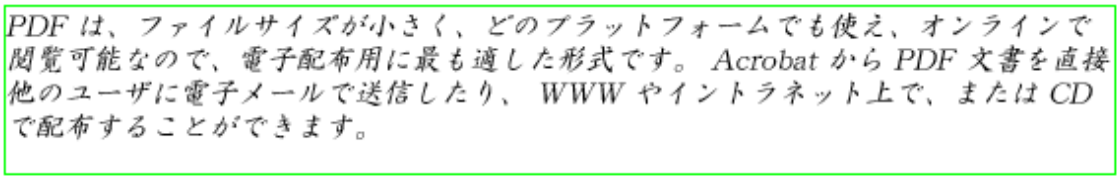


図 14-3 サンプル1

２．サンプル２

サンプル１に２行目に次のソースを追加し、中央揃え方式に設定されます。
textarea.setAlignMode(PDFAligned.MIDDLE_ALIGNED);

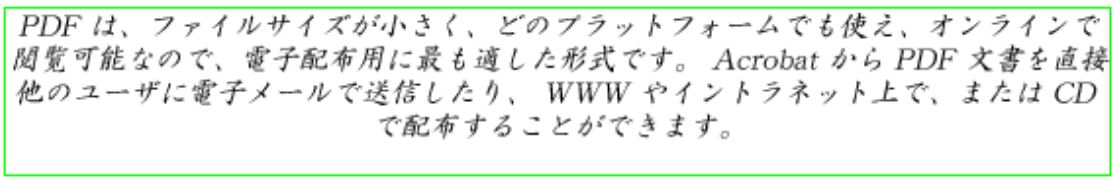


図 14-4 サンプル2

3．サンプル３

サンプル 1 の 2 行目に次のコードを追加し、右揃え方式を設定されます。図14－5を参照して下さい。

textarea.setAlignMode(PDFAligned.RIGHT_ALIGNED);

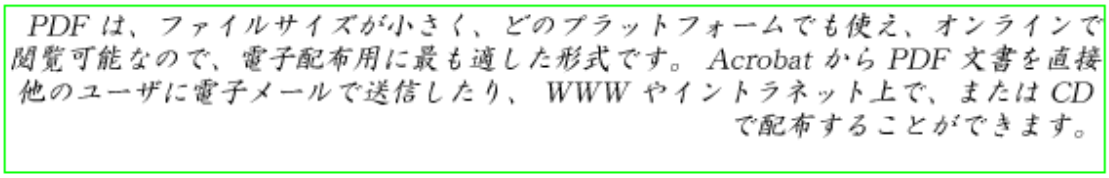


図 14-5 サンプル3

4．サンプル４

サンプル 1 の 2 行目に次のコードを追加し、段落の 2 行目から右にインデント8バイトになります。図14－6を参照して下さい。

textarea. setLeadingSpaceChar(8);

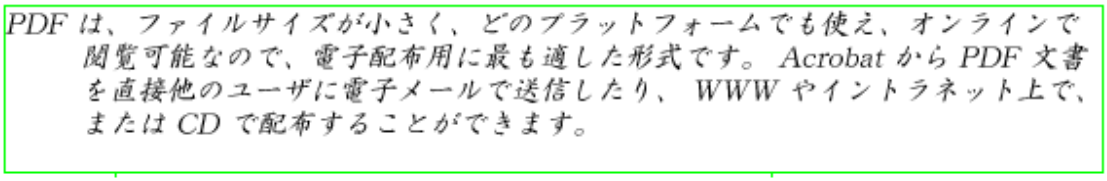


図 14-6 サンプル4

5．サンプル５

サンプル 1 の 2 行目に次のコードを追加し、テキスト表示領域の上下左右余白を設定されます。図14－7を参照して下さい。

textarea.tArea.setMargin(2, 2, 15, 15);

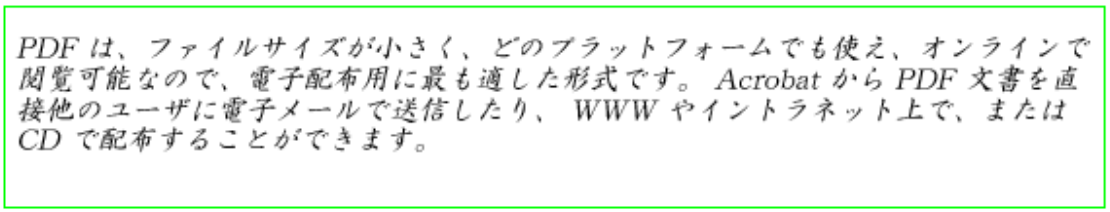


図 14-7 サンプル5

第 15 章 PDFTextV

15.1 概要

PDFTextV クラスは xinca.tools パッケージの縦テキストを作成するツールです。

15.2 PDFTextV のコンストラクタ

new PDFTextV(PDFText text)

◆機能◆ PDFTextV オブジェクトのインスタンスを宣言します。パラメータ text は PDFText オブジェクトのインスタンスです。

15.3 PDFTextV のメソッド

PDFPage.append(PDFTextV textv)

◆機能◆ PDFTextV オブジェクトを PDFPage に追加します。

15.4 サンプル

```
PDFText text = new PDFText();
text.setBasefont("Courier");
text.setCharSize(12);
text.setText("this is text");
text.setXY(280, 600);
PDFTextV textV = new PDFTextV(text);
page.append(textV);
```

図 15-1 PDFTextV サンプル

第 16 章 PDFTextAreaV

16.1 概要

xinca.tools パッケージ中の PDFTextArea クラスの機能と似ているテキストレイアウトツールです。相違点は PDFTextAreaV 中のテキストは縦のように右から左のレイアウト形式だけです。テキストの揃え方式も上、中と下という 3 つが有ります。図 16-1 は PDFTextAreaV クラスを使用するサンプルです。

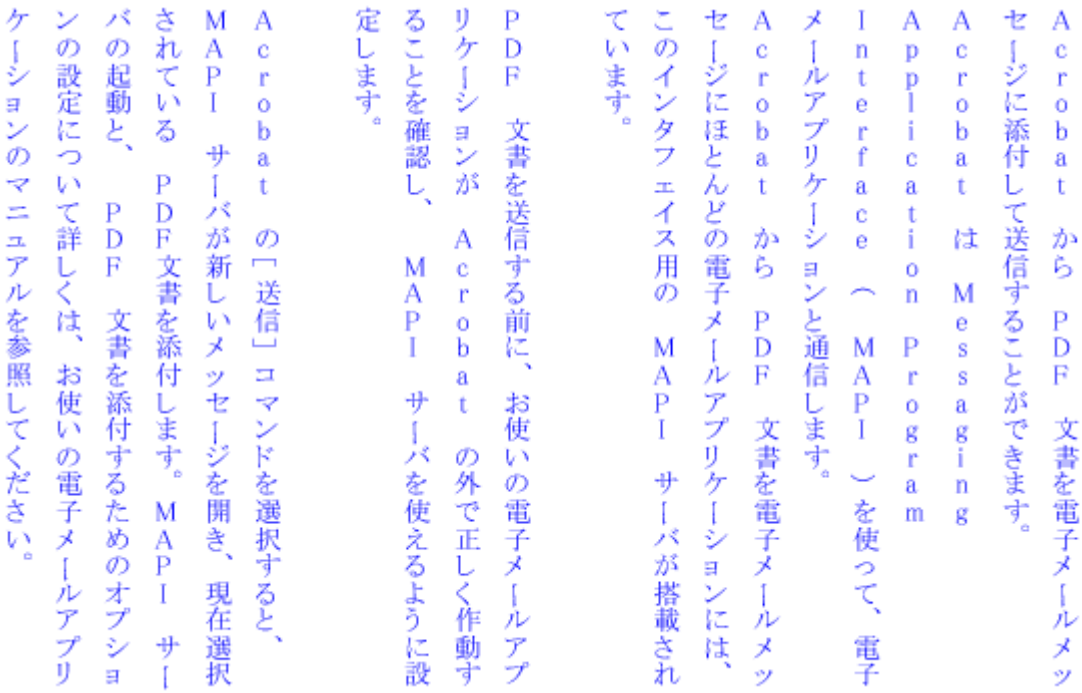


図 16-1 PDFTextV サンプル

16.2 PDFTextAreaV のコンストラクタ

new PDFTextAreaV()

◆機能◆ PDFTextAreaV オブジェクトのインスタンスを宣言します。

16.3 PDFTextAreaV のメソッド

void append(PDFText text)

◆機能◆ 1 つ PDFText オブジェクトを PDFTextAreaV オブジェクトに追加します。パラメータ text は追加される PDFTextV オブジェクトです。パラメータにテキストの属性とテキストの内容を含まれます。

void append(String s)

◆機能◆ PDFTextAreaV オブジェクトにテキストを追加します。当テキストの表示スタイルはこの前追加される PDFText オブジェクトの属性より決められます。パラメータ s は追加なされるテキスト内容です。

void setSize(int height)

◆機能◆ テキスト表示領域の高さを設定します。この時、テキスト表示領域の幅は当高さとテキスト内容及びテキスト属性により自動計算されます。パラメータ height はテキスト表示領域の高さです。

void setSize(int width, int height)

◆機能◆ テキスト表示領域の高さと幅を設定します。この時、テキストの実際幅は設定値より大きくければ、超える部分を表示されなくなります。パラメータ width は幅、height は高さです。

void setAlignMode(int mode)

◆機能◆ テキストの揃え方式を設定します。図 14-3～図 14-5 を参照してください。パラメータ mode は揃え方式です。値は PDFAligned が定義された 3 つの揃え方式を使えます。

PDFAligned.V_UPPER_ALIGNED	上揃え
PDFAligned.V_CENTER_ALIGNED	中央揃え
PDFAligned.V_DOWN_ALIGNED	下揃え

void setXY(int x, int y)

◆機能◆ PDFTextAreaV オブジェクトの左下角位置を設定します。パラメータ x、y は左下角の座標値です。

PDFPoint getSize()

◆機能◆ PDFTextAreaV オブジェクトのサイズを取得します。戻り値は
PDFPoint.x　PDFTextAreaV オブジェクトの幅
PDFPoint.y　PDFTextArea オブジェクトの高さです。

void setLineSpacing(int width)

◆機能◆ 列送りを設定します。パラメータ width は列送り値です。

void setLeadingSpaceChar(int n)

◆機能◆ 列インデントを設定します。設定は段落の 2 列目から適応されます。図 14-6 を参照して下さい。パラメータ n は 1 バイトを単位としてインデント値を設定されます。

void setColor(Color c)

◆機能◆ フォント色を設定します。パラメータ c は色値、java.lang.Color オブジェクトです。

void setBorder(Color c)

◆機能◆ PDFTextAreaV オブジェクトの枠を画きます。パラメータcは枠の色値、java.lang.Color オブジェクトです。デフォルトは枠を画きません。

void setBorder(Color c, boolean dashborder)

◆機能◆ PDFTextAreaV オブジェクトの枠を画きます。パラメータ *C*は枠の色値、java.lang.Color オブジェクトです。パラメータ *dashborde**r*の値は“true”であれば、枠を点線になります。デフォルトは枠を画きません。

16.4 サンプル

１．サンプル１

```
PDFTextextareaV textareav = new PDFTextextareaV();
PDFText text = new PDFText();
text.setBasefont("HG 正楷書体-PRO", Font.ITALIC);
text.setCharSize(13);
text.setText("PDF は、ファイルサイズが小さく、");
textareav.append(text);
textareav.append("どのプラットフォームでも使え、");
. . . . . .
textareav.setSize(150);
textareav.setBorder(Color.green);
textareav.setXY(50, 340);
page.append(textareav);
```

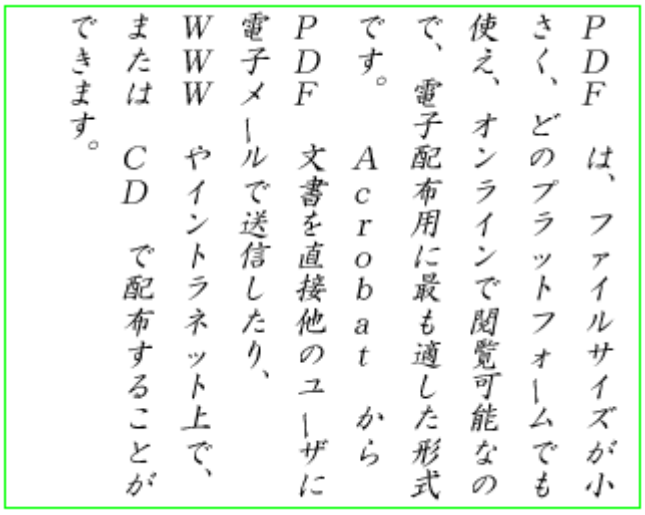


図 16-2 PDFTextV サンプル1

２．サンプル２

サンプル１に次のコードを追加されると、テキストは下揃えになります。

textareav.setAlignMode(PDFAligned.V_DOWN_ALIGNED);

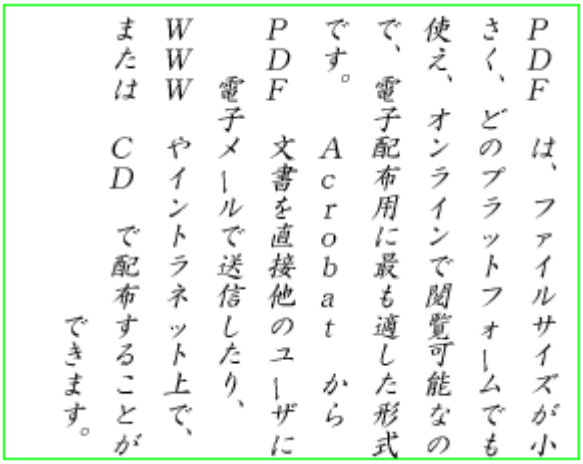


図 16-3 PDFTextV サンプル2

３．サンプル３

サンプル１に次のコードを追加されると、段落の2列目から上インデントの値は3になります。

textareav.setLeadingSpaceChar(3);

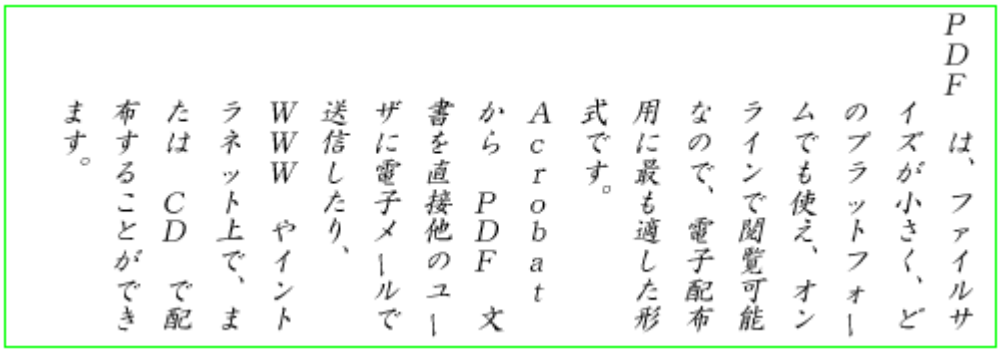


図 16-4 PDFTextV サンプル3

第 17 章 PDFTable

17.1 概要

PDFTable クラスは xinca.tools パッケージ中のテーブルを作成するツールです。テーブルを作成するには、シンプルなインタフェースを提供されています。ユーザーは PDFTable クラスを使用して、少ないコーディングによって、結構複雑な帳票を作成できます。図 17-1 にはサブテーブルとイメージを含まれる PDFTable クラスの応用例です。図 17-2 にはこのテーブルを作成するソースです。ご参考して下さい。

<table>
<tr><td>main table</td><td>insert a nested table in cell(6,1)</td><td>insert a image in cell(3,2)</td><td></td></tr>
<tr><td>PDF 帳票 ツール</td><td>abcd0123456789ABCD</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>cell31</td><td>  </td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td><table><tr><td></td><td>This a nested table</td><td></td></tr><tr><td>該当者一覧</td><td>sub1234567</td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td>アイ江魚</td></tr></table></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>

図17-1 テーブルサンプル

```
import xinca.*;
import xinca.tools.*;

public class first
{
    first(String s)
    {
        PDFFile     pdffile = new PDFFile(s);
        PDFPages pgs = new PDFPages(pdffile);

        int[]   colsWidth = {100, 50, 100, 60};
        // create a PDFTable instance
        PDFTable table = new PDFTable(10, 20, colsWidth);
        // set table style, default PDFTable.TABLE_STYLE_NONE
        table.setStyle(PDFTable.TABLE_STYLE_GRID7);
        // set cell's alignment style
        table.setColsAlign(PDFAligned.MIDDLE_ALIGNED);

        // set text cell
        table.setCellContent(0, 0, "main table");
        table.setCellContent(0, 1, "insert a nested table in cell(6,1)");
        table.setCellContent(0, 2, "insert a image in cell(3,2)");
        table.setCellContent(1, 0, "PDF 帳票 ツール");
        table.setCellContent(1, 1, "abcd0123456789ABCD");
        table.setCellContent(3, 1, "cell31");

        PDFImages img = new PDFImages("../images/zebralogo.jpg");

        // set image cell
        table.setCellContent(3, 2, img);

        PDFTable sub = subTable();
        // set table cell
        table.setCellContent(6, 1, sub);

        // set table position (the left-bottom corner)
        table.setXY(50, 50);

        PDFPage page = new PDFPage(pgs);
        // append this table to page
        page.append(table);

        pgs.append(page);
        pdffile.initialFile(pgs);
        pdffile.writeFile();
    }

    PDFTable subTable()
    {
        int[]   colsWidth = {50, 50, 50};
        PDFTable table = new PDFTable(4, 15, colsWidth);
        table.setStyle(PDFTable.TABLE_STYLE_GRID4);

        table.setCellContent(0, 1, "This a nested table");
        table.setCellContent(1, 0, "該當者一覧");
        table.setCellContent(1, 1, "sub1234567");
        table.setCellContent(3, 2, "アイ江魚");

        return table;
    }

    public static void main(String argv[ ]) {
        new first("sample.pdf");
    }
}
```

図17-2 ソース

17.2 主な機能

●28種類以上のテーブルフォーマットをサポートされます。(付録 C をご参考してください) よく組みたてられれば、色々な帳票を作れます。

●ユーザーはテーブルのスタイルをカスタマイズする事がでいます。図 17-8 を参照して下さい。

●コラム内容の揃え方式は水平と垂直の左、中央、右全部9種類です。

●セルに入れるコンテンツはテキスト、PDFText オブジェクト、PDFImages オブジェクトです。更に、ＰＤＦTable オブジェクトでもセルに入れます。

●セルのコンテンツはテキストであれば、セル幅よりテキストを自動改行されます。このセルが存在する行の高さも自動的に調整されます。

●セルのコンテンツは PDFImages と PDFTable オブジェクトであれば、コンテンツの高さと幅がセルより大きい場合、セルの高さと幅を自動的に調整してコンテンツに合わせます。

●テーブルの一行目 (常にタイトル) の水平と垂直の揃え方は両方とも中央揃えが採用されます。

●セル隣の水平あるいは垂直方向のセルと結合されます。それと同様に水平、垂直方向のセルと結合されます。図 17-5 を参照して下さい。

PDFTable クラスにより不規則的なテーブルレイアウトを作成できますが、図 17-6、図 17-7 の様な不規則テーブルは使用しなことをお勧めします。その様な不規則なテーブルの場合、セルのコンテンツは既定の処理ルーチンで処理できない恐れが有ります。特にコンテンツに画像など非テキストオブジェクトが含まれる場合に発生し易くなります。万一、この様な状況が発生すると、ユーザーが望むテーブルとは異なったテーブルになります。

17.3 PDFTableの座標システム

PDFTableには2種類の座標システムがあります。1つは“セル座標”と言いテーブルのセルに基づいたの相対座標です。テーブルの行と列は各々XとY座標に対応します。テーブルのコンテンツに対して操作する時は、この座標システムを使用します。テーブルの左上角のセルは座標の原始点です。X と Y 座標は各々右側と下側に増加します。例えば、1行目の1列目のセルは(0,0)、1行目2列目のセルは(0,1)、2行目1列目のセルは(1,0)です。このようにテーブルの各セルの座標を計算します。

もう1つは“クロス座標”と言います。テーブルの横罫と縦罫の交差点の位置を表示します。テーブルの左上角の交差点は座標の原始点となります。X と Y 座標は各々右側と下側に増加します。ユーザーはテーブルの形状に対して特別な処理が必要であれば、PDFTableのgetXYメソッド（クロス座標を使います）によって交差点はPDFページ上の絶対座標が取得できます。後は必要に応じてテーブルを作ります。

図 17-3 は PDFTable のこの 2 種類座標システムの見取り図です。

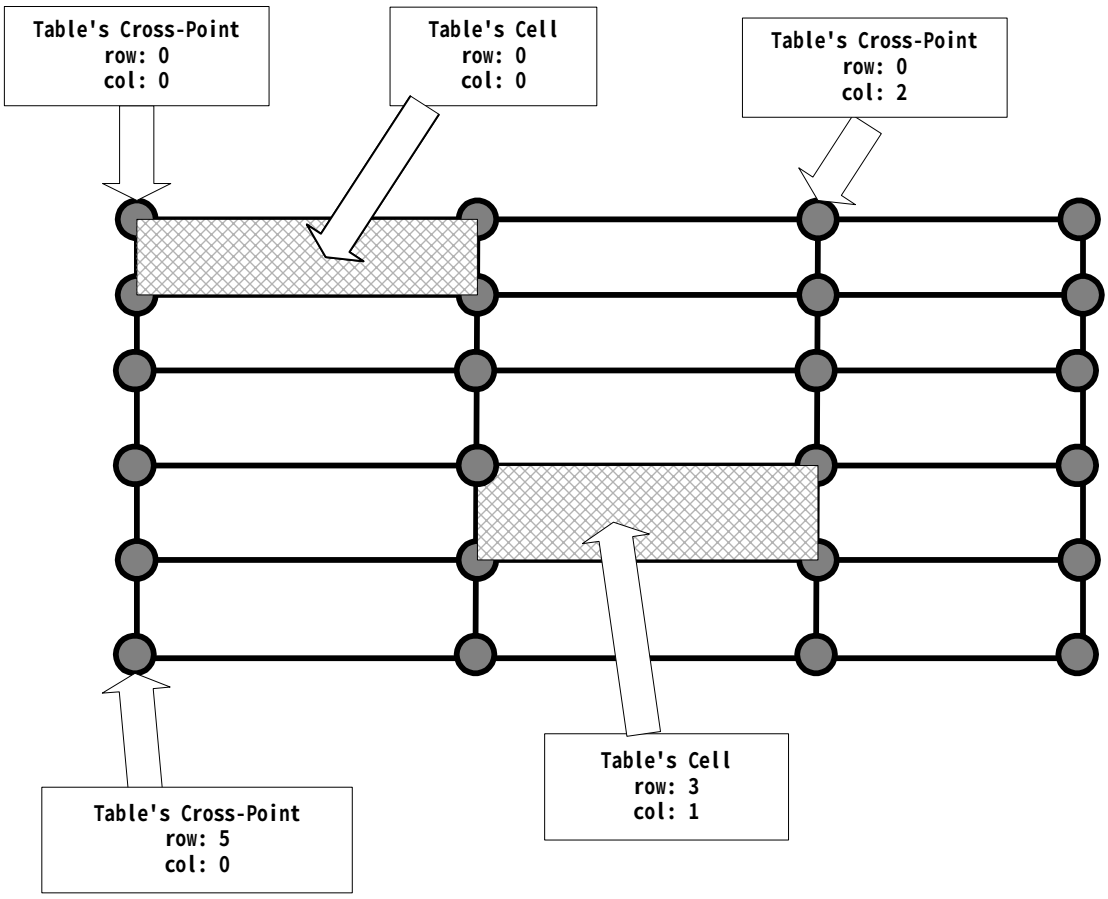


図17-3 PDFTableの座標システム見取り図

17.4 PDFTable のコンストラクタ

PDFTable(int x0, int y0, int rows, int rowHeight, int[] colsWidth)

◆機能◆ PDFTable オブジェクトのインスタンスを宣言します。5つのパラメータが有ります。各パラメータの機能は次の様になります。

x0 テーブルの左下角の X 座標の位置

y0 テーブルの左下角の Y 座標の位置

rows テーブルの行数

rowHeight 行の高さ

colsWidth テーブルのコラム数とコラムの幅。

PDFTable(int rows, int rowHeight, int[] colsWidth)

◆機能◆ PDFTable オブジェクトのインスタンスを宣言します。3つのパラメータが有ります。各パラメータは次の様になります。

rows テーブルの行数

rowHeight 行の高さ

colsWidth テーブルのコラム数とコラムの幅。

17.5 PDFTable のメソッド

void setXY(int x0, int y0)

◆機能◆ テーブル左下の角の位置を設定します。パラメータは2つ有ります。x0 はテーブル左下の角の X 座標の位置、y0 はテーブル左下の角の Y 座標の位置を示します。

void setColsAlign(int colsAlign)

◆機能◆ テーブル全体のコラムに対する揃え方を colsAlign によって指定します。パラメータ colsAlign は9つの値を指定できます。各々は PDFAligned で定義される以下の方式です。

PDFAligned.LEFT_UPPER_ALIGNED	水平は左揃え、垂直は上揃え
PDFAligned.LEFT_CENTER_ALIGNED	水平は左揃え、垂直は中揃え
PDFAligned.LEFT_DOWN_ALIGNED	水平は左揃え、垂直は下揃え
PDFAligned.CENTER_UPPER_ALIGNED	水平は中揃え、垂直は上揃え
PDFAligned.CENTER_CENTER_ALIGNED	水平は中揃え、垂直は中揃え
PDFAligned.CENTER_DOWN_ALIGNED	水平は中揃え、垂直は下揃え
PDFAligned.RIGHT_UPPER_ALIGNED	水平は右揃え、垂直は上揃え
PDFAligned.RIGHT_CENTER_ALIGNED	水平は右揃え、垂直は中揃え
PDFAligned.RIGHT_DOWN_ALIGNED	水平は右揃え、垂直は下揃え


デフォルトの揃え方は PDFAligned.LEFT_UPPER_ALIGNED です。

void setColsAlign(int col, int colsAlign)

◆機能◆ テーブル col 列目のコラムの揃え方を colsAlign に指定します。パラメータ colsAlign の値は上と同じです。

void setStyle(int style)

◆機能◆ テーブルスタイルを設定します。PDFTable には28種類のスタイルが定義できます。パラメータ style の値は下記リストから指定できます。付録 C のテーブルフォーマット定義を参照してください。

PDFTable.TABLE_STYLE_NONE
PDFTable.TABLE_STYLE_CUSTOM

PDFTable.TABLE_STYLE_CLASSIC
PDFTable.TABLE_STYLE_ELEGANT

PDFTable.TABLE_STYLE_LIST1
PDFTable.TABLE_STYLE_LIST2
PDFTable.TABLE_STYLE_LIST3
PDFTable.TABLE_STYLE_LIST4

PDFTable.TABLE_STYLE_GRID1
PDFTable.TABLE_STYLE_GRID2
PDFTable.TABLE_STYLE_GRID3
PDFTable.TABLE_STYLE_GRID4
PDFTable.TABLE_STYLE_GRID5
PDFTable.TABLE_STYLE_GRID6
PDFTable.TABLE_STYLE_GRID7
PDFTable.TABLE_STYLE_GRID8
PDFTable.TABLE_STYLE_GRID9
PDFTable.TABLE_STYLE_GRID10
PDFTable.TABLE_STYLE_GRID11
PDFTable.TABLE_STYLE_GRID12

PDFTable.TABLE_STYLE_SIMPLE1
PDFTable.TABLE_STYLE_SIMPLE2
PDFTable.TABLE_STYLE_SIMPLE3
PDFTable.TABLE_STYLE_SIMPLE4
PDFTable.TABLE_STYLE_SIMPLE5
PDFTable.TABLE_STYLE_SIMPLE6
PDFTable.TABLE_STYLE_SIMPLE7
PDFTable.TABLE_STYLE_SIMPLE8


デフォルトは PDFTable.TABLE_STYLE_NONE です。

void setStyle(int style, int substyle)

◆機能◆ テーブルのフォーマットのカスタマイズをします。パラメータ style は PDFTable.TABLE_STYLE_CUSTOM を固定し、substyle で、次の幾つかのフォーマットを組み合わせて使用します。

パラメータ style：PDFTable.TABLE_STYLE_CUSTOM

substyle：PDFTable は定義されたカスタマイズテーブルフォーマッタを組み合わせたものです。(複数のパラメータを選択をする時はパラメータ間に JAVA で用いられる区切り記号（“|”と“＋”）を使用します)

テーブルの枠	PDFTable.DRAW_THIN_BOX PDFTable.DRAW_THICK_BOX PDFTable.DRAW_DASH_THIN_BOX PDFTable.DRAW_DASH_THICK_BOX
テーブル枠の水平線	PDFTable.DRAW_THIN_HORIZONTAL_BORDER PDFTable.DRAW_THICK_HORIZONTAL_BORDER PDFTable.DRAW_DASH_THIN_HORIZONTAL_BORDER PDFTable.DRAW_DASH_THICK_HORIZONTAL_BORDER
テーブル枠の垂直線	PDFTable.DRAW_THIN_BOX PDFTable.DRAW_THIN_VERTICAL_BORDER PDFTable.DRAW_THICK_VERTICAL_BORDER PDFTable.DRAW_DASH_THIN_VERTICAL_BORDER PDFTable.DRAW_DASH_THICK_VERTICAL_BORDER
テーブル内の水平線	PDFTable.DRAW_HORIZONTAL_LINE PDFTable.DRAW_DASH_HORIZONTAL_LINE
テーブル内の垂直線	PDFTable.DRAW_VERTICAL_LINE PDFTable.DRAW_DASH_VERTICAL_LINE

void setDashLine()

◆機能◆ テーブルの水平線を点線に設定します。

void setDefaultFontSize(int size)

◆機能◆ テーブル中の文字のデフォルトフォントサイズを設定します。パラメータ size：フォントのサイズです。

boolean setCellContent(int row, int col, String s)

◆機能◆ 指定されたテーブルのセルに文字を追加します。3つのパラメータが有ります。row はテーブルの行番号（基底値 0）、col はテーブルのコラム番号（基底値 0）、s は文字列です。

boolean setCellContent(int row, int col, Object obj)

◆機能◆ 指定されたテーブルのセルにオブジェクトを追加します。3つのパラメータが有ります。row はテーブルの行番号（基底値 0）、col はテーブルのコラム番号（基底値 0）、obj は追加されるオブジェクトです。obj は PDFText、PDFImages、PDFTable、PDFTextArea と PDFCheckbox 等のオブジェクトが指定できます。

int appendRow()

◆機能◆ テーブルの最終行に行を一行追加します。戻り値は新しく追加された行番号（最大行数）となります。

PDFPoint getSize()

◆機能◆ テーブルのサイズを取得します。戻り値は PDFPoint オブジェクトです。

戻り値：PDFPoint.x：テーブルの幅、PDFPoint.y：テーブルの高さ

PDFPage.append(PDFTable table)

◆機能◆ table オブジェクトを PDFPage オブジェクトに追加します。

setTitleRowToMiddleAlign ()

◆機能◆ テーブルのタイトル行(一行目)を中央揃えに設定します。

setLineBaseWidth(int w)

◆機能◆ テーブルのベース線太さを設定します。テーブルに細い線と太い線を両方とも存在すると、細い線の太さはベース線太さになり、太い線の太さはベース線の太さプラス1です。パラメータ w はベース線の太さです。

setLineBaseWidth(double w)

◆機能◆ 同上、パラメータのデータタイプは倍精度になります。

setLineColor(Color c)

◆機能◆ テーブルの内側の線の色を設定します。パラメータ c は線色です。

PDFPoint getXY(int row, int col)

◆機能◆ 指定されたテーブルの交差点の座標を取得します。2つのパラメータが有ります。1番目の row はクロス座標のテーブル行番号（基底値 0）です。2番目の col はクロス座標のテーブルコラム番号（基底値 0）です。

戻り値は PDFPoint オブジェクトです。PDFPoint.x を使用して交差点の X 座標を取得し、PDFPoint.y を使用して交差点の Y 座標を取得します。

Object getCellContent(int row, int col)

◆機能◆ 指定されたテーブルセルの内容を取得します。2つのパラメータが有ります。1番目の row はセルの行番号（基底値 0）です。2番目の col はセルのコラム番号（基底値 0）です。

戻り値はオブジェクトタイプのセル内容です。

int getRowNumber()

◆機能◆ テーブルの総行数を取得します。

int getColNumber()

◆機能◆ テーブルの総数を取得します。

boolean copyRowTo(PDFTable dest, int srcRow, int dstRow)

◆機能◆ 指定したテーブルの行の内容を目的のテーブルの指定された行にコピーします。3つのパラメータが有ります。1番目のパラメータ dest は目標のテーブルオブジェクトのインスタンスです。2番目 srcRow はコピーされる行番号です。3番目の dstRow はコピー先の行番号です。

戻り値は“true”と“false”です。コピー操作が成功すれば、“true”を戻します。失敗した場合には“false”を戻します。

PDFTable cloneTableStructure()

◆機能◆ カレントテーブルと同じ構造を持った新規テーブルを生成します。但し、カレントテーブルのセルの内容はコピーされません。戻り値は新しく生成される PDFTable オブジェクトとなります。

int removeRow(int row)

◆機能◆ 指定された行を削除します。パラメータ row は削除される行番号です。戻り値は削除操作が完了した総行数となります。

int getColWidth(int col)

◆機能◆ 指定されたコラム幅を取得します。パラメータ col はコラム番号です。戻り値は指定されたコラム幅です。

int getRowHeight(int row)

◆機能◆ 指定された行の高さを取得します。パラメータ row は行番号です。戻り値は指定された行の高さです。

boolean spanRow(int row, int startCol, int endCol)

◆機能◆ 指定された行と隣接するセルを結合します。結合元の開始セル（row,startCol）の内容が結合後のセルの内容になります。3つのパラメータが有ります。1番目のパラメータ row は行番号です。2番目のパラメータ startCol は開始セルのコラム番号です。3番目のパラメータ endCol は終了セルのコラム番号です。

結合操作が成功であれば “true”を返し、失敗であれば“false”を返します。

boolean spanCol(int col, int startRow, int endRow)

◆機能◆ 指定された列と隣接セルを結合します。結合元の開始セル(col,startRow)の内容が結合後のセルの内容となります。3つのパラメータが有ります。1番目のパラメータ col は行番号です。2番目のパラメータ startRow は開始セルの行番号です。3番目のパラメータ endRow は終了セルの行番号です。

結合操作が成功であれば “true”を返し、失敗であれば“false”を返します。

boolean span(int startRow, int startCol, int endRow, int endCol)

◆機能◆ 指定された範囲（行列）内のセルを結合します。結合元の開始セル(startRow,startCol)の内容が結合後のセルの内容となります。4つのパラメータが有ります。1番目のパラメータ startRow は開始セルの行番号です。2番目のパラメータ startCol は開始コラム番号です。3番目のパラメータ endRow は終了行の番号です。4番目のパラメータ endCol は終了コラムの番号です。

結合操作が成功であれば “true”を返し、失敗であれば“false”を返します。

17.6 コーディングのステップ

1．xinca.tools をインポートします。

```
import xinca.tools.*;
```

2．PDFTableオブジェクトをコンストラクタします。

```
int[]    colsWidth = {100, 50, 100, 60};
PDFTable table = new PDFTable(10, 20, colsWidth);
```

3．テーブルのフォーマットを設定します。

```
table.setStyle(PDFTable.TABLE_STYLE_GRID7);
```

4．テーブルの揃え方を設定します。

```
table.setColsAlign(PDFAligned.MIDDLE_ALIGNED);
```

5．テーブルのセルの内容を追加します。

```
table.setCellContent(0, 0, "cell00 Content");
```

6．テーブル左下の角の位置をPDFページ上からの位置で設定します。

```
table.setXY(50, 100);
```

７．テーブルをPDFPageに追加します。

```
page.append(table);
```

17.7 サンプル

１．サンプル１

```
int[]    colsWidth = {100, 50, 100, 60};
PDFTable table = new PDFTable(10, 20, colsWidth);
table.setStyle(PDFTable.TABLE_STYLE_GRID8);
table.setColsAlign(PDFAligned.CENTER_CENTER_ALIGNED);

table.setCellContent(0, 0, "main table");
table.setCellContent(0, 1, "insert a nested table in cell(6,1)");
table.setCellContent(0, 2, "insert a image in cell(3,2)");
table.setCellContent(1, 0, "PDF 帳票 ツール");
table.setCellContent(1, 1, "abcd0123456789ABCD");
table.setCellContent(3, 1, "cell31");
table.setCellContent(4, 1, "cell41");
table.setCellContent(6, 1, "cell61");
table.setCellContent(7, 0, "cell70\ncell70");

PDFCheckbox checkbox;
checkbox = new PDFCheckbox("Xinca CheckBox Sample1");
table.setCellContent(3, 2, checkbox);

checkbox = new PDFCheckbox("Xinca CheckBox Sample2", true);
checkbox.setBoxSize(12);
table.setCellContent(3, 3, checkbox);

table.setXY(50, 100);
page.append(table);
```

main table	insert a nested table in cell(6,1)	insert a image in cell(3,2)	
PDF 帳票 ツール	abcd0123456789ABCD		
	cell31	☐ Xinca CheckBox Sample1	☑ Xinca CheckBox Sample2
	cell41		
	cell61		
cell70 cell70			

図17-4 サンプル1

２．サンプル２

サンプル1に次のコードを追加し、セルを結合します。

①table.spanRow(0, 1, 2);

②table.spanCol(3, 3, 4);

③table.span(6, 1, 8, 2);

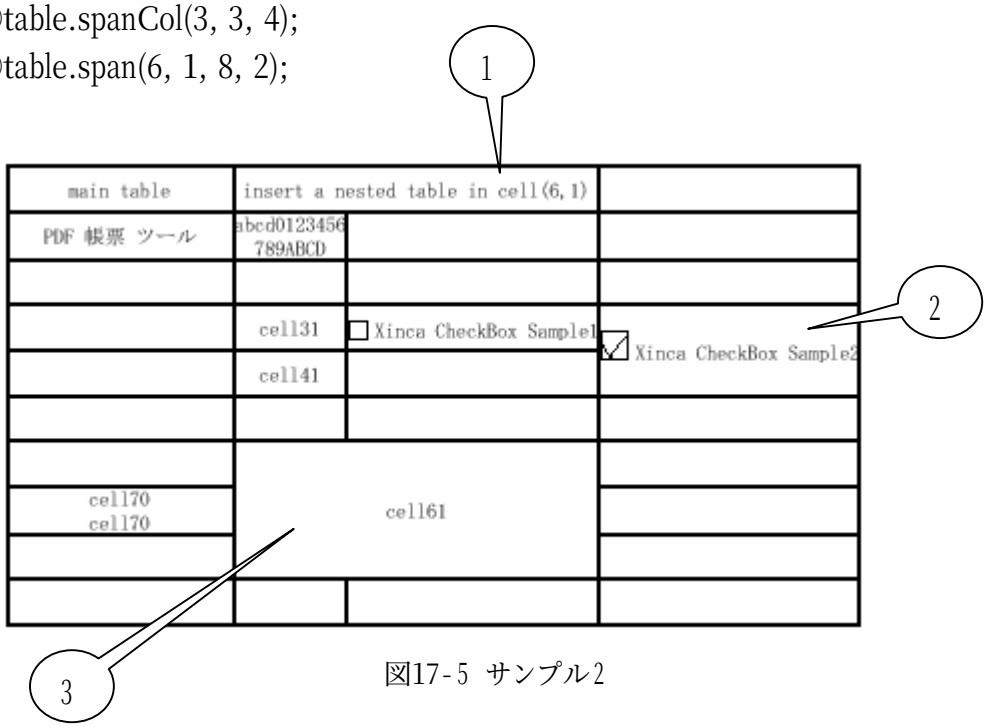


図17-5 サンプル2

３．サンプル３

サンプル1に次のコードを追加し、セルを結合します。

①table.spanRow(0, 1, 2);

②table.spanRow(4, 1, 3);

③table.spanRow(7, 1, 3);

④table.spanCol(2, 6, 8);

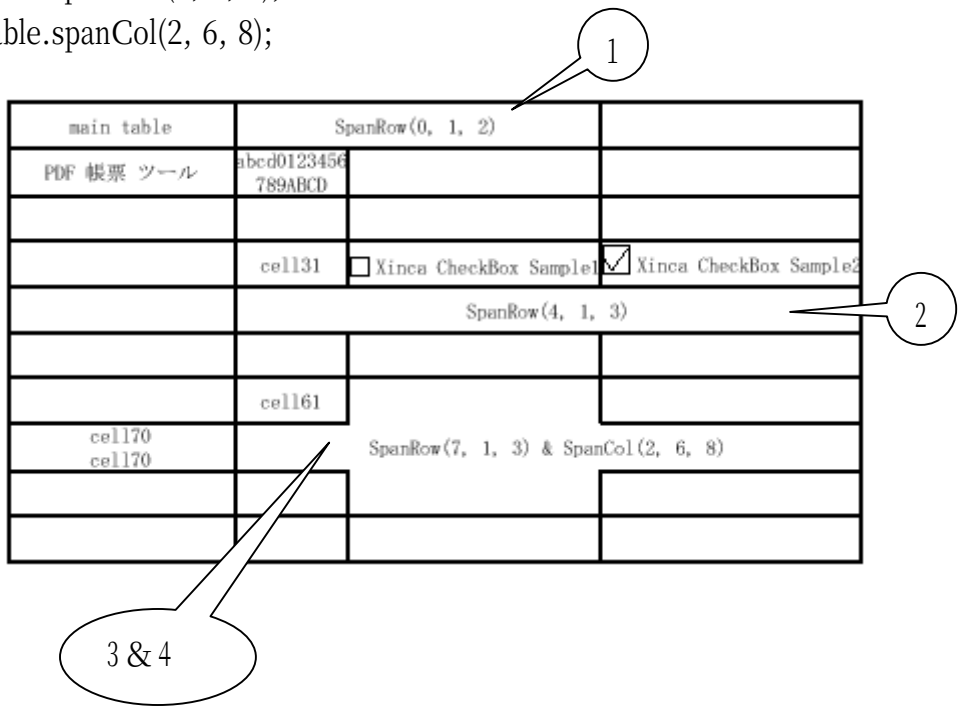


図17-6 サンプル3

４．サンプル４

サンプル3の④を次に直しますと、図17－7のようになります。

table.spanCol(3, 5, 7);

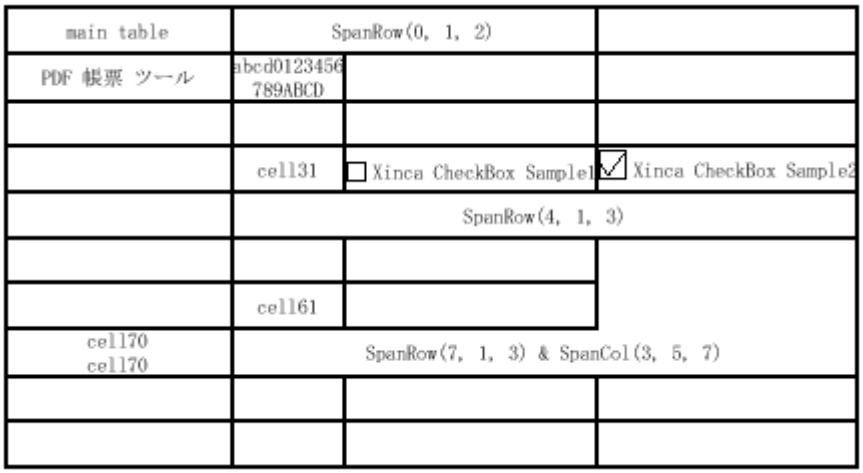


図17-7 サンプル4

５．サンプル５

これはカスタマイズフォーマットのサンプルです。

```
. . . . . .
int style = PDFTable.DRAW_THICK_HORIZONTAL_BORDER;
style |= PDFTable.DRAW_DASH_THIN_VERTICAL_BORDER;
style |= PDFTable.DRAW_HORIZONTAL_LINE;
style |= PDFTable.DRAW_DASH_VERTICAL_LINE;
table.setStyle(PDFTable.TABLE_STYLE_CUSTOM, style);
. . . . . .
```

複数のフォーマットを指定するには、2つ目の指定に対して“｜”あるいは“＋”を使用します。これは JAVA 言語の仕様に準拠しています。

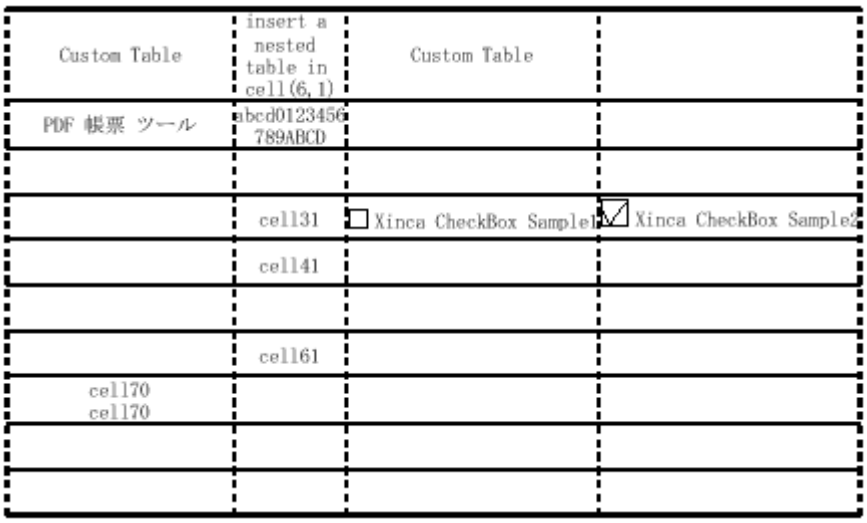
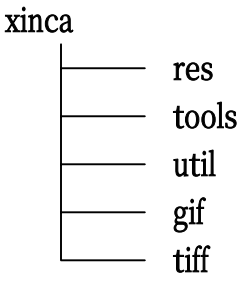


図17-8　サンプル5

第３部 XINCA のクラスインデックス

第3部は第18章～第20章から構成されております。XINCA SERVER を利用するのみ必要な全ての API クラスとメソッドのインデックスが作成れております。本部では XINCA SERVER のクラスとメソッドに対するインデックスですので、クラスに関する使用方法については第1部と第 2 部をご覧下さい。

第 18 章 XINCA ディレクトリ構造



次の様にパッケージごとに設置されるクラスの概要を説明します。

xinca：　XINCA SERVERのルートディレクトリにXINCA SERVERのコアクラスが設置されています
res：　フォント関連のクラス、XINCA SERVERのRESOURCEメッセージが設置されています

tools：　XINCA SERVERの拡張クラス、例えば、PDFTable，PDFCheckbox等が設置されています
util：　XINCA SERVER内部で使用される補助クラスが設置されています

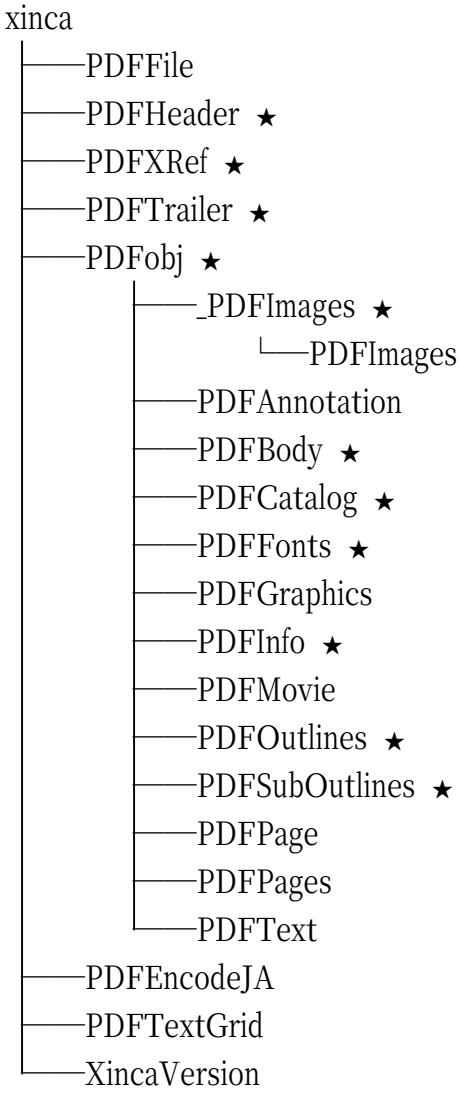
gif：　GIF格式の画像ファイルを解析するためのクラスが設置されています

tiff：　TIFF格式の画像ファイルを解析するためのクラスが設置されています

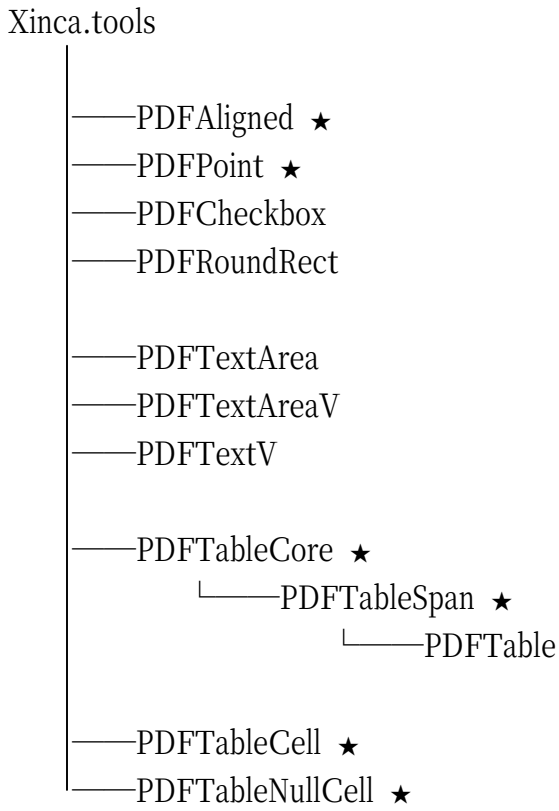
第 19 章 パッケージ及びクラスインデックス

19.1 xinca パッケージ

下記に列挙したクラス中で、本マニュアルでは説明していないクラス(★マークを付けた)が有ります。実はこれらのクラスは XINCA SERVER 内部で使用されます。但し、PDF ファイルでこれらと対応するオブジェクトが有ります。XINCA SERVER パッケージに対して十分にご理解いただく為に、ここに記載してあります。ユーザーはクラス名によって、そのクラスの機能が推測できると思います。

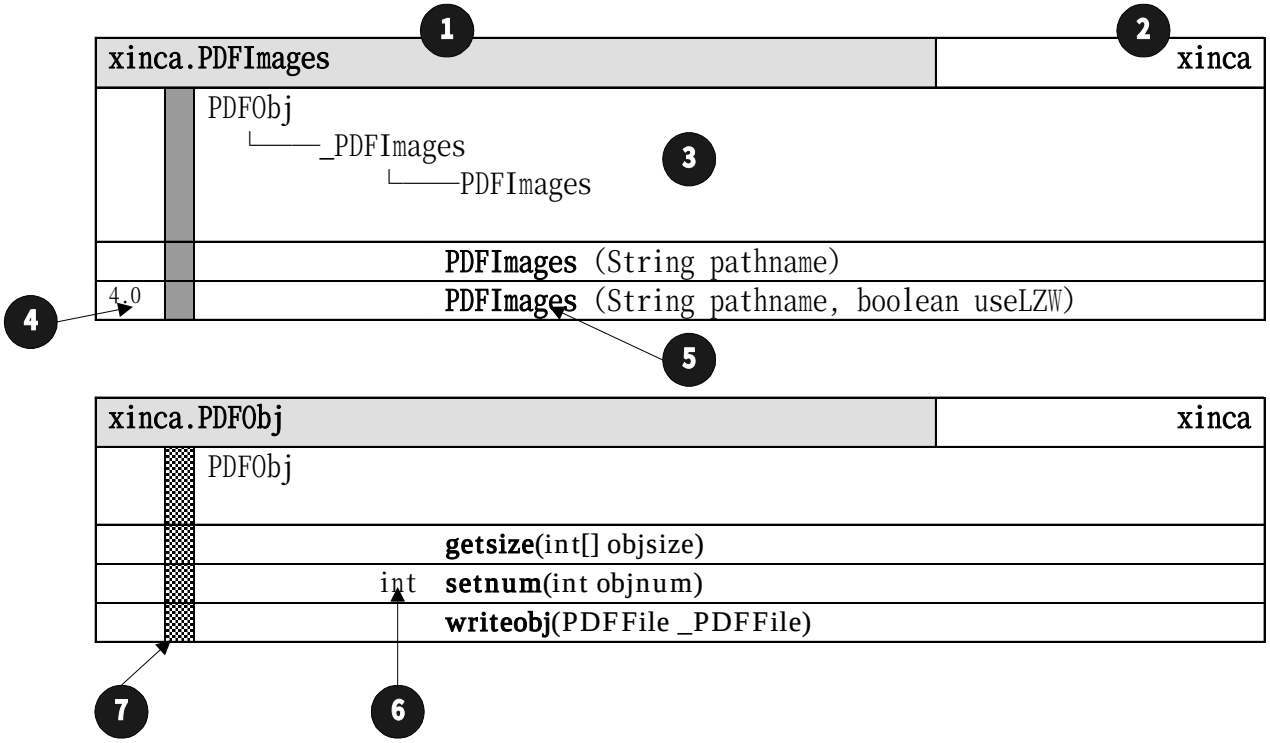


19.2 Xinca.tools パッケージ



第 20 章 クラス及びメソッドインデックス

20.1 図解



1: クラス及びインタフェース名
2: パッケージ名
3: クラスツリー
4: Xinca のバージョン、空欄であればバージョンは 3.x です。
5: メソッド及び属性名
6: 戻り値のタイプ
7: インタフェース

20.2 PDFAligned

xinca.tools.PDFAligned	xinca.tools
PDFAligned	
public static final int	H_LEFT_ALIGNED
public static final int	H_CENTER_ALIGNED
public static final int	H_RIGHT_ALIGNED
public static final int	V_UPPER_ALIGNED
public static final int	V_CENTER_ALIGNED
public static final int	V_DOWN_ALIGNED
public static final int	LEFT_UPPER_ALIGNED
public static final int	LEFT_CENTER_ALIGNED
public static final int	LEFT_DOWN_ALIGNED
public static final int	CENTER_UPPER_ALIGNED
public static final int	CENTER_CENTER_ALIGNED
public static final int	CENTER_DOWN_ALIGNED
public static final int	RIGHT_UPPER_ALIGNED
public static final int	RIGHT_CENTER_ALIGNED
public static final int	RIGHT_DOWN_ALIGNED
public static final int	LEFT_ALIGNED
public static final int	CENTER_ALIGNED
public static final int	MIDDLE_ALIGNED
public static final int	RIGHT_ALIGNED

20.3 PDFAnnotation

xinca.PDFAnnotation	xinca
PDFObj └———PDFAnnotation	
PDFAnnotation ( )	
setCotent (String content)	
setOpen ( )	
setUnicode ( )	
setXY (int x1, int y1, int x2, int y2)	

20.4 PDFCatalog

xinca.PDFCatalog			xinca
		PDFObj └———PDFCatalog	
4.0		public final static int	PAGEMODE_USENONE
4.0		public final static int	PAGEMODE_USEOUTLINES
4.0		public final static int	PAGEMODE_USETHUMBS
4.0		public final static int	PAGEMODE_FULLSCREEN

20.5 PDFCheckbox

xinca.tools.PDFCheckbox			xinca.tools
4.0		PDFCheckbox	
		public final static int	**CHECKED**
		public final static int	**UNCHECK**
			PDFCheckbox(String title)
			PDFCheckbox(PDFText title)
			PDFCheckbox(String title, boolean checkflag)
			PDFCheckbox(PDFText title, boolean checkflag)
			PDFCheckbox(int x0, int y0, PDFText title)
			PDFCheckbox(double x0, double y0, PDFText title)
			PDFCheckbox(int x0, int y0, PDFText title, boolean checkflag)
			PDFCheckbox(double x0, double y0, PDFText title, boolean checkflag)
			PDFCheckbox(int x0, int y0, String title)
			PDFCheckbox(double x0, double y0, String title)
			PDFCheckbox(int x0, int y0, String title, boolean checkflag)
			PDFCheckbox(double x0, double y0, String title, boolean checkflag)
			PDFCheckbox(int x0, int y0)
			PDFCheckbox(double x0, double y0)
		PDFPoint	**getSize**()
			setBoxColor(Color c)
			setBoxLeftSpace(int w)
			setBoxLeftSpace(double w)
			setBoxRightSpace(int w)
			setBoxRightSpace(double w)
			setBoxSize(int size)
			setChecked()
			setTitle(String title)
			setUncheck()
			setXY(int x0, int y0)
			setXY(double x0, double y0)

20.6 PDFEncodeJA

xinca.PDFEncodeJA			xinca
4.0		PDFEncodeJA	
		public static String **getStringSJIS**(byte[] bytes))	
		public static String **getStringEUCJIS**(byte[] bytes)	
		public static byte[] **getStringBytesSJIS**(String s)	

20.7 PDFFile

xinca.PDFFile			xinca
		PDFFile	
		PDFFile (String　filename)	
4.0		**PDFFile** (PrintWriter pw)	
4.0		**PDFFile** (Writer w)	
4.0		**PDFFile** ()	
		initialFile (PDFPages pages)	
4.0		boolean　**isOK** ()	
		setPageSize(String size)	
		setPageSize(String size, boolean horizontal)	
		setPageSize(int wide, int height)	
4.0		**setPageMode** (int mode)	
		wiriteFile ()	

20.8 PDFGraphics (—)

xinca.PDFGraphics			xinca
		PDFObj └———PDFGraphics	
		PDFGraphics ( )	
4.0		**closeDashLine** ()	
4.0		**drawArc** (int x, int y, int radius, int startAngle, int arcAngle)	
		drawBezier (double x0, double y0, double x1, double y1, double x2, double y2, double x3, double y3)	
		drawBezier (double x0, double y0, double x1, double y1, double x2,double y2)	
		drawCircle (int x0, int y0,int radius)	
		drawDashLine (int x0, int y0, int x1, int y1)	
4.0		**drawDashLine** (doublex0, doubley0, doublex1, doubley1)	
		drawEllips (int x0, int y0, double , doubleb)	
		drawPolygon (int[] x,int[] y, int number)	
		drawRectangle(int x, int y, int width, int Height)	
4.0		**drawRectangle**(double x, double y, double width, double Height)	
		drawSegment(int[] x, int[] y, int number)	
		drawScallop(int x0, int y0, int radius, int startAngle, int arcAngle)	
		fillBezier(double x0, double y0, double x1, double y1, double x2, double y2, double x3, double y3)	
		fillBezier(double x0, double y0, double x1, double y1, double x2, double y2)	
		fillCircle (int x0, int y0, int radius)	
		fillEllips (int x0, int y0, double a, double b)	
		fillPolygon (int[] x, int[] y, int number)	
		fillRectangle(int x, int y, int width, int Height)	
4.0		**fillRectangle**(double x, double y, double width, double Height)	
		fillScallop (int x0, int y0, int radius, int startAngle, int arcAngle)	
		resetLineStyle ( )	
		setArrow (int arrorwMode, int voneMode, int arrowAngle, int arrowHeight)	

20.9 PDFGraphics (二)

xinca.PDFGraphics			xinca
		PDFObj └PDFGraphics	
		setCapStyle (int capstyle)	
		setCurrentPoint (int x, int y)	
		setDashLine (int unit)	
		setDashLine (int black, int white)	
		setFillColor (Color color)	
		setFillColor (int red, int green, intbblue)	
		setFillMode (int mode)	
		setLineJoin (int mode)	
		setStrokeColor (Color color)	
		setStrokeColor (int red, int green, int blue)	
		setWidth (int width)	
4.0		**setWidth** (double width)	
		strokeBezier (double x0, double y0, double x1, double y1, double x2, double y2, double x3, double y3)	
		strokeBezier (double x0, double y0, double x1, double y1, double x2, double y2, double x3, double y3)	
		strokeCircle (int x0, int y0,int radius)	
		strokeCloseBezier (double x0, double y0, double x1, double y1, double x2, double y2, double x3, double y3)	
		strokeCloseBezier (double x0, double y0, double x1, double y1, double x2, double y2, int style)	
		strkoeEllips (int x0, int y0, double a, double b)	
		strokeLine (int x0, int y0, int x1, int y1)	
4.0		**strokeLine** (double x0, double y0, double x1, double y1)	
		strokePolygon (int[] x, int[] y, int number)	
		strokeRectangle(int x, int y, int width,int Height)	
4.0		**strokeRectangle**(double x, double y, double width, double Height)	
		strokeScallop(int x0, int y0, int radius, int startAngle, int arcAngle)	

20.10 _PDFImages

xinca._PDFImages			xinca
		PDFObj └———_PDFImages	
		_PDFImages (String pathname)	
4.0		**_PDFImages** (String pathname, boolean useLZW)	
4.0		PDFPoint **getSize** ()	
		linkWith (Object)	
		linkWith (String fileName, int pageNumber)	
		linkWith (String pathName)	
		setOffset (int x, int y)	
		setRotate (int angle)	
		setSize (int width, int height)	
		setXY (int y, int y)	

20.11 PDFImages

xinca.PDFImages			xinca
		PDFObj └─────_PDFImages └─────PDFImages	
		PDFImages (String pathname)	
4.0		**PDFImages** (String pathname, boolean useLZW)	

20.12 PDFMovie

xinca.PDFMovie	xinca
PDFObj └PDFMovie	
PDFMovie (String pathnam)	
setXY (int x, int y)	
setBoundingBox (int width, int height)	

20.13 PDFObj

xinca.PDFObj	xinca
PDFObj	
getsize (int[] objsize)	
int **setnum** (int objnum)	
writeobj (PDFFile _PDFFile)	

20.14 PDFPage

xinca.PDFPage			xinca
		PDFObj └──PDFPage	
		public final static int	**ASCII85**
		public final static int	**ASCIIHEX**
		public final static int	**DCT**
		public final static int	**FLATE**
		public final static int	**LZW**
4.0		public final static int	**NON_COMPRESSION**
		public final static int	**RUNLENGTH**
			PDFPage (PDFPages pdfpages)
			append (Object object)
4.0		PDFPoint	**getPageSize** ()
4.0			**nonCompress** ()
			setCompressMethod (int firstMethod, int second Method)
			setLabelName (String name)
			setPageSize(String size)
			setPageSize(String size, boolean horizontal)
			setPageSize(int wide, int height)

20.15 PDFPages

xinca.PDFPages	xinca
PDFObj └PDFPages	
PDFPages (PDFFile pdffile)	
append (PDFPage page)	
append (PDFPages pages)	
setLabelName (String name)	

20.16 PDFPoint

xinca.tools.PDFPoint			xinca.tools
4.0		PDFPoint	
		public int **x**	
		public int **y**	
		PDFPoint(int _x, int _y)	
		boolean **equals**(PDFPoint src)	

20.17 PDFRoundRect

xinca.tools.PDFRoundRect			xinca.tools
4.0		PDFRoundRect	
		PDFRoundRect ()	
		setDashLine(int　black, int　white)	
		setLineColor(Color　c)	
		setLineWidth(int　lineWidth)	
		setLineWidth(double　lineWidth)	
		setRectHeight(int　rectHeight)	
		setRectWidth(int　rectWidth)	
		setRound(int　round)	
		setXY(int　x0,int　y0)	
		setXYWHR(int　x0, int　y0, int　rectWidth, int　rectHeight, int　round)	

20.18 PDFTable (—)

xinca.tools.PDFTable			xinca.tools
4.0		PDFTableCore └———PDFTableSpan └———PDFTable	
		public static final int **TABLE_STYLE_NONE**	
		public static final int **TABLE_STYLE_CUSTOM**	
		public static final int **TABLE_STYLE_CLASSIC**	
		public static final int **TABLE_STYLE_ELEGANT**	
		public static final int **TABLE_STYLE_LIST1**	
		public static final int **TABLE_STYLE_LIST2**	
		public static final int **TABLE_STYLE_LIST3**	
		public static final int **TABLE_STYLE_LIST4**	
		public static final int **TABLE_STYLE_GRID1**	
		public static final int **TABLE_STYLE_GRID2**	
		public static final int **TABLE_STYLE_GRID3**	
		public static final int **TABLE_STYLE_GRID4**	
		public static final int **TABLE_STYLE_GRID5**	
		public static final int **TABLE_STYLE_GRID6**	
		public static final int **TABLE_STYLE_GRID7**	
		public static final int **TABLE_STYLE_GRID8**	
		public static final int **TABLE_STYLE_GRID9**	
		public static final int **TABLE_STYLE_GRID10**	
		public static final int **TABLE_STYLE_GRID11**	
		public static final int **TABLE_STYLE_GRID12**	
		public static final int **TABLE_STYLE_SIMPLE1**	
		public static final int **TABLE_STYLE_SIMPLE2**	
		public static final int **TABLE_STYLE_SIMPLE3**	
		public static final int **TABLE_STYLE_SIMPLE4**	
		public static final int **TABLE_STYLE_SIMPLE5**	
		public static final int **TABLE_STYLE_SIMPLE6**	
		public static final int **TABLE_STYLE_SIMPLE7**	
		public static final int **TABLE_STYLE_SIMPLE8**	
		public static final int **TABLE_STYLE_DEFAULT**	

20.19 PDFTable (二)

xinca.tools.PDFTable			xinca.tools
4.0		PDFTableCore └PDFTableSpan └PDFTable	
		public static final int	**DRAW_THIN_BOX**
		public static final int	**DRAW_THICK_BOX**
		public static final int	**DRAW_DASH_THIN_BOX**
		public static final int	**DRAW_DASH_THICK_BOX**
		public static final int	**DRAW_THIN_HORIZONTAL_BORDER**
		public static final int	**DRAW_THICK_HORIZONTAL_BORDER**
		public static final int	**DRAW_DASH_THIN_HORIZONTAL_BORDER**
		public static final int	**DRAW_THIN_VERTICAL_BORDER**
		public static final int	**DRAW_THICK_VERTICAL_BORDER**
		public static final int	**DRAW_DASH_THIN_VERTICAL_BORDER**
		public static final int	**DRAW_DASH_THICK_VERTICAL_BORDER**
		public static final int	**DRAW_HORIZONTAL_LINE**
		public static final int	**DRAW_VERTICAL_LINE**
		public static final int	**DRAW_DASH_HORIZONTAL_LINE**
		public static final int	**DRAW_DASH_VERTICAL_LINE**
			PDFTable(int _x0, int _y0, int _rows, int _rowHeight, int[] _colsWidth)
			PDFTable(int _rows, int _rowHeight, int[] _colsWidth)
		PDFTable	**cloneTableStructure**()
			setDashLine()
			setLineBaseWidth(int w)
			setLineBaseWidth(double w)
			setLineColor(Color c)
			setStyle(int _style
			setStyle(int _style, int _types)

20.20 PDFTableCore

xinca.tools.PDFTableCore			xinca.tools
4.0		PDFTableCore	
		PDFTableCore(int _x0, int _y0, int _rows, int _rowHeight, int[] _colsWidth)	
		PDFTableCore(int _rows, int _rowHeight, int[] _colsWidth)	
		int **appendRow**()	
		boolean **copyRowTo**(PDFTable dest, int srcRow, int dstRow)	
		Object **getCellContent**(int row, int col)	
		int **getColNumber**()	
		int **getRowNumber**()	
		int **removeRow**(int index)	
		boolean **setCellContent**(int row, int col, String s)	
		boolean **setCellContent**(int row, int col, Object obj)	
		setColsAlign(int[] _colsAlign)	
		setColsAlign(int _colsAlign)	
		setColsAlign(int col, int _colsAlign)	
		setDefaultFontSize(int _size)	
		setRowHeight(int height)	
		setTitleRowToMiddleAlign()	

20.21 PDFTableNullCell

xinca.tools.PDFTableNullCell		xinca.tools
4.0		PDFTableNullCell

20.22 PDFTableSpan

xinca.tools.PDFTableSpan			xinca.tools
4.0		PDFTableCore └────PDFTableSpan	
		PDFTableSpan(int _x0, int _y0, int _rows, int _rowHeight, int[] _colsWidth	
		PDFTableSpan(int _rows, int _rowHeight, int[] _colsWidth)	
		int **getColWidth**(int col)	
		int **getRowHeight**(int row)	
		PDFPoint **getSize**()	
		PDFPoint **getXY**(int row, int col)	
		boolean **span**(int row1, int col1, int row2, int col2)	
		boolean **spanCol**(int col, int startRow, int endRow)	
		boolean **spanRow**(int row, int startCol, int endCol)	
		setXY(int _x0, int _y0)	

20.23 PDFText (—)

xinca.PDFText			xinca
		PDFObj └———PDFText	
			PDFText ()
			PDFText(String orgencode, String dstencode)
4.0		double	**getStringFloatWidth**()
4.0		double	**getStringFloatWidth**(String s)
4.0		int	**getStringWidth**()
4.0		int	**getStringWidth**(String s)
			linkWith (Object)
			linkWith (String pathname, int pagenum)
			linkWith (String pathname)
			linkWith (String website)
			setBasefont (Font font)
			setBasefont (String fontname, int fontmode
			setCharSize (int size)
4.0			**setCharSize** (double size)
			setCharSpace (int value)
4.0			**setCharSpace** (double value)
			setcirRectangle ( )
			setcirRectangle (Color color)
			setcirRectangle (int width, int height)
			setcirRectangle (Color color, int windth, int height)
			setcirRectangle (Color color, int width, int height)
			setColor (Color color)
			setColor (int red, int green, int blue)
			setHorizontalScale (int value)
			setLineWidth (int width)
4.0			**setLineWidth** (double width)
			setRendering (int renderingMode)
			setRotation (int angle)
			setText (String text)

20.24 PDFText（二）

xinca.PDFText			xinca
		PDFObj └PDFText	
		setUnderLine ( )	
		setUnderLine (Color color)	
		setUnderLine (int width)	
4.0		**setUnderLine** (double width)	
		setUnderLine (Color color, int width)	
4.0		**setUnderLine** (Color color, double width)	
		setWordSpace (int value)	
4.0		**setWordSpace** (double value)	
		setXY (int x, int y)	
4.0		**setXY** (double x, double y)	

20.25 PDFTextArea

xinca.tools.PDFTextArea			xinca.tools
4.0		PDFTextArea	
		PDFTextArea()	
		append(PDFText text)	
		append(String s)	
		PDFPoint **getSize**()	
		setAlignMode(int mode)	
		setBorder(Color c)	
		setBorder(Color c, boolean dashborder)	
		setColor(Color c)	
		setLeadingSpaceChar(int n)	
		setLineSpacing(int height)	
		setLineSpacing(double height)	
		setMargin(int left, int right, int top, int bottom)	
		setSize(int width)	
		setSize(int width, int height)	
		setXY(int x, int y)	
		setXY(double x, double y)	

20.26 PDFTextAreaV

xinca.tools.PDFTextAreaV			xinca.tools
4.0		PDFTextAreaV	
		PDFTextAreaV()	
		append(PDFText text)	
		append(String s)	
		PDFPoint **getSize**()	
		setAlignMode(int mode)	
		setBorder(Color c)	
		setBorder(Color c, boolean dashborder)	
		setColor(Color c)	
		setLeadingSpaceChar(int n)	
		setLineSpacing(int height)	
		setSize(int width)	
		setSize(int width, int height)	
		setXY(int x, int y)	

20.27 PDFTextGrid

xinca.PDFTextGrid	xinca
PDFTextGrid	
public static final int **TopLeftAligned**	
public static final int **TopRightAligned**	
public static final int **TopMiddleAligned**	
public static final int **LeftAligned**	
public static final int **RightAligned**	
public static final int **MiddleAligned**	
public static final int **BottomLeftAligned**	
public static final int **BottomRightAligned**	
public static final int **BottomMiddleAligned**	
PDFTextGrid ( )	
appendText (PDFText text)	
setAlignMode (int mode)	
setLineSpacing (int width)	
setSize (int width, int height)	
setXY (int x, int y)	

20.28 PDFTextV

xinca.tools.PDFTextV			xinca.tools
4.0		PDFTextV	
		PDFTextV(PDFText text)	

20.29 XincaVersion

xinca.XincaVersion			xinca
4.0		XincaVersion	
		public static String **VERSION**	
		public static String **CREATED_DATE**	
		Piblic static void **main** (String[] argv)	
		XincaVersion ()	

第４部 ニューバージョンに追加クラス

第 21 章 xinca.util.Ruler

21.1 概要

単位を定義する、単位を取り替える。

21.2 Ruler の属性

public static int CM

◆機能◆　センチメートル単位。

public static int MM

◆機能◆　ミリメートルル単位

public static int INCH

◆機能◆　インチ単位

21.3 Rurler のメソッド

public static int toPixel(double c, String unit)

◆機能◆　指定するサイズをピクセルに取り替える。

public static int[] toPixel(double[] c, String unit)

◆機能◆　指定するサイズ配列をピクセル配列に取り替える。

public static double toFloatPixel(double c, String unit)

◆機能◆　指定するサイズを小数ピクセルに取り替える。

public　static　int　mmToPixel(double c)

◆機能◆　ミリメートルサイズをピクセルに取り替える。

public　static　int　cmToPixel (double c)

◆機能◆　センチメートルサイズをピクセルに取り替える。

public　static　int　inchToPixel (double c)

◆機能◆　インチサイズをピクセルに取り替える。

public　static　double mmToFloatPixel(double c)

◆機能◆　ミリメートルサイズを小数ピクセルに取り替える。

public　static　double cmToFloatPixel (double c)

◆機能◆　センチメートルサイズを小数ピクセルに取り替える。

public　static　double inchToFloatPixel (double c)

◆機能◆　インチサイズを小数ピクセルに取り替える。

public　static　double inchToCM (double inch)

◆機能◆　インチサイズをセンチメートルに取り替える。

public　static　double inchToMM (double inch)

◆機能◆　インチサイズをセンミリメートルに取り替える。

public　static　double cmToInch (double cm)

◆機能◆　センチメートルサイズをインチに取り替える。

public　static　double mmToInch (double mm)

◆機能◆　ミリメートルサイズをインチに取り替える。

第 22 章 xinca.font. FontName

22.1 概要

xinca のフォント名前を定義する。

22.2 FontName の属性

22.2.1 英語フォント

public　static　String　EN_ARIAL

◆機能◆　Arial フォント。

public　static　String　EN_ARIAL_BLACK

◆機能◆　ArialBlack フォント。

public　static　String　EN_BRUSH_SCRIPT_MT

◆機能◆　BrushScriptMT フォント。

public　static　String　EN_CENTURY_GOTHIC

◆機能◆　CenturyGothic フォント。

public　static　String　EN_COURIER

◆機能◆　Courier フォント。

public　static　String　EN_HELVETICA

◆機能◆　Helvetica フォント。

public　static　String　EN_SYMBOL

◆機能◆　Symbol フォント。

public　static　String　EN_TIMES_ROMAN

◆機能◆　Times-Roman フォント。

public　static　String　EN_ZAPF_DINGBATS

◆機能◆　ZapfDingbats フォント。

22.2.2 日本語フォント

public　static　String　JP_MS_MINTYOU

◆機能◆　MS 明朝フォント。

public　static　String　JP_MS_P_MINTYOU

◆機能◆　MS P 明朝フォント。

public　static　String　JP_MS_GOTHIC

◆機能◆　MS ゴシックフォント。

public　static　String　JP_MS_P_GOTHIC

◆機能◆　MS P ゴシックフォント。

public　static　String　P_MS_UI_GOTHIC

◆機能◆　MS UI ゴシックフォント。

public　static　String　JP_DF_POP

◆機能◆　ＤＦPOP 体フォント。

public　static　String　JP_DFP_POP

◆機能◆　ＤＦPPOP 体フォント。

public　static　String　JP_DF_GOTHIC

◆機能◆　ＤＦ特太ゴシック体フォント。

public　static　String　JP_DFP_GOTHIC

◆機能◆　ＤＦＰ特太ゴシック体フォント。

public　static　String　JP_HG_GOTHIC

◆機能◆　HGｺﾞｼｯｸ E-PRO フォント。

public　static　String　JP_HG_MARU_GOTHIC

◆機能◆　HG 丸ｺﾞｼｯｸ M-PRO フォント。

public　static　String　JP_HG_SEIKAI

◆機能◆　HG 正楷書体-PRO フォント。

22.2.3 韓国語フォント

public　static　String　KOREA_HYGOTHIC_MEDIUM_ACRO

◆機能◆　HYGoThic-Medium-Acro フォント。

22.2.4 中国語フォント

public static String CH_MS_SONG

◆機能◆　MS 宋体1フォント(簡体中文 Simplified Chinese Font)。

public static String CH_MSSONG

◆機能◆　MS 宋体2フォント(簡体中文 Simplified Chinese Font)。

public static String CH_MS_HEI

◆機能◆　黒体フォントフォント(簡体中文 Simplified Chinese Font)。

public static String CH_MS_BIG5

◆機能◆　Big5 フォント(繁体中文 Traditional Chinese Font)。

public static String CH_MING_LIU5

◆機能◆　明流フォント(繁体中文 Traditional Chinese Font)。

22.3 FontName のメソッド

第５部 バーコードクラス

Code39
CODABAR
CODE128
ITF
JAN

第２３章 xinca.barcode.BarcodeCODE39

BarcodeCODE39 extends xinca.barcode.BarCode

このクラスは PDF ファイルに埋め込む Barcode を取り扱う.

入力された文字列を CODE39 の Bar/SPACE パタンに変換, Xinca PDF page に挿入する。すべての ASCII コード（128 個）をサポートする。

CODE39 は、全英数字のキャラクタセットと、独自のスタート／ストップキャラクタ、数個の追加キャラクタからなるバーコードシンボル体系である。CODE39 の各々のキャラクタは全部で 9 個のエレメントからなり、そのうち 3 個が太エレメントである。この 9 個のエレメントは 5 個のバーと 4 個もスペースから構成される。

public BarcodeCODE39()
public BarcodeCODE39(byte text[], int length)
public BarcodeCODE39(java.lang.String text)

BarcodeCODE 39 のオブジェクトを生成する。

text：バーコード文字列

length：text の文字数

public void setChecksum(boolean _flag)

チェック用キャラクタを生成するかどうかを設定する。デフォルト値は生成しない。

flag＝true に設定する場合は、チェックキャラクタを生成。

public boolean create(xinca.PDFPage page)

バーコードの図形を生成し、PDFPage に挿入。

public void setText(byte text[], int length)
public void setText(java.lang.String text)

バーコードの文字列を設置。

public void setXUnitWidth(double w)

バーの幅を設定（inch 単位）

public void setYMultiplier(int n)

バーの高さを設定する。
高さ＝n＊バーの幅

public void setPosition(int x, int y)

バーコードの位置を設定する。

第２４章 xinca.barcode.BarcodeCODABAR

BarcodeCODABAR extends xinca.barcode.BarCode

このクラスは PDF ファイルに埋め込む Barcode を取り扱う.

入力された文字列を Codabar の Bar/SPACE パタンに変換, Xinca PDF page に挿入する。

Codabar は分離独立型のバーコードシンボル体系であり、0から9までの数字と“－＄：／．＋”からなる16種類のデータキャラクターを提供する。またAからDで示される4つの独立のスタート／ストップキャラクターもある。

このシンボル体系は可変長の、主に数字のシンボルを提供する。

public BarcodeCODABAR ()
public BarcodeCODABAR (byte text[], int length)
public BarcodeCODABAR (java.lang.String text)

BarcodeCODABAR のオブジェクトを生成する。

text：バーコード文字列

length：text の文字数

public void setStartSymbol(byte start)

CODABAR のスタートキャラクターを設定する。デフォルト値はAです。

public void setStopSymbol (byte stop)

CODABAR のストップキャラクターを設定する。デフォルト値はBです。

public void setChecksum(boolean _flag)

チェック用キャラクタを生成するかどうかを設定する。デフォルト値は生成しない。
flag＝true に設定する場合は、チェックキャラクタを生成。

public boolean create(xinca.PDFPage page)

バーコードの図形を生成し、PDFPage に挿入。

public void setText(byte text[], int length)
public void setText(java.lang.String text)

バーコードの文字列を設置。

public void setXUnitWidth(double w)

バーの幅を設定（inch 単位）

public void setYMultiplier(int n)

バーの高さを設定する。
高さ＝n＊Bar の幅

public void setPosition(int x, int y)

バーコードの位置を設定する。

第２５章 xinca.barcode. BarcodeCODE128

BarcodeCODE128 extends xinca.barcode.BarCode

このクラスは PDF ファイルに埋め込む Barcode を取り扱う.
入力された文字列を ITF の Bar/SPACE パターンに変換, Xinca PDFPage に挿入する。

CODE128は、フルASCIIの１２８キャラクタセット、１２８の拡張ＡＳＣＩＩキャラクタセット、ならびに4個の非データファンクションキャラクタをコード化することのできるバーコードシンボル体系である。数字データの場合、各シンボルキャラクタも2つの数字となるコンパクトな倍密度モードで表すこともできる。

各ＣＯＤＥ１２８シンボルは、パリテイによるキャラクタのセルフチェックとモジュラ１０３のチェックキャラクタという、２つの独立してセルフチェック機能を備えている。これによってリーダが誤読を起こすのを最小限にすることができる。外側のバーは２モジュール幅になっており、これによって多くのリーダで高密度のシンボルの読み取りが容易になる。

public BarcodeCODE128 ()
public BarcodeCODE128 (byte text[], int length)
public BarcodeCODE128 (java.lang.String text)

BarcodeCODE128 のオブジェクトを生成する。
text：バーコード文字列
length：text の文字数

public boolean create(xinca.PDFPage page)

バーコードの図形を生成し、PDFPage に挿入。

public void setText(byte text[], int length)
public void setText(java.lang.String text)

バーコードの文字列を設置。

public void setXUnitWidth(double w)

バーの幅を設定（inch 単位）

public void setYMultiplier(int n)

バーの高さを設定する。
高さ＝n＊Bar の幅

public void setPosition(int x, int y)

バーコードの位置を設定する。

第２６章 xinca.barcode. BarcodeITF

BarcodeITF extends xinca.barcode.BarCode

このクラスは PDF ファイルに埋め込む Barcode を取り扱う.

入力された文字列を ITF の Bar/SPACE パターンに変換, Xinca PDFPage に挿入する。

ITF(Interleaved 2-of-5)は数字キャラクタセット及び異なったスタート／ストップパターンを有するバーコードシンボル体系である。ITF シンボルは常に偶数個の数字で構成される。

このシンボルでは、2個のキャラクタがペアで使用され、最初のキャラクタはバーで表される、2番目のキャラクタはスペースで表される。個々のキャラクタはペアはシンボルキャラクタと呼ばれる。シンボルを構成するペアの各々のキャラクタは、データキャラクタまたはデジットと呼ばれる。

強化されたデータ安全性を必要とする応用状況では、チェックデジットを使用する必要がある。

public BarcodeITF()
public BarcodeITF(byte text[], int length)
public BarcodeITF (java.lang.String text)

BarcodeITF のオブジェクトを生成する。

text：バーコード文字列

length：text の文字数

public void setChecksum(boolean _flag)

チェック用キャラクタを生成するかどうかを設定する。デフォルト値は生成しない。

flag＝true に設定する場合は、チェックキャラクタを生成。

public boolean create(xinca.PDFPage page)

バーコードの図形を生成し、PDFPage に挿入。

public void setText(byte text[], int length)
public void setText(java.lang.String text)

バーコードの文字列を設置。

public void setXUnitWidth(double w)

バーの幅を設定（inch 単位）

public void setYMultiplier(int n)

バーの高さを設定する。
高さ＝n＊Bar の幅

public void setPosition(int x, int y)

バーコードの位置を設定する。

第２７章 xinca.barcode. BarcodeJAN

BarcodeJAN extends xinca.barcode.BarCode

このクラスは PDF ファイルに埋め込む Barcode を取り扱う.

入力された文字列を JAN の Bar/SPACE パタンに変換, Xinca PDF page に挿入する。標準と短縮の二つ JAN タイプが維持できる。

public BarcodeJAN(int makeCode, int itemCode, int type)

BarcodeJAN のオブジェクトを生成する。

makeCode：メーカーコード

itemCode：アイテムコード

type：　　バーコードのタイプ、

BarcodeJAN.TYPE_EAN_13：標準タイプ（13桁）

BarcodeJAN.TYPE_EAN_8：　短縮タイプ（8桁）

public void setCountryCode(int　countryCode)

国コードを設定し

CountryCode ：国コード、デフォルトは日本国のコード（４９）です。

public boolean create(xinca.PDFPage page)

バーコードの図形を生成し、PDFPage に挿入。

public void setText(byte text[], int length)
public void setText(java.lang.String text)

バーコードの文字列を設置。

public void setXUnitWidth(double w)

バーの幅を設定（inch 単位）

public void setYMultiplier(int n)

バーの高さを設定する。

高さ＝n＊Bar の幅

public void setPosition(int x, int y)

バーコードの位置を設定する。

第 28 章 xinca.barcode. BarcodeCODE93

BarcodeCODE93extends xinca.barcode.BarCode

このクラスは PDF ファイルに埋め込む Barcode を取り扱う.

入力された文字列を CODE93 の Bar/SPACE パタンに変換, Xinca PDF page に挿入する。

CODE93 は、４３個のデータキャラクタ（０～９、A～Z、6 個のシンボル、及びスペース）と、4 個のシフトキャラクタ、それに独自のスタート／ストップキャラクタをコード化するバーコードシンボル体系である。

CODE93 では、１２８のフル ASCII キャラクタはシフトキャラクタと基本のデータキャラクタの組み合わせで表される。

CODE93 の名称は、各キャラクタが 9 つのモジュールからなり、それらのモジュールが 3 本のバーとそれに接するスペースとして表現されるという事実に由来する。

各 CODE９３シンボルは二個のチェックキャラクタを含む。S

public BarcodeCODE93 ()
public BarcodeCODE93 (byte text[], int length)
public BarcodeCODE93 (java.lang.String text)

BarcodeCODE93 のオブジェクトを生成する。

text：バーコード文字列

length：text の文字数

public boolean create(xinca.PDFPage page)

バーコードの図形を生成し、PDFPage に挿入。

public void setText(byte text[], int length)
public void setText(java.lang.String text)

バーコードの文字列を設置。

public void setXUnitWidth(double w)

バーの幅を設定（inch 単位）

public void setYMultiplier(int n)

バーの高さを設定する。
高さ＝n＊バーの幅

public void setPosition(int x, int y)

バーコードの位置を設定する。

第六部　グラフパッケージ

XincaServer のグラフ基本パック(xinca.tools.chart)はグラフを生成するツールです。現在、サポートされているグラフは次の七種類です。

- 縦棒グラフ
- 横棒グラフ
- 積層形縦棒グラフ
- 積層形横棒グラフ
- 円グラフ
- 折線グラフ
- 散布図グラフ

特徴

様々な要求を満足するために、ユーザーはグラフのすべての属性を個別に設定できます。例えば、文字の色、フォント名とサイズ、線の色、幅と点線、グラフの各エリアのサイズ等です。属性は多いのですが、必要な要素はデフォルト値が設定されています。通常の場合、ユーザーは設定するパラメータは少ない。

更に、XincaServer のグラフ基本パックはグラフのレイアウトの自動計算機能があるので、グラフのエリアのサイズはこのエリアの内容の実際のサイズによって、自動に調整できます。例えば、指定されたエリアのサイズはエリアの内容の実際のサイズより小さい場合に、エリアのサイズは実際のサイズによって、変更できます。

クラス設計方法

グラフの属性は多いので、これらの属性を次の三種類に分けます。

① レイアウトの制御
グラフのレイアウトを管理します。

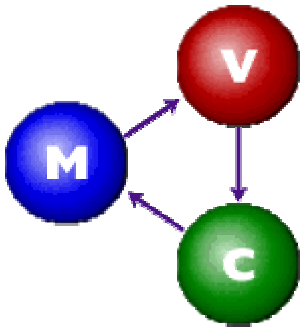
例えば、各エリアのサイズと位置等を設定します。

② グラフの凡例のレイアウトの制御
グラフの凡例のレイアウトを管理します。
例えば、凡例の高さ、幅、色等を設定します。

③データモデルの制御

グラフには表示する内容を管理します。
例えば、データ、タイトル文字と凡例文字等を設定します、および、データモデルでデータの統計方法等を設定します。

前文の分類は MVC (Model-View-Controller) 設計方法を利用して、レイアウトクラスは Controller 部分に対応し、データモデルは Model 部分に対応し、グラフの表示クラスは View 部分に対応します。



マニュアルの構成

この部分の構成は次のとおりです。

- **第29章：サポートされているグラフ仕様**
  サポートされているグラフ仕様について説明します。

- **第30章：XincaServer グラフ基本パック説明**
  グラフ基本パックのクラス関連図について説明します。

- **第31章：XincaServer グラフクラスの使い方**
  グラフクラスの使い方について説明します。

- **第32章：グラフのレイアウト**
  グラフのレイアウト、および関連するクラスとメソッドについて説明します。

- **第33章：グラフのレイアウト要素定義**
  グラフのレイアウト要素についての定義、および関連するクラスとメソッドについて説明します。

- **第34章：グラフの文字要素定義**
  グラフの文字要素についての定義、および関連するクラスとメソッドについて説明します。

- **第35章：グラフの凡例の共通属性定義**
  凡例の共通属性についての定義、および関連するクラスとメソッドについて説明します。

- **第36章：グラフの凡例クラスインターフェイス**
  凡例クラスのインターフェイスについて説明します。

- **第37章：グラフの棒凡例の属性定義**
  棒凡例の属性についての定義、および関連するクラスとメソッドについて説明します。

- **第38章：グラフの折線凡例の属性定義**
  折線凡例の属性について定義、および関連するクラスとメソッドについて説明します。

- **第39章：グラフの散布図凡例の属性定義**
  散布凡例の属性について定義、および関連するクラスとメソッドについて説明します。

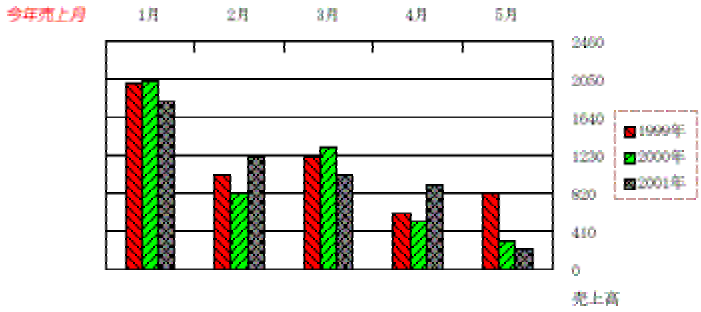
- **第40章： XincaServer グラフパッケージの API 説明**

xinca.tools.chart パッケージの API について説明します。

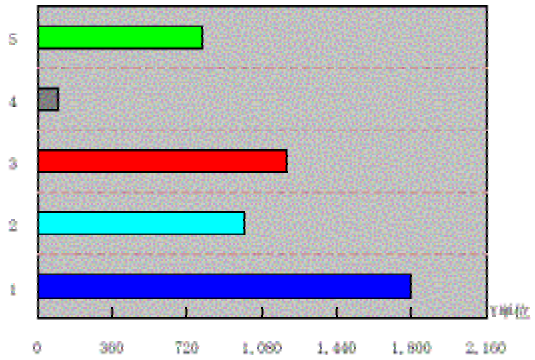
- 第41章 ～ 第45章:サンプル一 ～ サンプル五
  サンプルです。

第２９章　サポートされているグラフ仕様

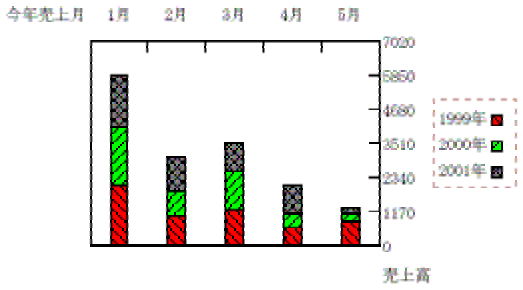
縦棒グラフ



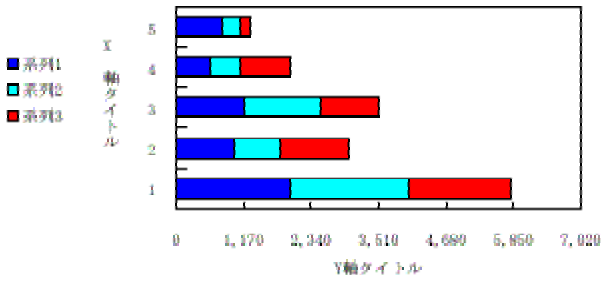
横棒グラフ



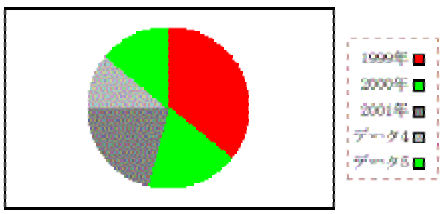
積層形縦棒グラフ



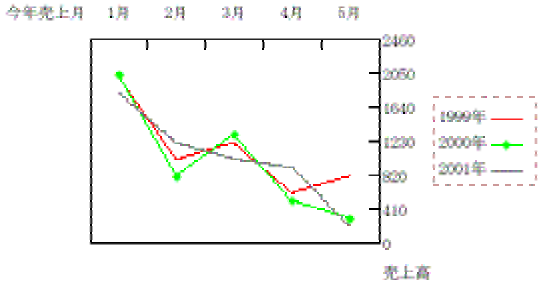
積層形横棒グラフ



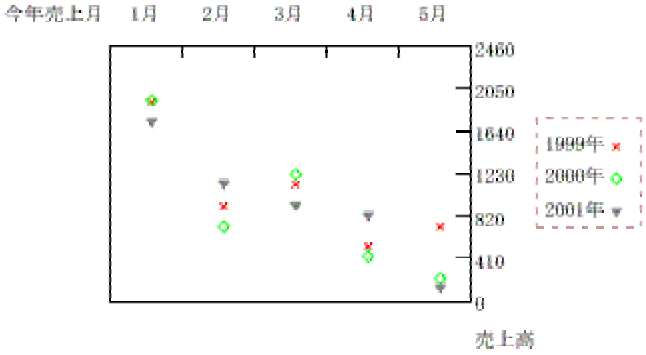
円グラフ



折線グラフ



散布図グラフ



第３０章　XincaServer グラフ基本パック説明

パッケージ名

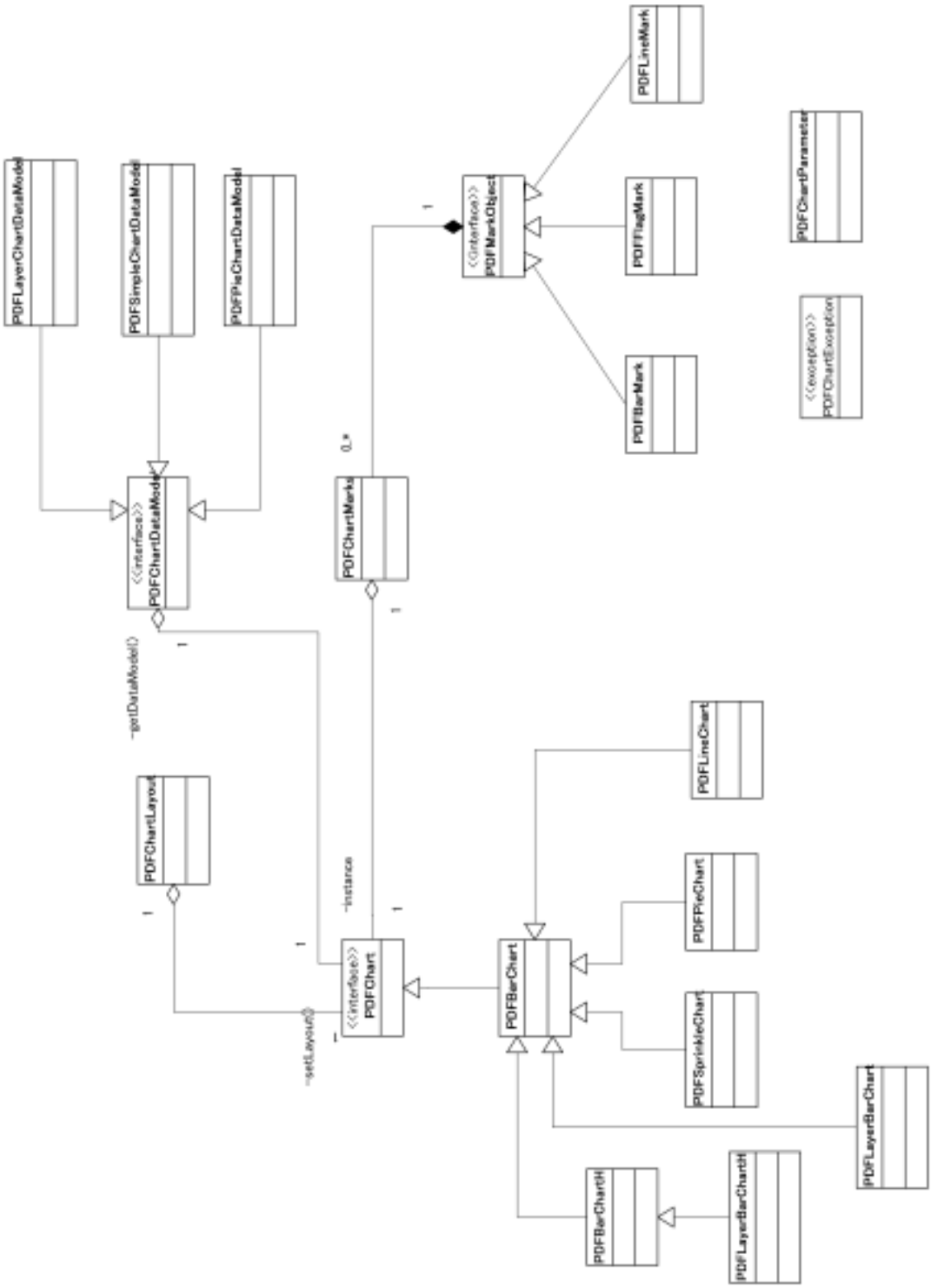
xinca.tools.chart

クラス一覧

クラス名	説明
PDFChartLayout	グラフのレイアウト属性のクラス
PDFChartDataModel	グラフのデータモデルの抽象基底クラス
PDFSimpleChartDataModel	棒、折線、散布図グラフのデータモデル
PDFLayerChartDataModel	積層形棒グラフのデータモデル
PDFPieChartDataModel	円グラフのデータモデル
PDFChart	グラフの抽象基底クラス
PDFBarChart	縦棒グラフ
PDFLayerBarChart	積層形縦棒グラフ
PDFBarChartH	横棒グラフ
PDFLayerBarChartH	積層形横棒グラフ
PDFSprinkleChart	散布図グラフ
PDFPieChart	円グラフ
PDFLineChart	折線グラフ
PDFChartMarks	凡例配列を格納するクラス
PDFMarkObject	凡例マークの抽象基底クラス
PDFBarMark	棒マークの属性を設定する
PDFFlagMark	散布図のマークの属性を設定する
PDFLineMark	線マークの属性を設定する
PDFChartParameter	共通定数を定義するクラス
PDFChartException	グラフの例外クラス

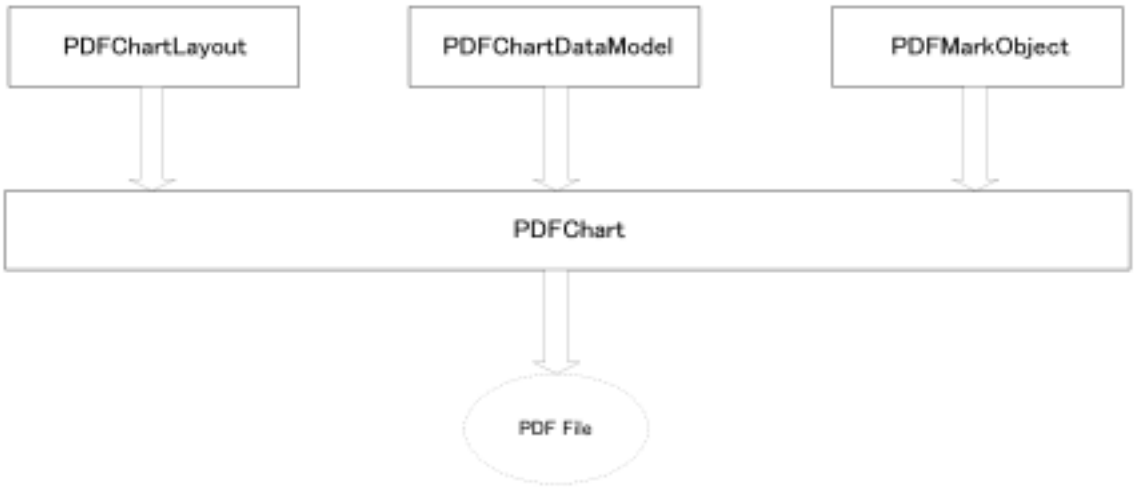
クラス関連図

次のページをご覧ください。


PDFLineMark
PDFChartParameter
PDFLayerChartDataModel
PDFSimpleChartDataModel
PDFPieChartDataModel
<<interface>>
PDFMarkObject
PDFFlagMark
<<exception>>
PDFChartException
PDFBarMark
0..*
PDFChartMarks
<<interface>>
PDFChartDataModel
-getDataModel()
PDFLineChart
PDFChartLayout
-instance
PDFPieChart
<<interface>>
PDFChart
PDFBarChart
PDFSprinkleChart
-setLayout()
PDFLayerBarChart
PDFBarChartH
PDFLayerBarChartH

第３１章　XincaServer グラフクラスの使い方

プログラミングモデル



PDFChart オブジェクトの取得

PDFChart は抽象基底クラスですから、PDFChart オブジェクトを取得ために、メソッド **getInstance** が提供されます：

```
PDFChart chart = PDFChart.getInstance(type)
```

type の値は PDFChart クラスには定義されます、次のとおりです。

PDFChart.VERTICAL_BAR_CHART	縦棒グラフ
PDFChart.HORIZONTAL_BAR_CHART	横棒グラフ
PDFChart.VERTICAL_LAYER_BAR_CHART	積層形縦棒グラフ
PDFChart.HORIZONTAL_LAYER_BAR_CHART	積層形横棒グラフ
PDFChart.PIE_CHART	円グラフ
PDFChart.LINE_CHART	折線グラフ
PDFChart.SPRINKLE_CHART	散布図グラフ

PDFChartDataMode オブジェクトの取得

```
PDFChartDataModel mode = PDFChart.getDataModel()
```

PDFMarkObject オブジェクトの取得

① 棒、積層形棒、円グラフの場合：

```
PDFBarMark mark = (PDFBarMark) PDFChart. newMarkInstance ()
```

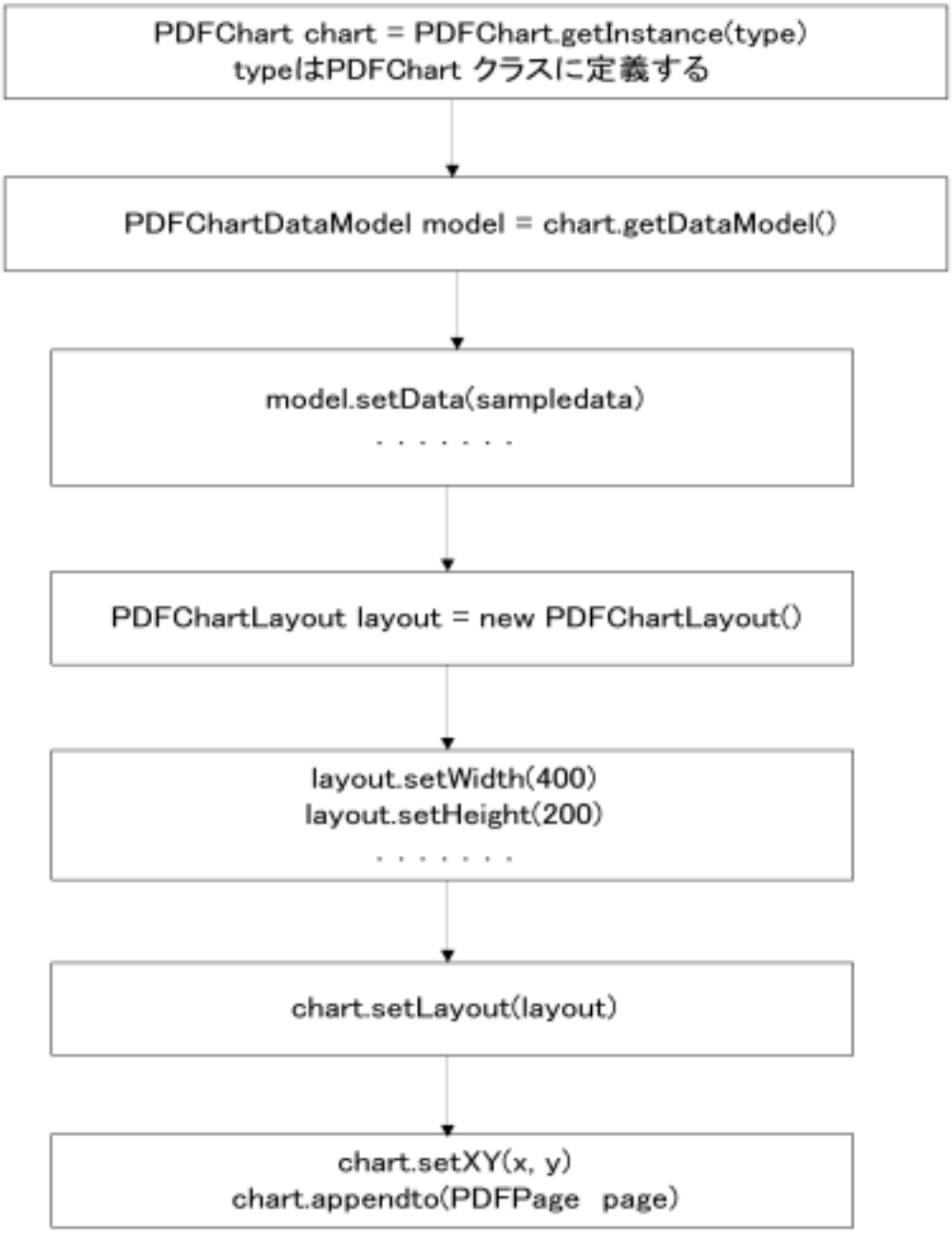
② 折線グラフの場合：

```
PDFLineMark mark = (PDFLineMark) PDFChart. newMarkInstance ()
```

③ 散布図グラフの場合：

```
PDFFlagMark mark = (PDFFlagMark) PDFChart. newMarkInstance ()
```

プログラミングフローチャート



サンプルソース

```
. . . . . .

import xinca.tools.chart.PDFChartDataModel;
import xinca.tools.chart.PDFChartParameter;
import xinca.tools.chart.PDFChartLayout;
import xinca.tools.chart.PDFChartException;

import xinca.tools.chart.PDFChart;

. . . . . .

        //縦棒統計図オブジェクトを取得します。
        PDFChart chart = PDFChart.getInstance(PDFChart.VERTICAL_BAR_CHART);

        //この統計図オブジェクトのデータ模型を取得します。
        PDFChartDataModel model = chart.getDataModel();

        double[] sample1 = {1800, 1000, 1200, 100, 800};
        //データを設定する。
        model.setData(sample1);

        //PDFChartLayout オブジェクトを構築します。
        PDFChartLayout layout = new PDFChartLayout();

        //統計グラフのレイアウトオブジェクトを設定する。
        chart.setLayout(layout);

        //グラフの左上角はPDFPageに座標を設定する。
        chart.setXY(100, 100);
        try
        {
                //PPDFChartObjectオブジェクトをPDFPageに書く。
                chart.appendto(page);
        }catch(PDFChartException e)
        {
                e.printStackTrace();
        }

. . . . . .
```

統計図パッケージをインポートする

Step 1

Step 2

Step 3

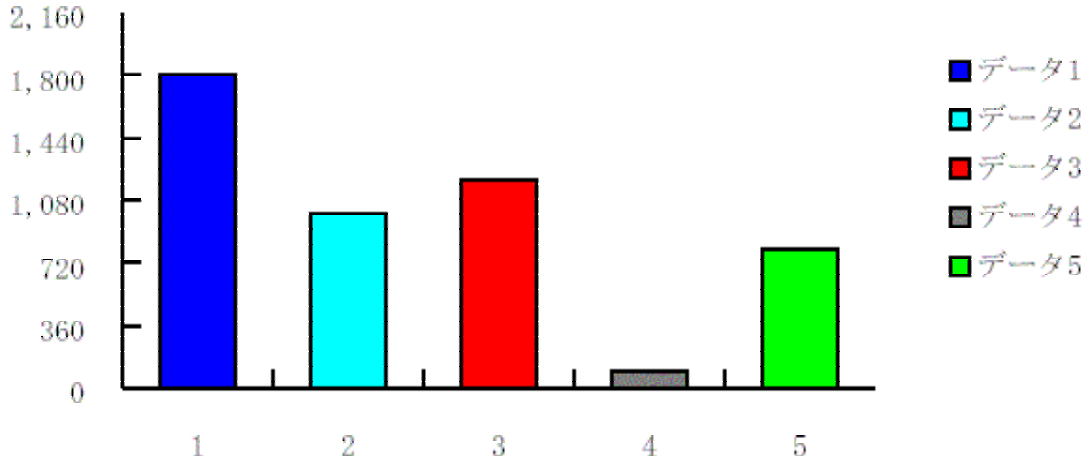
Step 4

Step 5

Step 6

Step 7

サンプルソースで生成されたグラフ

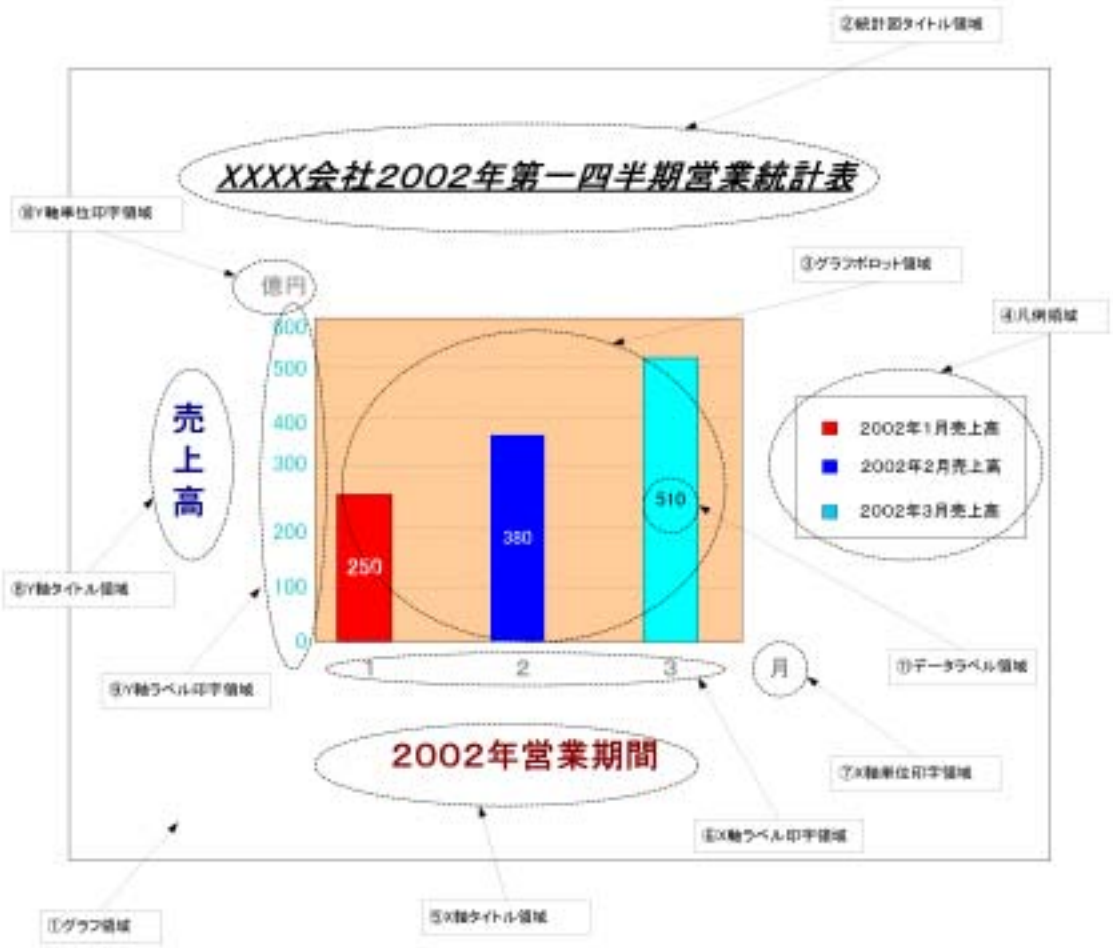


第３２章　グラフのレイアウト

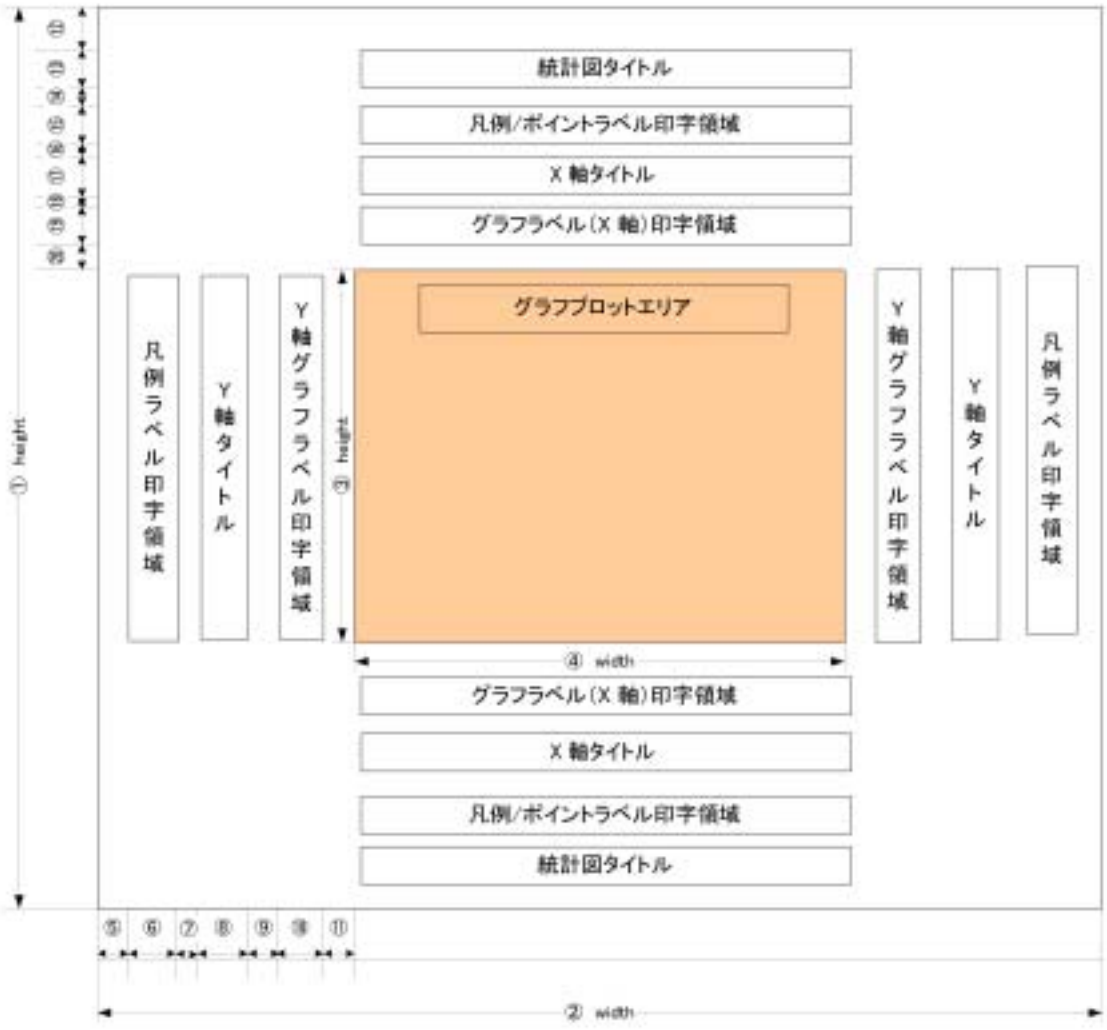
クラス名

xinca.tools.chart.PDFChartLayout

グラフの各領域の定義



グラフのレイアウト定義



グラフのレイアウトの属性説明

レイアウトの属性	関連メソッド
①グラフ領域の高さ	xxxHeight
②グラフ領域の幅	xxxWidth
③グラフプロットエリア領域の高さ	xxxHeightPlot
④グラフプロットエリア領域の幅	xxxWidthPlot
⑤グラフの横余白	xxxMarginOuterH
⑫グラフの縦余白	xxxMarginOuterV
⑥凡例/ポイントラベル領域の幅	xxxWidthMarkLabelH
⑮凡例/ポイントラベル領域の高さ	xxxHeightMarkLabelV
⑭グラフタイトル･凡例/ポイントラベル間の縦マージン	xxxMarginChartTitlekWithMarkV
⑦凡例/ポイントラベル･Y 軸タイトル間の横マージン	xxxMarginMarkWithYAxisTitleH
⑯凡例/ポイントラベル･X 軸タイトル間の縦マージン	xxxMarginMarkWithXAxisTitleV
⑧Y 軸タイトル領域の幅	xxxWidthYAxisTitleH
⑰X 軸タイトル領域の高さ	xxxHeightXAxisTitleV
⑨Y 軸タイトル･Y 軸グラフラベル間の横マージン	xxxMarginYAxisTitleWithYAxisScaleH
⑱X 軸タイトル･X 軸グラフラベル間の縦マージン	xxxMarginXAxisTitleWithXAxisScaleV
⑩ Y 軸グラフラベル領域の幅	xxxWidthAxisScaleLabelH
⑲ X 軸グラフラベル領域の高さ	xxxHeightAxisScaleLabelV
⑬ グラフタイトル領域の高さ	xxxHeightChartTitle
⑪ グラフ領域とプロットエリア間の横マージン	xxxMarginAxisScaleWithPlotH
⑳ グラフ領域とプロットエリア間の縦マージン	xxxMarginAxisScaleWithPlotV

注：xxx は get と set を表します。

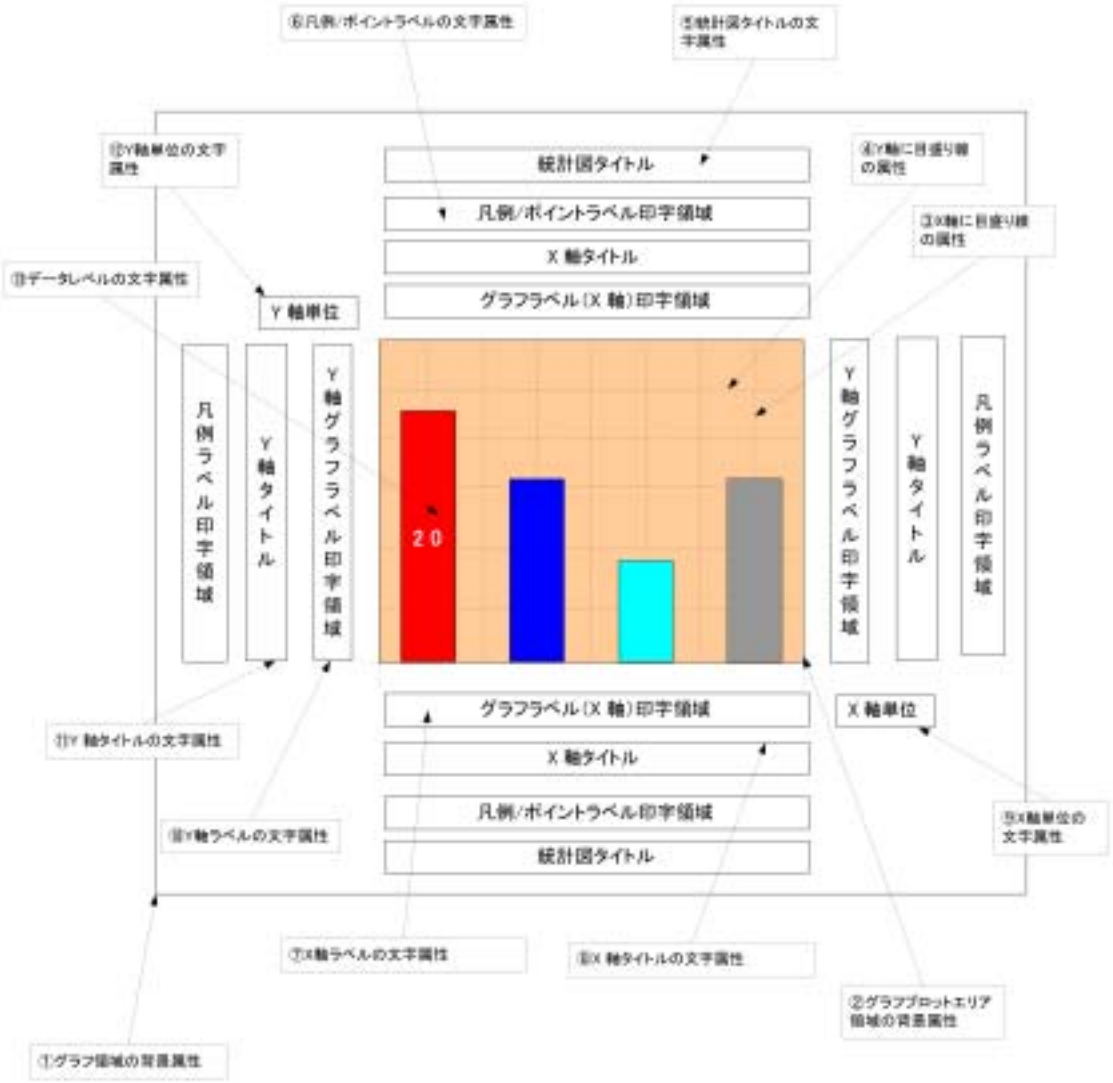
例えば、xxxWidthPlot は、PDFChartLayout.getWidthPlot と PDFChartLayout.setWidthPlot の二つメソッドを表します。

第３３章　グラフのレイアウト要素定義

クラス名

xinca.tools.chart.PDFChartLayout

グラフのレイアウトの要素



グラフのレイアウトの要素の説明

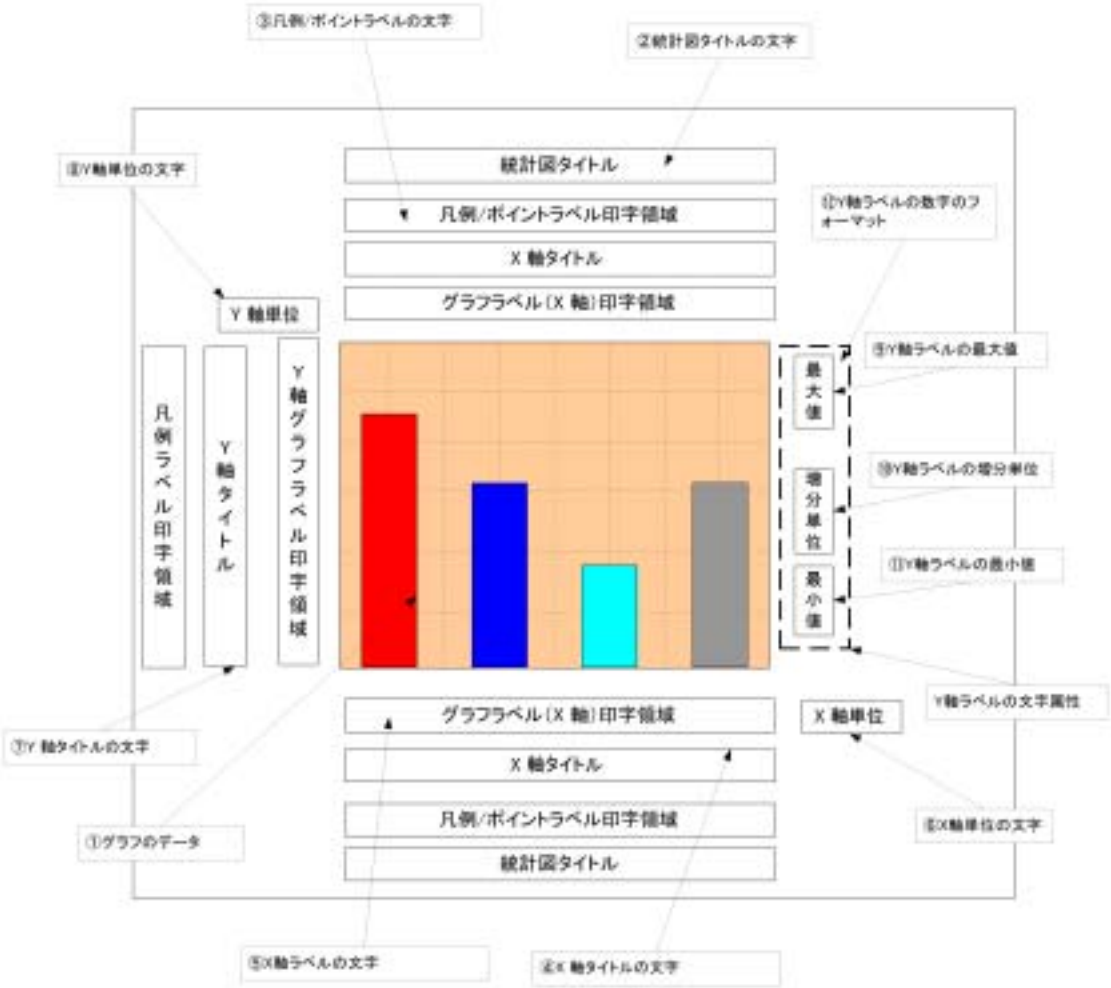
レイアウト要素の属性	関連メソッド
①グラフ領域の枠属性	xxxAttrBackgroundGraphic
②グラフプロットエリア領域の背景属性	xxxAttrPlotBackgroundGraphic
③X 軸に目盛り線の属性	xxxAttrXAxisGraduationLine
X 軸に目盛り線の有無	xxxDrawXAxisGraduationLine
④Y 軸に目盛り線の属性	xxxAttrYAxisGraduationLine
Y 軸に目盛り線の有無	xxxDrawYAxisGraduationLine
⑤グラフタイトルの文字属性	xxxAttrChartTitle
グラフタイトルの有無	xxxLabelChartTitle
グラフタイトルの位置	xxxPosChartTitle
⑥凡例/ポイントラベルの文字属性	xxxAttrMarkLabelText
凡例ラベルの有無	xxxLabelMark
凡例ラベルの位置	xxxPosMark
凡例ラベルの揃え方式	xxxAlignMark
凡例ラベルに文字の位置	xxxPosMarkText
⑦X 軸ラベルの文字属性	xxxAttrXAxisLabelText
⑧X 軸タイトルの文字属性	xxxAttrXAxisTitle
⑨X 軸単位の文字属性	xxxAttrXAxisUnit
X 軸単位の有無	xxxLabelXAxisUnit
X 軸単位の位置	xxxPosXAxisUnit
⑩Y 軸ラベルの文字属性	xxxAttrYAxisLabelText
⑪Y 軸タイトルの文字属性	xxxAttrYAxisTitle
⑫Y 軸単位の文字属性	xxxAttrYAxisUnit
Y 軸単位の有無	xxxLabelYAxisUnit
Y 軸単位の位置	xxxPosYAxisUnit
座標ラベルの有無	xxxLabelAxisScale
座標ラベルの位置	xxxPosAxisScale
⑬データレベルの文字属性	xxxAttrDataLebelText
データレベルの有無	xxxLabelDataLabel
データレベルの位置	xxxPosDataLabel

第３４章　グラフの文字要素定義

クラス名

xinca.tools.chart.PDFChartDataModel

グラフの文字の要素



グラフの文字要素の説明

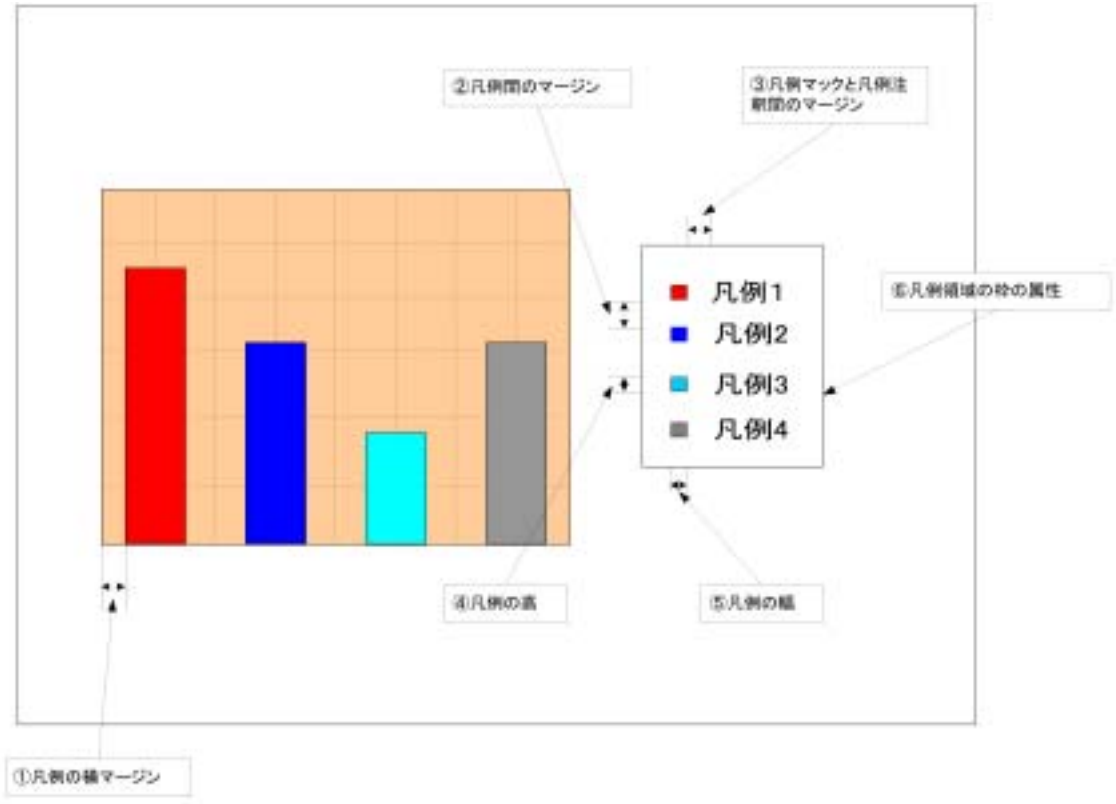
文字要素の属性	関連メソッド
①グラフのデータ	xxxData
②グラフタイトル文字	xxxTextChartTitle
③凡例/ポイントラベルの文字	xxxTextNote
④X 軸タイトル文字	xxxTextXAxisTitle
⑤X 軸のラベル文字	xxxXAxisLabelText
⑥X 軸の単位	xxxXAxisUnit
⑦Y 軸タイトル文字	xxxTextYAxisTitle
⑧Y 軸の単位	xxxYAxisUnit
⑨Y 軸の最大値	xxxYAxisMax
⑩Y 軸の増分単位	xxxYAxisStep
⑪Y 軸の最小値	xxxYAxisMin
⑫Y 軸の数字フォーマット	xxxYFormat

第３５章　グラフの凡例の共通属性定義

クラス名

xinca.tools.chart.PDFChart

グラフの凡例の共通属性



グラフの凡例の共通属性の説明

凡例の共通属性		関連メソッド
①凡例の横マージン		xxxMarginBar
②凡例マーク間のマージン		xxxSizeMarkSpace
③凡例マークと凡例注釈間のマージン		xxxMarginMarkWithText
④凡例の高さ		xxxHeightMark
⑤凡例の幅		xxxWidthMark
⑥凡例領域の枠の属性		xxxAttrFrame

第３６章　グラフの凡例クラスインターフェイス

クラス名

xinca.tools.chart.PDFMarkObject

グラフの凡例インターフェイスの説明

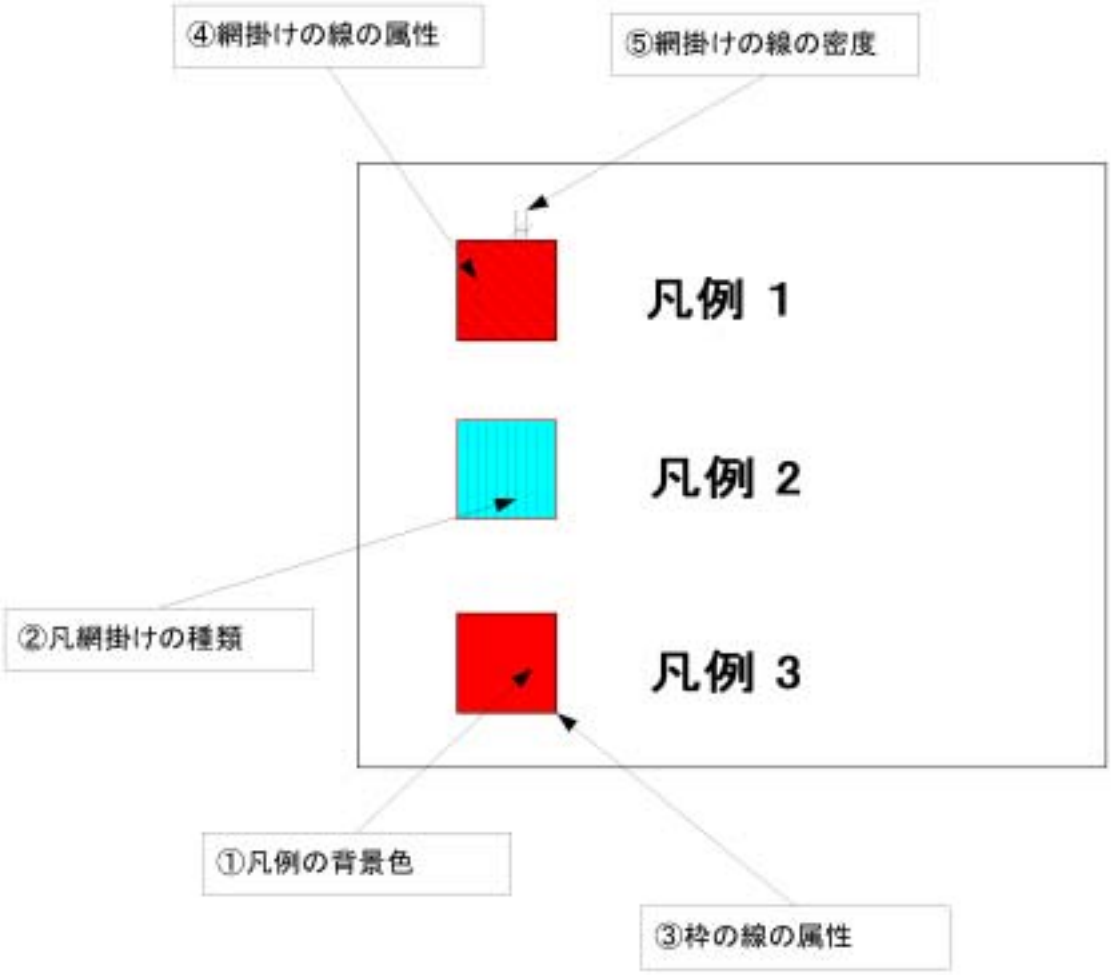
凡例の属性		関連メソッド
①凡例の種類		xxxTypeMark
②凡例の色		xxxColorMark

第３７章　グラフの棒凡例の属性定義

クラス名

xinca.tools.chart.PDFBarMark

グラフの棒凡例の属性



グラフの棒凡例の属性の説明

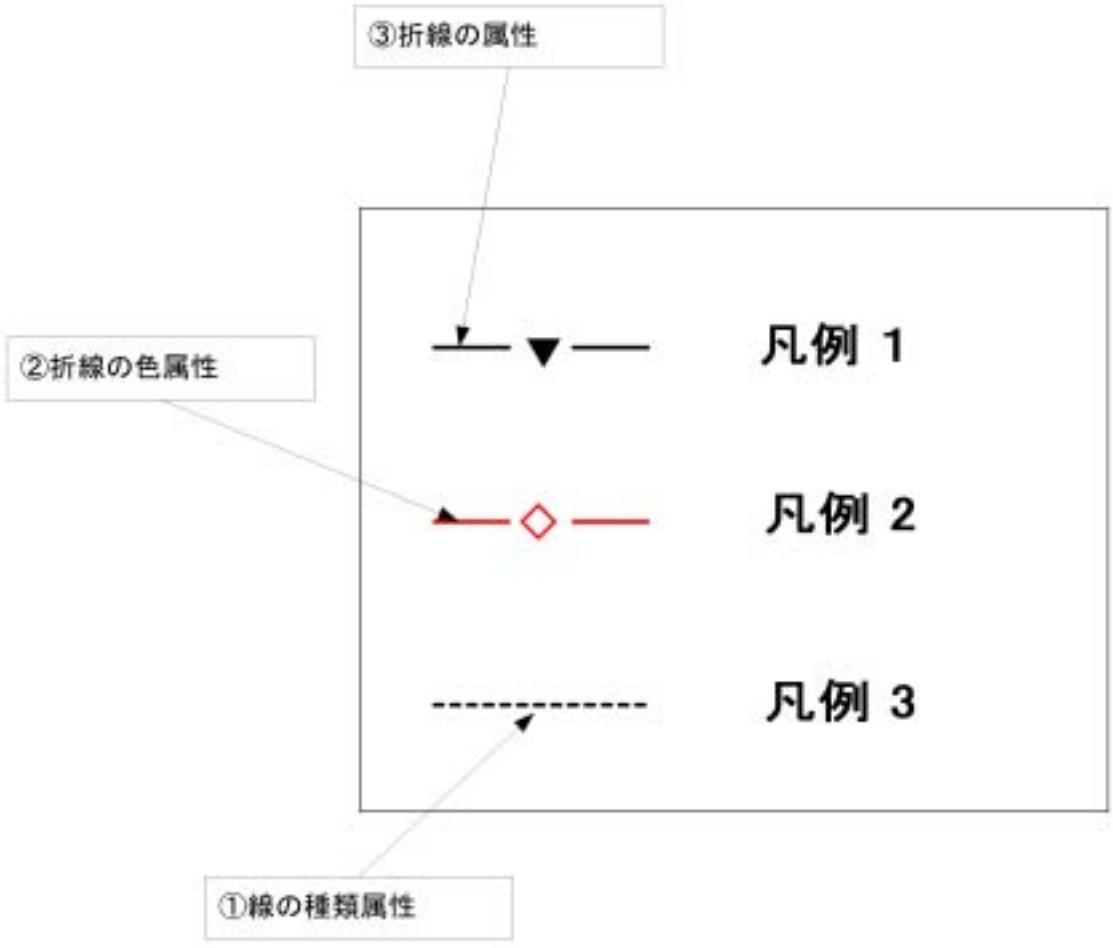
棒凡例の属性	関連メソッド
①凡例の背景色	xxxBackgroundColor
	xxxColorMark
②凡例の網掛けの種類	xxxModeBackground
	xxxTypeMark
③枠の線の属性	xxxAttrFrame
④網掛けの線の属性	xxxAttrBackground
⑤網掛けの線の密度	xxxDensityBackground

第３８章　グラフの折線凡例の属性定義

クラス名

xinca.tools.chart.PDFLineMark

グラフの折線凡例の属性



グラフの折線凡例の属性の説明

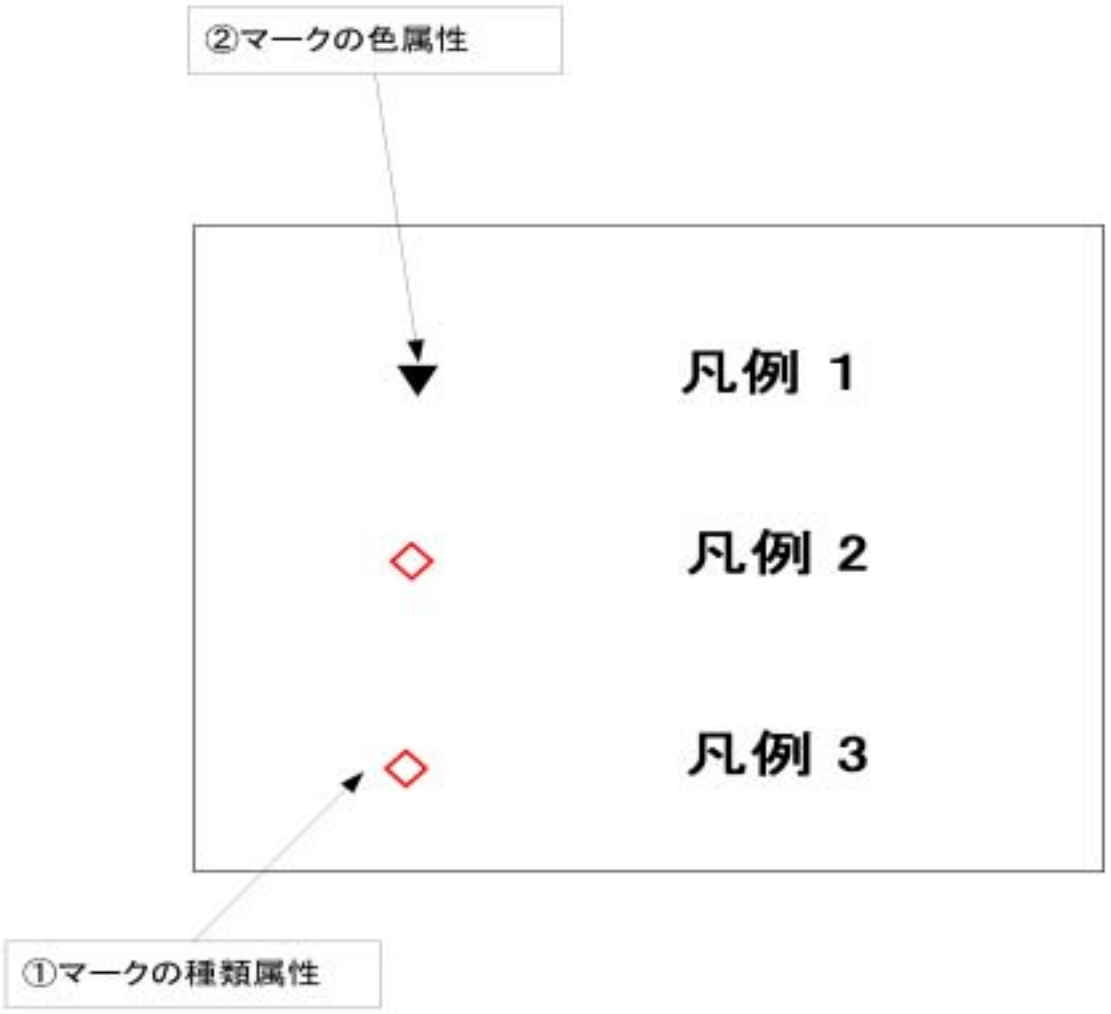
折線凡例の属性	関連メソッド
①線の種類	xxxTypeLine
	xxxTypeMark
②線の色	xxxColorLine
	xxxColorMark
③折線の属性	xxxAttrLine

第３９章　グラフの散布図凡例の属性定義

クラス名

xinca.tools.chart.PDFFlagMark

グラフの散布図凡例の属性



グラフの散布図凡例の属性の説明

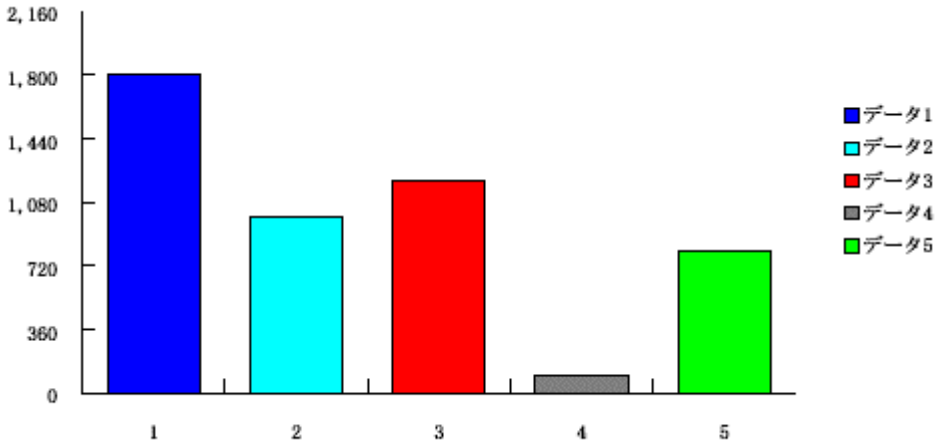
散布図凡例の属性	関連メソッド
①マークの種類	xxxTypeMark
②マークの色	xxxColorMark

第４０章　XincaServer グラフ基本パックの API 説明

XincaServer グラフ基本パックの API の詳細な説明は xinca.tools.chart の javadoc を参照してください。

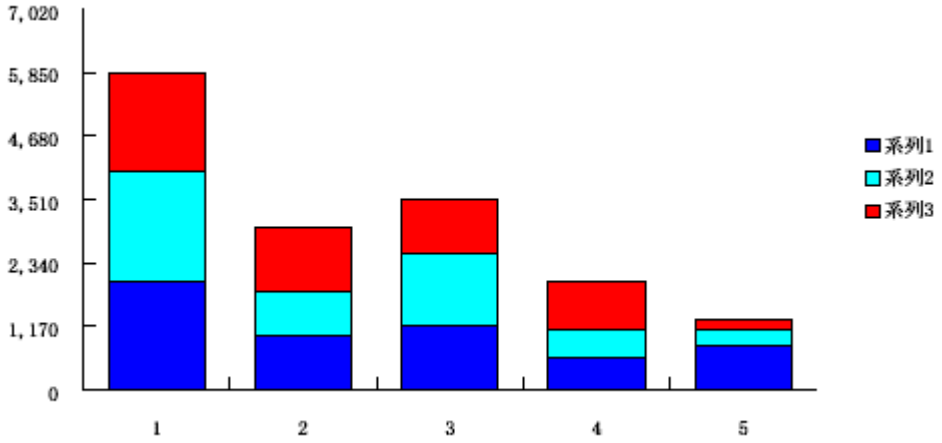
第４１章　サンプルー

```
import xinca.tools.chart.PDFChart;
import xinca.tools.chart.PDFChartDataModel;
import xinca.tools.chart.PDFChartParameter;
import xinca.tools.chart.PDFChartLayout;
import xinca.tools.chart.PDFChartException;

. . . . . .

        //縦棒グラフオブジェクトを取得する
          PDFChart chart = PDFChart.getInstance(PDFChart.VERTICAL_BAR_CHART);
          //このグラフオブジェクトのデータモデルを取得する
        PDFChartDataModel model = chart.getDataModel();

        model.setData(sample1);         //データを設定する

        //PDFChartLayout  オブジェクトを構築する
        PDFChartLayout layout = new PDFChartLayout();
        layout.setWidth(400);
        layout.setHeight(200);

        chart.setLayout(layout);
        chart.setXY(100, 100);

        try
        {
                chart.appendto(page);
        }catch(PDFChartException e)
        {
                e.printStackTrace();
        }

        double[] sample1 = {1800, 1000, 1200, 100, 800};
```


2,160
1,800
1,440
1,080
720
360
0
1
2
3
4
5
データ1
データ2
データ3
データ4
データ5

第４２章　サンプルニ

```
. . . . . .

//縦棒グラフオブジェクトを取得する
PDFChart chart = PDFChart.getInstance(PDFChart. VERTICAL_LAYER_BAR_CHART);
        //このグラフオブジェクトのデータモデルを取得する
PDFChartDataModel model = chart.getDataModel();

model.setData(sample2);         //データを設定する

//PDFChartLayout  オブジェクトを構築する
PDFChartLayout layout = new PDFChartLayout();
layout.setWidth(400);
layout.setHeight(200);

chart.setLayout(layout);
chart.setXY(100, 100);

try
{
            chart.appendto(page);
}catch(PDFChartException e)
{
            e.printStackTrace();
}
. . . . . .

double[][] sample2 = {{2000, 1000, 1200, 600, 800},
                      {2020, 800, 1300, 500, 300},
                      {1800, 1200, 1000, 900, 200}};
```


7,020
5,850
4,680
3,510
2,340
1,170
0
1
2
3
4
5
系列1
系列2
系列3

第４３章　サンプル三

```
import    java.awt.Color;

import    xinca.PDFFile;
import    xinca.PDFPages;
import    xinca.PDFPage;

import xinca.tools.PDFPoint;

import    xinca.util.PDFGraphicAttribute;

import xinca.tools.chart.PDFChartDataModel;
import xinca.tools.chart.PDFChartParameter;
import xinca.tools.chart.PDFChartLayout;
import xinca.tools.chart.PDFChartException;

import xinca.tools.chart.PDFChart;

public class ChartSample
{
        public static void main(String argv[ ])
        {
                    ChartSample inst = new ChartSample();
                    inst.create();
                    System.exit(0);
        }

        public    void create()
        {
            PDFFile    pdffile = new PDFFile("ChartSample.pdf");
             PDFPages pgs         = new PDFPages(pdffile);
```

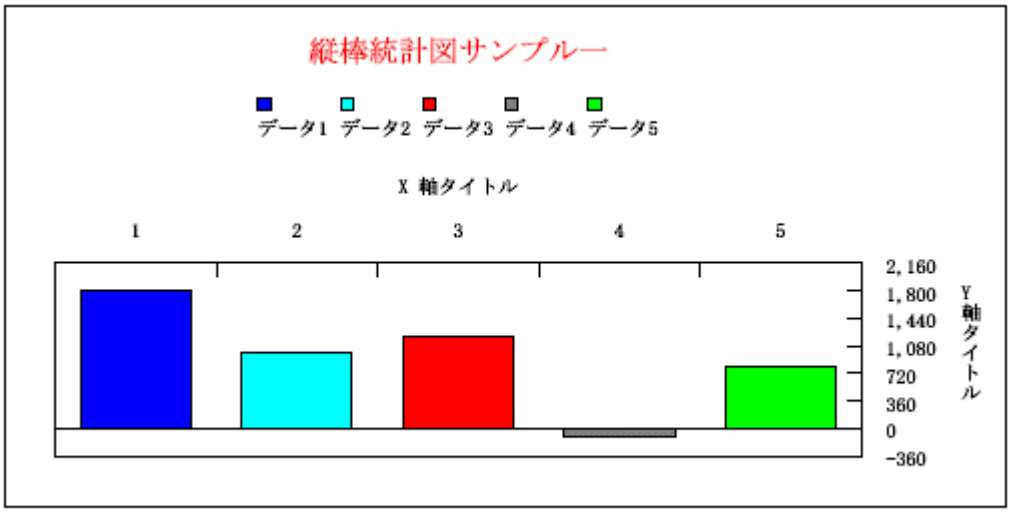
```
        pgs.append(createPage(pgs, PDFChart.VERTICAL_BAR_CHART,  "縦棒グラフ"));
        pgs.append(createPage(pgs, PDFChart.PIE_CHART,  "円グラフ"));
pgs.append(createPage(pgs, PDFChart.VERTICAL_LAYER_BAR_CHART, "積層形縦棒グラフ"));
        pgs.append(createPage(pgs, PDFChart.LINE_CHART,  "折線グラフ"));
        pgs.append(createPage(pgs, PDFChart.SPRINKLE_CHART,    "散布グラフ"));

            pdffile.initialFile(pgs);
            pdffile.setPageSize("a4");
            pdffile.writeFile();

        }

        private PDFPage createPage(PDFPages pgs, int type, String title)
        {
            PDFPage    page       = new PDFPage(pgs);
                PDFPoint pagesize = page.getPageSize();

                int x = 50;
                int y = pagesize.y - 100;

                //グラフサンプル一

                //指定された種類を使用してグラフオブジェクトを取得する
                PDFChart chart = PDFChart.getInstance(type);
                //このグラフオブジェクトのデータモデルを取得する
                PDFChartDataModel model = chart.getDataModel();

                //データを設定する
                model.setData(sample1);
                //グラフのタイトル文字を設定する
                model.setTextChartTitle(title+"サンプル一");
                //X 軸タイトル文字を設定する
                model.setTextXAxisTitle("X 軸タイトル");
                //Y 軸タイトル文字を設定する
                model.setTextYAxisTitle("Y 軸タイトル");
```

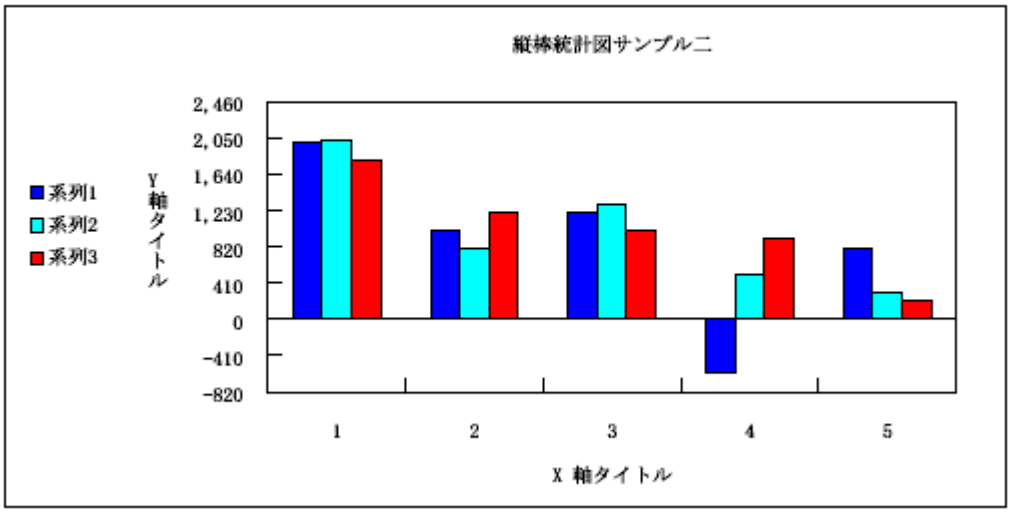
```
//PDFChartLayout オブジェクトを構築する
PDFChartLayout layout = new PDFChartLayout();
//グラフの幅を設定する
layout.setWidth(400);
//グラフの高さを設定する
layout.setHeight(200);
//凡例の位置を設定する
layout.setPosMark(PDFChartParameter.POSITION_UPPER);
//凡例に文字の位置を設定する
layout.setPosMarkText(PDFChartParameter.POSITION_DOWN);

//グラフの座標の位置を設定する
layout.setPosAxisScale(PDFChartParameter.POSITION_RIGHT +
PDFChartParameter.POSITION_UPPER);

//グラフのタイトル位置を設定する
layout.setPosChartTitle(PDFChartParameter.POSITION_UPPER);

layout.setLabelChartTitle(true);
layout.setLabelXAxisTitle(true);
layout.setLabelYAxisTitle(true);

//グラフのタイトルの文字サイズを設定する
layout.getAttrChartTitle().setSizeChar(12);
//グラフのタイトルの文字色を設定する
layout.getAttrChartTitle().setColor(Color.red);

layout.setAttrBackgroundGraphic(new PDFGraphicAttribute());
layout.setAttrPlotBackgroundGraphic(new PDFGraphicAttribute());

chart.setLayout(layout);
chart.setXY(x, y);

try
{
        chart.appendto(page);
```

```
}catch(PDFChartException e)
{
        e.printStackTrace();
}

//グラフサンプル二
chart = PDFChart.getInstance(type);
model = chart.getDataModel();
model.setData(sample2);

model.setTextChartTitle(title+"サンプル二");
model.setTextXAxisTitle("X 軸タイトル");
model.setTextYAxisTitle("Y 軸タイトル");

layout = new PDFChartLayout();
layout.setWidth(400);
layout.setHeight(200);
layout.setPosMark(PDFChartParameter.POSITION_LEFT);

layout.setLabelChartTitle(true);
layout.setLabelXAxisTitle(true);
layout.setLabelYAxisTitle(true);
chart.setLayout(layout);

layout.setAttrBackgroundGraphic(new PDFGraphicAttribute());
layout.setAttrPlotBackgroundGraphic(new PDFGraphicAttribute());

y -= chart.getSize().y + 50;
chart.setXY(x, y);
try
{
        chart.appendto(page);
}catch(PDFChartException e)
{
        e.printStackTrace();
}
```

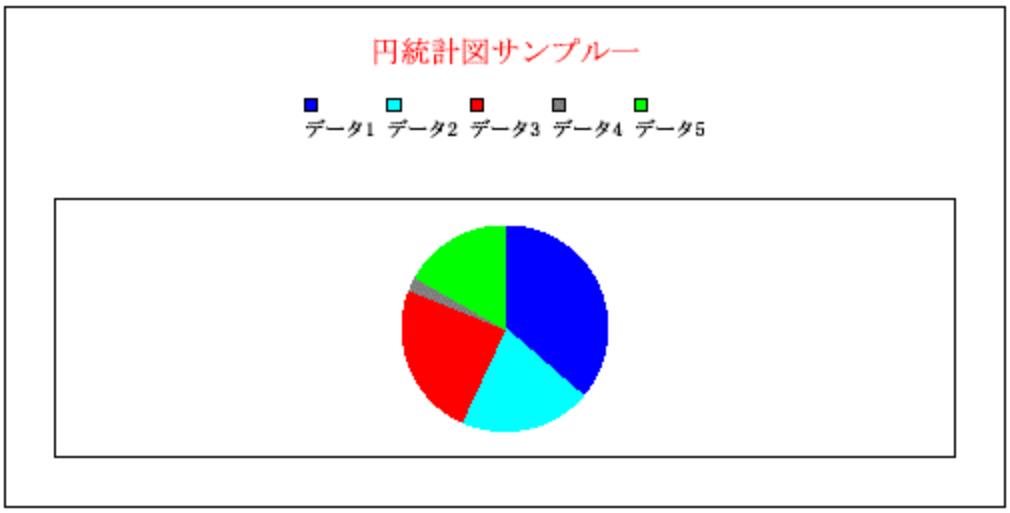
```
                return page;
        }

        double[] sample1 = {1800, 1000, 1200, -100, 800};

        double[][] sample2 = {{2000, 1000, 1200, -600, 800},
                              {2020, 800, 1300, 500, 300},
                              {1800, 1200, 1000, 900, 200}};

}
```

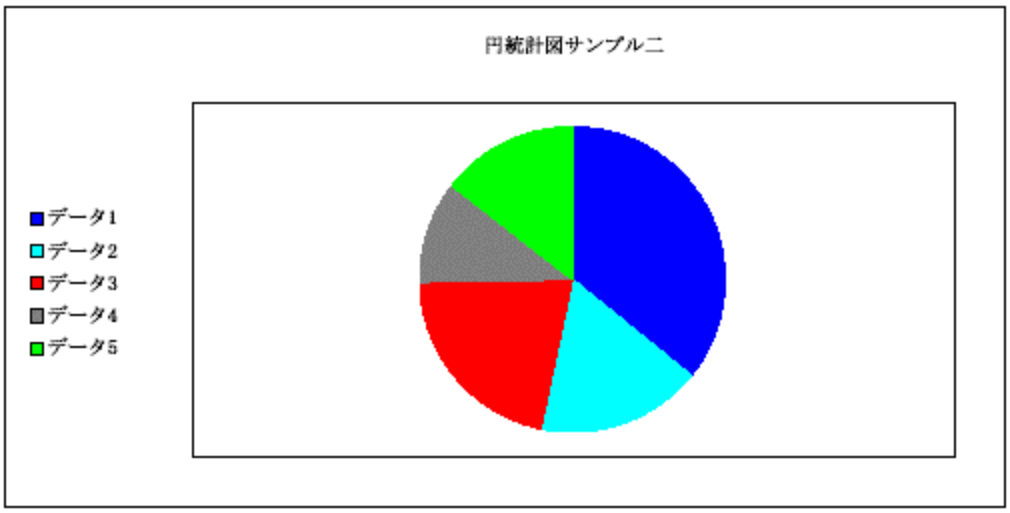
サンプルで生成された PDF ファイル



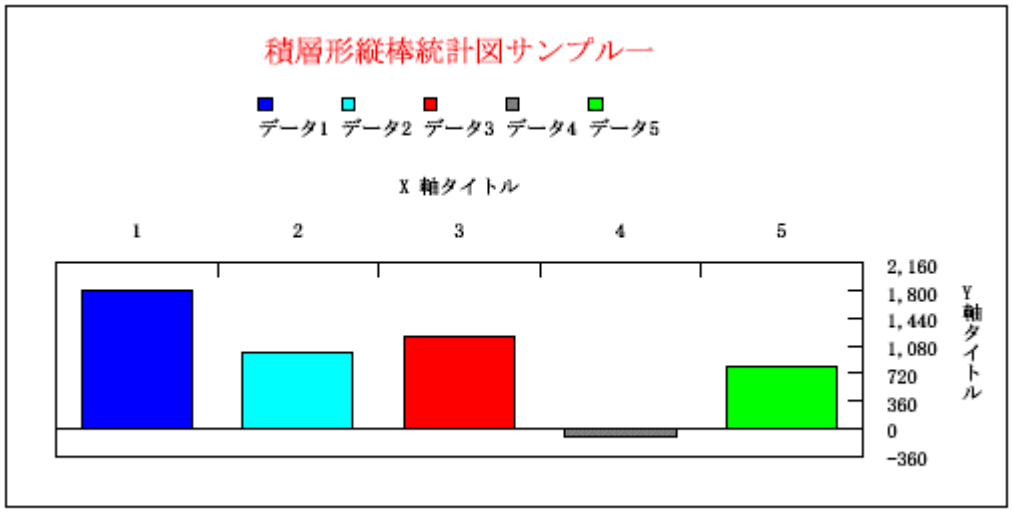


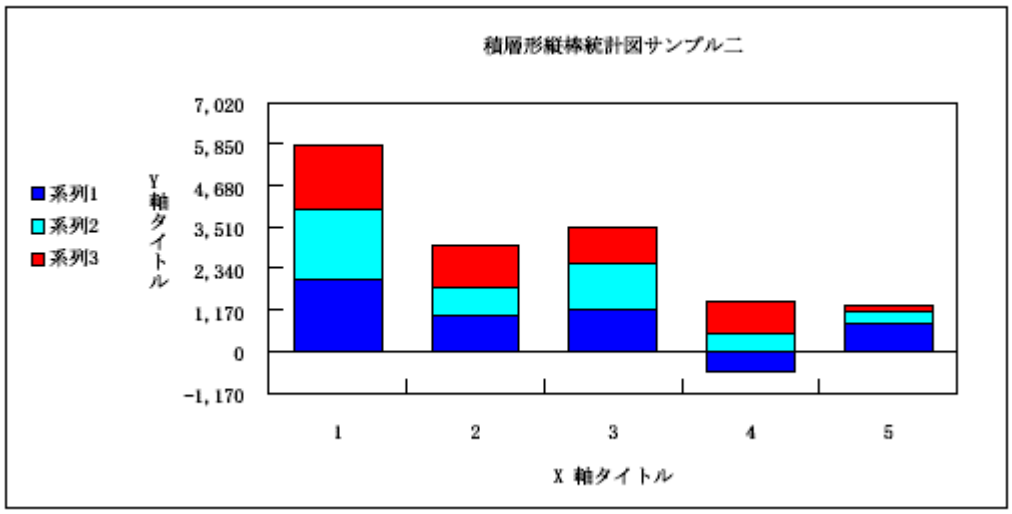
PDF ファイルのページ1



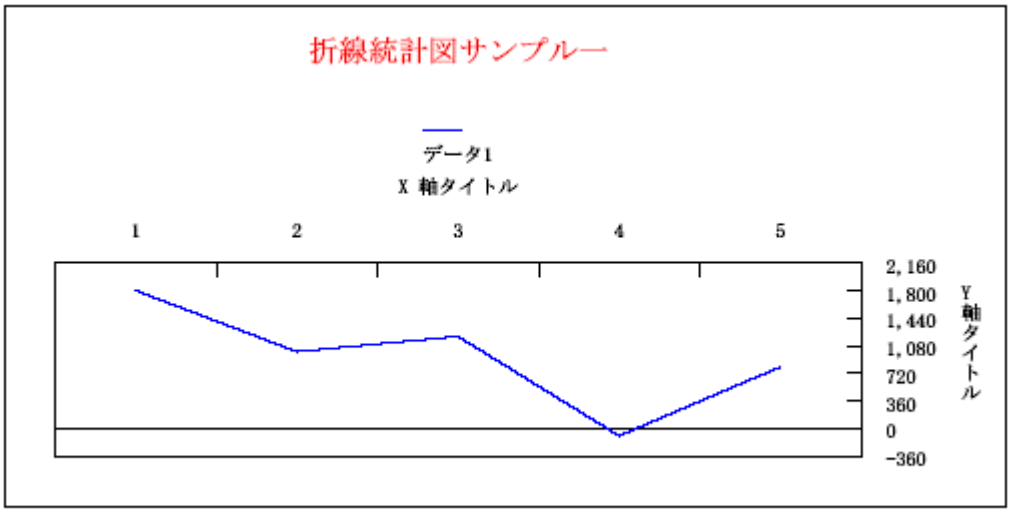


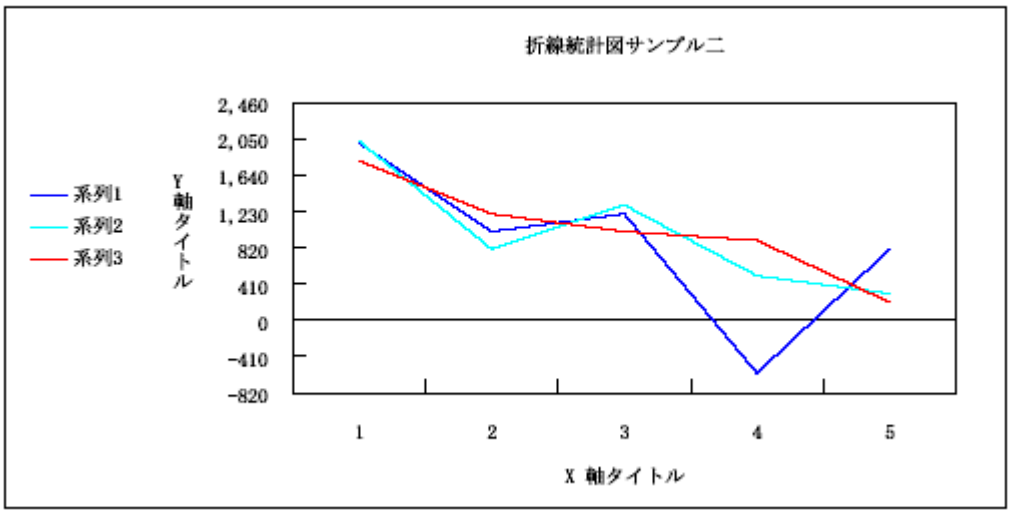
PDF ファイルのページ2



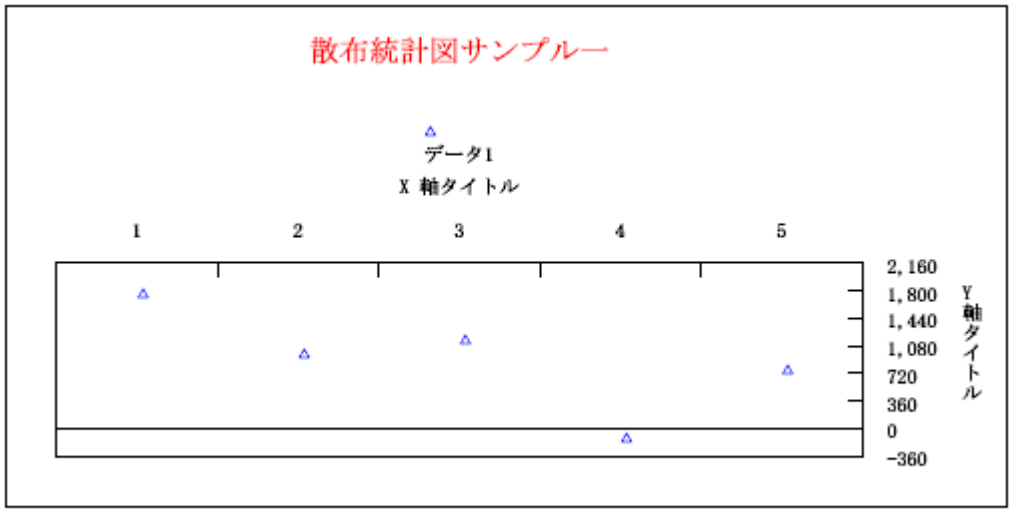


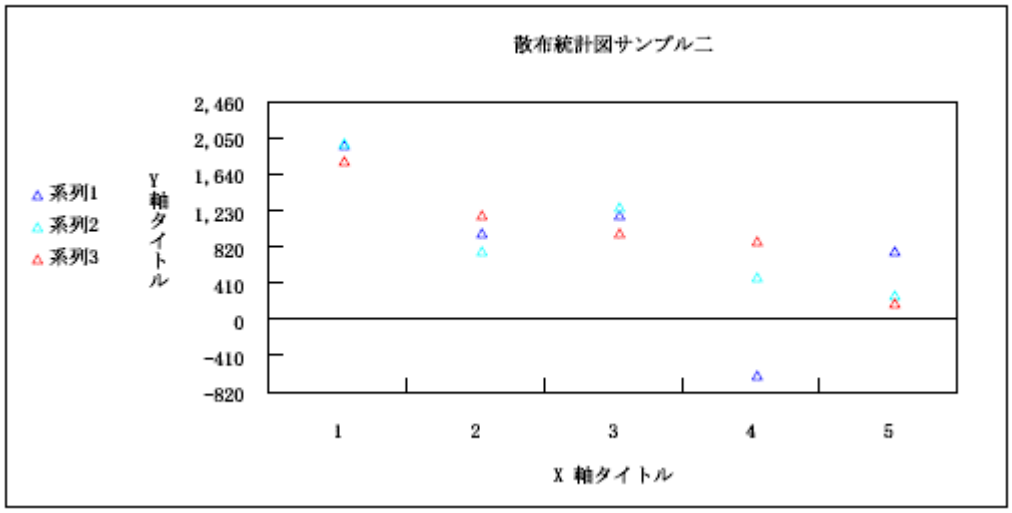
PDF ファイルのページ3





PDF ファイルのページ4



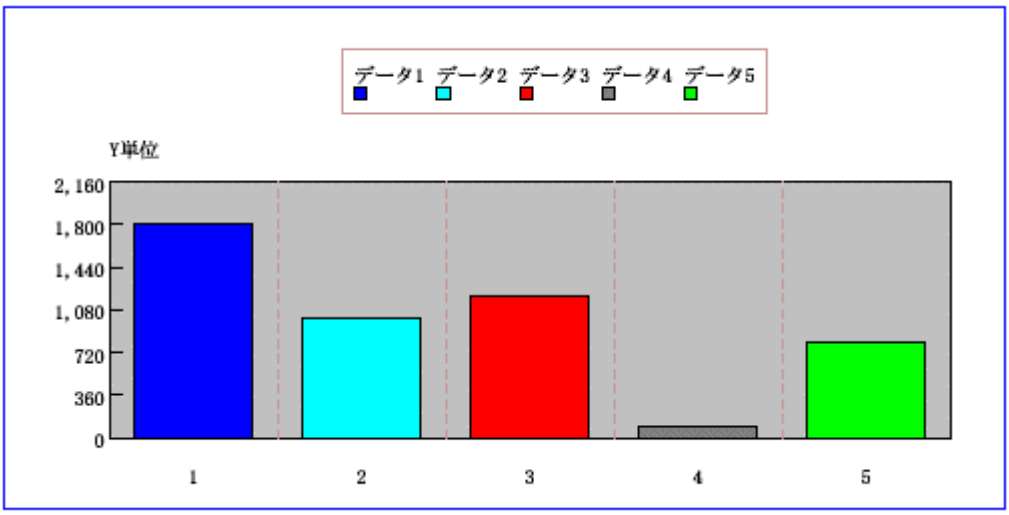


PDF ファイルのページ5

第４４章　サンプル四

```
//縦棒グラフオブジェクトを取得する
PDFChart chart
        = PDFChart.getInstance(PDFChart.VERTICAL_BAR_CHART);
//このグラフオブジェクトのデータモデルを取得する
PDFChartDataModel model = chart.getDataModel();

model.setData(sample1);
model.setXAxisUnit("X 単位");
model.setYAxisUnit("Y 単位");
model.setYFormat("#,###");

PDFChartLayout layout = new PDFChartLayout();
layout.setWidth(400);
layout.setHeight(200);
layout.setMarginAxisScaleWithPlotH(2);
layout.setMarginOuterH(20);
layout.setWidthMarkLabelH(40);
layout.setHeightMarkLabelV(40);
layout.setWidthAxisScaleLabelH(10);

//X 軸に目盛り線属性を表示に設定する
layout.setDrawXAxisGraduationLine(true);
//Y 軸に目盛り線属性を非表示に設定する
layout.setDrawYAxisGraduationLine(false);
layout.setLabelXAxisUnit(false);

//X 軸に目盛り線属性を設定する
attrGraph = new PDFGraphicAttribute();
attrGraph.setLineWidth(0.5);
attrGraph.setLineColor(new Color(200, 150, 150));
attrGraph.setDashLine(3, 3);
```

```
layout.setAttrXAxisGraduationLine(attrGraph);

//凡例の位置を設定する
layout.setPosMark(PDFChartParameter.POSITION_UPPER);
//凡例の揃え方式を設定する
layout.setAlignMark(PDFChartParameter.POSITION_CENTER);
//凡例文字の位置を設定する
layout.setPosMarkText(PDFChartParameter.POSITION_UPPER);

//グラフの枠属性を設定する
attr = new PDFGraphicAttribute();
attr.setLineColor(Color.blue);
layout.setAttrBackgroundGraphic(attr);

//グラフプロットエリア領域の背景属性を設定する
attr = new PDFGraphicAttribute();
attr.setLineColor(Color.black);
attr.setFillColor(Color.lightGray);
layout.setAttrPlotBackgroundGraphic(attr);

//凡例領域の枠を設定する
attrGraph = new PDFGraphicAttribute();
attrGraph.setLineWidth(0.8);
attrGraph.setLineColor(new Color(200, 150, 150));
chart.setAttrMarkFrame(attrGraph);

chart.setLayout(layout);
chart.setXY(100, 100);

try
{
        chart.appendto(page);
}catch(PDFChartException e)
{}

double[] sample1 = {1800, 1000, 1200, 100, 800};
```

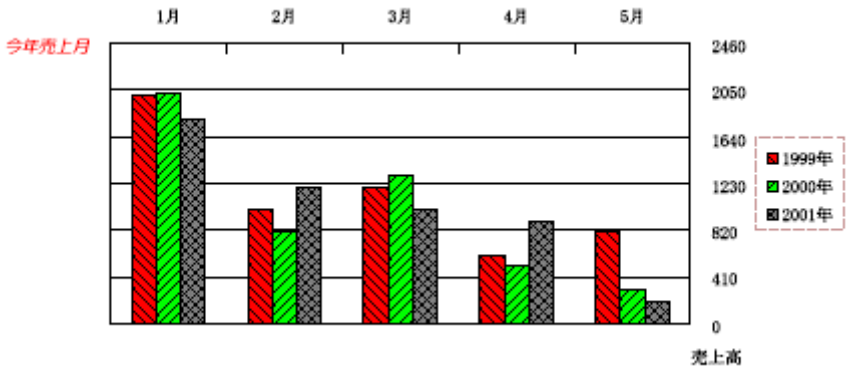

データ1 データ2 データ3 データ4 データ5
Y単位
2,160
1,800
1,440
1,080
720
360
0
1
2
3
4
5

第４５章　サンプル五

```
//縦棒グラフオブジェクトを取得する。
PDFChart chart
        = PDFChart.getInstance(PDFChart.VERTICAL_BAR_CHART);
//このグラフオブジェクトのデータモデルを取得する。
PDFChartDataModel model = chart.getDataModel();

model = chart.getDataModel();
model.setData(sample2);
//X 座標のラベルを設定する
model.setXAxisLabelText(xLabels);
model.setXAxisUnit("今年売上月");
model.setYAxisUnit("売上高");
model.setYFormat("####");
//凡例のラベルを設定する
model.setTextNote(units);

PDFChartLayout layout = new PDFChartLayout();
//座標単位の位置を設定する
layout.setPosXAxisUnit(PDFChartParameter.POSITION_LEFT);
layout.setPosYAxisUnit(PDFChartParameter.POSITION_DOWN);

layout.setHeight(180);
layout.setWidth(360);
layout.setMarginOuterH(50);
layout.setWidthPlot(200);

//グラフプロットエリア領域の背景属性を設定する
layout.setAttrPlotBackgroundGraphic(new PDFGraphicAttribute());

//X 座標単位の属性を設定する
PDFTextAttribute attrText = new PDFTextAttribute();
```

```
attrText.setFont(FontName.JP_HG_MARU_GOTHIC, Font.ITALIC);
attrText.setColor(Color.red);
layout.setLabelXAxisUnit(true);
layout.setAttrXAxisUnit(attrText);

//X 軸に目盛り線属性を非表示に設定する
        layout.setDrawXAxisGraduationLine(false);
//Y 軸に目盛り線属性を表示に設定する
layout.setDrawYAxisGraduationLine(true);

//座標ラベルの位置を設定する
layout.setPosAxisScale(PDFChartParameter.POSITION_RIGHT +
            PDFChartParameter.POSITION_UPPER);

layout.setPosMark(PDFChartParameter.POSITION_RIGHT);
layout.setAlignMark(PDFChartParameter.POSITION_CENTER);
layout.setPosMarkText(PDFChartParameter.POSITION_RIGHT);


//凡例領域の枠を設定する
attrGraph = new PDFGraphicAttribute();
attrGraph.setLineWidth(0.8);
attrGraph.setDashLine(4, 2);
attrGraph.setLineColor(new Color(200, 150, 150));
chart.setAttrMarkFrame(attrGraph);
chart.setLayout(layout);

//第一凡例の属性を設定する
PDFBarMark mark = (PDFBarMark)chart.newMarkInstance();
mark.getAttrBackground().setFillColor(Color.red);
mark.setModeBackground(PDFChartParameter.MODE1);
chart.setMark(0, mark);

//第二凡例の属性を設定する
mark = (PDFBarMark)chart.newMarkInstance();
mark.getAttrBackground().setFillColor(Color.green);
```

```
mark.setModeBackground(PDFChartParameter.MODE2);
chart.setMark(1, mark);

//第三凡例の属性を設定する
mark = (PDFBarMark)chart.newMarkInstance();
mark.getAttrBackground().setFillColor(Color.gray);
mark.setModeBackground(PDFChartParameter.MODE1 +
                PDFChartParameter.MODE2);
chart.setMark(2, mark);

chart.setXY(100, 100);
try
{
            chart.appendto(page);
}catch(PDFChartException e)
{
}

double[][] sample2 = {{2000, 1000, 1200, 600, 800},
                      {2020, 800, 1300, 500, 300},
                      {1800, 1200, 1000, 900, 200}};

String[] units = {"1999 年", "2000 年", "2001 年"};
String[] xLabels = {"1 月", "2 月", "3 月", "4 月", "5 月"};
```



付録 A フォント名

A.1 英語フォント名：

Arial
ArialBlack
BrushScriptMT
CenturyGothic
Courier
Helvetica
Times-Roman
Symbol
ZapfDingbats

A.2 日本語フォント名：

ＭＳ明朝
ＭＳＰ明朝
ＭＳゴシック
ＭＳＰゴシック
HG 正楷書体-PRO
ＤＦPOP 体
ＤＦＰPOP 体
ＤＦＰ特太ゴシック体
ＤＦ特太ゴシック体
HGｺﾞｼｯｸ E-PRO
HG 丸ｺﾞｼｯｸ M-PRO
MS UI Gothic

A3 中国語 Simplified フォント名：

MS Song
MSSong
MSHei

A4 中国語 Traditional フォント名：

MS Big5
MingLiU

A5 韓語フォント名：

HYGoThic-Medium-Acro

A6 日本語フォントの別名表

PDF Font Name	Xinca Font Name
MS-Mincho	ＭＳ明朝
	ＭＳ　明朝
	mincho
	mintyo
	ms mincho
	ms mintyo
	Mintyo
	Mincho
	MS Mintyo
	MS Mincho
	MSMincho
	MSMintyo
MS-PMincho	ＭＳＰ明朝
	ＭＳ　Ｐ明朝
	ms p mincho
	ms p mintyo
	MS P Mincho
	MS P Mintyo
	MSPMintyo
	MSPMincho
MS-Gothic	ＭＳゴシック
	ＭＳ　ゴシック
	gothic
	ms gothic
	MS Gothic
	MSGothic

PDF Font Name	Xinca Font Name
MS-PGothic	ＭＳＰゴシック
	ＭＳ　Ｐゴシック
	MS P Gothic
	MSPGothic
	ms p gothic
DFPOP1-SB-WIN-RKSJ-H	
	ＤＦＰＯＰ体
DFPPOP1-SB-WINP-RKSJ-H	DFPOP
	ＤＦＰＰＯＰ体
DFGothic-EB	DFPPOP
	ＤＦ特太ゴシック体
DFPGothic-EB-WINP-RKSJ-H	DFGothic
	ＤＦＰ特太ゴシック体
HGGothicEPRO	DFPGothic
HGMarugothicMPRO	ＨＧｺﾞｼｯｸ Ｅ-PRO
HGSeikaishotaiPRO	HG 丸ｺﾞｼｯｸ M-PRO
	HG 正楷書体-PRO

付録 B フォントサンプル

B.1 英語フォント：

ArialBlack with Font.PLAIN Style

ArialBlack with Font.BOLD Style

ArialBlack with Font.ITALIC Style

ArialBlack with Font.BOLD and Font.ITALIC Style

Arial with Font.PLAIN Style
Arial with Font.BOLD Style
Arial with Font.ITALIC Style
Arial with Font.BOLD and Font.ITALIC Style

BrushScriptMT with Font.PLAIN Style
BrushScriptMT with Font.BOLD Style
BrushScriptMT with Font.ITALIC Style
BrushScriptMT with Font.BOLD and Font.ITALIC Style

CenturyGothic with Font.PLAIN Style
CenturyGothic with Font.BOLD Style
CenturyGothic with Font.ITALIC Style
CenturyGothic with Font.BOLD and Font.ITALIC Style

Courier with Font.PLAIN Style
Courier with Font.BOLD Style
Courier with Font.ITALIC Style
Courier with Font.BOLD and Font.ITALIC Style

Helvetica with Font.PLAIN Style
Helvetica with Font.BOLD Style
Helvetica with Font.ITALIC Style
Helvetica with Font.BOLD and Font.ITALIC Style

Times-Roman with Font.PLAIN Style
Times-Roman with Font.BOLD Style
Times-Roman with Font.ITALIC Style
Times-Roman with Font.BOLD and Font.ITALIC Style

Symbol : Σψμβολ ωιτη Φοντ.ΠΛΑΙΝ Στψλε

ZapfDingbats :


B.2 日本語フォント：

ＭＳ 明朝 with Font..PLAIN Style
ＭＳ 明朝 with Font..BOLD Style
ＭＳ 明朝 with Font..ITALIC Style
ＭＳ 明朝 with Font..BOLD and Font.ITALIC Style

ＭＳＰ 明朝 with Font..PLAIN Style
ＭＳＰ 明朝 with Font..BOLD Style
ＭＳＰ 明朝 with Font..ITALIC Style
ＭＳＰ 明朝 with Font..BOLD and Font.ITALIC Style

ＭＳ ゴシック with Font..PLAIN Style
ＭＳ ゴシック with Font..BOLD Style
ＭＳ ゴシック with Font..ITALIC Style
ＭＳ ゴシック with Font..BOLD and Font.ITALIC Style

ＭＳＰ ゴシック with Font..PLAIN Style
ＭＳＰ ゴシック with Font..BOLD Style
ＭＳＰ ゴシック with Font..ITALIC Style
ＭＳＰ ゴシック with Font..BOLD and Font.ITALIC Style

HG 正楷書体- PRO with Font.PLAIN Style
HG 正楷書体- PRO with Font.BOLD Style
HG 正楷書体- PRO with Font.ITALIC Style
HG 正楷書体- PRO with Font.BOLD and Font.ITALIC Style

ＤＦＰＯＰ 体 with Font..PLAIN Style
ＤＦＰＯＰ 体 with Font..BOLD Style
ＤＦＰＯＰ 体 with Font..ITALIC Style
ＤＦＰＯＰ 体 with Font..BOLD and Font.ITALIC Style

ＤＦＰＰＯＰ 体 with Font..PLAIN Style
ＤＦＰＰＯＰ 体 with Font..BOLD Style
ＤＦＰＰＯＰ 体 with Font..ITALIC Style
ＤＦＰＰＯＰ 体 with Font..BOLD and Font.ITALIC Style

ＤＦＰ 特太ゴシック体 with Font..PLAIN Style
ＤＦＰ 特太ゴシック体 with Font..BOLD Style
ＤＦＰ 特太ゴシック体 with Font..ITALIC Style
ＤＦＰ 特太ゴシック体 with Font..BOLD and Font.ITALIC Style

ＤＦ 特太ゴシック体 with Font..PLAIN Style
ＤＦ 特太ゴシック体 with Font..BOLD Style
ＤＦ 特太ゴシック体 with Font..ITALIC Style
ＤＦ 特太ゴシック体 with Font..BOLD and Font.ITALIC Style

HG ｺﾞｼｯｸE- PRO with Font.PLAIN Style
HG ｺﾞｼｯｸE- PRO with Font.BOLD Style
HG ｺﾞｼｯｸE- PRO with Font.ITALIC Style
HG ｺﾞｼｯｸE- PRO with Font.BOLD and Font.ITALIC Style

HG 丸ｺﾞｼｯｸM- PRO with Font.PLAIN Style
HG 丸ｺﾞｼｯｸM- PRO with Font.BOLD Style
HG 丸ｺﾞｼｯｸM- PRO with Font.ITALIC Style
HG 丸ｺﾞｼｯｸM- PRO with Font.BOLD and Font.ITALIC Style

MS UI Gothic with Font.PLAIN Style
MS UI Gothic with Font.BOLD Style
MS UI Gothic with Font.ITALIC Style
MS UI Gothic with Font.BOLD and Font.ITALIC Style

付録 C　PDFTable のテーブルフォーマット

C.1 PDFTable のベースフォーマット

Table style: PDFTable.TABLE_STYLE_NONE

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_CLASSIC

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_ELEGANT

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_LIST1

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_LIST2

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_LIST3

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_LIST4

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID1

	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID2

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID3

	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID4

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID5

	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID6

	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID7

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID8

CELL01	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID9

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID10

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID11

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID12

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_SIMPLE1

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_SIMPLE2

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_SIMPLE3

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_SIMPLE4

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_SIMPLE5

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_SIMPLE6

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_SIMPLE7

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_SIMPLE8

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

C.2 PDFTable のベース仕様 ＋ PDFTable.setDashLine()

Table style: PDFTable.TABLE_STYLE_ELEGANT with dash line

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_LIST1 with dash line

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_LIST2 with dash line

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_LIST3 with dash line

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_LIST4 with dash line

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID1 with dash line

	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID2 with dash line

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID4 with dash line

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID5 with dash line

	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID7 with dash line

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID8 with dash line

CELL01	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_GRID10 with dash line

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_SIMPLE6 with dash line

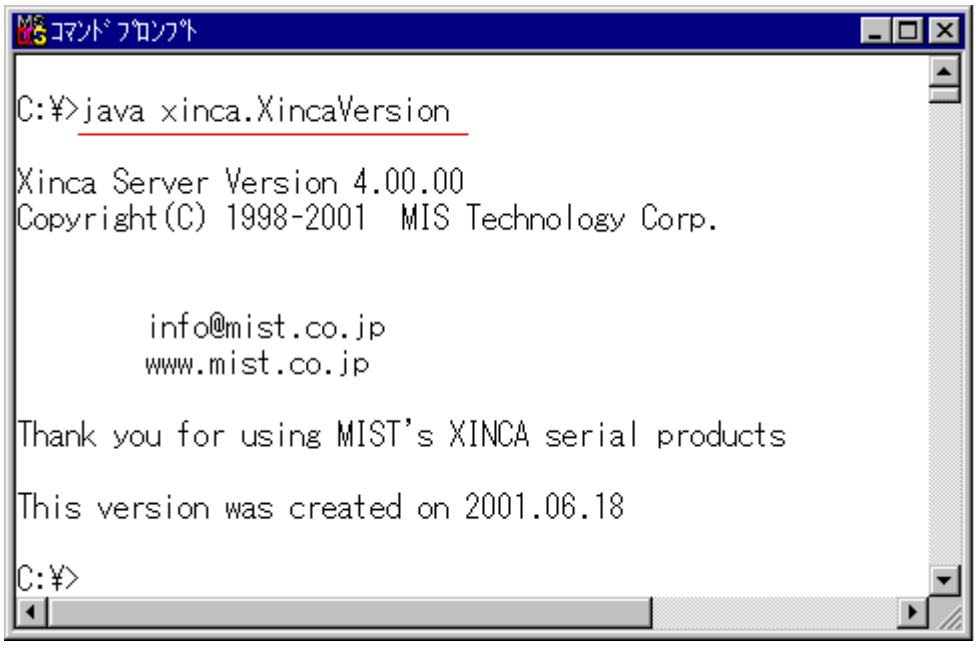
CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

Table style: PDFTable.TABLE_STYLE_SIMPLE8 with dash line

CELL00	CELL01	CELL02	CELL03	CELL04
CELL10	CELL11	CELL12	CELL13	CELL14
CELL20	CELL21	CELL22	CELL23	CELL24
CELL30	CELL31	CELL32	CELL33	CELL34
CELL40	CELL41	CELL42	CELL43	CELL44

付録 D XINCA SERVER バージョンメッセージ

XINCA SERVER パッケージに１つの JAVA アプリケーション――XincaVersion が有ります。これを使用すると XINCA SERVER のバージョンメッセージが表示できます。下図の様にコマンドプロンプトウィンドウにコマンドを入力する事でバージョンメッセージが表示できます。XINCA SERVER をご使用頂きお礼申し上げます。XINCA Server のご使用際し、万が一不備が発生した場合には弊社までご連絡頂ける様お願い申し上げます。ご連絡の際にはバージョンメッセージ、使用状況などを一緒にご提供頂ければ幸いです。



付録 E　XINCA SERVER VER4 追加機能

クラス名	
xinca.XincaVersion	XINCA バージョンの取得
xinca.PDFEncodeJA	文字コードの変換
xinca.tools.PDFTextArea	テキストレイアウトの設定
xinca.tools.PDFCheckbox	チェックボックスの作成
xinca.tools.PDFRoundRect	角丸四角形の作成
xinca.tools.PDFTextV	縦テキストの作成
xinca.tools.PDFTextAreaV	縦書テキスト領域
xinca.tools.PDFAligned	位置調整の設定
xinca.tools.PDFPoint	取得単位をＰＴに設定
xinca.tools.PDFTable	テーブルの作成
xinca.tools.PDFTableSpan	テーブルスパンの設定
xinca.tools.PDFTableCore	テーブルカラーの設定
xinca.tools.PDFTableCell	テーブルセルの設定
xinca.tools.PDFTableNullCell	テーブルのヌルセルを設定

E.2 追加メソッド

PDFFile: ＰＤＦファイル作成方法の設定

PDFFile(PrintWriter pw)
PDFFile(Writer w)
PDFFile()
Boolean isOK()
void setPageMode(int mode)

PDFPage: ＰＤＦページを非圧縮に設定

noCompress()

PDFImages: 画像圧縮の方式の設定･画像サイズの取得

PDFImages(String pathname, boolean useLZW)
PDFPoint getSize()

PDFText: テキスト装飾機能の設定（下線、二重下線、他）

void setLineWidth(double width)
void setCharSize(double cs)
void setCharSpace(double value)
void setWordSpace(double value)
void setXY(double x, double y)
void setUnderLine(Color color, double width)
void setUnderLine(double width)
int getStringWidth(String s)
int getStringWidth()
double getStringFloatWidth(String s)
double getStringFloatWidth()

PDFGraphics: グラフィック作成機能

void setWidth(double width)
void closeDashLine()
void drawDashLine(double x0, double y0, double x1, double y1)
void strokeLine(double x0, double y0, double x1, double y1)
void strokeRectangle(double x, double y, double width, double height)
void fillRectangle(double x, double y, double width, double height)
void drawArc(int x,int y,int radius,int startAngle,int arcAngle)

付録 F　XINCA SERVER VER4.1 追加機能

クラス名	
xinca.font.FontName	フォント名前を定義
xinca.util.Ruler	単位の変換

F .2 追加メソッド

PDFFile

PDFFile(OutputStream out)
void setTitle(String _title)
void setAuthor(String _author)

PDFPage

void setPageSize(double w, double h)
void setPageSize(double w, double h, String unit)

PDFImages

int　　getImageDPI()
void setMaskColor()
void setMaskColor(int red, int green, int blue)
void setMaskColor(int redFrom, int greenFrom, int blueFrom, int redTo, int greenTo, int blueTo)

付録 G　XINCA SERVER VER4.２追加機能

第五部バーコードと第六部グラフパッケージを追加。

追加クラス

クラス名	機能
xinca.barcode. BarcodeCODE39	CODE39
xinca.barcode. BarcodeCODABAR	CODEBAR
xinca.barcode. BarcodeCODE128	CODE128
xinca.barcode. BarcodeITF	ITF
xinca.barcode. BarcodeJAN	JAN

パッケージ名	機能
xinca.tools. chart	XINCA グラフ基本パッケージ