Java 用 バーコード作成ツール

Barcode.jar

説明書

Version 3.0.0

2019 年 10 月

Pao@Office

はじめに

Barcode.jar は、Java 環境下で動作する、バーコード作成ツール(クラス群)の総称です。

Barcode.jar は、次のことを念頭において開発いたしました。

１．精密なこと

単なるバーコードリーダでの検査でなく、RJS のレーザーインスペクター Model L2000 というバーコード検査機にて細かくバーコードの精度を検査しております。それにより、従来の他社のバーコード作成ツールに比べても精密なバーコードを作成することが可能です。

バーコード全体の幅を指定する方法以外にも、バーの最小幅を指定することにより、縮小することなく直接バーコードを描画し、より精度の高いバーコードを作成することが可能です。

２．使いやすいこと

わかりやすいクラスのインタフェイスになっております。

後の使用例でも書かれておりますが、2～3 Step のロジックでバーコードの印刷等を行うことができます。

３．軽いこと

何と言っても軽さが命です。Barcode.jar を利用してバーコード作成を行う場合、Barcode.jar 自体がシステムに与える負荷は微小です。ほんの数ＭＢのメモリ上で動作します。

４．汎用性があること(用途が様々)

バーコードのアウトプットは、Graphics2D オブジェクト、または、画像ファイルです。

従って、皆様がバーコードを作成するアプリケーションから、様々な用途で利用することが可能になっています。

Barcode.jar をご利用していただく皆さんが、Java 環境でのバーコードの生成(印刷)プログラムの作成作業に、楽しさを感じていただければ幸いです。

2016 年 4 月　作者

目次

1. Barcode.jar の動作環境・インストール方法

1-1. 動作環境

OS	JDK1.5 以上が正常に動作するものである事
動作に必要なメモリ	JDK1.5 以上が正常に動作するために 必要な容量
画面解像度	特に制限なし
開発環境	eclipse をご利用いただくとサンプルプログラムをお試しし易くなっております。

1-2. インストール方法

まず、以下の URL より試用版をダウンロードしてください、

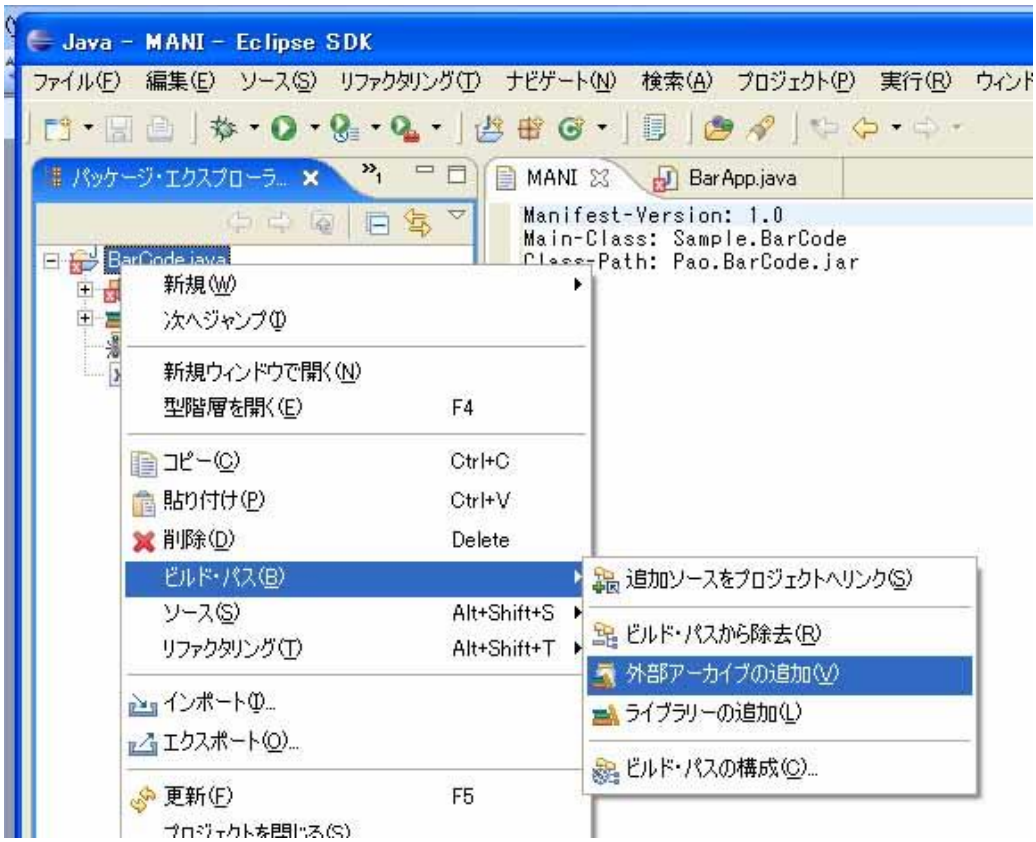
http://www.pao.ac/barcode.jar/#download

次にダウンロードしたファイルを任意の場所に解凍してください。

フォルダ内に Pao.Barcode.jar がございます。

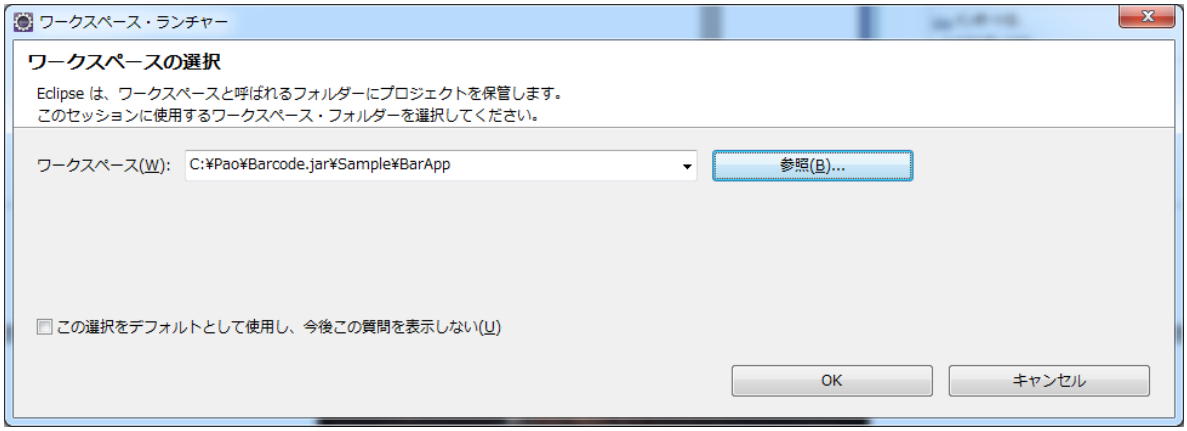
eclipse を使用する場合、「ビルドパス－外部アーカイブの追加」で、Pao.Barcode.jar を追加してください。

いくつかのサンプルプログラムが、解凍フォルダ内の Sample フォルダに格納されております。次のページでサンプルのご利用方法について記述があります。



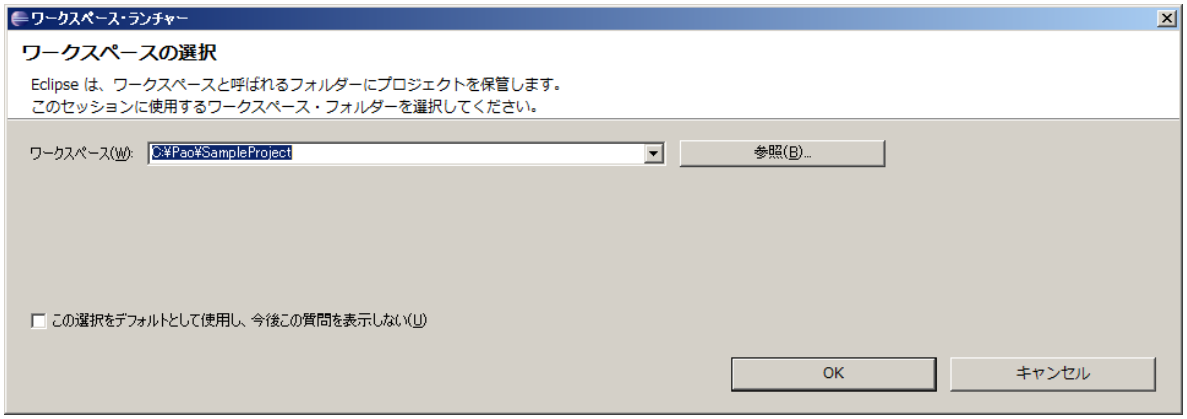
1-3. サンプルプログラムの使用方法

Eclipse 3.7(Indiego) 他、いくつかのバージョンをお使いの場合は、
Eclipse 起動時のワークスペースにサンプルフォルダを指定していただければ、結構です。

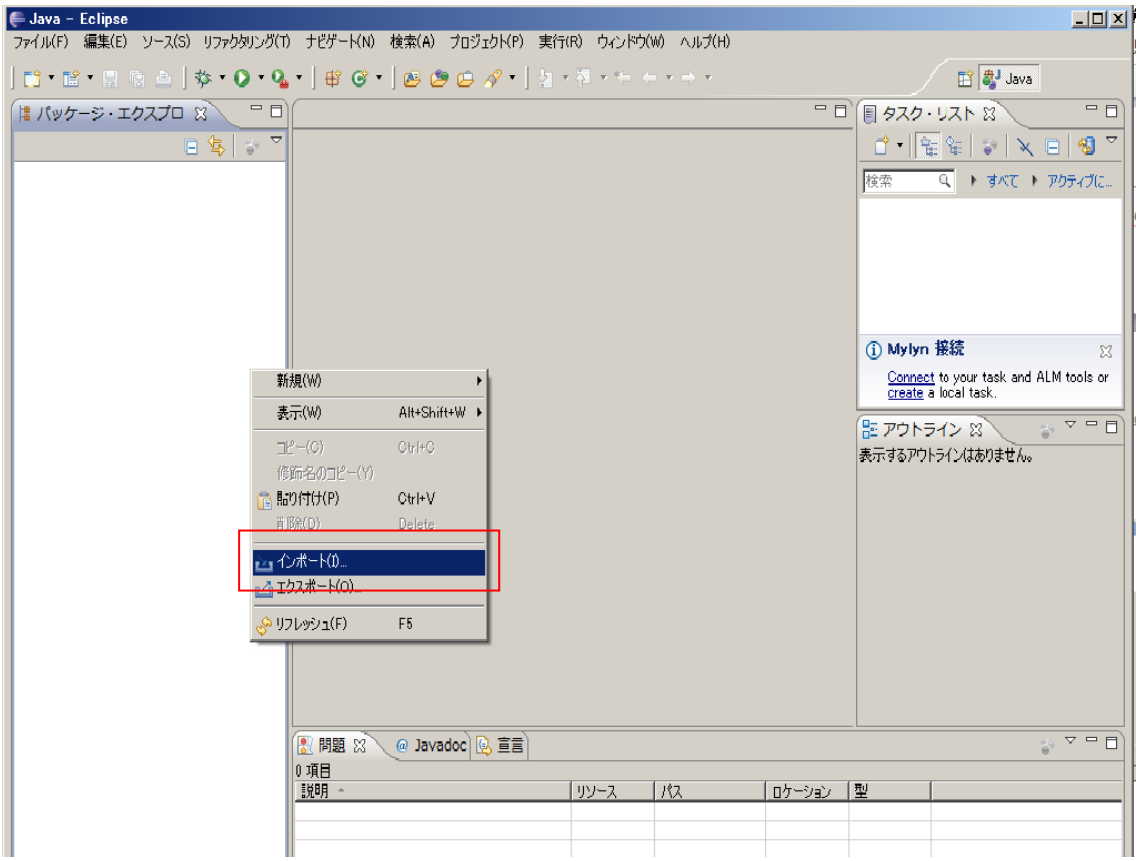


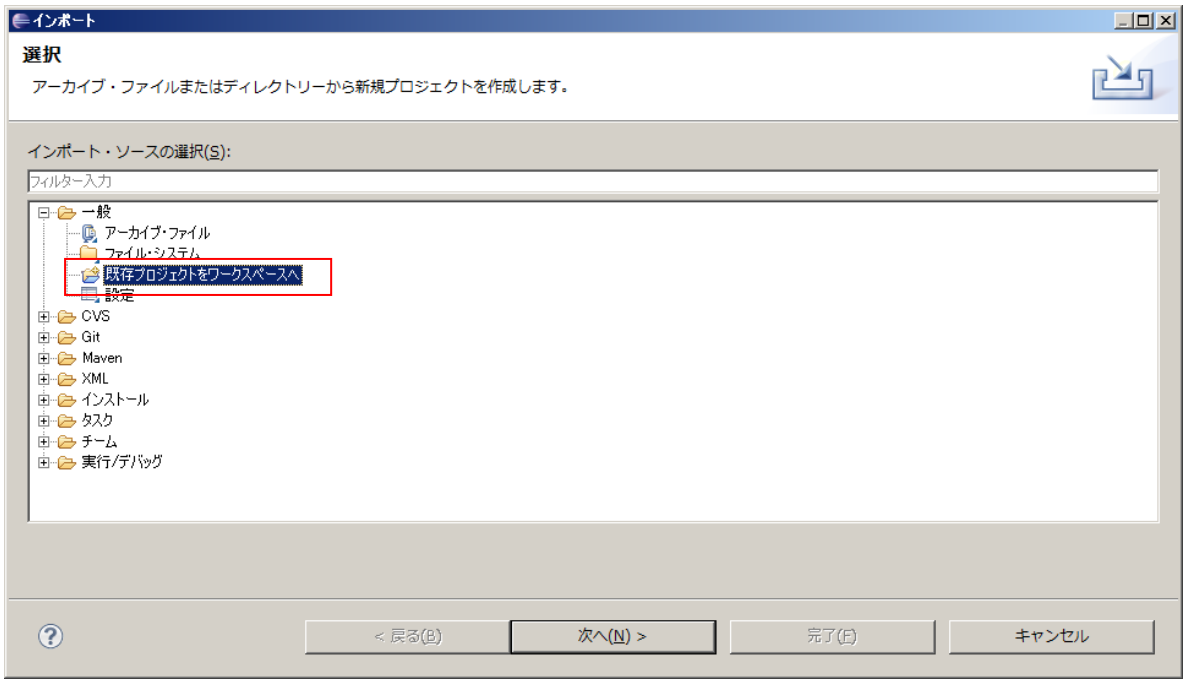
以降は、それではうまくサンプルワークスペースを開けない場合の操作の説明です。

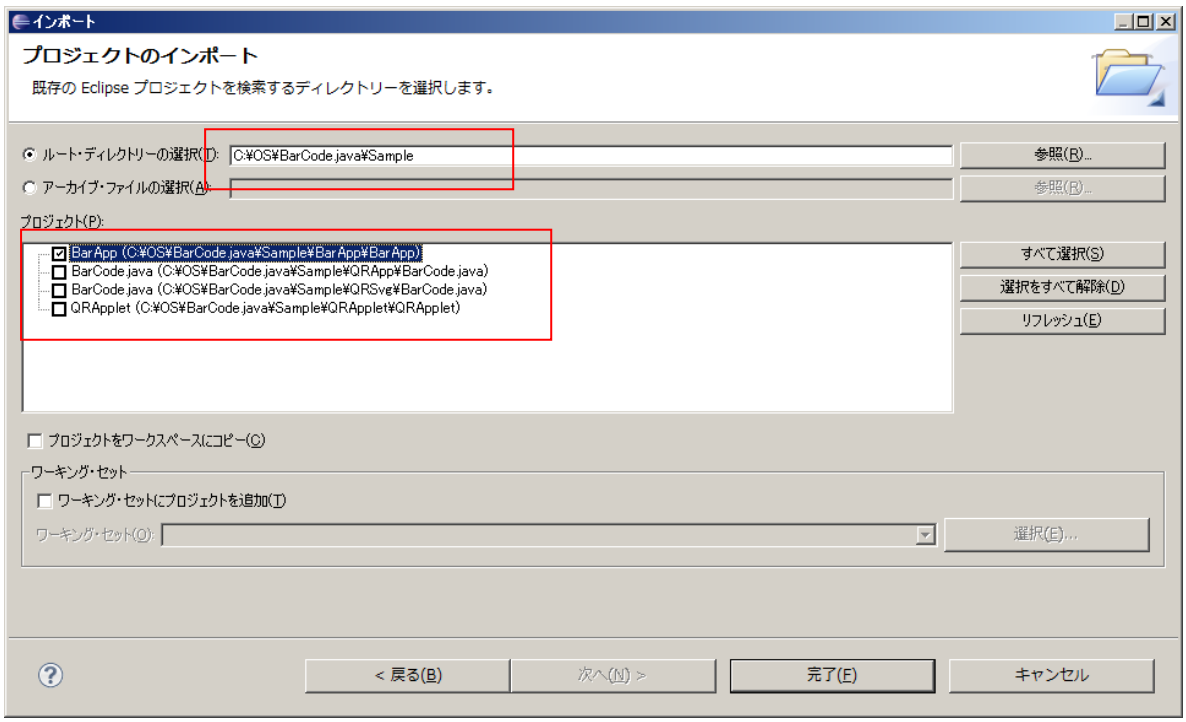
（1） 新規のワークスペースを作成してください。



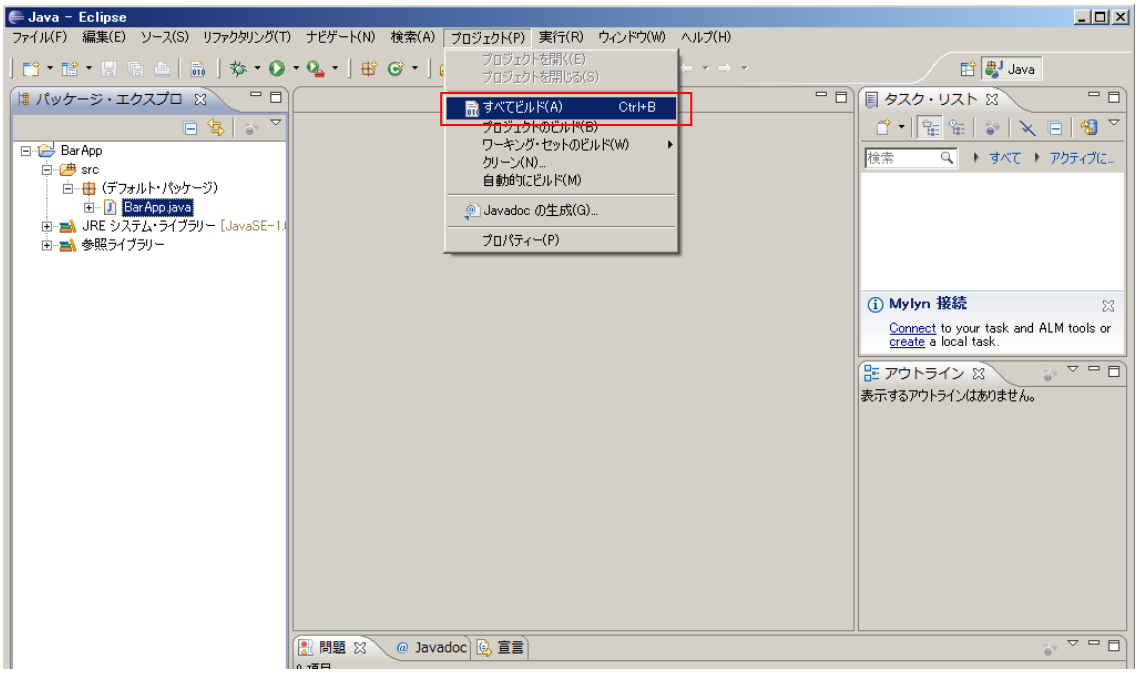
（２） サンプルプロジェクトをインポートしてください。
（こちらの例では、Sample¥BarApp を使用します。）



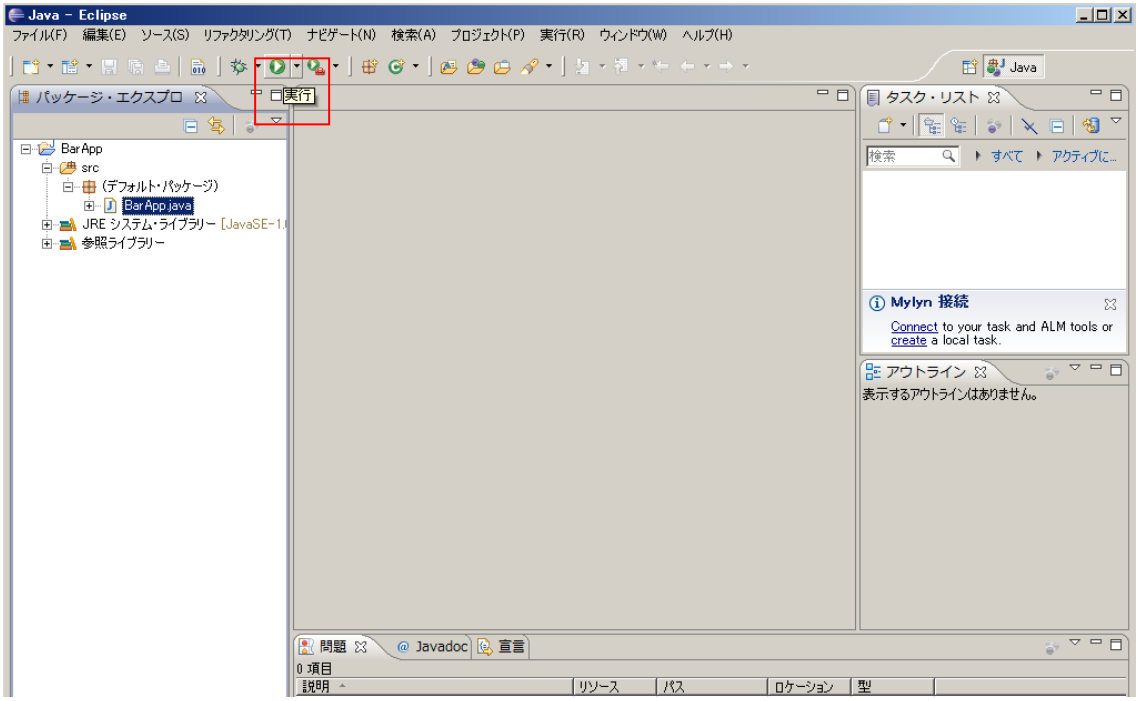



インポート
プロジェクトのインポート
既存の Eclipse プロジェクトを検索するディレクトリーを選択します。
ルート・ディレクトリーの選択(T):
C:¥OS¥BarCode.java¥Sample
参照(R)...
アーカイブ・ファイルの選択(A):
参照(R)...
プロジェクト(P):
BarApp (C:¥OS¥BarCode.java¥Sample¥BarApp¥BarApp)
BarCode.java (C:¥OS¥BarCode.java¥Sample¥QRApp¥BarCode.java)
BarCode.java (C:¥OS¥BarCode.java¥Sample¥QRSvg¥BarCode.java)
QRApplet (C:¥OS¥BarCode.java¥Sample¥QRApplet¥QRApplet)
すべて選択(S)
選択をすべて解除(D)
リフレッシュ(E)
プロジェクトをワークスペースにコピー(O)
ワーキング・セット
ワーキング・セットにプロジェクトを追加(T)
ワーキング・セット(O):
選択(E)...
< 戻る(B)
次へ(N) >
完了(F)
キャンセル

（３） ビルドを行います。


Java - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H)
プロジェクトを開く(E)
プロジェクトを閉じる(S)
すべてビルド(A) Ctrl+B
プロジェクトのビルド(B)
ワーキング・セットのビルド(W)
クリーン(N)...
自動的にビルド(M)
Javadoc の生成(G)...
プロパティー(P)
パッケージ・エクスプロ
BarApp
src
(デフォルト・パッケージ)
BarApp.java
JRE システム・ライブラリー
参照ライブラリー
タスク・リスト
Mylyn 接続
Connect to your task and ALM tools or create a local task.
アウトライン
表示するアウトラインはありません。
問題
Javadoc
宣言

（4）「実行」するとサンプルプログラムが起動します。



2. Barcode.jar の機能

2-1. 機能概要

Barcode.jar は、以下のバーコードの作成が可能です。

(1) JAN-13(EAN-13)
(2) JAN-8(EAN-8)
(3) UPC-A
(4) UPC-E
(5) ITF(インターリーブド 2 of 5)
(6) Matrix 2 of 5
(7) NEC 2 of 5 (Coop 2 of 5)
(8) NW-7(CODA-BAR)
(9) CODE39
(10) CODE93
(11) CODE128
(12) GS1-128 (UCC/EAN-128)
- コンビニ向け標準料金代理収納用バーコード
- 医療用 医薬品等のバーコード
- 医療用 医療材料等のバーコード
- 食肉標準物流バーコード「基本バーコード」

(13) 郵便カスタマバーコード
(14) GS1 Databar 標準型 (RSS-14) … **ver 3.0 において追加**
(15) GS1 Databar 限定型 (RSS Limited) … **ver 3.0 において追加**
(16) GS1 Databar 拡張型 (RSS Expanded) … **ver 3.0 において追加**
(17) QR コード
(18) 標準料金代理収納用バーコード(コンビニバーコード)
(19) DataMatrix (GS1 DataMatrix)
(20) PDF417

※郵便カスタマバーコード・GS1 Databar・二次元バーコード(QR / DataMatrix / PDF417)以外は、以降総称して「一次元バーコード」と呼びます。

Barcode.jar では、上記の各バーコードを作成するために、バーコードの種類ごとに全て別々のクラスとして利用することが可能となっております。
Barcode.jar の各バーコード作成クラスは２つのコンストラクタを用意しております。一つ目は、クラスのコンストラクタで java.awt.Graphics2D オブジェクトを受け取り、Graphics2D オブジェクトに対してバーコードを描画します。二つ目は、クラスのコンストラクタで保存する画像ファイルパスを受け取り画像ファイルにバーコードを出力します。

※1 画像ファイルの形式は、filename の拡張子で判断します。
※2 png / gif の背景は透明になります。他の画像形式の背景は白になります。
※3 画像解像度(dpi)の指定は行えません。
※4 描画単位は全て pixel になります。

また、QR コードは、SVG 形式のファイルへの出力が可能です。

2-1-1. 一次元バーコードの種類

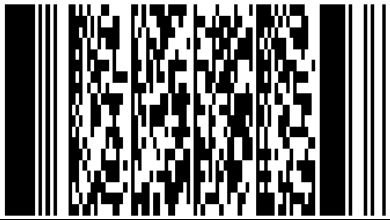
次の種類のバーコードを出力できます。

1．JAN-13(EAN-13)	2．JAN-8(EAN-8)
数字 13 桁。12 桁指定時、13 桁目のチェックディジットを自動計算付与。  	数字 8 桁。7 桁指定時、8 桁目のチェックディジットを自動計算付与。 
3．UPC-A	**4．UPC-E**
数字 12 桁。11 桁指定時、12 桁目のチェックディジットを自動計算付与。  	数字 7 桁。6 桁指定時、7 桁目のチェックディジットを自動計算付与。 
5．ITF(インターリーブド 2 of 5)	**6．Matrix 2 of 5**
使用可能文字：数字  	使用可能文字：数字 
7．NEC 2 of 5 (Coop 2 of 5)	**8．NW-7(CODA-BAR)**
使用可能文字：数字  	使用可能文字： ABCD.+:/$-0123456789  1 文字目 ABC いずれか入力した文字がスタート・ストップキャラクタとなる。入力がない場合 C が既定値。 **パラメータ：** スタート・ストップキャラクタ表示／非表示

９．CODE39	１０．CODE93
使用可能文字： 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ-. *$/+%  **パラメータ：** スタート・ストップキャラクタ表示／非表示	使用可能文字： 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ-. *$/+% 
１１．CODE128	**１２．GS1 128 (UCC/EAN-128)**
多くの半角英数文字記号・エスケープシーケンスが使用可能。 CODE A/B/C により出力文字・パターンを切り替えることが可能。  **パラメータ：** どのコードパターンを使用するか。 AUTO / CODE_A / CODE_B / CODE_C 既定値は、AUTO。AUTO は、数時 4 文字続けば CODE A にコードチェンジするなど、バーコードが一番小さくなるように自動調整する。	- コンビニ向け標準料金代理収納用バーコード - 医療用 医薬品等のバーコード - 医療用 医療材料等のバーコード - 食肉標準物流バーコード「基本バーコード」  コンビニバーコード 入力： {FNC1}9191234500000000000000452087500401310029500 6 
１３．郵便カスタマバーコード	
郵便番号＋住所の英数字部分のみ入力 例：27500263-29-2-401 **パラメータ**：8～11.5 ポイント(大きさ) 	

2-1-2. 二次元バーコードの種類

次の種類のバーコードを出力できます。

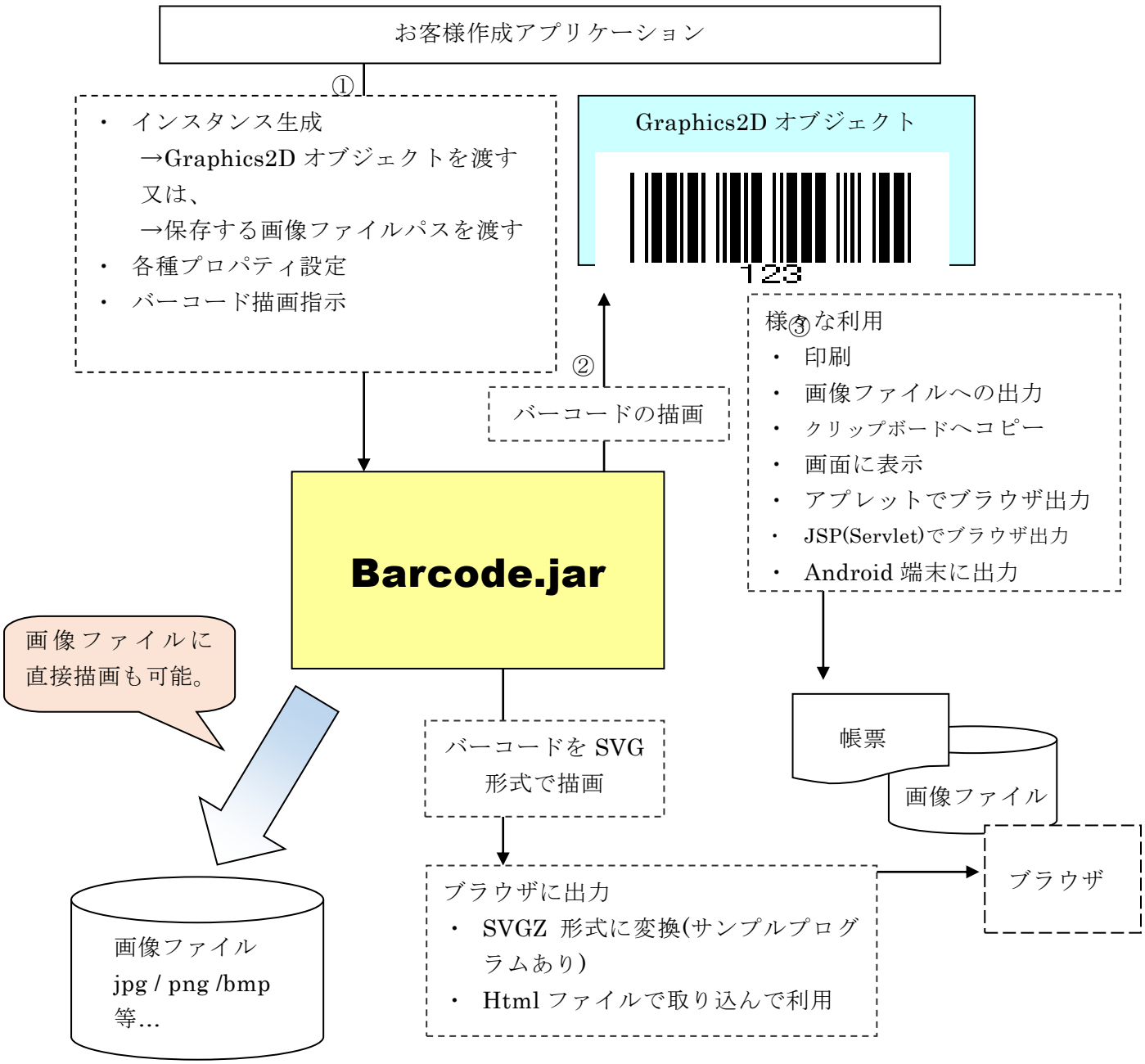
１．QR コード
パラメータ：(既定値はアンダーバー) (1) エラー訂正レベル：L / M / Q / H (2) 文字種：数字 / 大文字英数字 / 8 ビットバイトデータ(漢字含む) (3) バージョン： 1～40 5 (4) 全角文字エンコード： “shift-jis” / “utf8” /etc.. 
２．DataMatrix (GS1 DataMatrix)
パラメータ：(既定値はアンダーバー) (1) シンボルコードサイズ：自動 / 10x10～144x144 / 8x18～16x48 等 (2) 全角文字エンコード： “utf8” / “shift-jis” / etc.. 
３．PDF417
パラメータ：(既定値はアンダーバー) (1) エラー訂正レベル自動決定：する / しない (2) エラー訂正レベル：1～8 2 (3) アスペクト比：0.5 / 1.0 / 2.0 等 (4) 列・行数決定方法： アスペクト比より自動決定 / 列数指定 / 行数指定 / 列数行数指定 (5) 指定列数：1～30 5 (6) 指定行数：3～90 5 (7) 全角文字エンコード： “shift-jis” / “utf8” /etc.. 

2-1-3. GS1 Databar (RSS) の種類

次の種類のバーコードを出力できます。

<table>
<tr><th colspan="2">１．GS1 データバーOmni-directional / 表示桁数：数字 14 桁(GTIN)</th></tr>
<tr><td>Omni-directional(標準型)
 
高さが狭い Truncated(カット型)という種類もございますが、この標準型の高さを調整して出力してください。</td><td>Stacked（2 層型）
  </td></tr>
<tr><td>Stacked Omni-directional
(標準 2 層型)
  </td><td></td></tr>
<tr><th colspan="2">２．GS1 データバーLimited / 表示桁数：数字 14 桁(GTIN)</th></tr>
<tr><td colspan="2">Limited(限定型)
  </td></tr>
<tr><th colspan="2">３．GS1 データバーExpanded / 表示桁数：最大数字 74 桁または英字 41 文字</th></tr>
<tr><td>Expanded(拡張一層型)
  </td><td>Stacked Expanded(拡張多層型)
  </td></tr>
</table>

2-1-4.概要図


お客様作成アプリケーション
①
・ インスタンス生成
→Graphics2D オブジェクトを渡す
又は、
→保存する画像ファイルパスを渡す
・ 各種プロパティ設定
・ バーコード描画指示
Graphics2D オブジェクト
123
②
バーコードの描画
様々な利用
③
・ 印刷
・ 画像ファイルへの出力
・ クリップボードへコピー
・ 画面に表示
・ アプレットでブラウザ出力
・ JSP(Servlet)でブラウザ出力
・ Android 端末に出力
Barcode.jar
画像ファイルに
直接描画も可能。
バーコードを SVG
形式で描画
帳票
画像ファイル
ブラウザ
ブラウザに出力
・ SVGZ 形式に変換(サンプルプログラムあり)
・ Html ファイルで取り込んで利用
画像ファイル
jpg / png /bmp
等…

2-2. 一次元バーコード作成クラスの機能

Barcode.jar の各一次元バーコード作成クラスは、以下の機能を有します。

(1) バーコードの描画

コード、始点(左上の X/Y 座標)とバーコードの高さ・幅を指定してバーコードを描画します。幅の代わりに、バーコードの線幅の最小値を指定して描画することも可能です。その場合、より高い精度のバーコードが作成できますが、最終的に描画される幅の調整が必要になります。

もう一つの描画方法として、幅を指定していただきその中に納まる一番広い幅で、かつ、ドットにのった精度の高いバーコードを描画することも可能です。それぞれ順番に、draw / drawDelicate / drawDirect メソッドをご用意しております。

座標単位は、ver 2.0 で pixel に加え、mm / inch / point(1/72 インチ)の指定が可能になりました。

少し特殊な GS1-128(UCC/EAN128)において、AI(アプリケーション識別子)挿入方法は 2 通りございます。

(1) 可変長項目(データブロック)の後の AI には、FNC1 を挿入
⇒これまで通り"{FNC1}"を付ける。例 : "{FNC1}21"のようにコードを指定

(2) 固定長項目(データブロック)の後の AI には、固定長のため目印の FNC1 は不要
⇒新しく追加した"{AI}"を付ける。例 : "{AI}21" のようにコードを指定

"{AI}"を指定して FNC1 を挿入しない場合も、カッコ()付コード文字は出力されます。例えば入力コードに"{AI}21"を指定した場合、添え字には(21)と出力されます。

例) (01)04512345670016(21)1 ⇒(01)の前には FNC1 を挿入し(21)の前には挿入しない。
コード指定方法 → "{FNC1}0104512345670016{AI}211"

(2) 添字の描画

バーコードの下にコードの文字列自体を描画します(既定値)。プロパティの設定で描画をしないようにすることも可能です。

添字を、コードを意味するバーの位置に描画する(既定値)か、バーコード全体の幅に均等割付するかを指定することが可能です。

※ JAN(EAN)コードの場合、既定値の状態では商品コードのバーコードのような描画を行い、均等割付にすると書籍コードのバーコードのような描画を行います。

添字のフォントをプロパティで指定することも可能です。

CODE39／NW-7(CODABAR) のみスタート・ストップキャラクタを印字するかどうかをプロパティで指定することが可能です。既定値は印字しません。

(3) 回転描画

プロパティの設定により、バーコードの左上始点座標を中心に、
90 度／180 度／270 度　回転して描画することが可能です。既定値は 0 度です。

(4) 黒バー調整

プロパティの設定により、描画する黒バー幅をドット単位で微細調整できます。
既定値は、0 ドットです。
例えば、このプロパティに-1 を指定すると、バーコード内全ての黒バーの幅が 1 ドットずつ細くなります。
解像度（DPI）も指定できるプロパティを設けましたので
合わせて指定してください。
プリンタにより、微調整が必要な場合にこの機能を使用してください。

※この機能は。drawDirect / drawDelicate メソッドには有効ですが、
draw メソッドには無効ですのでご注意ください。

(5) 画像ファイルへのバーコード出力

バーコードを画像に保存する場合は、コンストラクタの引数に画像ファイル名(String filename)を指定してください。
バーコードの描画は、draw / drawDirect / drawDelicate メソッドをそのままお使いください。
ただし、画像ファイル名(String filename)を引数に持つ draw / drawDirect / drawDelicate メソッドもご用意してございます。バーコード出力ファイルを随時変更する場合は、こちらのオーバーロードメソッドをご利用ください。

※1 画像ファイルの形式は、filename の拡張子で判断します。
※2 png / gif の背景は透明になります。他の画像形式の背景は白になります。
※3 画像解像度(dpi)の指定は行えません。
※4 描画単位は全て pixel になります。
(mm 等の単位での画像出力は行えません。)

保存するバーコード画像の周りの余白部分の大きさを pixel 単位で指定することも可能です。
バーコード両端のクワイエットゾーン等にもお使いいただけます。
以下のプロパティをご利用ください。

setImgMargin(int marginPixel) / int getImgMargin()

2-3. コンビニ向け標準料金代理収納用バーコード(コンビニバーコード)

GS1-128(UCC/EAN128)バーコード作成クラスにコンビニバーコード描画メソッドを用意しました。コンビニバーコードメソッドは、以下の機能を有します。

コンビニバーコードは、GS1_128 クラスに含まれます。Barcode.jar のコンビニバーコードは、以下の機能を有します。

(1) コンビニバーコードの描画

コードと、始点(左上の X/Y 座標)及び、コンビニバーコードの高さを mm(ミリ)単位で指定し、コンビニバーコードを描画します。

コンビニバーコードの幅は、プリンタの解像度(dpi)により、以下の表の通り、自動的に決まります。

解像度	モジュール幅		バーコード部の幅
	ドット	mm	
300dpi	2	0.169	48.67mm
400dpi	3	0.190	54.72mm
480dpi	3	0.158	45.50mm
600dpi	4	0.169	48.67mm
300dpi の倍数	2 の倍数	0.169	48.67mm

ver 2.0 で、コンビニバーコードの幅を指定できるようにいたしました。ガイドラインはあくまでガイドですので、お客様が自由にバーコードの幅をご指定頂いて問題ございません。(単位:mm)

アプリケーション識別子(AI)の前に"{FNC1}"という文字列を指定することでGS1-128(UCC/EAN128)のコンビニバーコード作成を可能としています。

例) (91)912345　・・・() 付きがアプリケーション識別子(AI)

　　コード指定方法→「{FNC1}91912345」

(2) 添字の描画

コンビニバーコードの下にコードの文字列自体を描画します。この添字(コード文字列)は、ガイドラインに従い、左詰で以下のように描画します。

(91)912345-1234567890123456789211

020331-0-123456-2

なお、これは、コードで以下のように指定された場合です。

「{FNC1}9191234512345678901234567892110203310123456 2」

(3) 画像ファイルへのコンビニバーコード出力

コンビニバーコードを画像に保存する場合は、コンストラクタの引数に画像ファイル名(String filename)を指定してください。

バーコードの描画は、draw / drawDirect / drawDelicate メソッドをそのままお使いください。

ただし、画像ファイル名(String filename)を引数に持つ draw / drawDirect / drawDelicate メソッドもご用意してございます。コンビニバーコード出力ファイルを随時変更する場合は、こちらのオーバーロードメソッドをご利用ください。

※1 画像ファイルの形式は、filename の拡張子で判断します。

※2 png / gif の背景は透明になります。他の画像形式の背景は白になります。

※3 画像解像度(dpi)の指定は行えません。

※4 描画単位は全て pixel になります。
(mm 等の単位での画像出力は行えません。)

保存するコンビニバーコード画像の周りの余白部分の大きさを pixel 単位で指定することも可能です。

コンビニバーコード両端のクワイエットゾーン等にもお使いいただけます。

以下のプロパティをご利用ください。

setImgMargin(int marginPixel) / int getImgMargin()

2-4. 郵便カスタマバーコード作成クラスの機能

Barcode.jar の郵便カスタマバーコード作成クラスは、以下の機能を有します。

(1) 郵便カスタマバーコードの描画

コード、始点(左上の X/Y 座標)と大きさとしてポイント(8～30)を指定して郵便カスタマバーコードを描画します。

コードの表記は・・・

[郵便番号の数字部分 7 桁]+[郵便番号では不明部分の住所の英数字を「－」区切り]で、指定してください。

例）〒116-0013 東京都荒川区西日暮里五丁目 37 番 5 号スタートアップオフィスＡ－207 号室

コード指定方法→「11600135-37-5-A-207」

※詳しくは、旧郵政省の web ページにマニュアルがございますのでご覧になってください。

(2) 回転描画

プロパティの設定により、郵便カスタマバーコードの左上始点座標を中心に、90 度／180 度／270 度　回転して描画することが可能です。

既定値は、0 度です。

(3) 画像ファイルへの郵便カスタマバーコード出力

郵便カスタマバーコードを画像に保存する場合は、コンストラクタの引数に画像ファイル名(String filename)を指定してください。

バーコードの描画は、draw / drawDirect / drawDelicate メソッドをそのままお使いください。

ただし、画像ファイル名(String filename)を引数に持つ draw / drawDirect / drawDelicate メソッドもご用意してございます。郵便カスタマバーコード出力ファイルを随時変更する場合は、こちらのオーバーロードメソッドをご利用ください。

※1 画像ファイルの形式は、filename の拡張子で判断します。

※2 png / gif の背景は透明になります。他の画像形式の背景は白になります。

※3 画像解像度(dpi)の指定は行えません。

※4 描画単位は全て pixel になります。
(mm 等の単位での画像出力は行えません。)

保存する郵便カスタマバーコード画像の周りの余白部分の大きさを pixel 単位で指定することも可能です。以下のプロパティをご利用ください。

setImgMargin(int marginPixel)　/　int getImgMargin()

2-5. QR コード作成クラスの機能

Barcode.jar の QR コード作成クラスは、以下の機能を有します。

(1) QR コードの描画

コード、始点(左上の X/Y 座標)とバーコードを描画する最小値を指定して描画することが可能です。

プロパティで以下の項目を指定することが必要です。

- バージョン(1～40)　既定値：5
- エラー訂正レベル(L,M,Q,H)　既定値：M
- エンコードモード　既定値：Z
  (N：数字モード A：英数字モード　Z：その他(漢字等、8bit byte モード)
- 全角エンコーディング（“shift-jis”,”utf-8”,etc..）既定値：shift-jis

エンコードモードに漢字モードがありませんが、漢字の入力も「その他:8bit byte モード」を指定してください。”N”/”A”以外の文字であればなんでも OK です。※決めかねるときは、”Z”を使用してください。

(2) 画像ファイルへの QR コード出力

QR コードを画像に保存する場合は、コンストラクタの引数に画像ファイル名(String filename)を指定してください。

バーコードの描画は、draw / drawDirect / drawDelicate メソッドをそのままお使いください。

ただし、画像ファイル名(String filename)を引数に持つ draw / drawDirect / drawDelicate メソッドもご用意してございます。QR コード出力ファイルを随時変更する場合は、こちらのオーバーロードメソッドをご利用ください。

※1 画像ファイルの形式は、filename の拡張子で判断します。
※2 png / gif の背景は透明になります。他の画像形式の背景は白になります。
※3 画像解像度(dpi)の指定は行えません。
※4 描画単位は全て pixel になります。
(mm 等の単位での画像出力は行えません。)

保存する QR コード画像の周りの余白部分の大きさを pixel 単位で指定することも可能です。以下のプロパティをご利用ください。

setImgMargin(int marginPixel)　/　int getImgMargin()

2-6. DataMatrix 作成クラスの機能

Barcode.jar の DataMatrix 作成クラスは、以下の機能を有します。

DataMatrix の描画

コード、始点(左上の X/Y 座標)、幅・高さを指定します。

DataMatrix の描画方法は一次元バーコード同様、3 種類あります。

- **幅・高さ指定ぴったり描画(Draw)**

  指定幅・高さにぴったり描画する。

  幅・高さに合わせるためバーコード画像の最低限の拡大縮小が発生

  特徴) 指定サイズぴったりに描画できる。精度は他より劣る。

- **最小値指定直接描画(DrawDelicate)**

  描画する最小値を指定して精度の高いバーコードを描画。

  特徴) 描画しないとサイズがわからない。ドットにのった高い精度。

- **幅指定直接描画(DrawDirect)**

  指定幅以内で一番大きく、かつ、ドットにのった正方形、または、シンボルコードサイズ通りの比率の長方形を描画。

  特徴) 指定サイズ(以内)で描画できる。ドットにのった高い精度。

バーコードを描画する座標・幅・高さ・最小値の単位は、座標単位は、pixel・、mm / inch / point(1/72 インチ)の指定が可能です。

プロパティで以下の項目を指定することができます。

・ シンボルコードサイズ　　既定値：DxCodeSize.DxSzAuto

```
(enum) DxCodeSize.
  DxSzRectAuto,        DxSz24x24,        DxSz88x88,
  DxSzAuto,            DxSz26x26,        DxSz96x96,
  DxSzShapeAuto,       DxSz32x32,        DxSz104x104,
                       DxSz36x36,        DxSz120x120,
  DxSz10x10,           DxSz40x40,        DxSz132x132,
  DxSz12x12,           DxSz44x44,        DxSz144x144,
  DxSz14x14,           DxSz48x48,
  DxSz16x16,           DxSz52x52,        DxSz12x36,
  DxSz18x18,           DxSz64x64,        DxSz16x36,
  DxSz20x20,           DxSz72x72,        DxSz16x48
  DxSz22x22,           DxSz80x80,
```

・ 全角エンコーディング（"utf-8", "shift-jis", etc..) 既定値：utf-8

2-7. PDF417 作成クラスの機能

Barcode.jar の PDF417 作成クラスは、以下の機能を有します。

PDF417 の描画

コード、始点(左上の X/Y 座標)、幅・高さを指定します。

PDF417 の描画方法は一次元バーコード同様、3 種類あります。

- **幅・高さ指定ぴったり描画(Draw)**

  指定幅・高さにぴったり描画する。

  幅・高さに合わせるためバーコード画像の最低限の拡大縮小が発生

  特徴) 指定サイズぴったりに描画できる。精度は他より劣る。

- **最小値指定直接描画(DrawDelicate)**

  描画する最小値を指定して精度の高いバーコードを描画。

  特徴) 描画しないとサイズがわからない。ドットにのった高い精度。

- **幅指定直接描画(DrawDirect)**

  指定幅以内で一番大きく、かつ、ドットにのった長方形を描画。

  長方形の 2 編の長さの比率は、プロパティの縦横アクセプト比とサイズ種別により決定される。

  特徴) 指定サイズ(以内)で描画できる。ドットにのった高い精度。

バーコードを描画する座標・幅・高さ・最小値の単位は、座標単位は、pixel・、mm / inch / point(1/72 インチ)の指定が可能です。

プロパティで以下の項目を指定することができます。

- サイズ種別　…データ列数・行数決定方法

自動	Pdf417.SIZE_KIND.AUTO　(既定値)
データ列数指定	Pdf417.SIZE_KIND.COLUMNS
データ行数指定	Pdf417.SIZE_KIND.ROWS
データ列数・行数指定	Pdf417.SIZE_KIND.COLUMNS_AND_ROWS

- 行数　…出力データ行数指定

  サイズ種別が、

  出力行数指定の場合＝(自動サイズでない・列数指定でない場合)有効

  3～90　　既定値：5

- 列数　…出力データカラム数指定

  サイズ種別が、

  出力列数指定の場合＝(自動サイズでない・行数指定でない場合)有効

  1～30

・ エラーレベル
0～8　　既定値：0

・ エラーレベル自動決定
自動でエラー訂正レベルを決定(する・しない) 既定値：true(する)

・ 縦横アクセプト比
シンボルの縦横比、シンボル アスペクト レシオ（比）
既定値：0.5

・ 全角エンコーディング（”utf-8”, “shift-jis”, etc..）
既定値：utf-8

2-8. GS1 データバー(RSS)作成クラスの機能

Barcode.jar の各 GS1 データバー(RSS)作成クラスは、以下の機能を有します。

(1) バーコードの描画

コード、始点(左上の X/Y 座標)とバーコードの高さ・幅を指定してバーコードを描画します。幅の代わりに、バーコードの線幅の最小値を指定して描画することも可能です。その場合、より高い精度のバーコードが作成できますが、最終的に描画される幅の調整が必要になります。

もう一つの描画方法として、幅を指定していただきその中に納まる一番広い幅で、かつ、ドットにのった精度の高いバーコードを描画することも可能です。それぞれ順番に、draw / drawDelicate / drawDirect メソッドをご用意しております。

標・高さ・幅の単位は、pixel / mm / inch / point(1/72 インチ)の指定が可能です。

(2) バーコードの高さ指定とその出力結果

高さ指定とその出力結果について Barcode.jar 独自の仕様がございます。

次の表にまとめました。

１．GS1 データバーOmni-directional / 表示桁数：数字 14 桁(GTIN)
Omni-directional （標準型） 一次元バーコードなので、幅高さは通常の一次元バーコードと同様に自在です。 
Stacked（2 層型） バーコード上段・センター部分・下段の高さ割合が決まっているため、高さの指定があっても割合をキープして出力することが基本です。 Barcode.jar では、drawDirect / drawDelicate メソッドを使用した時に、センター部分を含む 3 段の割合を確実にキープします。 draw メソッドの場合は、センターの小さな正方形群の各正方形割合はキープしながら、ユーザより引数で指定された高さにより上段・下段の割合に従って上段下段の高さを決定します。従って draw メソッドでは、本来決まっているセンター部分を含む 3 段の高さ割合は保たれません。指定された高さにより、上・下段が決められている割合を保ちながら伸び縮みします。 
Stacked Omni-directional （標準 2 層型） バーコードの上段・下段の割合は同一で、センター部分(小さな正方形 3 段)がある仕様のバーコードです。従ってセンター部分の高さを固定し、ユーザより引数で指定された高さにより、上・下段がその割合によって同じ高さで伸び縮みします。 
２．GS1 データバーLimited / 表示桁数：数字 14 桁(GTIN)
Limited （限定型） 一次元バーコードなので、幅高さは通常の一次元バーコードと同様に自在です。 

３．GS1 データバーExpanded / 表示桁数：最大数字 74 桁または英字 41 文字	
Expanded(拡張一層型) 一次元バーコードなので、幅高さは通常の一次元バーコードと同様に自在です。  	Stacked Expanded(拡張多層型) このバーコードは、各段(層)の割合は同一な仕様です。段と段(層と層)間の小さな 3 段の正方形の高さを固定し、ユーザより引数で指定された高さにより、各段(層)がその割合によって同じ高さで伸び縮みします。 

(3) 拡張型の AI 識別子(ファンクションコード)について

決められている AI 識別子(ファンクションコード)は、何も指定しなくても、Barcode.jar がコード体系から判断し自動的に挿入します。

ただし、任意で AI 識別子を挿入する場合は、{AI}を入力してください。

例：(01)商品識別コード＋任意の AI

0100012345678905{AI}10ABC123 ⇒ (01)00012345678905(10)ABC12

(4) 添字の描画

バーコードの下にコードの文字列自体を描画します(既定値)。プロパティの設定で描画をしないようにすることも可能です。

添字を、コードを意味するバーの位置に描画するか(JAN 既定値)、バーコード全体の幅に均等割付するか(通常既定値)を指定することが可能です。

※ JAN(EAN)コードの場合、既定値の状態では商品コードのバーコードのような描画を行い、均等割付にすると書籍コードのバーコードのような描画を行います。

添字のフォントをプロパティで指定することも可能です。

(5) 回転描画

プロパティの設定により、バーコードの左上始点座標を中心に、

90 度／180 度／270 度　回転して描画することが可能です。既定値は 0 度です。

3. アプリケーションプログラムから Barcode.jar の使用方法

3-1. クラス仕様

3-1-1. 概要

Barcode.jar は、以下のそれぞれバーコードごとに独立したクラスで構成されております。

pao.barcode
- **pao.barcode.Jan13**
- **pao.barcode.Jan8**
- **pao.barcode.UpcA**
- **pao.barcode.UpcE**
- **pao.barcode.ITF**
- **pao.barcode.Matrix2of5**
- **pao.barcode.NW7**
- **pao.barcode.Code39**
- **pao.barcode.Code93**
- **pao.barcode.Code128**
- **pao.barcode.GS1_128**
- **pao.barcode.EAN128**
- **pao.barcode.YubinCustomer**
- **pao.barcode.QRCode**
- **pao.barcode.DataMatrix**
- **pao.barcode.Pdf417**
- **pao.barcode. Gs1Databar14**
- **pao.barcode. Gs1DatabarLimited**
- **pao.barcode. Gs1DatabarExpanded**

コンビニバーコード以外の一次元バーコードのクラスは、基本的に同一名のプロパティやメソッドといったメンバを所有し、それらの機能も同一です。
そこで以降の各メンバ(メソッドやプロパティ)の説明では、
- 一次元バーコードのクラス
- コンビニバーコードのメンバ(GS1_128 クラス)
- 郵便カスタマバーコードクラス
- 二次元バーコードのクラス (QR コード / DataMatrix / PDF417)
- GS1 データバー(RSS)のクラス(標準型 / 限定型 / 拡張型)

の５つに分けてご説明いたします。

3-1-2. 一次元バーコードクラスメンバ

3-1-2-1. コンストラクタ

初期処理を行う。
バーコードの種類別に以下の２インタフェイスが存在します。

(a) System.drawing.Graphics オブジェクトへの描画

(1) public JAN13(java.awt.Graphics2D g)
(2) public JAN8(java.awt.Graphics2D g)
(3) public UpcA(java.awt.Graphics2D g)
(4) public UpcE(java.awt.Graphics2D g)
(5) public ITF(java.awt.Graphics2D g)
(6) public Matrix2of5(java.awt.Graphics2D g)
(7) public NEC2of5(java.awt.Graphics2D g)
(8) public NW7(java.awt.Graphics2D g)
(9) public Code39(java.awt.Graphics2D g)
(10) public Code93(java.awt.Graphics2D g)
(11) public Code128(java.awt.Graphics2D g)
(12) public GS1_128(java.awt.Graphics2D g)
(13) public EAN128(java.awt.Graphics2D g)

・引数

java.awt.Graphics2D g
バーコードの描画を行う Graphics2D を指定します。

(b) 画像ファイルへの描画

(1) public JAN13 (String imgFilePath)
(2) public JAN8 (String imgFilePath)
(3) public UpcA (String imgFilePath)
(4) public UpcE (String imgFilePath)
(5) public ITF (String imgFilePath)
(6) public Matrix2of5 (String imgFilePath)
(7) public NEC2of5 (String imgFilePath)
(8) public NW7 (String imgFilePath)
(9) public Code39 (String imgFilePath)
(10) public Code93 (String imgFilePath)
(10) public Code128 (String imgFilePath)
(11) public GS1_128 (String imgFilePath)
(12) public EAN128 (String imgFilePath)

・引数

String imgFilePath

バーコードを描画するデフォルトファイルパスを指定します。

※画像ファイルの拡張子から画像フォーマットを特定します。

こちらのコンストラクタでインスタンスを作成した場合、
通常の draw メソッドで画像ファイルへバーコード出力を行います。
もし、draw するタイミングで出力画像ファイルを変更される場合は、
drawメソッドの引数に画像ファイルパスを指定できる方のオーバーロードメソッドを使用してください。

3-1-2-2. メソッド

(1) **public void draw(String code, float x, float y, float width, float height)**

バーコードの描画を行います。指定幅を正確に合わせるためにいったん描画したバーコードを縮小して描画しなおします。そのため、多少、精度が劣化します。ドット単位で正確な精度を期待する場合は、指定した幅と正確に一緒にはなりませんが、drawDirect / drawDelicate メソッドを使用してください。

・引数

① **String code**

描画を行うバーコードのコードを文字列で指定します。

GS1-128(UCC/EAN128)のコード指定については、アプリケーション識別子(AI)の前に"{FNC1}"という文字列を指定することで GS1-128(UCC/EAN128)のバーコード作成を可能としています。

例) (01)04912345678904 (10)0211 (3103)001528　・・・()付きがアプリケーション識別子(AI)

コード指定方法→「{FNC1} 0104912345678904{FNC1} 100211{FNC1} 3103001528」

② **float x**

描画位置の始点(左上)の X 座標を指定します。

単位は、pixel です。

③ **float y**

描画位置の始点(左上)の Y 座標を指定します。

単位は、pixel です。

④ **float width**

バーコードの全体の幅を指定します。

単位は、pixel です。

⑤ **float height**

バーコードのバーの高さを指定します。

単位は、pixel です。

・戻り値

なし

・例外の種類

予測可能割込発生エラーを次ページでクラス別に記述します。

○ JAN13

① public ErrJAN13BadChar()
数字以外の文字が使用されました。
使用できる文字は数字のみです。

② public ErrJAN13BadLen()
コードの桁数は、13 桁か、12 桁を指定してください。
12 桁の場合チェックキャラクタを自動付与します。

③ public ErrJAN13CheckDigit()
コード末尾のチェックデジットが誤っています。

○ JAN8

④ public ErrJAN8BadChar()
数字以外の文字が使用されました。
使用できる文字は数字のみです。

⑤ public ErrJAN8BadLen()
コードの桁数は、8 桁か、7 桁を指定してください。
7 桁の場合チェックキャラクタを自動付与します。

⑥ public ErrJAN8CheckDigit()
コード末尾のチェックデジットが誤っています。

○ UPC-A

⑦ public ErrUPC_ABadChar()
数字以外の文字が使用されました。
使用できる文字は数字のみです。

⑧ public ErrUPC_ABadLen()
コードの桁数は、12 桁か、11 桁を指定してください。
11 桁の場合チェックキャラクタを自動付与します。

⑨ public ErrUPC_ACheckDigit()
コード末尾のチェックデジットが誤っています。

○ UPC-E

⑩ public ErrUPC_EBadChar()
数字以外の文字が使用されました。
使用できる文字は数字のみです。

⑪ public ErrUPC_EBadLen()
コードの桁数は、6 桁か、7 桁か、8 桁を指定してください。
6 桁の場合、頭に 0 と末尾にチェックキャラクタを自動付与します。
7 桁の場合、末尾にチェックキャラクタを自動付与します。

⑫ public ErrUPC_ECheckDigit()
コード末尾のチェックデジットが誤っています。
(8 桁の数値が指定されている時のエラーです。)

⑬ public ErrUPC_E_BadCodeForCheckDigit ()
チェックディジットを計算できるコード体系ではありません。
1 桁目=0 / 7 桁目=0～9 でなければいけません。
(7 桁以上の数値が指定されている時のエラーです。)

○ ITF

⑭ public ErrITFBadChar()

数字以外の文字が使用されました。
使用できる文字は数字のみです。

○ Matrix2of5

⑮ public ErrMatrix2of5BadChar()

数字以外の文字が使用されました。
使用できる文字は数字のみです。

○ NEC2of5

⑯ public ErrNEC2of5BadChar()

数字以外の文字が使用されました。
使用できる文字は数字のみです。

○ NW7

⑰ public ErrNW7BadChar()

利用できない文字 ='? ' が使用されました。
使用できる文字は"ABCD.+:/$-0123456789"です。

○ Code39

⑱ public ErrCode39BadChar()

利用できない文字 ='? ' が使用されました。
使用できる文字は
"1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ-. *$/+%"です。

○ Code128

予測可能割り込み発生エラーなし。

○ GS1_128 / EAN128

予測可能割り込み発生エラーなし。

(2) public void draw
(String code, float x, float y, float width, float height
, [pao.barcode.]GraphicsUnit gu, int dpi)

(1)の draw メソッドのオーバーロードメソッドで、pixel 以外の mm 等の単位を指定してバーコードの描画を行う。

引数で指定された解像度(dpi)を元に、引数で指定された単位(mm 等)で、x,y の位置に width,height の幅高さのバーコードを描画する。
単位は、mm / inch /point / pixel を指定可能。(Pao.barcode.GraphicsUnit)
ver 2.0 で追加されたオーバーロードメソッドです。
他の機能について draw メソッドと同様となります。

・引数
① String code
描画を行うバーコードのコードを文字列で指定します。
float x
描画位置の始点(左上)の X 座標を指定します。
単位は、引数 GraphicsUnit gu の通り。
② float y
描画位置の始点(左上)の Y 座標を指定します。
単位は、引数 GraphicsUnit gu の通り。
③ float width
バーコードの全体の幅を指定します。
単位は、引数 GraphicsUnit gu の通り。
④ float height
バーコードのバーの高さを指定します。
単位は、引数 GraphicsUnit gu の通り。
⑤ GraphicsUnit gu
描画する座標・幅・高さの単位。以下の列挙体を指定可能。
Pao.barcode.GraphicsUnit.MM / .INCHI / .POINT / .PIXEL
⑥ int dpi
描画(出力)デバイスの解像度。

・戻り値
なし

・例外の種類
draw メソッドと同様。

(3) public void draw
(String code, float x, float y, float width, float height, String imgFilePath)

(1)の draw メソッドのオーバーロードメソッドでバーコードを画像ファイルに出力します。

imgFilePath で指定したファイルへバーコードの描画を行います。
ver 2.0 で追加されたオーバーロードメソッドです。
他の機能について draw メソッドと同様となります。

・引数

① String code
描画を行うバーコードのコードを文字列で指定します。

② float x
0 のみが指定可能です。

③ float y
0 のみが指定可能です。

④ float width
バーコードの全体の幅を指定します。
単位は、pixel です。

⑤ float height
バーコードのバーの高さを指定します。
単位は、pixel です。

⑥ String imgFilePath
バーコード出力を行う画像ファイルパスを指定します。
※画像ファイルパスはコンストラクタで指定されているため不要ですが、出力する画像ファイルへバーコードを描画するたびに変更する場合に、こちらの画像ファイルパスを指定可能なメソッドをお使いください。

・戻り値
なし

・例外の種類
draw メソッドと同様。

(4) public void drawDirect
(String code, float x, float y, float width, float height)
バーコードの描画を行います。
指定幅以内で最も広い幅でバーコードを直接描画します。
ドット単位での描画精度を実現します。

・引数
① String code
描画を行うバーコードのコードを文字列で指定します。
GS1-128(UCC/EAN128)のコード指定については、アプリケーション識別子(AI)の前に"{FNC1}"という文字列を指定することで GS1-128(UCC/EAN128)のバーコード作成を可能としています。
例) (01)04912345678904 (10)0211 (3103)001528 ・・・()付きがアプリケーション識別子(AI)
コード指定方法→「{FNC1} 0104912345678904{FNC1} 100211{FNC1} 3103001528」
② float x
描画位置の始点(左上)の X 座標を指定します。
単位は、pixel です。
③ float y
描画位置の始点(左上)の Y 座標を指定します。
単位は、pixel です。
④ float width
バーコードの全体の幅を指定します。
指定した幅以内で最も広い幅のバーコードを描画します。
単位は、pixel です。
⑤ float height
バーコードのバーの高さを指定します。
単位は、pixel です。

・戻り値
なし

・例外の種類
draw メソッドと同様。

(5) public void drawDirect
(String code, float x, float y, float width, float height
, [Pao.barcode.]GraphicsUnit gu, int dpi)

(4)の drawDirect メソッドのオーバーロードメソッドで pixel 以外の mm 等の単位を指定してバーコードの描画を行う。

引数で指定された解像度(dpi)を元に、引数で指定された単位(mm 等)で、x,y の位置に Width,Height の幅高さのバーコードを描画する。
単位は、mm / inch /point / pixel を指定可能。(Pao.barcode.GraphicsUnit)
ver 2.0 で追加されたオーバーロードメソッドです。
他の機能について drawDirect メソッドと同様となります。

・引数
① String code
描画を行うバーコードのコードを文字列で指定します。
float x
描画位置の始点(左上)の X 座標を指定します。
単位は、引数 GraphicsUnit gu の通り。
② float y
描画位置の始点(左上)の Y 座標を指定します。
単位は、引数 GraphicsUnit gu の通り。
③ float width
バーコードの全体の幅を指定します。
指定した幅以内で最も広い幅のバーコードを描画します。
単位は、引数 GraphicsUnit gu の通り。
④ float height
バーコードのバーの高さを指定します。
単位は、引数 GraphicsUnit gu の通り。
⑤ GraphicsUnit gu
描画する座標・幅・高さの単位。以下の列挙体を指定可能。
[Pao.barcode.] GraphicsUnit.MM / .INCHI / .POINT / .PIXEL
⑥ int dpi
描画(出力)デバイスの解像度。

・戻り値
なし

・例外の種類
draw メソッドと同様。

(6) public void drawDirect
(String code, float x, float y, float width, float height, String imgFilePath)

(4)の drawDirect メソッドのオーバーロードメソッドでバーコードを画像ファイルに出力する。

imgFilePath で指定したファイルへバーコードの描画を行います。
ver 2.0 で追加されたオーバーロードメソッドです。
他の機能について drawDirect メソッドと同様となります。

・引数
① String code
描画を行うバーコードのコードを文字列で指定します。
② float x
0 のみが指定可能です。
③ float y
0 のみが指定可能です。
④ float width
バーコードの全体の幅を指定します。
指定した幅以内で最も広い幅のバーコードを描画します。
単位は、pixel です。
⑤ float height
バーコードのバーの高さを指定します。
単位は、pixel です。
⑥ String imgFilePath
バーコード出力を行う画像ファイルパスを指定します。
※画像ファイルパスはコンストラクタで指定されているため不要ですが、出力する画像ファイルへバーコードを描画するたびに変更する場合に、こちらの画像ファイルパスを指定可能なメソッドをお使いください。

・戻り値
なし

・例外の種類
draw メソッドと同様。

(7) public void drawDelicate(string code, float x, float y, float minLineWidth, float height)

バーコードの描画を行います。

draw メソッドとの違いは、バーコード全体の幅を指定するのではなく、バーを描画する一番細い線の幅を指定します。

draw メソッドに比べて、**drawDirect** メソッドと同様に精度の高いバーコードを描画することが可能です。ただし、バーコード全体の幅の調整が必要になります。

※この **drawDelicate** メソッドを使用してバーの最小幅を指定した場合に精度が高くなる理由は、直接、Graphics オブジェクトに線を描画するためです。

draw メソッドを使用して全体の幅を指定した場合、一旦、仮想空間に描画したバーコードを指定された全体の幅に対して当てはまるように Graphics オブジェクトに縮小描画しております。

・引数

① string code

描画を行うバーコードのコードを文字列で指定します。

GS1-128(UCC/EAN128)のコード指定については、アプリケーション識別子(AI)の前に"{FNC1}"という文字列を指定することで GS1-128(UCC/EAN128)のバーコード作成を可能としています。

例) (01)04912345678904 (10)0211 (3103)001528　・・・()付きがアプリケーション識別子(AI)

コード指定方法→「{FNC1}010491234567890 4{FNC1}100211{FNC1}3103001528」

② float x

描画位置の始点(左上)の X 座標を指定します。

単位は、pixel です。

③ float y

描画位置の始点(左上)の Y 座標を指定します。

単位は、pixel です。

④ float minLineWidth

バーコードを描画するバーの最小幅の値を指定します。

単位は、pixel です。

⑤ float height

バーコードのバーの高さを指定します。

単位は、pixel です。

・戻り値

なし

・例外の種類

draw メソッドと同様。

(8) public void drawDelicate(string code, float x, float y, float minLineWidth, float height, String imgFilePath)

(7)の drawDelicate メソッドのオーバーロードメソッドで、バーコードを画像ファイルに出力する。

imgFilePath で指定したファイルへバーコードの描画を行います。
ver 2.0 で追加されたオーバーロードメソッドです。
他の機能について drawDelicate メソッドと同様となります。

・引数

① String code
描画を行うバーコードのコードを文字列で指定します。

② float x
0 のみが指定可能です。

③ float y
0 のみが指定可能です。

④ float minLineWidth
バーコードを描画するバーの最小幅の値を指定します。
単位は、pixel です。

⑤ float height
バーコードのバーの高さを指定します。
単位は、pixel です。

⑥ String imgFilePath
バーコード出力を行う画像ファイルパスを指定します。
※画像ファイルパスはコンストラクタで指定されているため不要ですが、出力する画像ファイルへバーコードを描画するたびに変更する場合に、こちらの画像ファイルパスを指定可能なメソッドをお使いください。

・戻り値
なし

・例外の種類
draw メソッドと同様。

3-1-2-3. プロパティ

(1) public boolean getTextWrite()
public void setTextWrite(boolean value)
true： 添字の描画を行う。(既定値)
false： 添字を描画しない。

(2) public boolean getTextKintou ()
public void setTextKintou(boolean value)
true： 添字の描画は、バーコード全体の幅に均等割付で行う。
false： 添字を描画は、コードを意味するバーの位置に行う。(既定値)

(3) public Font getTextFont ()
public void setTextFont(Font value)
添字のフォント。既定値は、”ＭＳ ゴシック　9 ポイント　標準”。

(4) public float getRotateAngl ()
public void setRotateAngle(float value)
回転角度を数値で指定。左下を軸に右回転して描画を行う。
既定値は、0 度。

(5) public boolean getDispStartStopCode ()
public void setDispStartStopCode (boolean value)
Code39/NW7 のみ使用可能なプロパティ
true： スタート／ストップコードの描画を行う。
false： スタート／ストップコードを描画しない。(既定値)

(6) public int getKuroBarChousei ()
public void setKuroBarChousei (int value)
黒バーの幅を微細調整(加減)するドット数を指定する。
マイナスの値で黒バーを細くすることも可能。既定値は 0 (pixel)。

(7) public int getDPI ()
public void setDPI (int value)
黒バーの微細調整を行うための DPI を指定する。既定値は 600DPI。

(8) public int getImgMargin ()
public void setImgMargin (int value)
バーコードを画像保存する際の
バーコード周囲の余白を pixel 単位で指定する。既定値余白は 1 (pixel)。

◇次のプロパティは、CODE128 / GS1_128(EAN128)のみで使用するプロパティです。

```
public CodeSet128  getCodeABC()
public void  setCodeABC(CodeSet128 value)
```

CODE128 のコードセットには、次の種類があります。

CODE-A： フルアスキー

CODE-B: 1 桁の アルファ・ニューメリック

CODE-C: 2 桁の数字

本プロパティでは、次の指定が可能です。

列挙値　Pao.BarCode.CodeSet128	
AUTO	Barcode.net が自動でコードセットを組み合わせ最小幅のバーコードとする。
CODE_A	CODE-A
CODE_B	CODE-B
CODE_C	CODE-C

3-1-3. コンビニエンスストア標準料金代理収納用バーコードメソッド

3-1-3-1. コンストラクタ

一次元バーコードの GS1_128(EAN128)のインスタンスを使用する。

(a) System.drawing.Graphics オブジェクトへの描画

(1) public GS1_128(java.awt.Graphics2D g)

(2) public EAN128(java.awt.Graphics2D g)

・引数

java.awt.Graphics2D g

バーコードの描画を行う Graphics2D を指定します。

(b) 画像ファイルへの描画

(1) public GS1_128 (String imgFilePath)

(2) public EAN128 (String imgFilePath)

・引数

String imgFilePath

バーコードを描画するデフォルトファイルパスを指定します。

※画像ファイルの拡張子から画像フォーマットを特定します。

こちらのコンストラクタでインスタンスを作成した場合、
通常の draw メソッドで画像ファイルへバーコード出力を行います。
もし、draw するタイミングで出力画像ファイルを変更される場合は、
draw メソッドの引数に画像ファイルパスを指定できる方のオーバーロードメソッドを使用してください。

3-1-3-2. メソッド

(1) public void drawConvenience
(String code, float x, float y, float width, float height)
コンビニバーコードの描画を行います。指定幅を正確に合わせるためにいったん描画したバーコードを縮小して描画しなおします。
そのため、多少、精度が劣化します。
ドット単位で正確な精度を期待する場合は、指定した幅と正確に一緒にはなりませんが、drawConvenienceDirect / drawConvenienceDelicate メソッドを使用してください。
特に UCC/EAN-128 標準料金代理収納ガイドライン(財団法人 流通システム開発センター)」に準拠する場合は、drawConvenienceDelicate メソッドを使用してください。

・引数
① String code
描画を行うバーコードのコードを文字列で指定します。
アプリケーション識別子(AI)の前に"{FNC1}"という文字列を指定することで GS1-128 のコンビニバーコード作成を可能としています。
例) (91)91234500000000000000452087500401310029500 6
・・・()カッコ付きがアプリケーション識別子(AI)
コード指定方法↓
「{FNC1}9191234500000000000000452087500401310029500 6」
② float x
描画位置の始点(左上)の X 座標を指定します。
単位は、pixel です。
③ float y
描画位置の始点(左上)の Y 座標を指定します。
単位は、pixel です。
④ float width
バーコードの全体の幅を指定します。
単位は、pixel です。
⑤ float height
バーコードのバーの高さを指定します。
単位は、pixel です。

・戻り値
なし

・例外の種類
予測可能割込発生エラーは、draw メソッド 参照。

(2) public void drawConvenience
(String code, float x, float y, float width, float height
, [Pao.barcode.]GraphicsUnit gu, int dpi)

(1)の drawConvenience メソッドのオーバーロードメソッドで、pixel 以外の mm 等の単位を指定してコンビニバーコードの描画を行う。
引数で指定された解像度(dpi)を元に、引数で指定された単位(mm 等)で、x,y の位置に Width,Height の幅高さのバーコードを描画する。
単位は、mm / inch /point / pixel を指定可能。(Pao.barcode.GraphicsUnit)
ver 2.0 で追加されたオーバーロードメソッドです。
他の機能について drawConvenience メソッドと同様となります。

・引数
① String code
描画を行うバーコードのコードを文字列で指定します。
アプリケーション識別子(AI)の前に"{FNC1}"という文字列を指定することで GS1-128 のコンビニバーコード作成を可能としています。
例) (91)912345 ・・・() 付きがアプリケーション識別子(AI)
コード指定方法→「{FNC1}91912345」
② float x
描画位置の始点(左上)の X 座標を指定します。
単位は、引数 GraphicsUnit gu の通り。
③ float y
描画位置の始点(左上)の Y 座標を指定します。
単位は、引数 GraphicsUnit gu の通り。
④ float width
バーコードの全体の幅を指定します。
単位は、引数 GraphicsUnit gu の通り。
⑤ float height
バーコードのバーの高さを指定します。
単位は、引数 GraphicsUnit gu の通り。
⑥ GraphicsUnit gu
描画する座標・幅・高さの単位。以下の列挙体を指定可能。
[Pao.barcode.] GraphicsUnit.MM / .INCHI / .POINT / .PIXEL
⑦ int dpi
描画(出力)デバイスの解像度。

・戻り値
なし

・例外の種類
予測可能割込発生エラーは、draw メソッド 参照

(3) public void drawConvenience
(String code, float x, float y, float width, float height
, String imgFilePath)

(1)の drawConvenience メソッドのオーバーロードメソッドでコンビニバーコードを画像ファイルに出力する。

imgFilePath で指定したファイルへバーコードの描画を行います。
ver 2.0 で追加されたオーバーロードメソッドです。
他の機能について drawConvenience メソッドと同様となります。

・引数
① String code
描画を行うバーコードのコードを文字列で指定します。
アプリケーション識別子(AI)の前に”{FNC1}”という文字列を指定することで GS1-128 のコンビニバーコード作成を可能としています。
例) (91)91234500000000000000452087500401310029500 6
・・・()カッコ付きがアプリケーション識別子(AI)
コード指定方法↓
「{FNC1}9191234500000000000000452087500401310029500 6」
② float x
0 のみが指定可能です。
③ float y
0 のみが指定可能です。
④ float width
バーコードの全体の幅を指定します。
単位は、pixel です。
⑤ float height
バーコードのバーの高さを指定します。
単位は、pixel です。
⑥ String imgFilePath
バーコード出力を行う画像ファイルパスを指定します。
※画像ファイルパスはコンストラクタで指定されているため不要ですが、出力する画像ファイルへバーコードを描画するたびに変更する場合に、こちらの画像ファイルパスを指定可能なメソッドをお使いください。

・戻り値
なし

・例外の種類
draw メソッドと同様。

(4) public void drawConvenienceDirect
(String code, float x, float y, float width, float height)
コンビニバーコードの描画を行います。
指定幅以内で最も広い幅でバーコードを直接描画します。
ドット単位での描画精度を実現します。

・引数
① String code
描画を行うバーコードのコードを文字列で指定します。
アプリケーション識別子(AI)の前に"{FNC1}"という文字列を指定することで GS1-128 のコンビニバーコード作成を可能としています。
例) (91)91234500000000000000452087500401310029500 6
・・・()カッコ付きがアプリケーション識別子(AI)
コード指定方法↓
「{FNC1}9191234500000000000000452087500401310029500 6」
② float x
描画位置の始点(左上)の X 座標を指定します。
単位は、pixel です。
③ float y
描画位置の始点(左上)の Y 座標を指定します。
単位は、pixel です。
④ float width
バーコードの全体の幅を指定します。
指定した幅以内で最も広い幅のバーコードを描画します。
単位は、pixel です。
⑤ float height
バーコードのバーの高さを指定します。
単位は、pixel です。

・戻り値
なし

・例外の種類
draw メソッドと同様。

(5) public void drawConvenienceDirect
(String code, float x, float y, float width, float height
, [Pao.barcode.]GraphicsUnit gu, int dpi)

(4)の drawConvenienceDirect メソッドのオーバーロードメソッドで pixel 以外の mm 等の単位を指定してバーコードの描画を行う。

引数で指定された解像度(dpi)を元に、引数で指定された単位(mm 等)で、x,y の位置に Width,Height の幅高さのバーコードを描画する。
単位は、mm / inch /point / pixel を指定可能。(Pao.barcode.GraphicsUnit)
ver 2.0 で追加されたオーバーロードメソッドです。
他の機能について drawConvenienceDirect メソッドと同様となります。

・引数

① String code
描画を行うバーコードのコードを文字列で指定します。
アプリケーション識別子(AI)の前に"{FNC1}"という文字列を指定することで GS1-128 のコンビニバーコード作成を可能としています。
例) (91)912345 ・・・() 付きがアプリケーション識別子(AI)
コード指定方法→「{FNC1}91912345」

② float x
描画位置の始点(左上)の X 座標を指定します。
単位は、引数 GraphicsUnit gu の通り。

③ float y
描画位置の始点(左上)の Y 座標を指定します。
単位は、引数 GraphicsUnit gu の通り。

④ float width
バーコードの全体の幅を指定します。
指定した幅以内で最も広い幅のバーコードを描画します。
単位は、引数 GraphicsUnit gu の通り。

⑤ float height
バーコードのバーの高さを指定します。
単位は、引数 GraphicsUnit gu の通り。

⑥ GraphicsUnit gu
描画する座標・幅・高さの単位。以下の列挙体を指定可能。
[Pao.barcode.] GraphicsUnit.MM / .INCHI / .POINT / .PIXEL

⑦ int dpi
描画(出力)デバイスの解像度。

・戻り値
なし

(6) public void drawConvenienceDirect
(String code, float x, float y, float width, float height, String imgFilePath)

(4)の drawConvenienceDirect メソッドのオーバーロードメソッドでバーコードを画像ファイルに出力する。

imgFilePath で指定したファイルへバーコードの描画を行います。
ver 2.0 で追加されたオーバーロードメソッドです。
他の機能について drawConvenienceDirect メソッドと同様となります。

・引数

① String code
描画を行うバーコードのコードを文字列で指定します。
例) (91)91234500000000000000452087500401310029500 6
・・・()カッコ付きがアプリケーション識別子(AI)
コード指定方法↓
「{FNC1}9191234500000000000000452087500401310029500 6」

② float x
0 のみが指定可能です。

③ float y
0 のみが指定可能です。

④ float width
バーコードの全体の幅を指定します。
指定した幅以内で最も広い幅のバーコードを描画します。
単位は、pixel です。

⑤ float height
バーコードのバーの高さを指定します。
単位は、pixel です。

⑥ String imgFilePath
バーコード出力を行う画像ファイルパスを指定します。
※画像ファイルパスはコンストラクタで指定されているため不要ですが、出力する画像ファイルへバーコードを描画するたびに変更する場合に、こちらの画像ファイルパスを指定可能なメソッドをお使いください。

・戻り値
なし

・例外の種類
draw メソッド 参照。

(7) public void drawConvenienceDelicate (String code, float x, float y , float minLineWidth, float height)

コンビニバーコードの描画を行います。
例えば、以下のような手順でバーコードを作成します。
① getdrawConvenienceDelicateWidth() を minLineWidth=1 で呼び出し、(image)画像の横幅を取得します。
② ①で取得した image(画像)の Graphics に対して、コンビニバーコードを描画します。具体的には、この drawConvenienceDelicate()メソッドを minLineWidth=1 で呼び出します。
③ 描画した image(画像)を jpeg 等画像ファイルに保存します。
④ 保存した画像データを印刷データとして何らかの手法で取り扱います。
例えば・・・
・Excel ファイルに画像データとして挿入する。
・ラスタイメージの印刷データの一部として。
なお上記手順は、付属のサンプルプログラム(BarApp)で実現しております。

・引数
① String code
描画を行うバーコードのコードを文字列で指定します。
アプリケーション識別子(AI)の前に"{FNC1}"という文字列を指定することで GS1-128(UCC/EAN128)のバーコード作成を可能としています。
例) (91)91234500000000000000452087500401310029500 6
・・・()カッコ付きがアプリケーション識別子(AI)
コード指定方法↓
「{FNC1}9191234500000000000000452087500401310029500 6」
② float x
描画位置の始点(左上)の X 座標を指定します。単位は、pixel です。
③ float y
描画位置の始点(左上)の Y 座標を指定します。単位は、pixel です。
④ float minLineWidth
バーコードを描画するバーの最小幅の値をドット単位で指定します。
単位は、pixel です。
⑤ float height
バーコードのバーの高さを指定します。単位は、pixel です。

・戻り値
なし

(8) **public static float getdrawConvenienceDelicateWidth**
(String code, float minLineWidth, int dpi, int dotKuroBarChousei)

※ver 2.0 以降、画像保存を行うことができるようになったため、このメソッドは基本的に不要なメソッドです。下位互換のために残してあります。

コンビニバーコードを画像ファイルに保存するため、実際に描画した時のバーコード幅を事前に取得するためのメソッドです。
バーコード幅は事前にわからないためこのメソッドが用意されています。
(バーコードの高さは、drawConvinienceDelicate()で指定したとおりに描画されます。)

・引数

① **String code**
描画を行うバーコードのコードを文字列で指定します。
アプリケーション識別子(AI)の前に"{FNC1}"という文字列を指定することで GS1-128(UCC/EAN128)のバーコード作成を可能としています。
例) (91)91234500000000000004520875004013100295006
・・・()カッコ付きがアプリケーション識別子(AI)
コード指定方法↓
「{FNC1}9191234500000000000004520875004013100295006」

② **float minLineWidth**
バーコードを描画するバーの最小幅の値をドット単位で指定します。

③ **int dpi**
黒バーの微細調整を行うための DPI を指定します。
既定値は、600DPI ですので、不明な時は 600 を指定してください。

④ **int dotKuroBarChousei**
黒バーの微細調整のためにドット数を指定する。
既定値は、0 ドットですので、不明な時は 0 を指定してください。

3-1-3-3. プロパティ

3-1-2.一次元バーコードと同様のため説明を割愛します。

3-1-4. GS1 データバー(RSS) クラスメンバ

GS1 データバーのクラスのインタフェイス(コンストラクタ・メソッド・プロパティ)は、基本的に一次元バーコードのものと変わりません。クラス自体も同じ Interface で実装し、特別な機能のプロパティのみ追加されております。
従って、メッソッドやプロパティの説明は、冗長にならない目的のためにも割愛させていただいている部分がございます。

3-1-4-1. コンストラクタ

初期処理を行う。
バーコードの種類別に以下の２インタフェイスが存在します。

(a) System.drawing.Graphics オブジェクトへの描画

(1) public Gs1Databar14 (java.awt.Graphics2D g)
(2) public Gs1DatabarExpanded (java.awt.Graphics2D g)
(3) public Gs1DatabarLimited (java.awt.Graphics2D g)

・引数

java.awt.Graphics2D g
バーコードの描画を行う Graphics2D を指定します。

(b) 画像ファイルへの描画

(1) public Gs1Databar14 (String imgFilePath)
(2) public Gs1DatabarExpanded (String imgFilePath)
(3) public Gs1DatabarLimited (String imgFilePath)

・引数

String imgFilePath
バーコードを描画するデフォルトファイルパスを指定します。
※画像ファイルの拡張子から画像フォーマットを特定します。

こちらのコンストラクタでインスタンスを作成した場合、
通常の draw メソッドで画像ファイルへバーコード出力を行います。
もし、draw するタイミングで出力画像ファイルを変更される場合は、
draw メソッドの引数に画像ファイルパスを指定できる方のオーバーロードメソッドを使用してください。

3-1-4-2. メソッド

(1) **public void draw(String code, float x, float y, float width, float height)**

バーコードの描画を行います。指定幅を正確に合わせるためにいったん描画したバーコードを縮小して描画しなおします。そのため、多少、精度が劣化します。ドット単位で正確な精度を期待する場合は、指定した幅と正確に一緒にはなりませんが、drawDirect / drawDelicate メソッドを使用してください。

・引数

① **String code**

描画を行うバーコードのコードを文字列で指定します。

《拡張型の AI 識別子(ファンクションコード)について》

決められている AI 識別子(ファンクションコード)は、何も指定しなくとも、コード体系から自動的に挿入されます。

ただし、任意で AI 識別子を挿入する場合は、{AI}を入力してください。

例：(01)商品識別コード＋任意の AI

0100012345678905{AI}10ABC123 ⇒ (01)00012345678905(10)ABC12

② **float x**

描画位置の始点(左上)の X 座標を指定します。

単位は、pixel です。

③ **float y**

描画位置の始点(左上)の Y 座標を指定します。

単位は、pixel です。

④ **float width**

バーコードの全体の幅を指定します。

単位は、pixel です。

⑤ **float height**

バーコードのバーの高さを指定します。

単位は、pixel です。

GS1 RSS の高さについては、Barcoe.net 独自に次の仕様がございます。

◇ **通常の一次元バーコードと同じ**

種類：標準一層型 / 限定型 / 拡張一層型

引数で指定された高さは、そのままバーコードの高さとなります。

◇ **各層が均一の伸縮**

種類：標準二層型 / 拡張多層型

各段(層)の割合は同一な仕様です。段と段(層と層)間の小さな正方形部分の高さを固定し、引数で指定された高さを全体の高さ都市、各段(層)がその割合によって同じ高さで伸び縮みします。

◇ 2 層とセンター部の割合が固定

種類：二層型(標準)

バーコード上段・センター部分・下段の高さ割合が決まっているため、高さの指定があっても割合をキープして出力することが基本です。

Barcode.net では、DrawDirect / DrawDelicate メソッドを使用した時に、センター部分を含む 3 段の割合を確実にキープします。

本 Draw メソッドの場合は、センターの小さな正方形群の各正方形割合はキープしながら、ユーザより引数で指定された高さにより上段・下段の割合に従って上段下段の高さを決定します。従って Draw メソッドでは、本来決まっているセンター部分を含む 3 段の高さ割合は保たれません。指定された高さにより、上・下段が決められている割合を保ちながら伸び縮みします。

・戻り値

なし

(2) `public void draw (String code, float x, float y, float width, float height , [pao.barcode.]GraphicsUnit gu, int dpi)`

(1)draw メソッドのオーバーロードメソッドで、pixel 以外の mm 等の単位を指定してバーコードの描画を行う。

引数については、3-1-1.一次元バーコードのメソッドと同様ですので、説明を割愛します。

(3) `public void draw (String code, float x, float y, float width, float height, String imgFilePath)`

(1)draw メソッドのオーバーロードメソッドで、バーコードを画像ファイルに出力します。

imgFilePath で指定したファイルへバーコードの描画を行います。

他の引数については、(1)の Draw メソッドと同様ですので割愛します。

(4) `public void drawDirect (String code, float x, float y, float width, float height)`

バーコードの描画を行います。

指定幅以内で最も広い幅でバーコードを直接描画します。

ドット単位での描画精度を実現します。

引数については、(1)の Draw メソッドと同様ですので割愛します。

(5) public void drawDirect
(String code, float x, float y, float width, float height
, [Pao.barcode.]GraphicsUnit gu, int dpi)
(4)drawDirect メソッドのオーバーロードメソッドで、pixel 以外の mm 等の単位を指定してバーコードの描画を行う。
引数については、3-1-1.一次元バーコードのメソッドと同様ですので、説明を割愛します。

(6) public void drawDirect
(String code, float x, float y, float width, float height, String imgFilePath)
(4)drawDirect メソッドのオーバーロードメソッドで、バーコードを画像ファイルに出力します。
imgFilePath で指定したファイルへバーコードの描画を行います。
他の引数については、(1)の Draw メソッドと同様ですので割愛します。

(7) public void drawDelicate
(string code, float x, float y, float minLineWidth, float height)
バーコードの描画を行います。
draw メソッドとの違いは、バーコード全体の幅を指定するのではなく、バーを描画する一番細い線の幅を指定します。
draw メソッドに比べて、drawDirect メソッドと同様に精度の高いバーコードを描画することが可能です。ただし、バーコード全体の幅の調整が必要になります。
※この drawDelicate メソッドを使用してバーの最小幅を指定した場合に精度が高くなる理由は、直接、Graphics オブジェクトに線を描画するためです。
draw メソッドを使用して全体の幅を指定した場合、一旦、仮想空間に描画したバーコードを指定された全体の幅に対して当てはまるように Graphics オブジェクトに縮小描画しております。
minLineWidth 以外の引数は、(1) Draw メソッドと同様ですので割愛します。

(8) public void drawDelicate
(string code, float x, float y, float minLineWidth, float height, String imgFilePath)
(7) drawDelicate のオーバーロードメソッドで、バーコードを画像ファイルに出力します。
imgFilePath で指定したファイルへバーコードの描画を行います。
他の引数については、(1)の Draw メソッドと同様ですので割愛します。

3-1-4-3. プロパティ

◇次のプロパティは、一次元バーコードと同様です。
クラスとしても同じ Interface を実装しています。

(1) `public boolean getTextWrite()`
`public void setTextWrite(boolean value)`
true： 添字の描画を行う。(既定値)

(2) `public boolean getTextKintou ()`
`public void setTextKintou(boolean value)`
true： 添字の描画は、バーコード全体の幅に均等割付で行う。(既定値)

(3) `public Font getTextFont ()`
`public void setTextFont(Font value)`
添字のフォント。既定値は、”ＭＳ ゴシック　9 ポイント　標準”。

(4) `public float getRotateAngl ()`
`public void setRotateAngle(float value)`
回転角度を数値で指定。左下を軸に右回転して描画を行う。既定値は 0 度。

(5) `public int getKuroBarChousei ()`
`public void setKuroBarChousei (int value)`
黒バーの幅を微細調整(加減)するドット数を指定する。
マイナスの値で黒バーを細くすることも可能。既定値は 0 (pixel)。

(6) `public int getDPI ()`
`public void setDPI (int value)`
黒バーの微細調整を行うための DPI を指定する。既定値は 600DPI。

(7) `public int getImgMargin ()`
`public void setImgMargin (int value)`
バーコードを画像保存する際の
バーコード周囲の余白を pixel 単位で指定する。既定値余白は 1 (pixel)。

◇次のプロパティは、GS1 データバー(RSS)のみで使用するものです。

(1) `public DatabaType getSymbolType()`
`public setSymbolType(DatabaType value)`

GS1 データバーのタイプです。
限定型では使用しません。標準型・拡張型で次のように定義されています。
出力するデータバーのタイプを標準型・拡張型、各々で指定します。

GS1 Databa RSS 14 のタイプ	
enum Pao.BarCode.Gs1Databar14.DatabarType	
OMNIDIRECTIONAL	標準型
STACKED	二層型
STACKED_OMNIDIRECTIONAL	標準二層型

GS1 Databa RSS Expanded のタイプ	
enum Pao.BarCode.Gs1DatabarExpanded.DatabarType	
UNSTACKED	一層型
STACKED	多層型

3-1-5. 郵便カスタマバーコードクラスメンバ

3-1-5-1. コンストラクタ

初期処理を行う。
public YubinCustomer (java.awt.Graphics2D g)
通常のコンストラクタ： Graphics2D オブジェクトにバーコードを描画する際に使用してください。(印刷・画面へのバーコード出力)

・引数
java.awt.Graphics2D g
バーコードの描画を行う Graphics2D を指定します。

public YubinCustomer (String imgFilePath)
画像ファイルにバーコードを出力する際に使用してください。
ver 2.0 で追加されたコンストラクタです。

・引数
① String imgFilePath
バーコードを描画するデフォルトファイルパスを指定します。
※画像ファイルの拡張子より、画像フォーマットを特定します。

こちらのコンストラクタでインスタンスを作成した場合、
通常の draw メソッドで画像ファイルへバーコード出力を行います。
もし、draw するタイミングで出力画像ファイルを変更される場合は、
draw メソッドの引数に画像ファイルパスを指定できる方のオーバーロードメソッドを使用してください。

3-1-5-2. メソッド

(1) public void draw(String code, float x, float y, float point)

郵便カスタマバーコードの描画を行います。

・引数

① String code

描画を行うバーコードのコードを文字列で指定します。

コードは・・・

[郵便番号の数字部分 7 桁]+[郵便番号では不明部分の住所の英数字を「－」区切り]

で、指定してください。

例）〒116-0013 東京都荒川区西日暮里五丁目 37 番 5 号スタートアップオフィスＡ－207 号室

コード指定方法→「11600135-37-5-A-207」

② float x

描画位置の始点(左上)の X 座標を指定します。

③ float y

描画位置の始点(左上)の Y 座標を指定します。

④ float point

バーコード大きさを表すポイントを指定します。

ポイントは、目安の大きさです。大体 5～30 の範囲内の数値で指定してください。

・戻り値

なし

・例外の種類

public ErrYubinBadChar ()

半角英数字－(ハイフォン)以外の文字が使用されました。

(2) public void draw(String code, float x, float y, float point
, String imgFilePath)

郵便カスタマバーコードを画像ファイルに出力します。
(1)の draw メソッドのオーバーロードメソッドです。

imgFilePath で指定したファイルへバーコードの描画を行います。
ver 2.0 で追加されたオーバーロードメソッドです。
他の機能について draw メソッドと同様となります。

・引数

② String code

描画を行うバーコードのコードを文字列で指定します。
コードは・・・
[郵便番号の数字部分 7 桁]+[郵便番号では不明部分の住所の英数字を「－」区切り]
で、指定してください。
例）〒116-0013 東京都荒川区西日暮里五丁目 37 番 5 号スタートアップオフィスＡ－207 号室
コード指定方法→「11600135-37-5-A-207」

② float x

描画位置の始点(左上)の X 座標を指定します。

③ float y

描画位置の始点(左上)の Y 座標を指定します。

④ float point

バーコード大きさを表すポイントを指定します。
ポイントは、目安の大きさです。大体 5～30 の範囲内の数値で指定してください。

⑤ String imgFilePath

バーコード出力を行う画像ファイルパスを指定します。
※画像ファイルパスはコンストラクタで指定されているため不要ですが、出力する画像ファイルへバーコードを描画するたびに変更する場合に、こちらの画像ファイルパスを指定可能なメソッドをお使いください。

・戻り値

なし

・例外の種類

public ErrYubinBadChar ()

半角英数字－(ハイフォン)以外の文字が使用されました。

3-1-5-3. プロパティ

(1) public float getRotateAngl()
public void setRotateAngle(float value)
回転角度を数値で指定。左下を軸に右回転して描画を行う。
既定値は、０度。

(2) public int getDPI ()
public void setDPI (int value)
出力デバイスの解像度(DPI)を指定する。
印刷を行う場合はプリンタの解像度。
既定値は、600DPI。

(3) public int getImgMargin ()
public void setImgMargin (int value)
バーコードを画像保存する際の
バーコード周囲の余白を pixel 単位で指定する。
既定値余白は、1 (pixel)。

(4) public static float getdrawWidth(String code, float point)

※ver 2.0 以降、画像保存を行うことができるようになったため、このメソッドは基本的に不要なメソッドです。下位互換のために残してあります。

郵便カスタマバーコードを画像ファイルに保存するため、draw ()で実際に描画した時のバーコード幅を事前に取得するためのメソッドです。
実際に描画される郵便カスタマバーコード幅は事前にわからないためこのメソッドが用意されています。

・引数
① String code
描画を行うバーコードのコードを文字列で指定します。
コードは・・・
[郵便番号の数字部分 7 桁]+[郵便番号では不明部分の住所の英数字を「－」区切り]
で、指定してください。
例）〒116-0013 東京都荒川区西日暮里五丁目 37 番 5 号スタートアップオフィスＡ－207 号室
コード指定方法→「11600135-37-5-A-207」
描画位置の始点(左上)の Y 座標を指定します。
② float point
バーコード大きさを表すポイントを指定します。
ポイントは、5～30 の範囲内の数値で指定してください。

3-1-6. QR コード / DataMatrix / PDF417 クラスメンバ

3-1-6-1. QR コードコンストラクタ

初期処理を行う。

(1) public QRCode(java.awt.Graphics2D g)

通常のコンストラクタ： Graphics2D オブジェクトにバーコードを描画する際に使用してください。(印刷・画面へのバーコード出力)

・引数

java.awt.Graphics2D g

バーコードの描画を行う Graphics2D を指定します。

(2) public QRCode(String imgFilePath)

画像ファイルに QR コードを出力する際に使用してください。
ver 2.0 で追加されたコンストラクタです。

・引数

String imgFilePath

QR コードを描画するデフォルトファイルパスを指定します。
※画像ファイルの拡張子より、画像フォーマットを特定します。
こちらのコンストラクタでインスタンスを作成した場合、
通常の draw メソッドで画像ファイルへバーコード出力を行います。
もし、draw するタイミングで出力画像ファイルを変更される場合は、
draw メソッドの引数に画像ファイルパスを指定できる方のオーバーロードメソッドを使用してください。

3-1-6-2. DataMatrix コンストラクタ

初期処理を行う。

public DataMatrix (System.Drawing.Graphics g)

・引数

System.Drawing.Graphics g

DataMatrix の描画を行う **Graphics** を指定します。

・戻り値

なし。

・割込発生エラー

なし。

public DataMatrix (String imgFilePath, ImageFormat imgFormat)

画像ファイルに DataMatrix を出力する際に使用してください。

・引数

① String imgFilePath

バーコードを描画するデフォルトファイルパスを指定します。

② ImageFormat imgFormat

出力をする ImageFormat を指定します。プロパティの DPI を有効化する為には、png/jpeg フォーマットを指定してください。

・戻り値

なし。

・割込発生エラー

なし。

3-1-6-3. PDF417 コンストラクタ

初期処理を行う。

public Pdf417(System.Drawing.Graphics g)

・引数

System.Drawing.Graphics g

PDF417 の描画を行う **Graphics** を指定します。

・戻り値

なし。

・割込発生エラー

なし。

public Pdf417 (String imgFilePath, ImageFormat imgFormat)

画像ファイルに PDF417 を出力する際に使用してください。

・引数

① String imgFilePath

バーコードを描画するデフォルトファイルパスを指定します。

② ImageFormat imgFormat

出力をする ImageFormat を指定します。プロパティの DPI を有効化する為には、png/jpeg フォーマットを指定してください。

・戻り値

なし。

・割込発生エラー

なし。

3-1-6-4. QR コード / DataMatrix / PDF417 メソッド (Draw 系同一)

(1) `public void draw(String code, float x, float y, float width, float height)`

バーコードの描画を行います。

`public void draw(String code, float x, float y, float width, float height, String imgFilePath)`

バーコードを画像ファイルへ出力します。

・ 引数

① `String code`

描画を行うバーコードのコードを文字列で指定します。

② `float x`

描画位置の始点(左上)の X 座標を指定します。

③ `float y`

描画位置の始点(左上)の Y 座標を指定します。

③ `float width`

バーコードの全体の幅を指定します。

通常、height と、同じ値を指定します。

⑥ `float height`

バーコードの全体の高さを指定します。

通常、width と、同じ値を指定します。

⑥ `String imgFilePath`

バーコードを出力する画像ファイルパス。

※画像ファイルパスはコンストラクタで指定されているため不要ですが、出力する画像ファイルへバーコードを描画するたびに変更する場合に、こちらの画像ファイルパスを指定可能なメソッドをお使いください。

・戻り値

なし

・例外の種類

指定されたコードの文字数が、バーコードに格納できる文字数をオーバーした。

(2) public void drawDirect (String code, float x, float y, float width, float height)

バーコードの描画を行います。

public void drawDirect (String code, float x, float y, float width, float height , String imgFilePath)

バーコードを画像ファイルへ出力します。

指定幅以内で最も広い幅でバーコードを直接描画します。
ドット単位での描画精度を実現します。

・ 引数

① String code

描画を行うバーコードのコードを文字列で指定します。

② float x

描画位置の始点(左上)の X 座標を指定します。

③ float y

描画位置の始点(左上)の Y 座標を指定します。

④ float width

バーコードの全体の幅を指定します。
通常、height と、同じ値を指定します。

⑤ float height

バーコード全体の高さを指定します。
通常、width と、同じ値を指定します。

⑥ String imgFilePath

バーコードを出力する画像ファイルパス。
※画像ファイルパスはコンストラクタで指定されているため不要ですが、出力する画像ファイルへバーコードを描画するたびに変更する場合に、こちらの画像ファイルパスを指定可能なメソッドをお使いください。

・戻り値

なし

・例外の種類

指定されたコードの文字数が、バーコードに格納できる文字数をオーバーした。

(3) public void drawDelicate(String code, float x, float y, float minLineWidth)

バーコードの描画を行います。

public void drawDelicate(String code, float x, float y, float minLineWidth
, String imgFilePath)

バーコードを画像ファイルへ出力します。

draw メソッドとの違いは、バーコード全体の幅を指定するのではなく、バーを描画する一番細かい単位を指定します。
draw メソッドに比べて、drawDirect メソッドと同様に精精度の高いバーコードを描画することが可能です。その一方、バーコード全体の幅の調整が必要になります。
おおよその全体幅も指定したい場合は、本メソッドではなく drawDirect()メソッドをご利用ください。

・ 引数

① String code

描画を行うバーコードのコードを文字列で指定します。

② float x

描画位置の始点(左上)の X 座標を指定します。

③ float y

描画位置の始点(左上)の Y 座標を指定します。

④ float minLineWidth

バーコードを描画する最小値を指定します。

⑤ String imgFilePath

バーコードを出力する画像ファイルパス。

※画像ファイルパスはコンストラクタで指定されているため不要ですが、出力する画像ファイルへバーコードを描画するたびに変更する場合に、こちらの画像ファイルパスを指定可能なメソッドをお使いください。

・戻り値

なし

・例外の種類

指定されたコードの文字数が、バーコードに格納できる文字数をオーバーした。

(3) public void WriteSVG
(String code, float x, float y, float width, float height, String filePath)
SVG ファイルへの QR コードの出力を行います。

・ 引数

① String code
描画を行うバーコードのコードを文字列で指定します。

② float x
描画位置の始点(左上)の X 座標を指定します。

③ float y
描画位置の始点(左上)の Y 座標を指定します。

④ float width
バーコードの全体の幅を指定します。
ただし、SVG ファイルは、ブラウザ表示用のため、画面の pixel とバーコードの線が一致しないといけないため、指定された幅以下のサイズに最適化されます。
通常、height と、同じ値を指定します。

⑤ float height
バーコードのバーの高さを指定します。
ただし、SVG ファイルは、ブラウザ表示用のため、画面の pixel とバーコードの線が一致しないといけないため、指定された幅以下のサイズに最適化されます。
通常、width と、同じ値を指定します。

⑥ String filePath
SVG ファイルのファイル名をフルパスで指定してください。

・戻り値
なし

・例外の種類
public ErrQRCodeOverLenght ()
指定されたコードの文字数が、指定されたバージョンの QR コードに格納できる文字数をオーバーした。

3-1-6-5. QR コードプロパティ

(1) `public int getVersion()`
`public void setVersion(int value)`
バージョン 1～40 を指定・取得。

(2) `public String getErrorCorrect()`
`public void setErrorCorrect(String value)`
エラー訂正レベル：”L”/”M”/”Q”/”H” のいずれかを指定・取得。

(3) `public String getEncodeMode()`
`public void setEncodeMode(String value)`
エンコードモードを指定・取得します。
“N”:数字モード
“A”:英数字モード
その他:8bit byte モード
エンコードモードに漢字モードがありませんが、漢字の入力も「その他:8bit byte モード」を指定してください。”N”/”A”以外の文字であればなんでも OK です。

(4) `public float getRotateAngl()`
`public void setRotateAngle(float value)`
回転角度を数値で指定。左下を軸に右回転して描画を行う。
既定値は、０度。

(5) `public int getImgMargin ()`
`public void setImgMargin (int value)`
バーコードを画像保存する際の
バーコード周囲の余白を pixel 単位で指定する。
既定値余白は、1 (pixel)。

3-1-6-6. DataMatrix プロパティ

(1)
```
public DxCodeSize getCodeSize()
public void setCodeSize(DxCodeSize value)
```

シンボルコードサイズ　　既定値：DxCodeSize.DxSzAuto(自動)

(public enum) DxCodeSize.

DxSzRectAuto,	DxSz24x24,	DxSz88x88,
DxSzAuto,	DxSz26x26,	DxSz96x96,
DxSzShapeAuto,	DxSz32x32,	DxSz104x104,
	DxSz36x36,	DxSz120x120,
DxSz10x10,	DxSz40x40,	DxSz132x132,
DxSz12x12,	DxSz44x44,	DxSz144x144,
DxSz14x14,	DxSz48x48,	
DxSz16x16,	DxSz52x52,	DxSz12x36,
DxSz18x18,	DxSz64x64,	DxSz16x36,
DxSz20x20,	DxSz72x72,	DxSz16x48
DxSz22x22,	DxSz80x80,	

(2)
```
public String getStringEncoding()
public void setStringEncoding(String value)
```
全角エンコーディング（”utf-8”, “shift-jis”, etc..） 既定値：utf-8

(3)
```
public float getRotateAngl()
public void setRotateAngle(String value)
```
回転角度を数値で指定。左下を軸に右回転して描画を行う。
既定値は、０度。

(4)
```
public int getImgMargin ()
public void setImgMargin(String value)
```
バーコードを画像保存する際の
バーコード周囲の余白を pixel 単位で指定する。
既定値余白は、1 (pixel)。

3-1-6-6. PDF417 プロパティ

(1) `public SIZE_KIND getSizeKind()`
`public void setSizeKind(SIZE_KIND sizeKind)`
サイズ種別　…データ列数・行数決定方法

```
/*
 *  データ列数・行数決定方法
 */
public enum SIZE_KIND
{
    /*
     * 自動サイズ(アスペクト比より決定)
    */
    AUTO
    ,
    /*
     * 指定列数(dataColumns)に従う
    */
    COLUMNS
    ,
    /*
     * 指定行数(dataRows)に従う
    */
    ROWS
    ,
    /*
     * 指定列数・行数両方に従う(長方形の最小値)
    */
    COLUMNS_AND_ROWS
}
```

(2) `public int getCodeRows()`
`public void setCodeRows(int codeRows)`
行数　…出力データ行数指定
サイズ種別が、
出力行数指定の場合＝(自動サイズでない・列数指定でない場合)有効
3～90　　既定値：5

(3) `public int getCodeColumns()`
`public void setCodeColumns(int codeColumns)`
列数　…出力データカラム数指定
サイズ種別が、
出力列数指定の場合＝(自動サイズでない・行数指定でない場合)有効
1～30　　既定値：5

(4) `public int getErrorLevel()`
`public void setErrorLevel(int errorLevel)`
エラー訂正レベル　0～8　　既定値：0

(5) `public boolean getUseAutoErrorLevel()`
`public void setUseAutoErrorLevel(boolean useAutoErrorLevel)`
エラー訂正レベル自動決定
自動でエラー訂正レベルを決定(する・しない)　　既定値：true(する)

(6) `public float getAspectRatio()`
`public void setAspectRatio(float aspectRatio)`
縦横アクセプト比　既定値：0.5
シンボルの縦横比、シンボル アスペクト レシオ（比）

(7) `public String getStringEncoding()`
`public void setStringEncoding(String value)`
全角エンコーディング（”utf-8”, “shift-jis”, etc..）既定値：utf-8

(8) `public float getRotateAngl()`
`public void setRotateAngle(String value)`
回転角度を数値で指定。左下を軸に右回転して描画を行う。
既定値は、０度。

(9) `public int getImgMargin ()`
`public void setImgMargin(String value)`
バーコードを画像保存する際の
バーコード周囲の余白を pixel 単位で指定する。
既定値余白は、1 (pixel)。

3-2. 使用例(CODE39 の例)

ここでは、CODE39 のバーコード印刷を例にして簡単な使用方法を説明します。

このサンプルプログラムは、swing を利用しています。

```
//ボタンをクリックした時の処理
public class myListener implements ActionListener {
    public void actionPerformed(ActionEvent e){


        Graphics g = null;

        if(e.getActionCommand() == "execute") {
        //画面に描画

               g = getGraphics();
               drowBarCode(g);

        } else {
        //印刷

           try
           {
              PrinterJob job = PrinterJob.getPrinterJob();
              job.setPrintable(canvas);
              if (job.printDialog(attributes))
              {
                 job.print(attributes);
              }
           }
           catch (PrinterException exception)
           {
              JOptionPane.showMessageDialog(
                         BarApp.this, exception);
           }
        }

    }

}

//バーコードの描画
void drowBarCode(Graphics g)
{
    try {
        g.setColor(Color.WHITE);
        g.fillRect(0,125,1000,575);

        Code39 barcode = new Code39((Graphics2D) g);
        barcode.draw("1234567890", 10, 10, 200, 50);
    } catch (Exception ex) {
        lblErr.setText(ex.getMessage());
        ex.printStackTrace();
    }
}
```

詳しくは、サンプルプログラムを複数ご用意させていただきましたので、じっくり弄繰り回してください。

3-3. サンプルプログラム

java で作成した Barcode.jar を利用したサンプルプログラムを 5 本ご用意いたしました。

http://www.pao.ac/barcode.jar/#download

より、Barcode.jar(バージョン).zip をダウンロードして解凍すると、Pao.Barcode.jar と一緒に 5 つのフォルダが作成されると思います。それぞれに、サンプルプログラムが入っております。Pao.Barcode.jar を 5 つのサンプルプログラムで「ビルドパスー外部アーカイブの追加」をして、起動してみてください。

(1) バーコードの印刷・プレビューサンプル

eclipse から BarApp フォルダを指定して起動する事により、このサンプルプログラムを開くことができます。

サンプルプログラムは、Barcode.jar の機能をフルに利用した印刷・プレビュー処理を実現しています。サンプルプログラムの割には、市販のバーコード作成ソフトにも見劣りしない多くの機能を実装しております。このままでも色々な用途があると思いますが、是非、弄繰り回して改造して遊んでください。

(2) QR コードの描画・印刷・プレビューサンプル

eclipse から QrApp フォルダを指定して起動する事により、このサンプルプログラムを開くことができます。

サンプルプログラムは、Barcode.jar の機能をフルに利用した印刷・プレビュー処理を実現しています。

(3) SVG / SVGZ 出力・ブラウザ表示サンプル

eclipse から QRSvg フォルダを指定して起動する事により、このサンプルプログラムを開くことができます。

QR コードの SVG /SVGZ 出力とブラウザ表示を行うサンプルです。

(4) アプレットサンプル

eclipse から QRApplet フォルダを指定して起動する事により、このサンプルプログラムを開くことができます。

QR コードのアプレット出力を行うサンプルです。

(5) JSP を使って QR コードブラウザ出力するサンプル

http://www.pao.ac:8080/examples/barcode.jar/QRJsp.html

にて、このサンプルの動作確認は行えます。

参考のため、ソースコードを QRJsp フォルダにおいてあります。

是非、環境を作って、お客様の環境で動作確認してください。

4. 使用条件等

4-1. お試し版と製品版

http://www.pao.ac/barcode.jar/#download
に試用版として最新バージョンを常にご用意させていただいております。
試用版の制限は、バーコードに控えめに「SAMPLE」という文字が入ります。

製品版は、Barcode.jar を購入(ユーザ登録)された方にオーダーメイドで作成し、メールでお送りいたします。
バージョンアップの際は、Web サイトにてお知らせいたします。
アップグレードをご希望されるお客様は、メールにてご依頼ください。
最新版をメールにてお送りいたします。無償でございます。

4-2. 使用許諾

Barcode.jar の使用について、Barcode.jar の使用者(以下「利用者様」と称します)と有限会社パオ・アット・オフィス(以下「弊社」と称します)は、以下の各項目についての内容に同意するものとします。

1.Barcode.jar の使用に関する使用許諾書

この使用許諾書は、利用者様がお使いのパソコンにおいて、Barcode.jar を使用する場合に同意しなければならない契約書です。

2.使用許諾書の同意

利用者様が Barcode.jar を使用する時点で、本使用許諾書に同意されたものとします。同意されない場合は、Barcode.jar を使用する事はできません。

3.ライセンス(使用権)の購入

利用者様が Barcode.jar の製品版を使用して開発を行う場合には、1 台の開発用コンピュータで Barcode.jar を使用するにあたり、1 ライセンスを購入する必要があります。

お客様環境等、開発コンピュータでないマシンで Barcode.jar を使用する場合ライセンスは必要ありません。ランタイムライセンスフリーでございます。

4.著作権

Barcode.jar 及の著作権は、いかなる場合においても弊社に帰属いたします。

5.免責

Barcode.jar の使用によって、直接的、あるいは、間接的に生じた、いかなる損害に対しても、弊社は補償賠償の責任を負わないものとします。

6.禁止事項

Barcode.jar 及びその複製物を第三者に譲渡・貸与する事は出来ません。Barcode.jar を開発ツールとして再販/再配布することを禁止します。なお、モジュールとして組み込みを行い再販/再配布する場合は、開発ツールとしての再販/再配布には含まれませんので、OK です。

7.保証の範囲

弊社は Barcode.jar の仕様を予告無しに変更することがあります。その場合の利用者様に対する情報提供は、弊社 WebSite にて行う事とします。

8.適用期間

本使用許諾条件は利用者様が Barcode.jar を使用した日より有効です。利用者様が本使用許諾条件のいずれかの条項に違反した場合、又は、本許諾条件に同意出来ない場合は、利用者様は Barcode.jar を一切使用出来ないものとします。

4-3. 代金支払い方法(ユーザ登録の方法)

Barcode.jar の製品版をご利用頂ける場合は、ライセンスを購入して頂く必要があります。ライセンス形態及び代金支払方法は以下のとおりです。

- 必要なライセンス数の数え方
  - Barcode.jar で開発を行うパソコンの台数。
- 1 ライセンス当たりの価格
  - 20,000 円(税込:22,000 円)
  - ソースコード付：92,000 円(税込:99,360 円)
    - ✧ バグフィックス等のバージョンアップは原則として無償とさせて頂きます。
    - ✧ 大幅な機能追加等によるバージョンアップの場合には別ライセンスとさせていただく場合がございます。
    - ✧ 本価格は Barcode.jar の使用権に対するも　のです。カスタマイズや保守等の費用は一切含まれておりません。
- お支払方法･･･先払い方法。後払い(納品後支払)方法は後に説明がございます。
  (22,000 円 or 99,360 円×ライセンス数)を下記口座へ銀行振込、
  または、郵便振替による送金をして下さい。

銀行名	支店名　(コード)	口座番号	名義
三菱ＵＦＪ銀行	新宿支店　(341)	普通 3831891	ユ）パオアツトオフイス
ジャパンネット銀行	すずめ支店　(002)	普通 6461359	ユ）パオアツトオフイス

郵便口座番号	名義
００１５０－０－５７６８４５	有限会社　パオ・アット・オフィス

  - ✧ 振込手数料は利用者様負担でお願い致します。

- お支払いの通知と製品の送付
  - 振り込みが完了した時点で、必ず弊社 WebSite の「Barcode.jar 入金連絡フォーム」から入金のご連絡をお願いいたします。
    http://www.pao.ac/barcode.jar/buy.html#form
  - 弊社では上記連絡を受けて入金確認を行い、Barcode.jar の製品自体を利用者様へ電子メールにてお送りさせて頂きます。

- 見積書/納品書/請求書/領収証の発行、納品後のお支払いについて
  見積書/納品書/請求書/領収証の発行は可能でございます。本製品納品後のお支払いも可能でございます。
  http://www.pao.ac/barcode.jar/buy.html
  上記サイトでの手続きにより、弊社からの見積書/納品書/請求書/領収証の発行、及び、納品後にお客様からお支払いが行えるようになっております。