Java 用 バーコード作成ツール

Barcode.jar

説明書

Version 2.0.0

2013 年 4 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。

また ver 2.0 以降で、画像ファイルをバーコードのアウトプットとすることができるようになりました。

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。

2013 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	JDK1.4 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	JDK1.4 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、

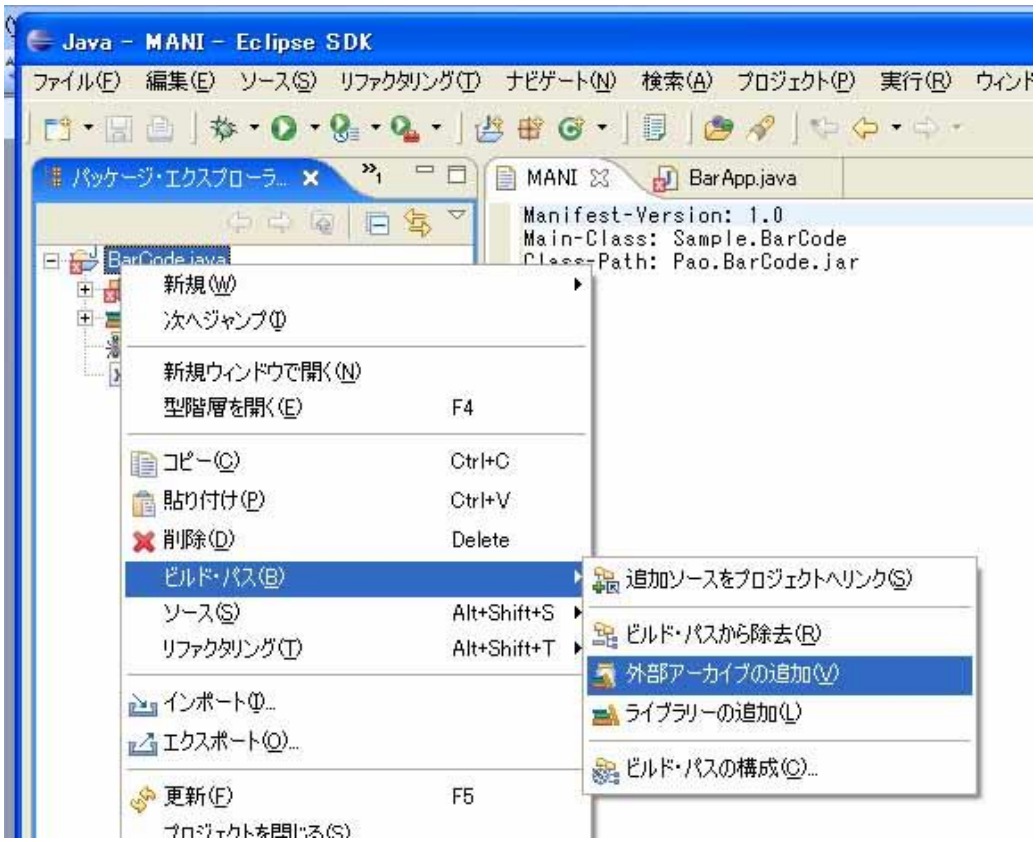
http://www.pao.ac/barcode.jar/#download

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。

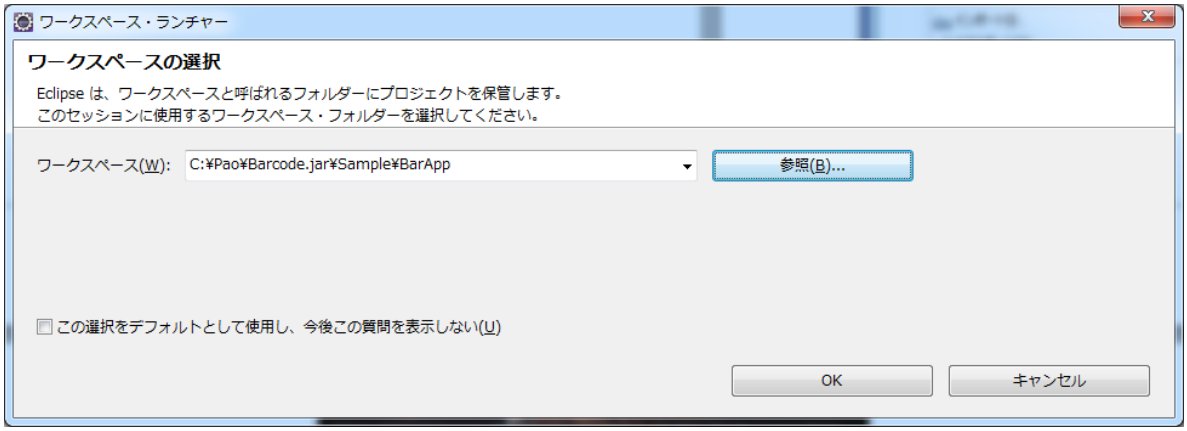
eclipse を使用する場合、「ビルドパス－外部アーカイブの追加」で、Pao.Barcode.jar を追加してください。

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。



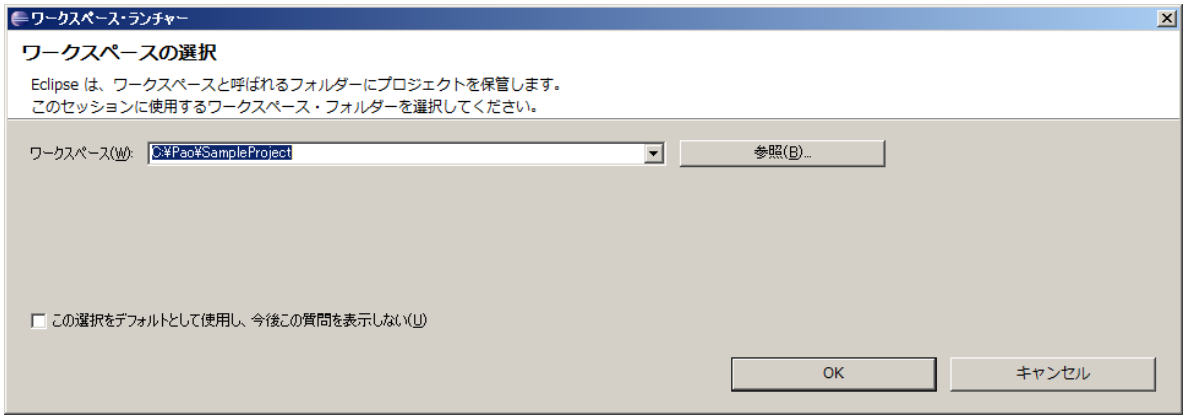
1-3. サンプルプログラムの使用方法

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。

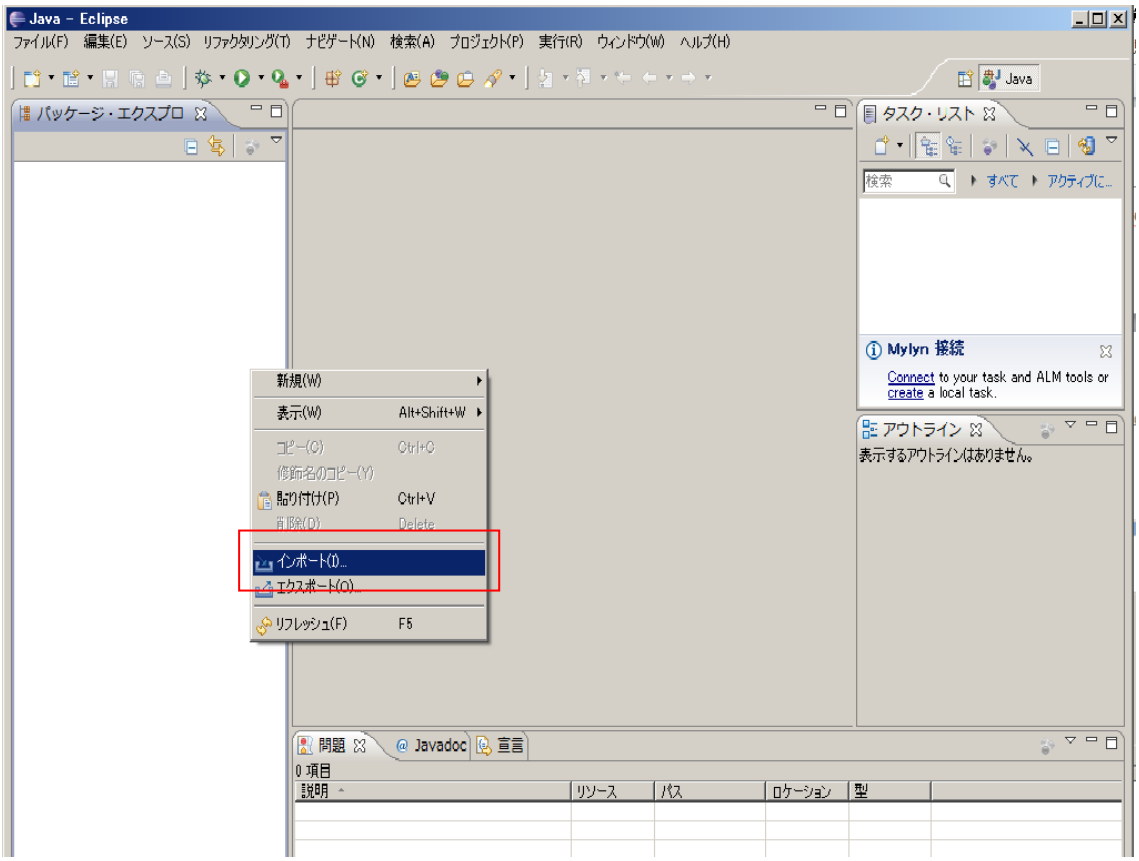


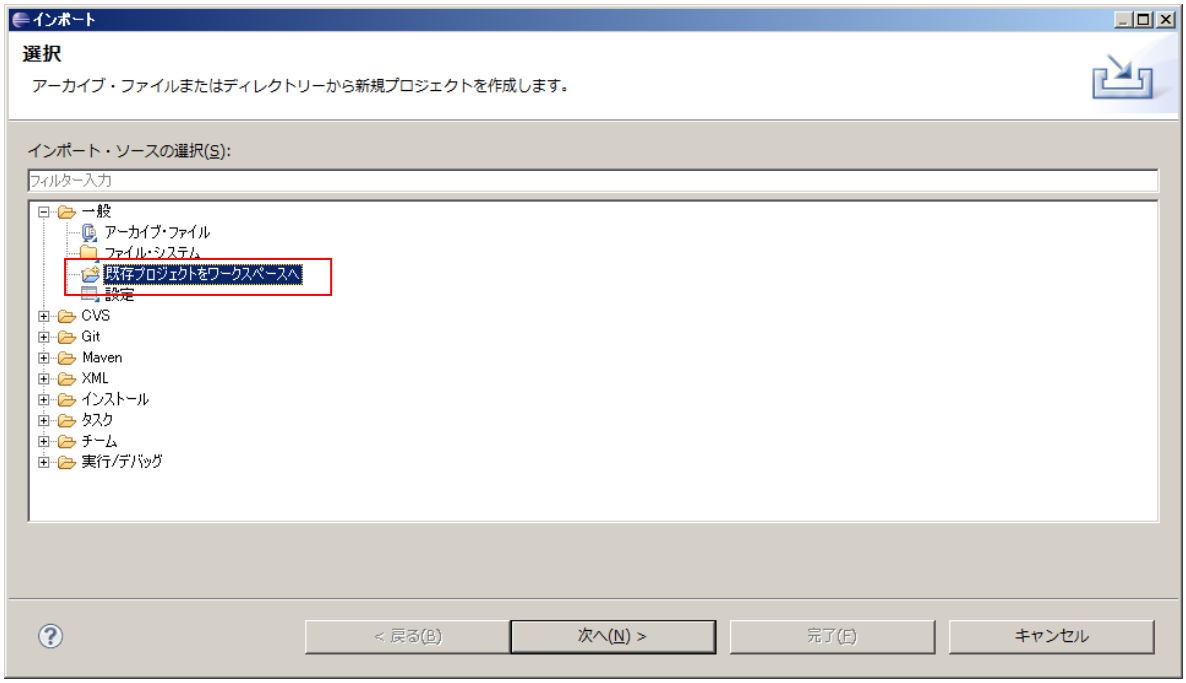
以降は、それではうまくサンプルワークスペースを開けない場合の操作の説明です。

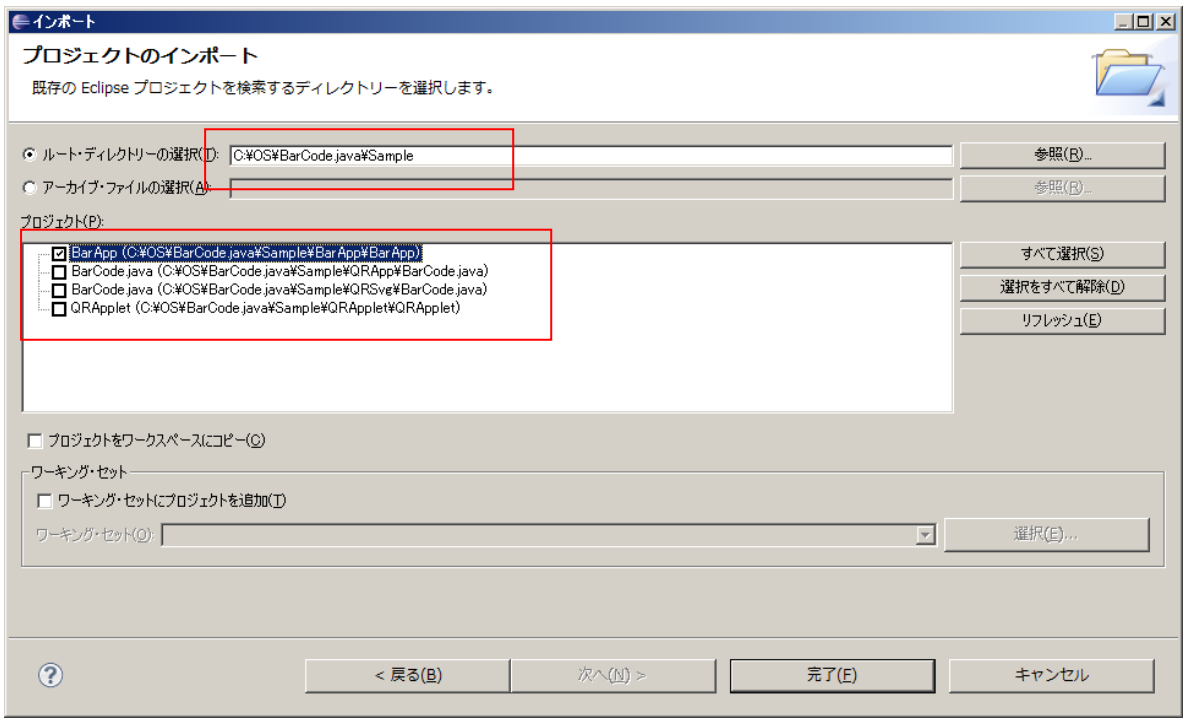
（1） 新規のワークスペースを作成してください。



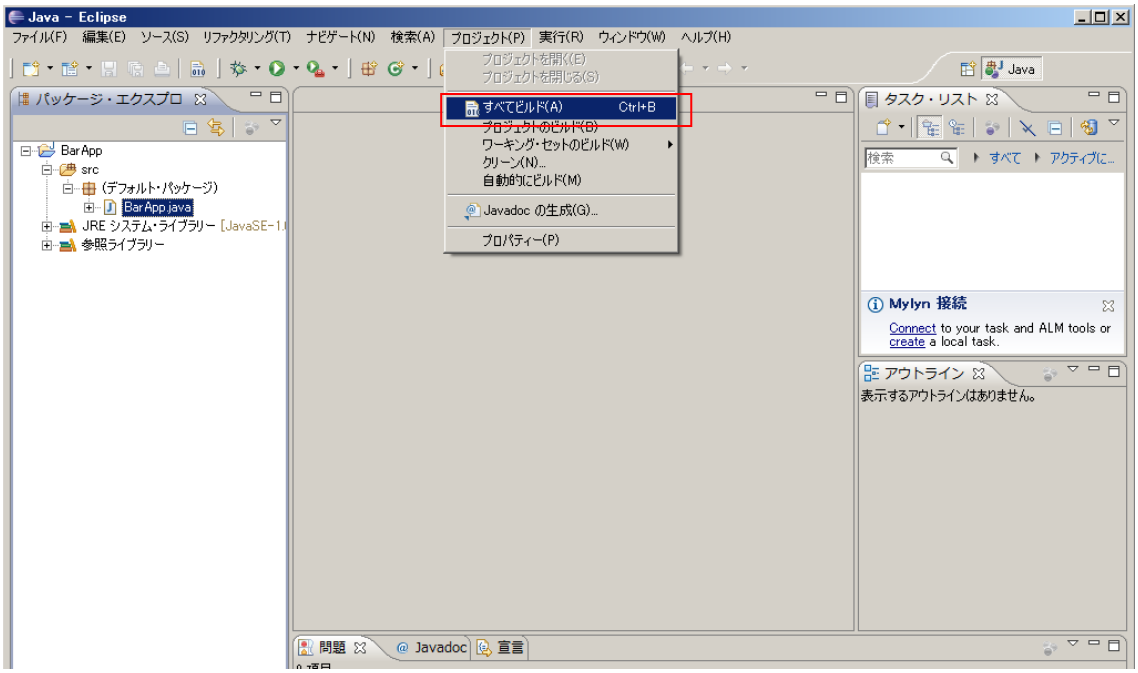
（２） サンプルプロジェクトをインポートしてください。
（こちらの例では、Sample¥BarApp を使用します。）



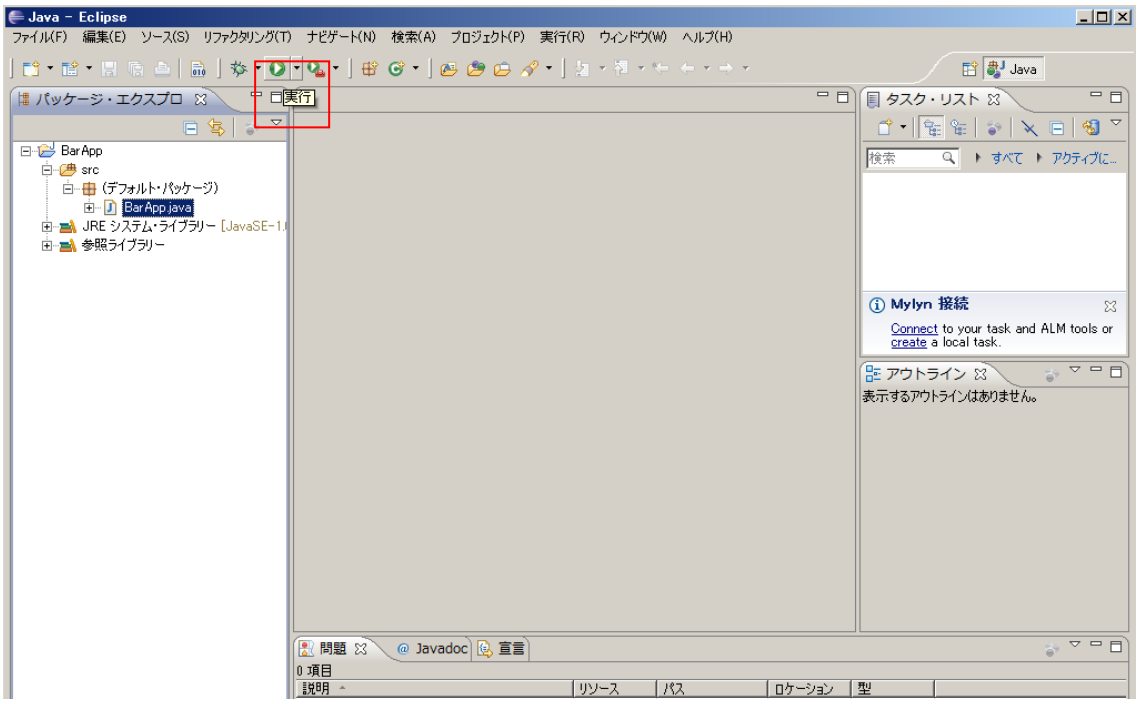



インポート
プロジェクトのインポート
既存の Eclipse プロジェクトを検索するディレクトリーを選択します。
ルート・ディレクトリーの選択(T):
C:¥OS¥BarCode.java¥Sample
参照(R)...
アーカイブ・ファイルの選択(A):
参照(R)...
プロジェクト(P):
BarApp (C:¥OS¥BarCode.java¥Sample¥BarApp¥BarApp)
BarCode.java (C:¥OS¥BarCode.java¥Sample¥QRApp¥BarCode.java)
BarCode.java (C:¥OS¥BarCode.java¥Sample¥QRSvg¥BarCode.java)
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 11 種類のバーコードの作成が可能です。

(1) JAN13(EAN13)
(2) JAN8(EAN8)
(3) ITF(インターリーブド 2 of 5)
(4) Matrix 2 of 5
(5) NEC 2 of 5
(6) NW7(Codebar)
(7) Code39
(8) Code128
(9) UCC/EAN128(GS1-128)
(10) コンビニエンス料金代理収納用バーコード(GS1-128)
(11) 郵便カスタマバーコード
(12) QR コード

※郵便カスタマバーコード・QR コード以外は、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、ver 2.0.0 以降に追加した機能で、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。

2-2. 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。

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)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) 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／NW7(Codebar) 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、EAN128 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)
- エラー訂正レベル(L,M,Q,H)
- エンコードモード

(N：数字モード A：英数字モード　その他：漢字等、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。※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()

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.ITF**
- **Pao.BarCode.Matrix2of5**
- **Pao.BarCode.NW7**
- **Pao.BarCode.Code39**
- **Pao.BarCode.Code128**
- **Pao.BarCode.EAN128**
- **Pao.BarCode.YubinCustomer**
- **Pao.BarCode.QRCode**

コンビニバーコード(EAN128 に含まれる)・郵便カスタマ(YubinCustomer)・QR コード(QRCode) 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(EAN128 クラス)
- 郵便カスタマバーコードクラス
- QR コードのクラス

の４つに分けて説明いたします。

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 ITF(java.awt.Graphics2D g)
(4) public Matrix2of5(java.awt.Graphics2D g)
(5) public NEC2of5(java.awt.Graphics2D g)
(6) public NW7(java.awt.Graphics2D g)
(7) public Code39(java.awt.Graphics2D g)
(8) public Code128(java.awt.Graphics2D g)
(9) 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 ITF (String imgFilePath)
(4) public Matrix2of5 (String imgFilePath)
(5) public NEC2of5 (String imgFilePath)
(6) public NW7 (String imgFilePath)
(7) public Code39 (String imgFilePath)
(8) public Code128 (String imgFilePath)
(9) 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。

○ ITF

⑦ 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

⑧ 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

⑨ 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

⑩ 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

⑪ 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。

○ 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。

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

(10) public boolean getTextKintou ()
public void setTextKintou(boolean value)
draw： 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)

(11) 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”。

(12) 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。

(13) public boolean getDispStartStopCode ()
public void setDispStartStopCode (boolean value)
Code39/NW7 のみ使用可能なプロパティ
draw： 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)

(14) 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)。

(15) 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。

(16)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)。

(17) public static float getdrawDirectWidth
(String code, float width, 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。

バーコードを画像ファイルに保存するため、drawDirect()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。
バーコード幅は事前に指定した幅以内で出力デバイスのドットのはまるいちばん大きな幅が採用され描画されます。
そのため、実際に描画される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、drawDirect()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。
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。
⑤ bool flgKintou (Jan13 のみ)
JAN コード(13 ケタ)の均等割り付けするかしないか？を指定する。
均等割り付けしない場合、巷にある商品コードのようにいちばん左端の添え字がバーコードの外側に出力されるためバーコードの幅が変わってくる。そのため、JAN コード(13)桁に限りこのパラメータを指定してください。draw=均等割りする / false=均等割りしない(巷の商品コード)

(18) public static float getdrawDelicateWidth
(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。

バーコードを画像ファイルに保存するため、drawDelicate()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。
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、drawDelicate()で指定したとおりに描画されます。)
ただし、このメソッドで取得できる幅はあくまでもバーコードの幅(モジュール幅)です。添字部分を含んでおりません。添字部分を考慮する場合は、取得した幅よりも少し大きめの画像サイズとするなどの調整をしてください。

・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。
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。

⑤ bool flgKintou (Jan13 のみ)
JAN コード(13 ケタ)の均等割り付けするかしないか？を指定する。
均等割り付けしない場合、巷にある商品コードのようにいちばん左端の添え字がバーコードの外側に出力されるためバーコードの幅が変わってくる。そのため、JAN コード(13)桁に限りこのパラメータを指定してください。draw=均等割りする / false=均等割りしない(巷の商品コード)

3-1-2-3. 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 (EAN128 クラス)

(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 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

⑥ **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 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」

⑦ **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。

⑧ **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。

⑨ **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. 郵便カスタマバーコードクラスメンバ

3-1-3-1. 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。

3-1-3-2. 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) 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。

(4) 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。

(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)。

(6) 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。

3-1-4. QR コードクラスメンバ

3-1-4-1. 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。

3-1-4-2. メソッド

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

QR 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)`

QR コードを画像ファイルへ出力します。(ver 2.0 追加メソッド)

・ 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`

QR 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`

QR 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`

QR 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 QR 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 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。

(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)

QR コードを画像ファイルへ出力します。(ver 2.0 追加メソッド)

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 QR 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

QR 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

QR 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

QR 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 QR 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 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) 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)

QR コードを画像ファイルへ出力します。(ver 2.0 追加メソッド)

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
QR 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 QR 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 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) public **static float getdrawDelicateWidth**(String code, float width, 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。

QR コードを画像ファイルに保存するため、drawDelicate()で実際に描画した時の QR コード幅を事前に取得するためのメソッドです。
QR 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。
(QR コードの高さは、QR コード幅と同一であることを前提にしております。)

・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。
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。

(4) public void WriteSVG
(String code, float x, float y, float width, float height, String filePath)
SVG ファイルへのバー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。

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

(6) 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。

(7) 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。

(8) 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。

(9) 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)。

3-2. 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。

QR コードのお試し版の制限は、指定したコードの先頭に半角の「9」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 人または 1 台のコンピュータで 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。同時に複数の人が複数のコンピュータで使用する場合はどちらか少ない方のライセンスを購入しなければなりません。

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 を使用するするパソコンの台数。
- 1 ライセンス当たりの価格
  - 20,000 円(税込:21,000 円)
  - ソースコード付：95,000 円(税込:99,750 円)
    - ✧ 本価格には消費税を含みません。
    - ✧ 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。

(20,000 円 or 95,000 円×ライセンス数)+5%(消費税)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	口座番号	名義
三菱東京 UFJ 銀行	新宿	普通 3831891	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、納品後のお客様からのお支払いを行えるようになっております。