Forte™ for Java™ 4, Mobile Edition
チュートリアル

Forte for Java 4

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 U.S.A.
650-960-1300

Part No. 816-7460-10
2002 年 6 月 Revision A

原典： *Forte for Java 4, Mobile Edition Tutorial*
Part No: 816-5081-10
Revision A

目次

図目次

はじめに

このマニュアルは、Forte™ for Java™ 4、Mobile Edition の統合開発環境 (IDE) のためのチュートリアルです。IDE を学習する読者を対象としています。このチュートリアルには、次の項目が含まれています。

- タスクの設定 - ファイルシステムのマウント、パッケージの作成、エミュレータのインストール、デフォルトエミュレータや各エミュレータのデフォルトデバイススキンの設定など
- MIDlet の作成、コンパイルおよびテスト
- MIDlet スイートの作成と、 スイートへのMIDlet の追加
- MIDlet のデバッグ
- プロジェクトの設定

このマニュアルで説明しているプログラム例は、実際に作成することができます。作業環境については、以下の Web サイトにあるリリースノートを参照してください。

`http://sun.co.jp/forte/ffj/documentation/index.html`

使用するプラットフォームによって、マニュアルに掲載している画面イメージと異なることがあります。その場合でも表示上の違いはわずかであり、内容を理解するのには問題ありません。ほとんどの手順で Forte for Java のユーザーインタフェースを使用しますが、場合によっては、コマンド行にコマンドを入力する必要があります。その場合は、次のように、Microsoft Windows の「コマンドプロンプト」ウィンドウでのプロンプトと構文が例として示されています。

```
c:\>cd MyWorkSpace\MyPackage
```

UNIX® や Linux 環境では、次のようなプロンプトとなり、¥ マーク (またはバックスラッシュ) ではなくスラッシュを使用します。

```
% cd MyWorkSpace/MyPackage
```

お読みになる前に

このチュートリアルを使用するためには、Forte for Java 4, Mobile Edition をインストールする必要があります。インストール方法に関する情報は、『Forte for Java 4, Mobile Edition インストールガイド』を参照してください。

Forte for Java 4, Mobile Edition IDE（以降、Mobile Edition IDE）は、Forte for Java 4 IDE に Java 2 Micro Edition (J2ME™) プラットフォームおよび J2ME Wireless Toolkit (WTK) を統合したものです。Mobile Edition IDE は、携帯端末や無線機器のアプリケーションの開発をサポートします。

現在 IDE は、CLDC(Connected, Limited Device Configuration) 上の MIDP(Mobile Information Device Profile) 用の J2ME プラットフォームをサポートしています。CLDC は、リソースに制限がある小さな装置のために設計されたもので、移植可能な最小容量の Java 環境が標準化されています。CLDC は、Sun の KVM (K Virtual Machine) 上で実行されます。MIDP は、CLDC で動作する Java API のセットで携帯電話や双方向携帯端末のようなモバイル情報装置に対して、完全な J2ME アプリケーションの実行時環境を提供します。MIDP API はユーザーインタフェース、永続記憶、ネットワーク接続およびアプリケーションのライフサイクルといったさまざまな問題を扱う標準的な方法を提供します。

このチュートリアルは、以下の知識があることを前提としています。

- Java プログラミング言語
- Java 2 Micro Edition プラットフォーム
- MIDP アプリケーション
- J2EE アプリケーションアセンブリと配備の概念

また、J2ME テクノロジとワイヤレスの概念に関する知識も要求されます。詳細情報は次の Web サイトを参照してください。

`http://java.sun.com/j2me/`

注 - Sun では、本マニュアルに掲載した第三者の Web サイトのご利用に関しましては責任はなく、保証するものでもありません。また、これらのサイトあるいはリソースに関する、あるいはこれらのサイト、リソースから利用可能であるコンテンツ、広告、製品、あるいは資料に関して一切の責任を負いません。Sun は、これらのサイトあるいはリソースに関する、あるいはこれらのサイトから利用可能であるコンテンツ、製品、サービスのご利用あるいは信頼によって、あるいはそれに関連して発生するいかなる損害、損失、申し立てに対する一切の責任を負いません。

このマニュアルの構成

第 1 章では、Forte for Java 4, Mobile Edition の設定に必要な情報、MIDlet と MIDlet スイートの作成およびテストのための Mobile Edition IDE の使用方法を説明します。

第 2 章では、Mobile Edition IDE デバッガを使用して MIDlet アプリケーションをデバッグする方法を説明します。

第 3 章では、OTA (Over-the-Air) のサポートやプロジェクトの操作を含む、より高度な機能について説明します。

付録 A では、サンプルの ConverterMIDlet アプリケーションと Java クラスのソースコードをまとめています。

書体と記号について

次の表と記述は、このマニュアルで使用している書体と記号について説明しています。

書体または記号	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コーディング例。	.login ファイルを編集します。 ls -a を使用してすべてのファイルを表示します。 machine_name% You have mail.
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して表わします。	machine_name% **su** Password:
AaBbCc123 または ゴシック	コマンド行の可変部分。実際の名前または実際の値と置き換えてください。	rm *filename* と入力します。 rm **ファイル名** と入力します。
『 』	参照する書名を示します。	『Solaris ユーザーマニュアル』
「 」	参照する章、節、または、強調する語を示します。	第 6 章「データの管理」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合、バックスラッシュは、継続を示します。	machinename% grep '^#define \ XV_VERSION_STRING'
➤	階層メニューのサブメニューを選択することを示します。	作成:「返信」➤「送信者へ」

シェルプロンプトについて

シェル	プロンプト
UNIX の C シェル	`machine_name%`
UNIX の Bourne シェルと Korn シェル	`machine_name$`
スーパーユーザー (シェルの種類を問わない)	#

関連マニュアル

Forte for Java のマニュアルは、Acrobat Reader (PDF) ファイル、オンラインヘルプ、サンプルアプリケーションの `Readme` ファイル、Javadoc™ 文書の形式で提供しています。

オンラインで入手可能なマニュアル

次のマニュアルは、Forte for Java のポータルサイトおよび `docs.sun.com` の Web サイトから入手できます。

Forte for Java ポータルサイトでのマニュアルの入手先は、`http://sun.co.jp/forte/ffj/documentation/index.html` です。`docs.sun.com` の URL は、`http://docs.sun.com` です。

- リリースノート (HTML 形式)

  Forte for Java の Edition ごとに用意されています。このリリースでの変更情報と技術上の注意事項を説明しています。

- インストールガイド (PDF 形式)

  Forte for Java の Edition ごとに用意されています。対応プラットフォームへの Forte for Java のインストール手順を説明しています。さらに、システム要件、アップグレード方法、Web サーバーやアプリケーションサーバーのインストール、コマンド行での操作、インストールされるサブディレクトリ、Javadoc の設定、データベースの統合、アップデートセンターの使用方法などが含まれます。

- Forte for Java プログラミングシリーズ (PDF 形式)

  Forte for Java の各機能を使用して優れた J2EE アプリケーションを開発するための方法を詳細に説明しています。

  - 『Web コンポーネントのプログラミング』

    JSP ページ、サーブレット、タグライブラリを使用し、クラスやファイルをサポートする Web アプリケーションを J2EE Web モジュールとして構築する方法を説明しています。

  - 『J2EE アプリケーションのプログラミング』

    EJB モジュールや Web モジュールを J2EE にアセンブルする方法を説明しています。また、J2EE アプリケーションの配備や実行についても説明しています。

  - 『Enterprise JavaBeans コンポーネントのプログラミング』

    Forte for Java の EJB ビルダーウィザードや、他の IDE コンポーネントを使用し、EJB コンポーネント (コンテナ管理や Bean 管理の持続性の機能を持つセッション Bean やエンティティ Bean) を作成する方法を説明しています。

  - 『Web サービスのプログラミング』

    Web サービスモジュールが提供するツールを使用して Web サービスを構築する方法を説明しています。Web サービスは、XML (Extensible Markup Language) 文書の形式で提供されるアプリケーションビジネスサービスであり、HTTP を介して配信されます。

  - 『Java DataBase Connectivity の使用』

    Forte for Java の JDBC 生産性向上ツールを使用し、JDBC アプリケーションを作成する方法について説明しています。

- Forte for Java チュートリアル (PDF 形式)

  Forte for Java の Edition ごとに用意されています。Forte for Javaのツールを使用してアプリケーションを作成する方法を、順を追って説明しています。

  チュートリアルアプリケーションは、以下のサイトからもアクセスできます。
  `http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`

`docs.sun.com` (`http://docs.sun.com`) の Web サイトでは、他のサンのマニュアルの参照、印刷、購入をすることもできます。

オンラインヘルプ

オンラインヘルプは、Forte for Java 開発環境内から参照できます。ヘルプキー (Solaris オペレーティング環境では Help キー、Windows および Linux 環境では F1 キー) を押すか、「ヘルプ」 >「内容」を選択します。ヘルプの目次を表示したり、検索することができます。

プログラム例

Forte for Java の機能を紹介したプログラム例とチュートリアルアプリケーション (各 Edition のチュートリアルで説明されているアプリケーションを含む) を、以下の Forte for Java のポータルサイトからダウンロードすることができます。

`http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`

Javadoc

Javadoc 形式のマニュアルは、Forte for Java の多くのモジュールに用意されており、IDE の中で参照できます。このマニュアルの使用方法については、リリースノートを参照してください。IDE を起動すると、エクスプローラの「Javadoc」タブで Javadoc マニュアルを参照できます。

ご意見の送付先

Sun のマニュアルについてのご意見やご要望をお寄せください。今後のマニュアル作成の参考にさせていただきます。次のアドレスまで電子メールをお送りください。

`docfeedback@sun.com`

電子メールのタイトルに、対象マニュアルの Part No. を明記してください。

第1章

MIDlet と MIDlet スイートの作成

このチュートリアルでは、通貨換算アプリケーションの作成方法を学びます。チュートリアルに入る前に、以下の Forte for Java のポータルサイトからソースコードをダウンロードしてください。

`http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`

この章では、Forte for Java 4, Mobile Edition ツールに取り組みます。具体的には、このツールを使用した J2ME MIDP/CLDC アプリケーションの作成に必要な作業を、以下の順に説明します。

- コードを作成する
- デバイスエミュレータでコードをテストする
- MIDlet をパッケージ化する

このチュートリアルで取り上げる Converter という名前の通貨換算アプリケーションは、ある通貨の金額を別の通貨の金額に変換します。"from" や "to" の通貨カテゴリに表示する通貨 (ユーロや円、ドルなど) は変更することができます。アプリケーションは "from" に任意の通貨の金額を入力させ、別の通貨でその金額に相当する金額を表示します。このサンプルアプリケーションは 4 つの Java ファイルを含んでいます。

- `ConverterMIDlet.java` - MIDlet クラスのコード
- `Converter.java` - 通貨の換算を行う Java クラス
- `Currencies.java` - 通貨やレートを保持する Java クラス
- `Settings.java` - "from" や "to" の通貨などアプリケーションの設定を記録する Java クラス

Mobile Edition IDE では、特定の機能を実現する方法が複数あることがあります。このチュートリアルでは、機能を実現する方法の一つを説明しますが、同じ機能を別の方法で実現できることもあります。たとえば、ドロップダウンメニューにある機能は

通常、構成要素の右クリックで表示されるコンテキストメニューから選択することによって利用することができます。Mobile Edition ツールに習熟するにしたがって、もっとも快適な操作モードが分かるようになります。

設定作業

サンプルの MIDlet アプリケーションを作成・構築するには、Mobile Edition IDE 上で基本的な設定作業を行う必要があります。具体的には、ファイルシステムのマウントやエミュレータの追加、エミュレータのデフォルトスキンの設定などの作業を行います。

ファイルシステムのマウント

最初に、サンプルのアプリケーションコードを保存するファイルシステム (またはディレクトリ) をマウントします。ファイルシステムをマウントすることによって、Mobile Edition IDE からファイルの表示や操作をすることができます。マウントしたファイルシステムは Java クラスパスに取り込まれます。これは、コードのコンパイルや実行、デバッグに必要です。

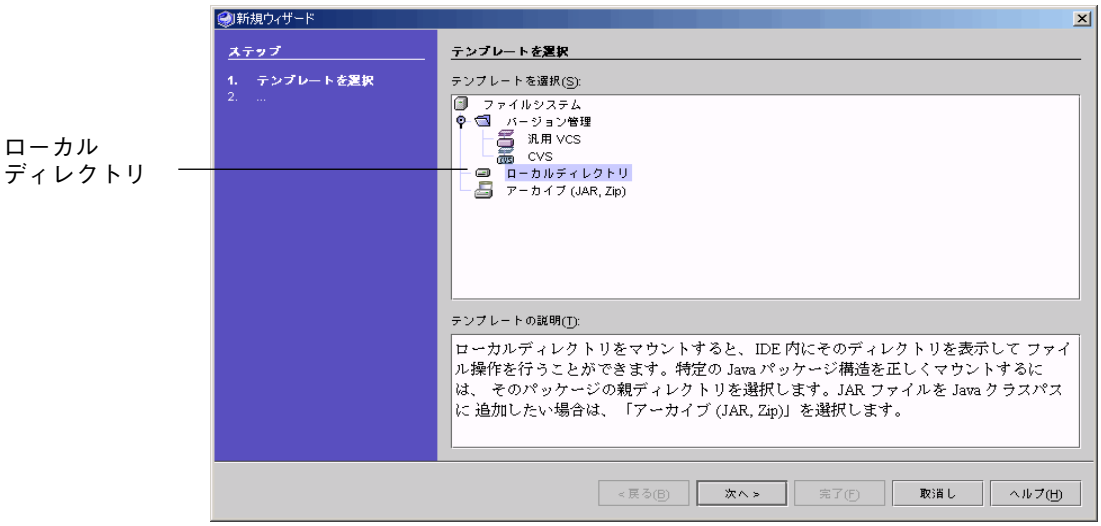
注 - Converter.zip ファイルをダウンロードして unzip すると、ソースファイルが *<download-directory>*¥exampledir¥converter に展開されます。このチュートリアルでは、ファイルのダウンロード先を D:¥ffjme ディレクトリと仮定しています。

ファイルシステムをマウントするには、次の操作を行います。

1. 「ファイル」->「ファイルシステムをマウント」を選択します。

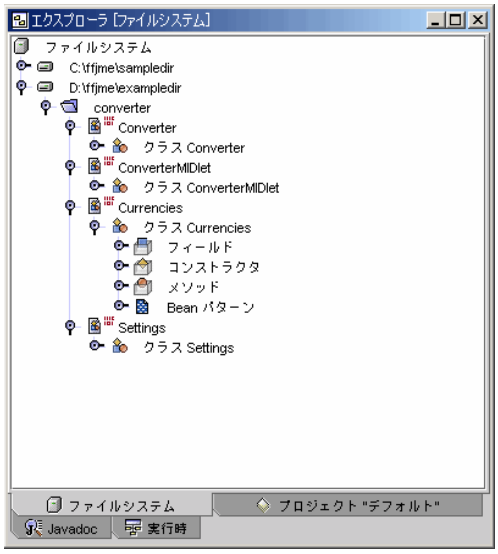
   ウィザードが開きます。このウィザードからファイルシステムのテンプレートを選択します。

2. 「ローカルディレクトリ」を選択し、「次へ」をクリックします。



3. ウィザードを使用して、`D:¥ffjme¥exampledir` ディレクトリに移動します。このディレクトリを選択して「完了」をクリックし、マウント処理を完了します。

   「エクスプローラ」の「ファイルシステム」タブに `D:¥ffjme¥exampledir` ディレクトリが表示されます。これでマウントが完了し、`D:¥ffjme¥exampledir` ディレクトリの内容を見ることができます。



デフォルトエミュレータの設定

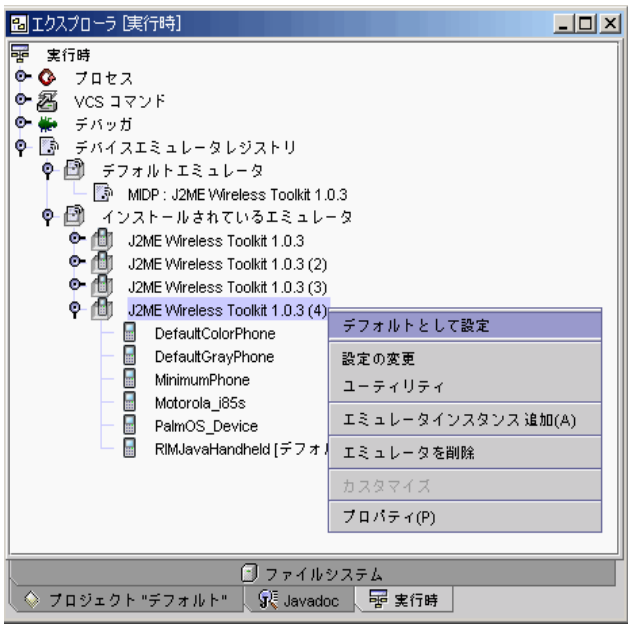
エミュレータは、ターゲットデバイス上でアプリケーションの実行をシミュレートすることによって、ユーザーと同様の経験をすることを可能にします。エミュレータを使用することによって、IDE 上でアプリケーションを実行およびデバッグすることができます。一般にエミュレータには、エミュレート対象のサンプルデバイスがいくつか含まれており、「スキン」と呼ばれています。J2ME Wireless Toolkit エミュレータはエミュレータの 1 つで、Motorola i85s フォン、Palm OS Device、Default Color Phone などのサンプルデバイス (スキン) を含んでいます。いずれのサンプルデバイスでもアプリケーションをテストすることができます。この機能により、デバイス間でアプリケーションの移植が可能かどうかをテストすることができます。

エミュレータについての詳細は、55 ページの「エミュレータの設定」を参照してください。この節では、複数のエミュレータインスタンスを設定して、さまざまなデバイス上でのアプリケーションのテストを高速かつ簡単に行う方法を紹介します。実際に見ていただくために、このチュートリアルでは、J2ME Wireless Toolkit エミュレータの 3 つのインスタンスを作成します。

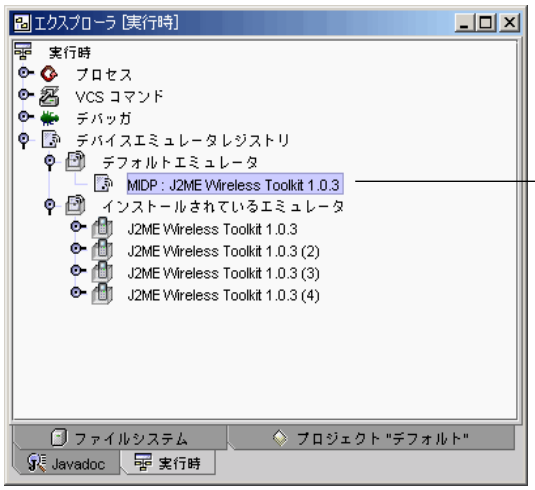
デフォルトのエミュレータは、アプリケーションをテストするときに呼び出されるインストール済みのエミュレータです。MIDP アプリケーションの場合、Mobile Edition ツールは一度に 1 つのデフォルトエミュレータを保持します。デフォルトのエミュレータは、MIDlet スイートに含まれていない MIDlet や、新規に作成された MIDlet スイートの実行に使用されます。

デフォルトのエミュレータを設定するには、次の操作を行います。

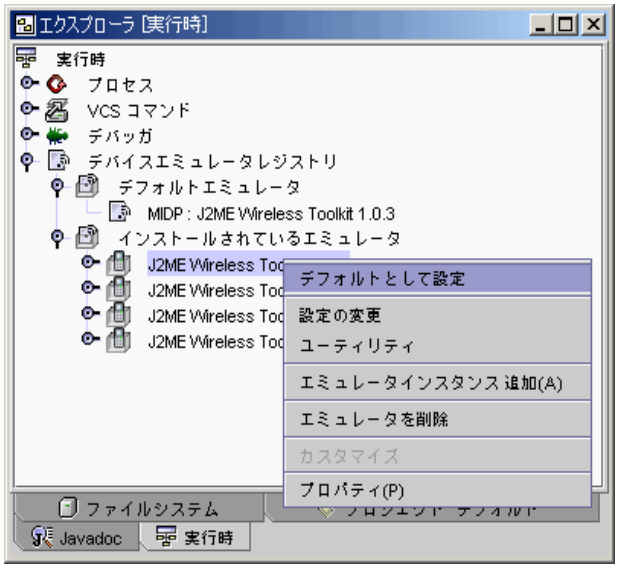
1. エクスプローラの「実行時」タブで「インストールされているエミュレータ」ノードを開いて、デフォルトエミュレータとして設定するエミュレータを右クリックし、「デフォルトとして設定」を選択します。



この例では、デフォルトのエミュレータとして J2ME Wireless Toolkit 1.0.3 (4) を選択しています。



2. エクスプローラで 最初の「J2ME Wireless Toolkit」インスタンスを右クリックし、メニューから「プロパティ」を選択します。



3. プロパティシートで、このエミュレータのプロパティを設定します。

   - プロパティ値を変更するには、そのプロパティの右の列のボックスをクリックします。矢印をクリックして選択可能な値のリストを表示するか、ボックスに有効な値を入力します。
   - 選択不可表示になっているプロパティは、編集できません。
   - プロパティシートの上部にある機能ボタンを使うと、プロパティをソートしたり編集可能なプロパティだけ表示したりできます。
   - プロパティシートには、独自のタブが設定されていることがあります。

図 1-1 はエミュレータのプロパティシートの例です。

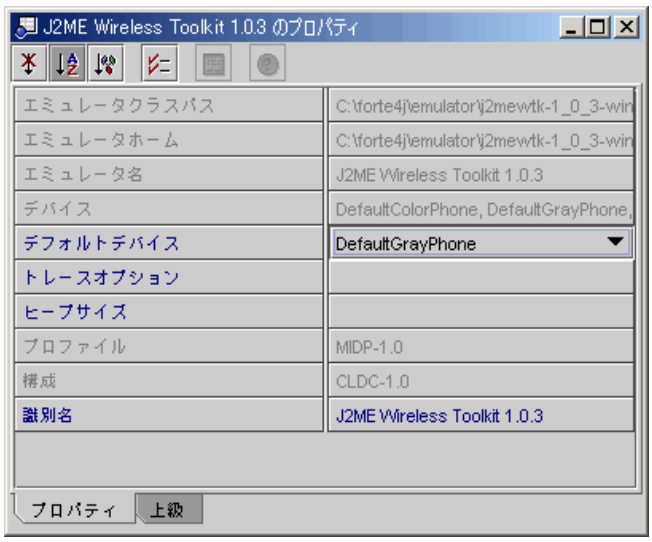


図 1-1　エミュレータのプロパティシート

エミュレータのデフォルトスキンの設定

エミュレータには、スキンと呼ばれるサンプルデバイスがいくつか含まれています。そのうちの 1 つをデフォルトスキンに設定したり、デフォルトスキンを変更したりできます。デフォルトスキンは、デフォルトエミュレータでアプリケーションをテストするときに表示されるデバイスです。

Wireless Toolkit エミュレータのそれぞれのバージョンに、異なるデフォルトスキンを設定することができます。このため、インストールされているエミュレータインスタンスにそれぞれ異なるデフォルトスキンを設定しておくと、複数の異なるデバイスでのアプリケーションテストを簡単に行うことができます。その場合には、アプリケーションをテストする前にデフォルトエミュレータを変更します。また、デフォルトエミュレータを変更せずに、デフォルトスキンを変更する方法もあります。

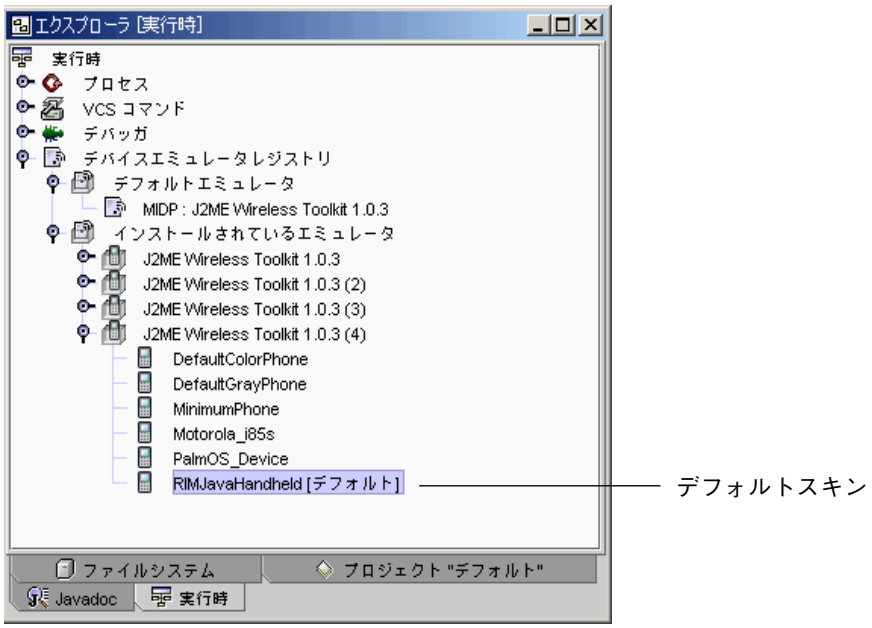
エミュレータのデフォルトスキンを設定するには、次の操作を行います。

1. エクスプローラの「実行時」タブで「インストールされているエミュレータ」を開きます。新規追加された「J2ME Wireless Toolkit (4)」エミュレータを開いて、そのスキンを表示します。

   どのデバイスもデフォルトスキンとして選択されていません。

2. スキンの 1 つを右クリックし、「デフォルトとして設定」を選択します。

この例では、RIMJavaHandheld デバイスをこのエミュレータのデフォルトスキンに設定します。このIDE では、[デフォルト] という表示でデフォルトスキンであることを示します。



コード補完機能の設定

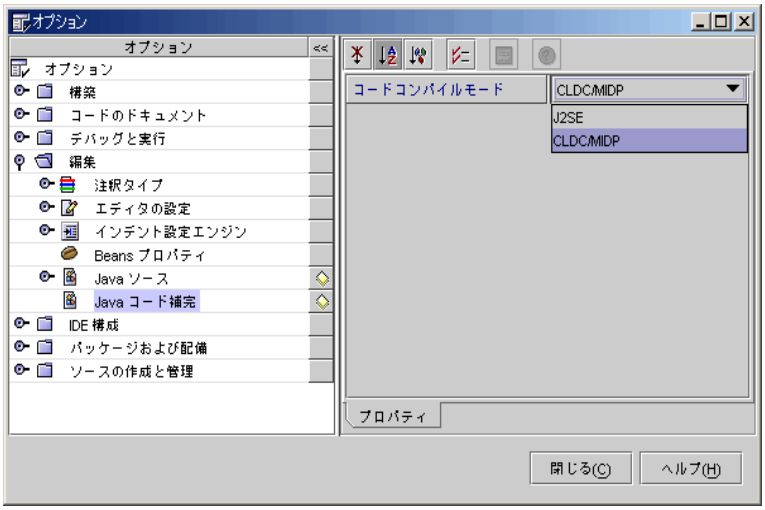
「ソースエディタ」ウィンドウを使用してコード入力すると、Mobile Edition IDE のコード補完機能を利用することができます。MIDlet の作成では、CLDC/MIDP にコード補完機能を設定できます 。コード補完機能は、MIDlet クラスにメソッドを追加するときに使用します。

コード補完機能を設定するには、次の操作を行います。

1. メインウィンドウの「ツール」->「オプション」を選択します。
「オプション」ウィンドウが表示されます。

2. 「オプション」ウィンドウで「編集」ノードを開き、「Java コード補完」をクリックします。

3. Java コード補完のプロパティシートでコードコンパイルモードとして「CLDC/MIDP」オプションを選択します。



「ソースエディタ」ウィンドウにおけるコードの入力では、コード補完機能を使用して、コードを補完することができます (図 1-2 を参照)。

コード補完機能を利用したコードの補完

1. コード補完機能を使用するには、コードの一部を入力し、Control (Ctrl) キーに続けてスペースキーを押します。Solaris の場合は、Contol (Ctrl) キーに続けて \ (バックスラッシュ) キーを押します。

   入力されたコードを補完する項目 (候補値) が表示されます。

2. リストから適切な項目をクリックしてEnter キーを押すと、項目が適用されます。

3. リストを絞り込むには、コードの入力を続けます。

   Enter キーが押されるか、現在のコード行が完成するまで、コード補完表示は開いたままになります。

注 - リストの項目を強調表示せずに Enter キーを押すと、ソースコードにコード補完が適用されないままコード補完表示が閉じます。

コード補完表示の項目は、コード行の入力状態に応じて絶えず変化します。

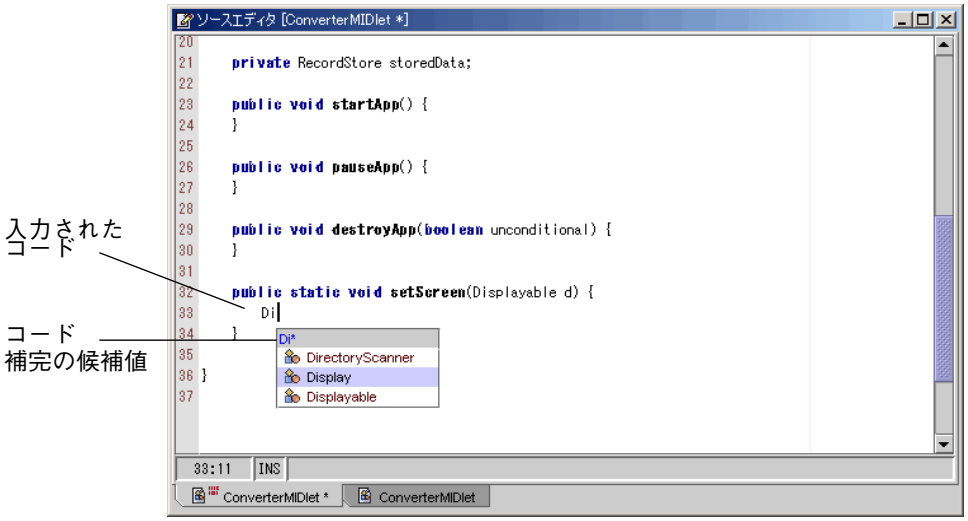


図 1-2　コード補完機能の使用方法

MIDlet のコンパイルとテスト

この節では、ひとつの MIDlet を作成し、コンパイルとテストをする方法を学びます。MIDlet クラスを作成し、ソースエディタとさまざまなツール機能を使用してコードを表示して編集します。その後、MIDlet スイートを作成し、前段階で作成した既存の MIDlet を MIDlet スイートに追加します。

このチュートリアルでは、新しい MIDlet を `myconverter` というパッケージディレクトリに作成します。MIDlet の新規作成では、MIDlet テンプレートを使用して、適切なクラス定義やパッケージ、インポート文、必要な MIDlet メソッドのスタブなどを設定します 。

MIDlet パッケージの作成

パッケージは、関連する Java コードを 1 つにまとめておく手段で、ディレクトリと同等視されることもあります。同じアプリケーションの Java クラスは 1 つのパッケージにまとめておくことを推奨します。

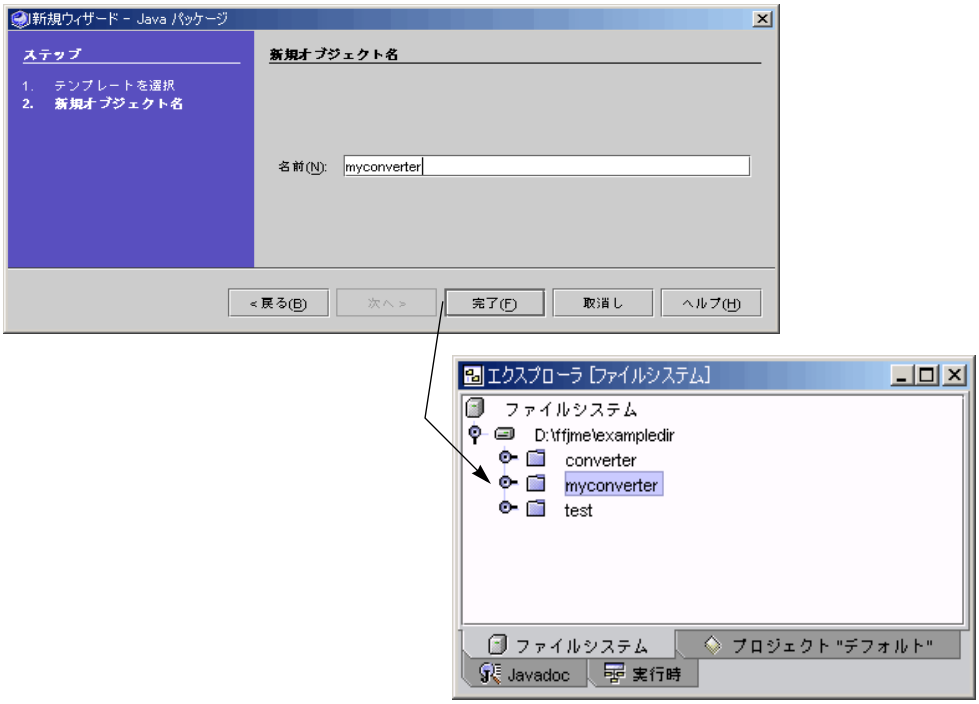
MIDlet のパッケージを作成するには、次の操作を行います。

1. エクスプローラの「ファイルシステム」タブをクリックし、D:¥ffjme¥exampledir ディレクトリを右クリックして、「新規」->「Java パッケージ」を選択します。

   新しいパッケージは、ファイルシステム構造内の任意のディレクトリに作成することができます。

2. 「新規ウィザード」で「名前」フィールドに myconverter と入力し、「完了」をクリックします。

   D:¥ffjme¥exampledir ディレクトリに myconverter という新規パッケージが作成されます。「エクスプローラ」ウィンドウでは、フォルダアイコンで 1 つのディレクトリとして表示されます。

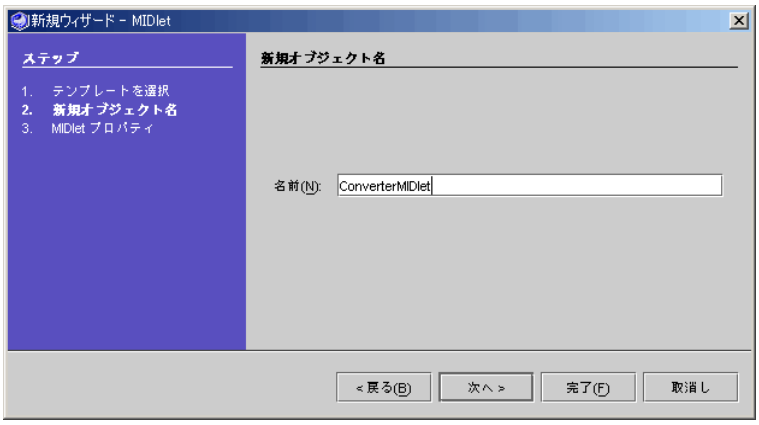


ConverterMIDlet MIDlet の作成

1. エクスプローラで「myconverter」パッケージを右クリックし、コンテキストメニューから「新規」->「MIDP」->「MIDlet」を選択します。

   MIDlet テンプレートを選択すると、MIDlet を新規作成するためのウィザードが起動されます。

2. 「名前」フィールドに、新規 MIDlet 名として `ConverterMIDlet` を入力し、「次へ」をクリックします。



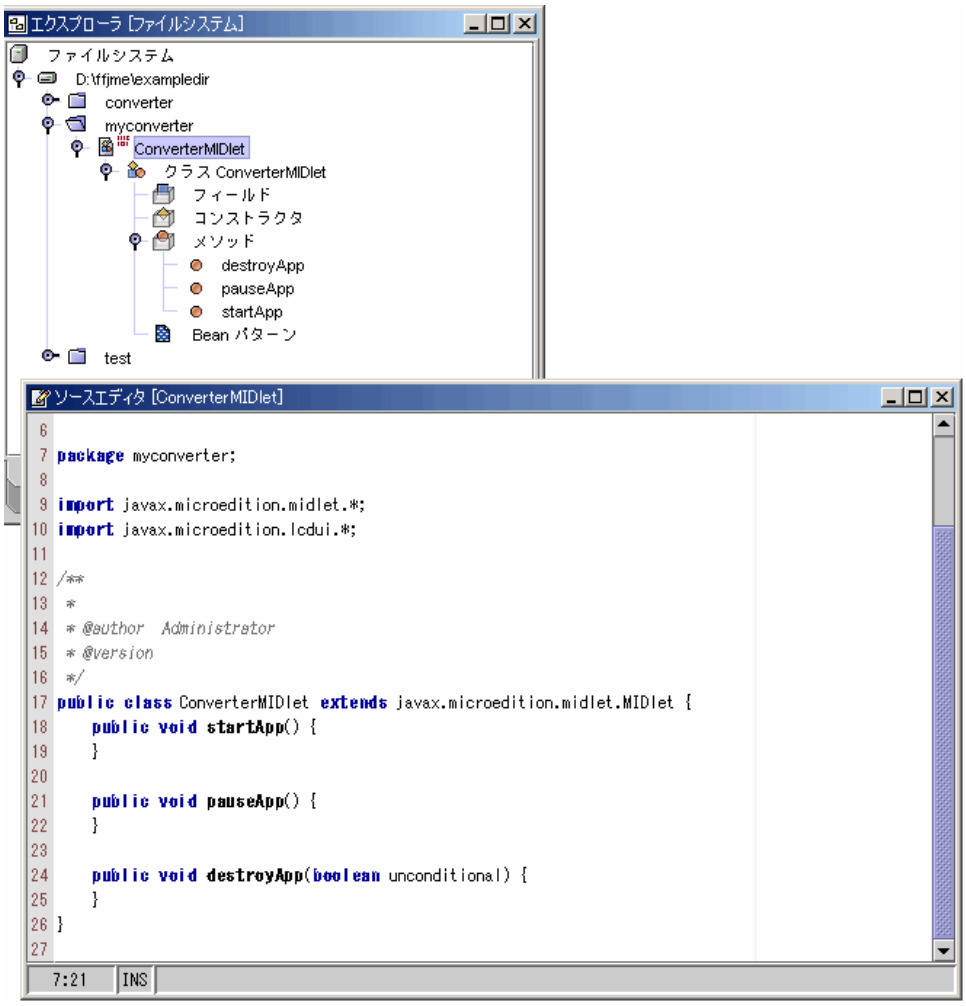
3. 「完了」をクリックして MIDlet を作成します。

4. 「エクスプローラ」ウィンドウの「ファイルシステム」タブでは作成された MIDlet を、「ソースエディタ」ウィンドウではそのコードを、それぞれ表示します。

   ConverterMIDlet は `javax.microedition.midlet.MIDlet` を拡張しており、`startApp`, `pauseApp`, `destroyApp` などの基本的な MIDlet メソッドが含まれています。

   「ConverterMIDlet」アイコンの横にある数字の 1 と 0 のパターンは、その MIDlet がコンパイルされていないことを示します。MIDlet クラスが正しくコンパイルされると、このパターンは消えます。

5. 「ConverterMIDlet」ノードを開くと、ConverterMIDlet クラス、フィールド、コンストラクタおよびメソッドが表示されます。



MIDlet のコーディング

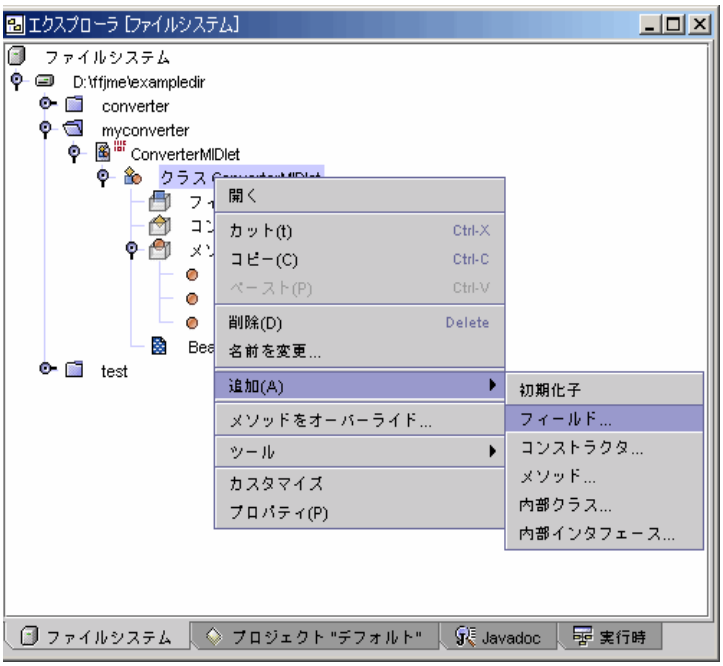
MIDlet のコードを記述する方法は 2 つあります。1 つはソースエディタでコードを直接入力する方法、もう 1 つはツールの機能を使用して、メソッド、フィールド、コンストラクタ、初期化子、クラスおよびインタフェースを追加する方法です。　一般的

に、クラスへの新規フィールドやメソッドの追加、または既存のフィールドやメソッドの変更には Mobile Edition ツールを使用し、あとでソースエディタで直接コードを修正します。

ここでは、Mobile Edition ツールとソースエディタの両方を利用してコードの入力や変更をする手順を紹介しますが、時間と労力を節約するために、通貨換算コードを myconverter ディレクトリにコピーすることもできます。

MIDlet にフィールドを追加

1. エクスプローラの「ファイルシステム」タブで「ConverterMIDlet」クラスを右クリックし、「追加」->「フィールド」を選択します。

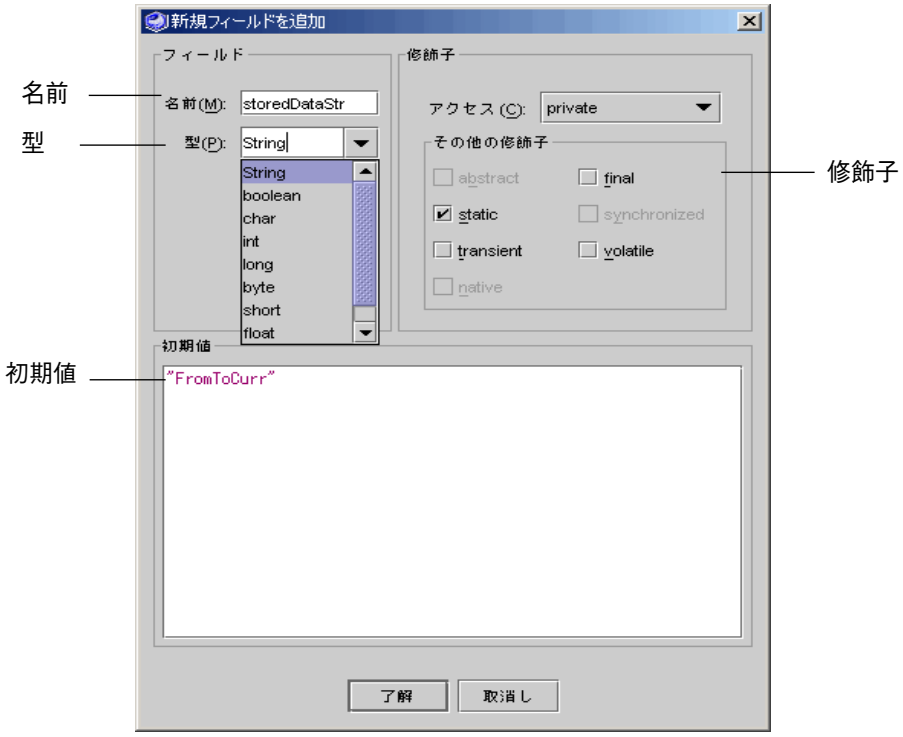


「新規フィールドを追加」ダイアログが表示されます。

2. 以下の手順で、「新規フィールドを追加」ダイアログを完成します。

   - 「名前」に新規フィールド名 `storedDataStr` を入力し、戻り値の型のコンボボックスから「String」を選択します。
   - 「修飾子」区画にあるアクセスのコンボボックスから、フィールドのアクセスタイプとして「private」を選択します。

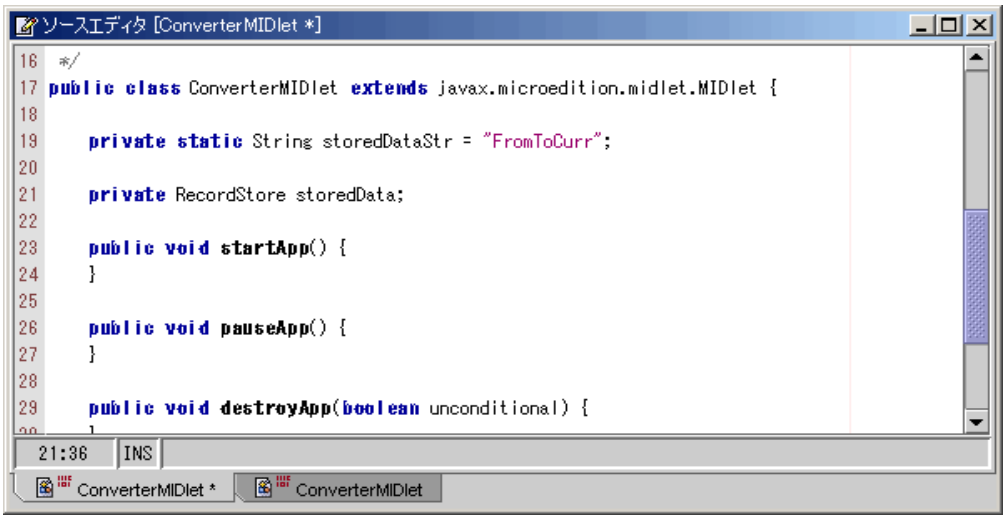
- 「その他の修飾子」から「static」を選択します。
- `storedDataStr` の初期値を "`FromToCurr`" に設定します。



3. 「エクスプローラ」ウィンドウで「ConverterMIDlet」クラスをダブルクリックして、「ソースエディタ」ウィンドウでクラスを開きます。

4. 新規フィールドの宣言を直接コードに入力します。「ソースエディタ」ウィンドウで次のように `storedData` フィールドの宣言を入力してください。

```
private RecordStore storedData;
```



MIDlet にメソッドを追加

MIDlet にメソッドを追加するには、次の操作を行います。

1. エクスプローラの「ファイルシステム」タブで「ConverterMIDlet」クラスを右クリックし、「追加」->「メソッド」を選択します。

   「新規メソッドを追加」ダイアログが表示されます。

2. 以下の手順で、「新規メソッドを追加」の追加ダイアログを完成します。

   a. 「名前」に新規メソッド名 setScreen を入力し、戻り値の型のコンボボックスから「void」を選択します。

   b. 「修飾子」区画にあるアクセスのコンボボックスから、「private」を選択します。

   c. 「その他の修飾子」から「static」を選択します。

   d. メソッドのパラメータの「追加」ボタンをクリックし、型に「Displayable」、名前に「d」を入力します。

3. 「setScreen」メソッドをダブルクリックして、ソースエディタで ConverterMIDlet クラスを開きます。コード補完機能を使用して、setScreen メソッドの実装コードを入力します。

   次の実装コードを利用してください。

```
Display.getDisplay(instance).setCurrent(d);
```

4. ConverterMIDlet クラスの入力を続けるか、「ソースエディタ」ウィンドウにソースコードをコピーします。

   Mobile Edition ツールの機能を使用する練習をしながら、ConverterMIDlet クラスの残りのコードを入力してください。コンバータパッケージのソースコードをソースエディタにコピーしてもかまいません。コードをコピーする場合は、パッケージ名が自分のパッケージと同じであることを確認してください。

MIDlet に追加クラスを追加

ConverterMIDlet クラスは 3 つの追加する Java クラス (Converter、Currencies、Settings) に依存します。これらは Java クラスではありますが、MIDlet ではありません。ConverterMIDlet クラスを作成したのと同様に、ウィザードを使用してこれらのクラスを作成することができます。

- ツールのコピー&ペースト機能を使用して、通貨換算アプリケーションに追加する 3 つのクラスを定義します。

  Mobile Edition ツールのコピー&ペースト機能を使用して、converter ディレクトリにある Java コードを myconverter ディレクトリ (ConverterMIDlet コードがある場所) にコピーすることができます。コードをコピー&ペーストすると、正しいパッケージ名が確実に設定できます。

  a. コピーするクラスを右クリックして、「コピー」を選択します。

  b. コピーしたクラスのペースト先のファイルシステムを右クリックして、「ペースト」->「コピー」を選択します。

MIDlet のコンパイル

MIDlet の作成が終了したら、コンパイルしてテストします。コンパイルする必要がある MIDlet クラスのアイコンの横には、数字の 0 と 1 のパターンが付いています。図 1-3 を参照してください。

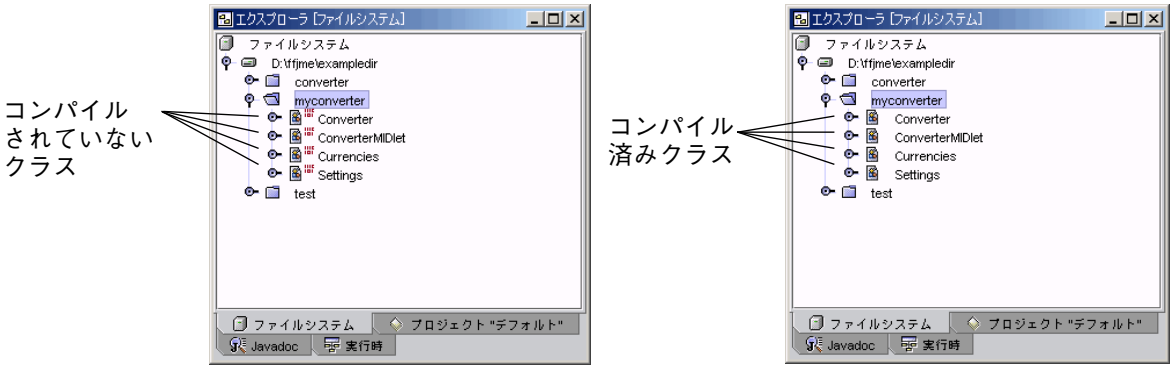


図 1-3　コンパイルされていない MIDlet クラスのアイコン

Mobile Edition ツールの MIDP コンパイラでは、他の方法では別々に分けて実行しなければならない手順を、連続する 1 つの手順にします。すなわち、コンパイラは MIDlet の `.java` ファイルをコンパイルして、`.class` バイナリファイルを生成し、コンパイル済みクラスの動作を事前に検証します。この事前検証は、MIDlet のコンパイル後に行われます。Mobile Edition IDE からコンパイル済みクラスファイルが渡されると、事前検証機能は CLDC 仮想マシンに MIDlet を準備します。コンパイル済みクラスのバイトコードを再編成して、CLDC 仮想マシン上での最終段階のバイトコード事前検証を簡単に行えるようにします。また、CLDC がサポートしない仮想マシン機能を利用していないかどうかを調べます。

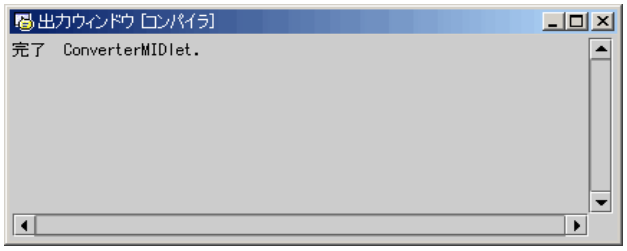
MIDlet をコンパイルするには、次の操作を行います。

1. エクスプローラで「ConverterMIDlet」を右クリックし、「コンパイル」を選択します。

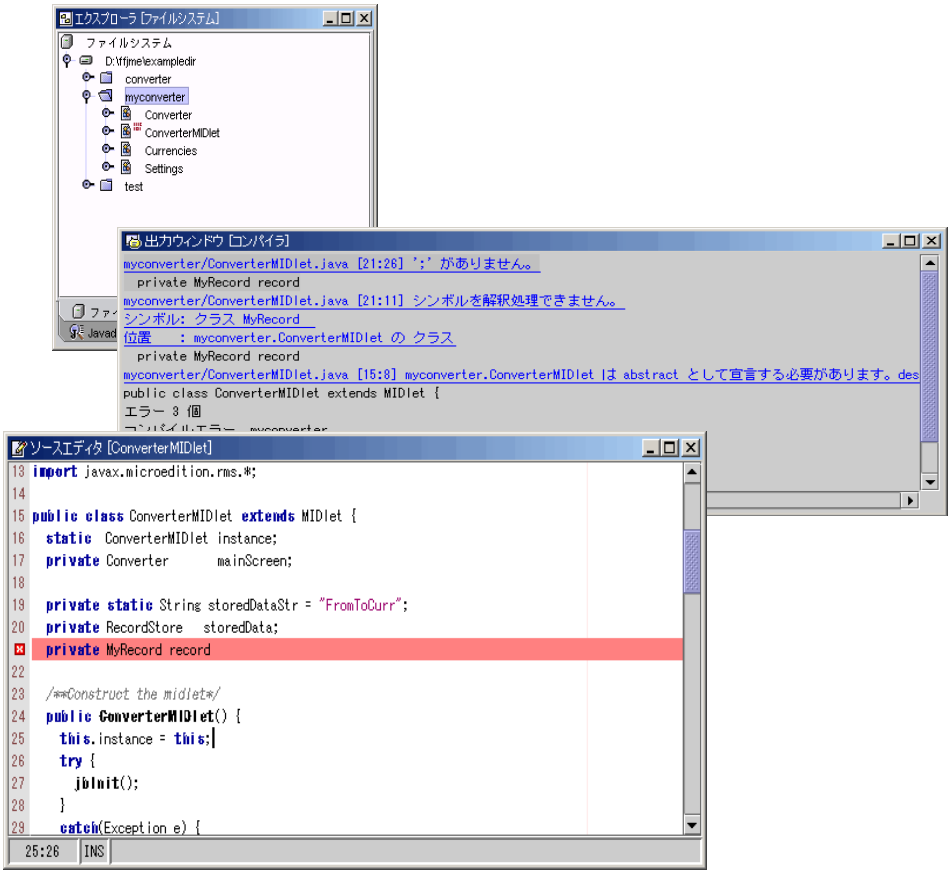
   ConverterMIDlet クラスと関係する 3 つの Java クラスのコンパイルは、「コンパイル」と「構築」のどちらのメニュー項目からでも起動できます。

2. 「出力」ウィンドウでコンパイルの状態を確認します。

コンパイラの「出力」ウィンドウには、コンパイラの状態として、コンパイルが成功したかエラーが発生したかが表示されます。たとえばコンパイラが ConverterMIDlet のコンパイルに成功すると、「完了 ConverterMIDlet」というメッセージが「出力」ウィンドウに表示されます。またこのとき、エクスプローラウィンドウからクラスアイコンの横にあった数字の 0 と 1 のパターンが消えます。



コンパイルエラーが発生すると、エクスプローラウィンドウのクラスアイコンには、赤丸で囲まれた「X」が表示されます。同時に「出力」ウィンドウにはそのエラーが含まれている行が表示され、ソースエディタではエラーを含む行が赤く強調表示されます。次の図は、ConverterMIDlet のコード行が不完全な場合の各ウィンドウの状態を表しています。



MIDlet のテスト

正常にコンパイルが終了すると、MIDlet をテストすることができます。コンパイル済みの MIDlet を実行するには、「実行」メニューを使用します。MIDlet アプリケーションはデフォルトエミュレータの現在 (デフォルト) のデバイススキンを使用して実行されます。

MIDlet の実行

- エクスプローラで「ConverterMIDlet」を右クリックし、「実行」を選択します。

  ConverterMIDlet アプリケーションでは、ある通貨の金額を別の通貨の金額に変換することができます。たとえばこのテストでは、日本円をユーロに換算するようにアプリケーションが設定されています。現在のデフォルトエミュレータは DefaultGrayPhone スキンを使用するように設定されています。

  アプリケーションが起動すると、エミュレータデバイスの画面上部にアプリケーションの名前 (Converter) が表示されます。

  エミュレータのデバイススキンで MIDlet をテストするには、次の操作を行います。

1. MIDlet を実行すると、Converter アプリケーションが動作し、最初のプロンプトまたは画面が表示されます。
2. エミュレータの数字キーをクリックして、換算する金額を入力します。
3. 「Settings」または「Exit」オプションを選択するには、オプション名の下にあるエミュレータのボタンをクリックします。
4. 円からドルというように通貨の種類を変更するには、「Settings」を選択してから、上下矢印キーを使用します。入力した金額を別の通貨に換算するには、「Settings」オプションから切り替わる「Save」オプションを使用します。

MIDlet スイートを使用したMIDlet のパッケージ化

MIDlet を個々に扱って開発やテストすることもできますが、一番良いのは MIDlet スイート内に MIDlet を作成することです。MIDlet スイートは、MIDlet アプリケーションをパッケージ化して、配備するのに役立ちます。

MIDlet スイートを使うと、MIDP アプリケーションに対してより多くの制御が行えるようになります。MIDlet スイートは MIDP アプリケーションにソース、ファイルおよび属性ファイルを編成します。MIDlet スイートを作成すると、Mobile Edition ツールは複数のアプリケーションファイルで構成される JAR ファイルを自動的に作成します。また、アプリケーションの配備に必要なアプリケーション記述子 (JAD) ファイルを作成します。

MIDlet スイートの作成

1. `D:¥ffjme¥exampledir` に `MyTestSuite` という新しいディレクトリを作成します。

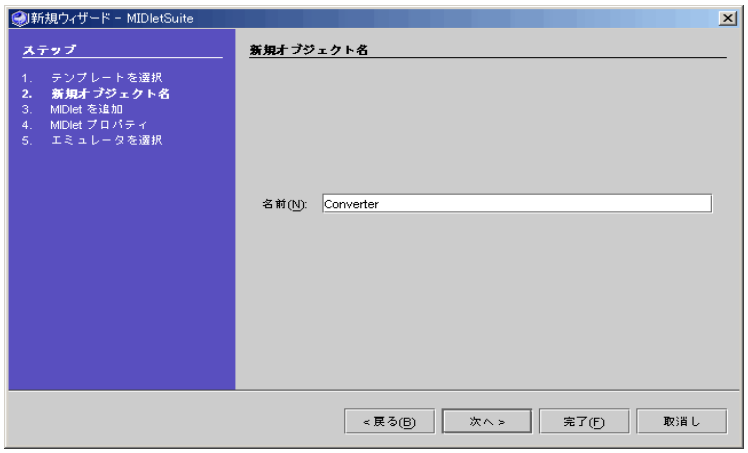
   このディレクトリに MIDlet スイートを作成します。

2. エクスプローラで「MyTestSuite」を右クリックし、コンテキストメニュー から「新規」->「MIDP」->「MIDletSuite」を選択します。

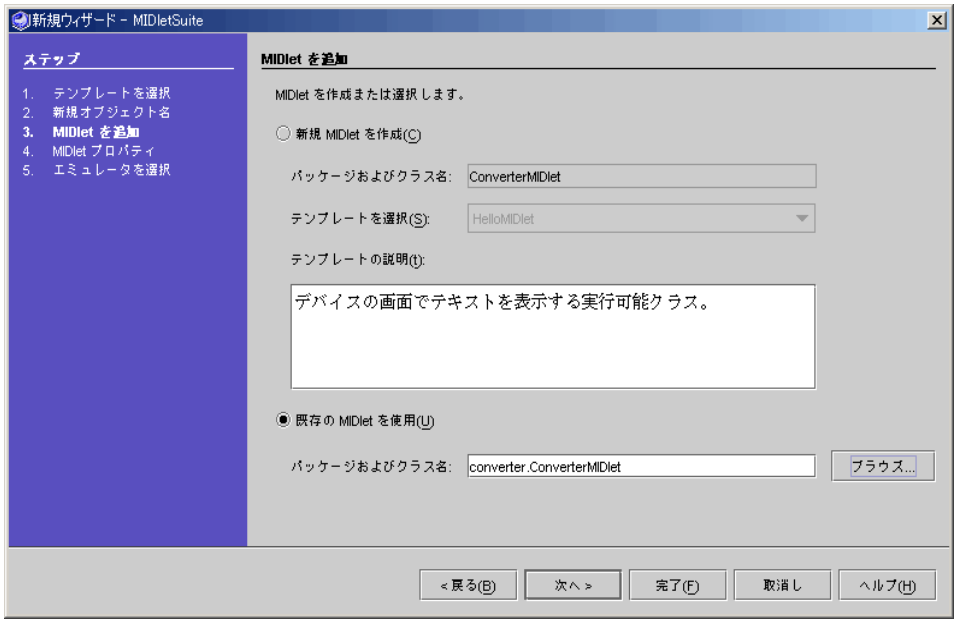
   MIDletSuite の新規ウィザードが表示されます。手順に従って MIDlet スイートを作成します。

3. MIDletSuite の新規ウィザードで新規 MIDlet スイートの名前を入力して、「次へ」をクリックします。

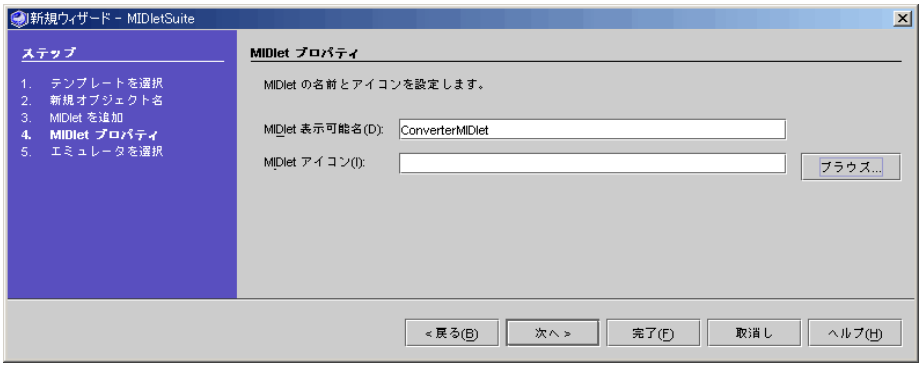
   Converter という名前を入力します。



4. 「既存の MIDlet を使用」を選択します。MIDlet クラス名を入力するか、「ブラウズ」ボタンを使って MIDlet クラスを表示し、「converter.ConverterMIDlet」クラスを選択し、「次へ」をクリックします。

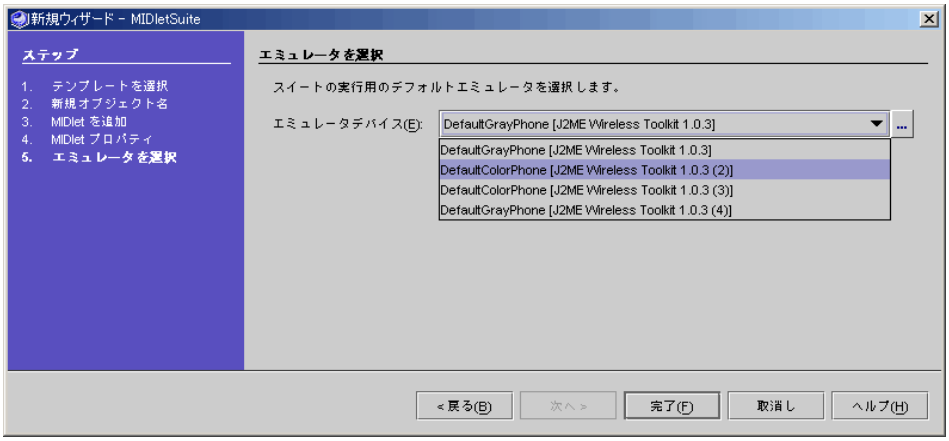


5. 「MIDlet プロパティ」区画の設定では、この MIDlet の表示可能名として `ConverterMIDlet` を入力し、「次へ」をクリックします。



6. MIDlet スイートのデフォルトのエミュレータを選択し、「完了」をクリックします。

   「エミュレータを選択」区画で、「J2ME Wireless Toolkit 1.0.3 (2)」のデフォルトのデバイススキンである「DefaultColorPhone」を選択します。

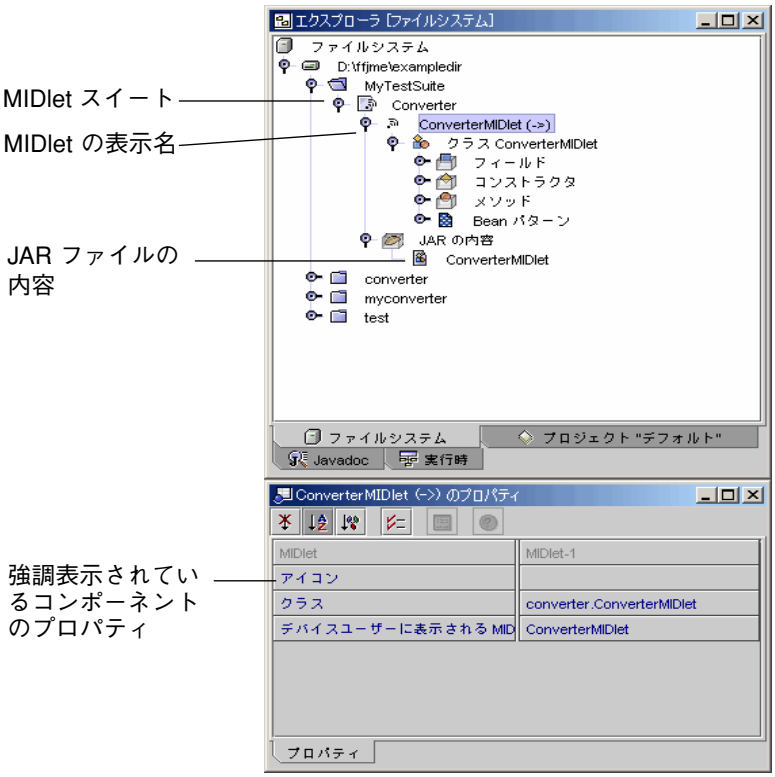


7. 「MyTestSuite」ノードを開いて、内容を確認します。

8. 「エクスプローラ」ウィンドウの「ファイルシステム」タブから、「Converter」スイートの「ConverterMIDlet」アイコンを右クリックします。

   MIDletSuite の新規ウィザードによって Converter スイートが作成されています。Converter スイートには ConverterMIDlet が含まれ、この ConverterMIDlet に ConverterMIDlet クラスとそのフィールド、コンストラクタ、メソッド、Bean パターンが含まれています。Converter スイートにはまた、ConverterMIDlet の実行可能

ファイルである JAR ファイルも含まれています。ConverterMIDlet のプロパティは、このウィンドウの下部または ConverterMIDlet のプロパティウィンドウで見ることができます。



MIDlet スイートのパッケージ化

前述のように、通貨換算アプリケーションは 1 つの MIDlet と 3 つの Java クラスで構成されています。このアプリケーションを実行するには、Java クラスとともに JAR ファイル内に MIDlet をパッケージ化する必要があります。 MIDlet スイートは、必要なパッケージ化を行って MIDlet アプリケーションを配備できるようにします。MIDlet スイートにはまた、JAD (Java Application Descriptor) ファイルも含まれます。MIDlet アプリケーションをパッケージ化するには、JAR ファイルと JAD ファイルの両方が必要です。

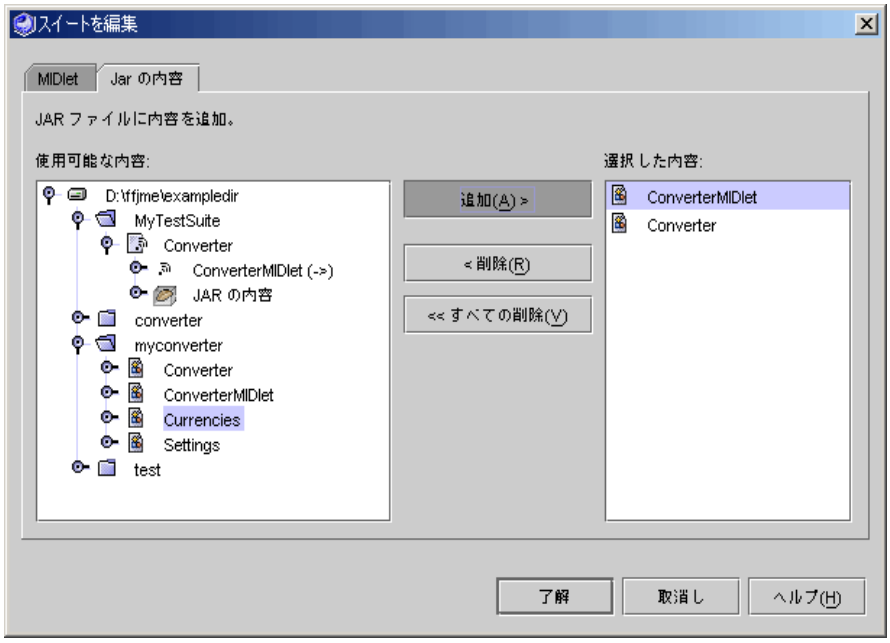
JAD ファイルには、MIDlet スイート内の MIDlet に事前に定義されている属性セットが含まれます。それらの属性によってアプリケーション管理ソフトウェアは MIDlet を識別して取り出し、インストールすることができます。また、アプリケーションに固有の独自の属性を定義して、JAD ファイルに追加することができます。

これに対し JAR (Java Archive) ファイルには、MIDlet スイートの各 MIDlet が使う Java クラスやリソースファイルが含まれます。リソースファイルとは、Java 以外のファイルです。JAR ファイルにはまた、その内容の記述やインストール固有の属性からなるマニフェストも含まれます。

MIDlet スイートをパッケージ化するには、次の操作を行います。

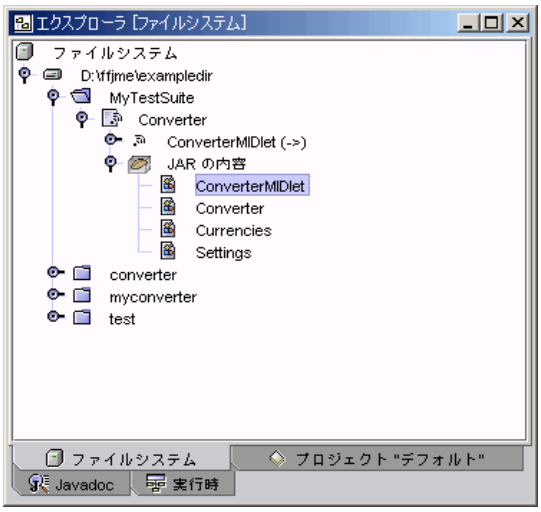
1. Converter MIDlet スイートを右クリックし、「スイートを編集」を選択します。「スイートを編集」ウィンドウで「Jar の内容」タブをクリックし、MIDlet スイートの JAR ファイルの内容を表示します。
2. JAR ファイルに追加する 「Converter」、「Currencies」、「Settings」クラスを順に選択して、「追加」をクリックし、「了解」をクリックします。

   JAR ファイルの内容を削除するには、「削除」または「すべてを削除」を使用します。

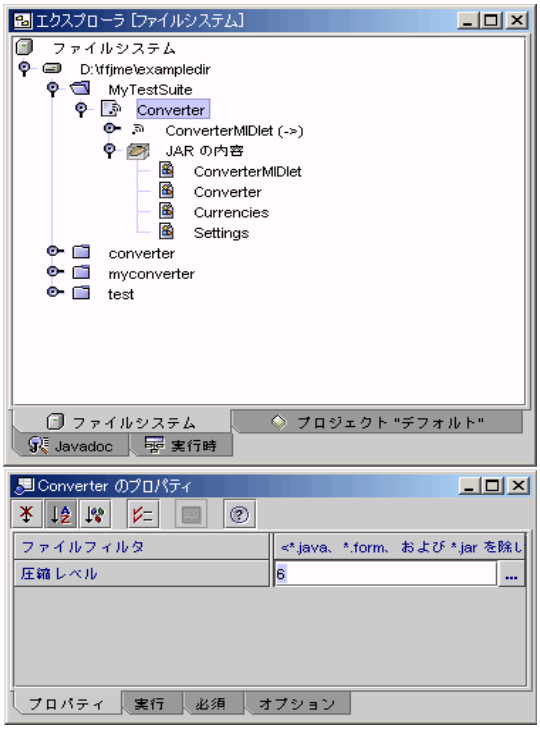


3. Converter MIDlet スイートにアプリケーションのすべてのクラスが追加されたことを確認します。

   JAR ファイルに通貨換算アプリケーションの Java クラスの追加が終了していると、Jar の内容として `ConverterMIDlet`、`Converter`、`Currencies`、`Settings` の 4 つのアプリケーションクラスが表示されます。



4. Converter MIDlet スイートのプロパティを変更するには、「エクスプローラ」ウィンドウで「Converter」MIDlet スイートを選択し、そのプロパティシートでプロパティ値を変更します。



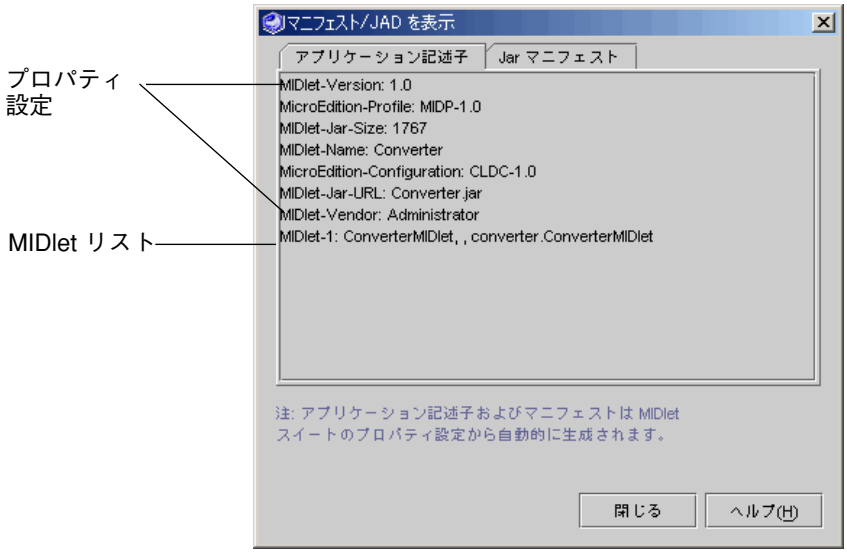
5. 圧縮レベルなどのプロパティ値の変更を終えたら、「Converter」MIDlet スイートを右クリックし、「JAR の更新」を選択して、新しい値を適用します。

6. ConverterMIDlet の JAD ファイルと JAR マニフェストファイルを確認するには、「エクスプローラ」ウィンドウの「ファイルシステム」タブで「Converter」MIDlet スイートを選択し、「マニフェスト /JAD を表示」を選択します。

   「マニフェスト /JAD を表示」ダイアログが表示されます。

7. JAD ファイルの内容を表示するには、「マニフェスト /JAD を表示」ダイアログで「アプリケーション記述子」タブをクリックします。また、JAR マニフェストファイルの内容を表示するには、「Jar マニフェスト」タブをクリックします。

   IDE は、自動的にMIDlet スイートのプロパティ設定と MIDlet からアプリケーション記述子とマニフェストを生成します。



MIDlet スイート内の MIDlet のテスト

MIDlet をテストするには、次の操作を行います。

1. 「エクスプローラ」ウィンドウの「ファイルシステム」タブで「MyTestSuite」(Converter MIDlet スイートが含まれているディレクトリ) を右クリックします。コンテキストメニューから「コンパイル」または「すべてをコンパイル」(「構築」->「すべてを構築」でもよい) を選択して、Converter MIDlet スイート内の ConverterMIDlet アプリケーションをコンパイルします。

2. 「エクスプローラ」ウィンドウの「ファイルシステム」タブにある「Converter」MIDlet スイートを右クリックし、コンテキストメニューから「実行」を選択して、ConverterMIDlet アプリケーションをテストします。

   MIDlet アプリケーションの実行前にコンパイルが必要な場合は、実行機能によってコンパイルされます。

MIDlet スイートの ConverterMIDlet アプリケーションが、エミュレータのデフォルトとして設定されているデバイススキン上で実行されます。ここでは、J2ME Wireless Toolkit 1.0.3 (2) の DefaultColorPhone スキンです。図 1-4 を参照してください。

図 1-4　エミュレータを使用した MIDlet のテスト

エミュレータデバイスは、通貨換算アプリケーションをただちに起動するのではなく、アプリケーションのリストを表示して、起動するアプリケーションの選択を求めます。MIDlet スイート内から MIDlet アプリケーションを起動しているので、エミュレータデバイスは MIDlet スイートに含まれているすべてのアプリケーションをリストで表示して、そこから実行するアプリケーションを選択できるようになっています。この時点では、Converter MIDlet スイートに含まれている MIDlet は 1 種類だけなので、リストに表示される MIDlet は ConverterMIDlet だけです。

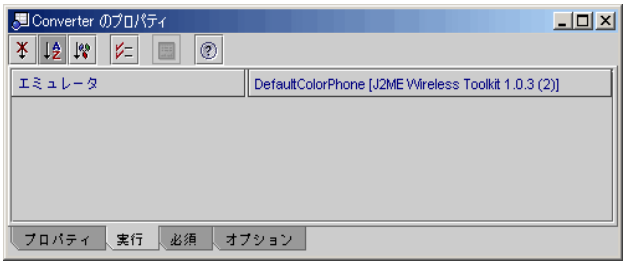
MIDlet スイートのターゲットエミュレータの変更

MIDlet スイートのターゲットエミュレータおよびデバイススキンは変更することができます。変更は、その MIDlet スイートのエミュレータにのみ適用されます。MIDlet スイートはそれぞれに異なるエミュレータやデバイススキンを設定でき、これは Mobile Edition IDE のデフォルトとして定義されているエミュレータと異なります。

1. 「エクスプローラ」ウィンドウの「ファイルシステム」タブにある「Converter」MIDlet スイートを右クリックして、「プロパティ」を選択します。

2. MIDlet スイートのプロパティシートにある「実行」タブをクリックします。

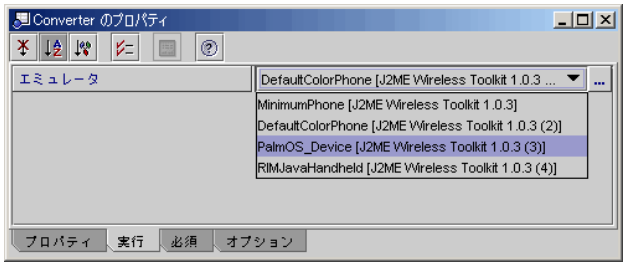
   プロパティシートの「実行」タブには、エミュレータプロパティと現在の設定が表示されます。Converter MIDlet スイートは、インストールされている J2ME Wireless Toolkit 1.0.3 エミュレータの 2 つ目のインスタンスを使用するように設定されています。また、そのデフォルトのデバイススキンは DefaultColorPhone が選択されています。



3. エミュレータ設定値のコンボボックスをクリックします。

   現在インストールされているエミュレータのいずれかを選択することができます。メニューには、インストールされている各エミュレータに対して設定されているデフォルトのデバイスが表示されます。

   エミュレータインスタンスのデフォルトのデバイススキンを変更することによって、MIDlet スイートのエミュレータデバイスを変更することもできます。

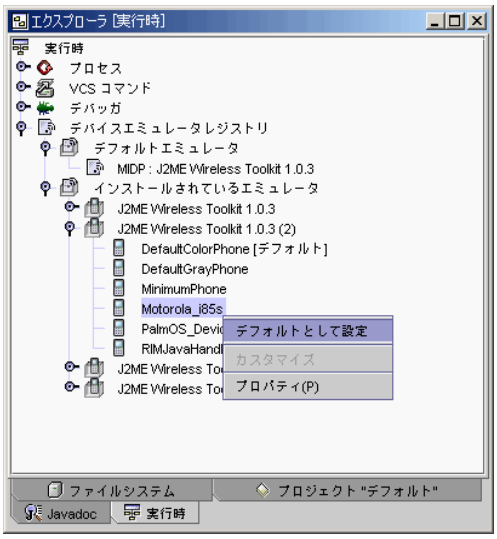


   デバイススキンを変更することによって MIDlet スイートのターゲットエミュレータを変更するには、次の操作を行います。

1. 「エクスプローラ」ウィンドウの「実行時」タブにある「インストールされているエミュレータ」ノードを開き、変更したいエミュレータインスタンスを開きます。

2. エミュレータのデバイススキンのリストから該当のデバイスを右クリックして、「デフォルトとして設定」オプションを選択します。

   この例では、J2ME Wireless Toolkit 1.0.3 (2) インスタンスのデフォルトスキンを Motorola_i85s に変更しています。7 ページの「エミュレータのデフォルトスキンの設定」を参照してください。



3. デフォルトのデバイススキンが変更されたことを確認します。「エクスプローラ」ウィンドウの「ファイルシステム」で「Convert」MIDlet スイートを右クリックし、「プロパティ」を選択してください。

   Converter の「実行」タブのエミュレータプロパティに、デフォルトデバイスとして Motorola_i85s が表示されています (図 1-5 を参照)。これで、Converter MIDlet スイートアプリケーションを実行すると、Motorola デバイススキン上で MIDlet をテストすることができます。

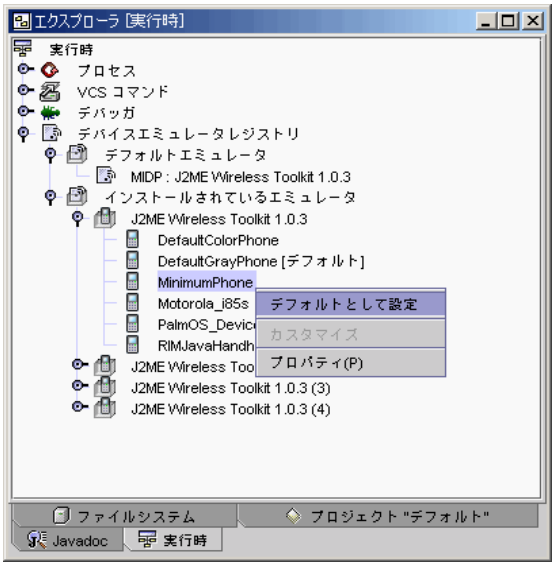
図 1-5　Converter MIDlet スイートの実行プロパティ

他のデバイスエミュレータでの MIDlet の実行

エミュレータのデフォルトデバイスを変更することによって、別のデバイスエミュレータ上で MIDlet を実行することができます。MIDlet を再コンパイルする必要はありません。

エミュレータのデフォルトデバイスを変更するには、次の操作を行います。

1. 「エクスプローラ」ウィンドウの「実行時」タブで「デバイスエミュレータレジストリ」ノードを開きます。
2. 「インストールされているエミュレータ」ノードを開き、インストールされているエミュレータのリストからデフォルトのエミュレータを見つけます。この例では、デフォルトのエミュレータは J2ME Wireless Toolkit 1.0.3 です。
3. 新しいデフォルトとして使用するデバイスを右クリックし、「デフォルトとして設定」を選択します。この例では、「MinimumPhone」を右クリックし、デフォルトデバイスとして設定します。



4. 「エクスプローラ」ウィンドウの「ファイルシステム」タブから「Converter」MIDlet スイートを実行し、MinimumPhone デバイススキン上で通貨換算アプリケーションをテストします。

図 1-6 に示すように、MinimumPhone デバイススキン上でアプリケーションが実行されます。

図 1-6　MinimumPhone デバイスエミュレータ上で動作する Converter プログラム

第2章

MIDlet と MIDlet スイートのデバッグ

この章では、MIDlet および MIDlet スイートのデバッグ方法を説明します。Mobile Edition IDE のデバッグ機能では、デバッグプロセスを完全に制御することができます。具体的に以下のことを行うことができます。

- ブレークポイントとウォッチポイントを設定する
- 変数値を調査し、変更する
- コードをステップ実行する
- クラスを調査する
- 呼び出しスタックを利用する
- IDE の他のデバッグ機能を利用する

デバッガの基礎

デバッグ作業スペースを開くには、メインウィンドウの「デバッグ」タブをクリックします。

この作業スペースは、「デバッガウィンドウ」、「出力ウィンドウ」、「ソースエディタ」ウィンドウで構成されます。ただし、メインウィンドウから「デバッグ」->「デバッガウィンドウ」を選択して「デバッガウィンドウ」だけ開くこともできます。この章では、これらのウィンドウを簡単に説明します。デバッグ作業スペースについての詳細は、オンラインヘルプを参照してください。

「デバッガウィンドウ」には、アプリケーションで発生している状況を表示する 7 つのビューがあります。これらのビューでは、アプリケーションの以下の項目を監視します。

- セッション、スレッド、アプリケーションの現在の呼び出しスタックに設定されているブレークポイント
- コードに設定されているウォッチポイント
- 変数値
- クラス

これらのビューの表示は、「デバッガウィンドウ」のツールバーを使って変更することができます。「デバッガウィンドウ」の各ビュー内のアイコンは、ウィンドウ上部の 1 行目のツールバーのアイコンと同じです。ウィンドウの 2 行目のツールバーでは、「変数」「ウォッチポイント」「スレッド」「呼び出しスタック」ビューに表示する内容を制御することができます。Mobile Edition IDE の他のウィンドウ同様、ウィンドウ内の項目を右クリックすると、その項目のコンテキストメニューが表示されます。図 2-1 を参照してください。

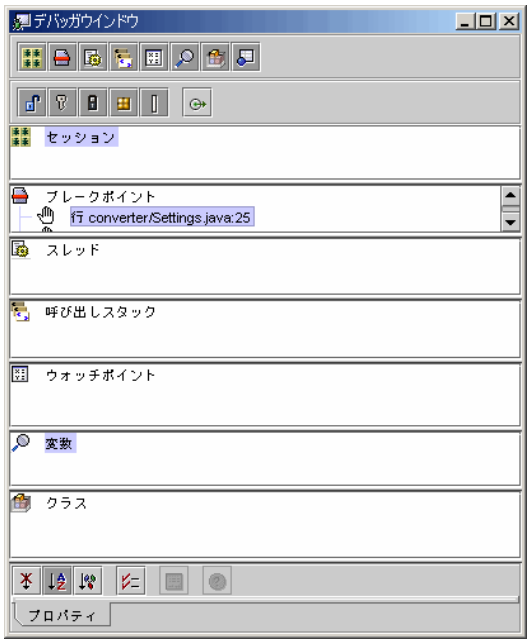


図 2-1　「デバッガウィンドウ」とそのビュー

「出力ウィンドウ」には 2 つの区画があります。左の区画にはアプリケーションの出力、右の区画にはデバッガからの状態メッセージとブレークポイントおよびスレッドの詳細情報が表示されます。

MIDlet をデバッグするには、必ず IDE デバッガをエミュレータデバッガに設定します。

IDE デバッガの設定を確認するには、次の操作を行います。

1. メインウィンドウで「ツール」->「オプション」を選択します。

   「オプション」ダイアログが開きます。

2. 「オプション」->「デバッグと実行」フォルダを開きます。

3. 「デバッガの種類」を開いて、「エミュレータデバッガ」ノードを選択します。

   ウィンドウの右の区画に、エミュレータデバッガのプロパティが表示されます。

4. デバッガの識別名がエミュレータデバッガであることを確認します。

   図 2-2 は、エミュレータデバッガを設定した「オプション」ウィンドウの表示例です。

   MIDlet プロパティシートの「実行」タブでは、クラス別にデバッグオプションを表示または設定することができます。

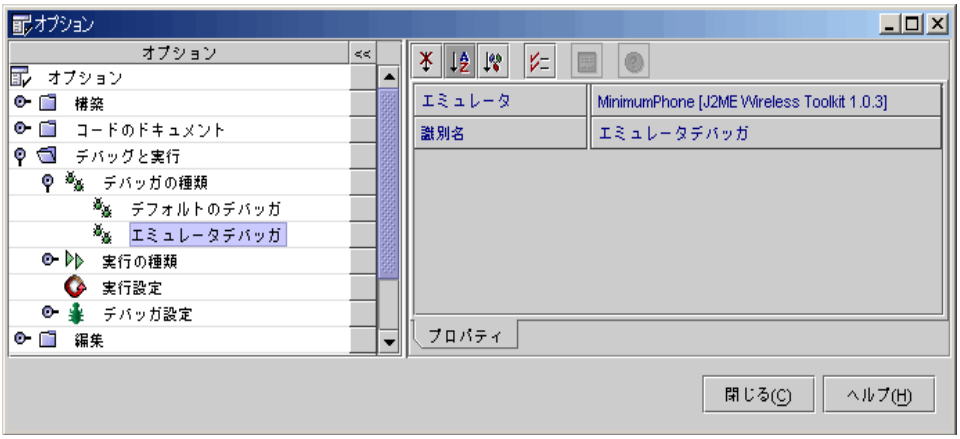


図 2-2　MIDlet をデバッグする際のエミュレータデバッガの設定

この章では、1 つの MIDlet をデバッグしますが、Mobile Edition IDE では、複数の MIDlet を同時にデバッグすることができます。この場合、「デバッガウィンドウ」の「セッション」ビューには、現在デバッグ中の MIDlet が一覧表示されるので、「セッション」ビューでコンテキストメニューを使用したり、セッションのプロパティを編集したりすることによって、デバッグセッションを管理することができます。こうしたオプションについては、オンラインヘルプで詳しく説明しています。

プログラムのデバッグ

ここでは、アプリケーションのコードにブレークポイントを設定したうえで、Mobile Edition IDE でデバッグセッションを開始し、ブレークポイントに達するまでアプリケーションを実行してみます。

ブレークポイントの設定

MIDlet のデバッグを開始するには、ConverterMIDlet プログラムにブレークポイントを設定します。

ブレークポイントを特定の行番号、メソッド名、例外、クラス、変数、スレッドに設定することによってプログラムの実行が停止します。変数、メソッド、行および例外に対するブレークポイントの設定では、条件式の評価が真（`true`）のときにブレークポイントをトリガーする条件を設定することもできます。

ブレークポイントに達すると、デバッガはデバッグを一時停止したり、「出力ウィンドウ」にメッセージを表示したりします。こうしたアクションは、ブレークポイントの設定で制御します。具体的には、「ブレークポイントを追加」ダイアログでブレークポイントを設定し、その条件とアクションを指定してください。このダイアログは、メインウィンドウの「デバッグ」メニューから開くことができます。別の設定方法としては、「ソースエディタ」ウィンドウ上で、ブレークポイントを設定する位置でカーソルを右クリックし、コンテキストメニューから「ブレークポイントの切り替え」を選択する方法があります。

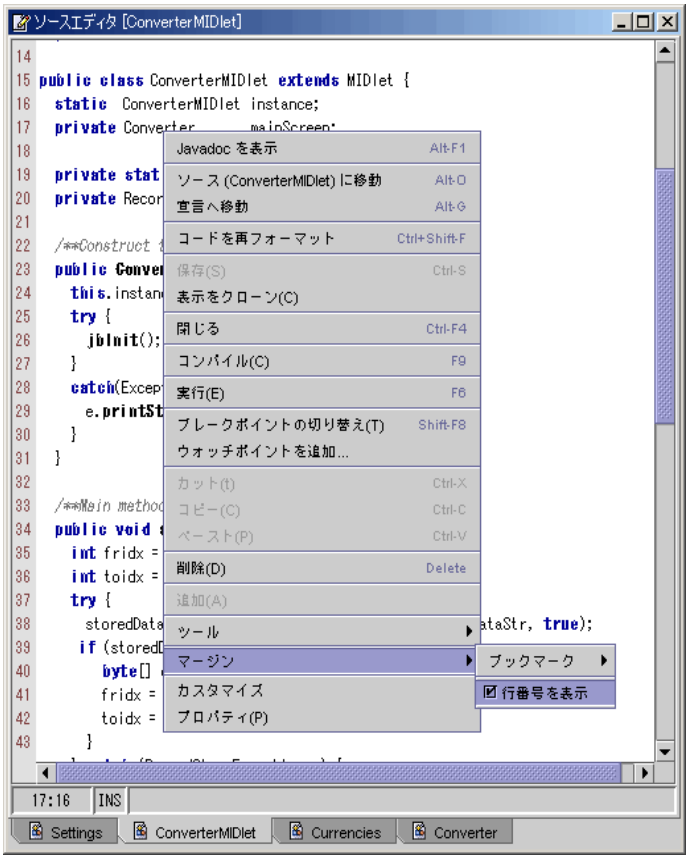
ブレークポイントを設定するには、次の操作を行います。

1. 「エクスプローラ」ウィンドウの「ファイルシステム」タブから、「`myconverter`」パッケージの「ConverterMIDlet」のソースコードを「ソースエディタ」ウィンドウ上に開きます。

   このためには、「`myconverter`」を開いて、「ConverterMIDlet」をダブルクリックします。

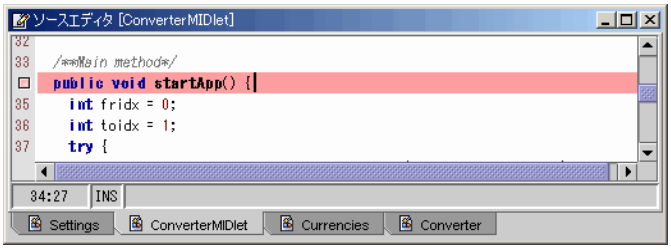
2. 「ソースエディタ」ウィンドウで ConverterMIDlet コードの行番号を表示するように設定します。「ソースエディタ」ウィンドウ上で右クリックし、コンテキストメニューから「マージン」を選択して、「行番号を表示」チェックボックスを選択します。

   プログラムのデバッグでは、特に特定の行にブレークポイントを設定する場合は、ソースコードの行番号を表示した方が便利なことがよくあります。行番号の表示の切り替えは、「行番号を表示」チェックボックスの選択と、選択を解除することで行うことができます。



3. startApp メソッドにブレークポイントを設定します。「ソースエディタ」ウィンドウで行番号 34 の startApp メソッドの左側のマージン部分をクリックします。

   ブレークポイントが設定されたコード行が強調表示され、左マージンに適切なブレークポイントのアイコンが表示されます 。さまざまなブレークポイントのアイコンについては、表にまとめてオンラインヘルプで説明しています。



デバッグモードでのプログラムの実行

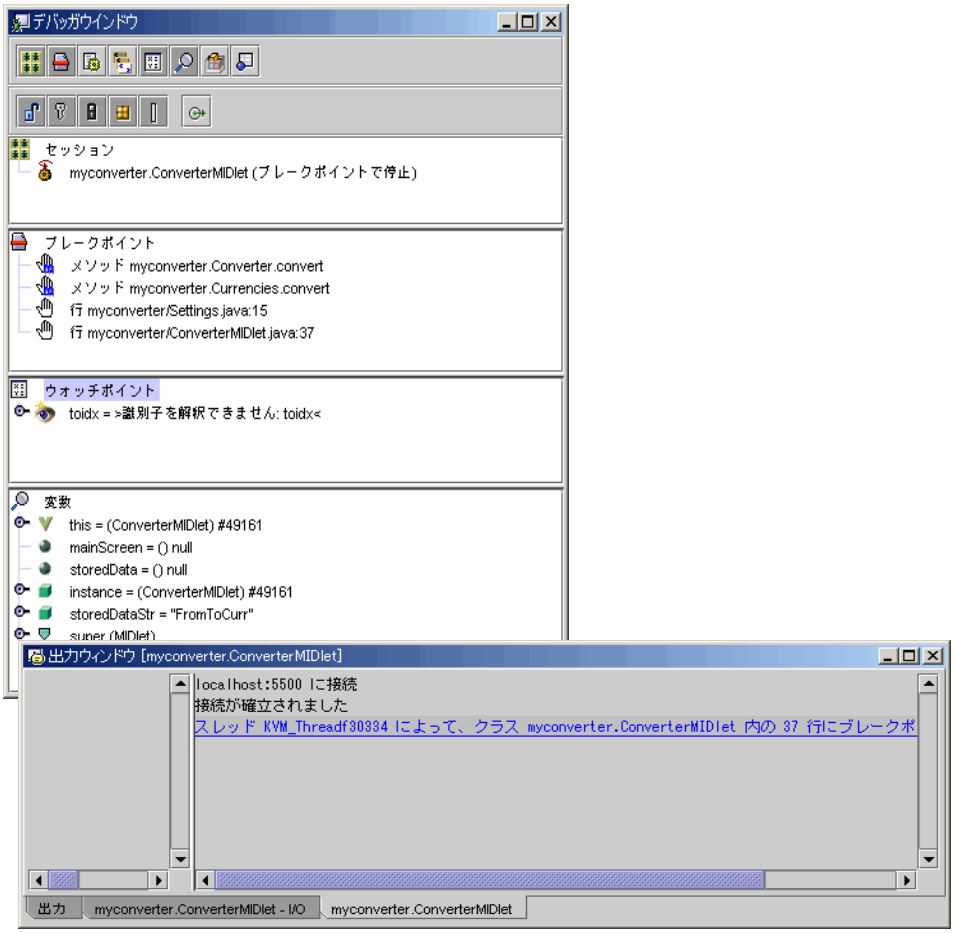
ここでは、メインウィンドウの「デバッグ」メニューから、ConverterMIDlet プログラムをデバッグモードで実行します。「編集」または「実行」作業スペースで「デバッグ」メニューオプションを選択すると、自動的にデバッグ作業スペースに切り替わり、プログラムがデバッグモードで起動します。

プログラムの起動

- 「エクスプローラ」ウィンドウの「ファイルシステム」タブで 「myconverter」パッケージを開いて「ConverterMIDlet」を選択し、メインウィンドウから「デバッグ」->「開始」を選択します。

  「デバッグ」->「開始」を選択すると、デバッグ作業スペースに切り替わり、「デバッガウィンドウ」が開きます。そしてメインウィンドウの「デバッグ」タブ右側のメッセージバーには、デバッガの現在の状態 (実行中、ブレークポイントの位置、完了など) が表示されます。こうした状態メッセージは、「出力ウィンドウ」の該当する区画にも表示されます。プログラムは、Settings クラスの行番号 37 の位置にあるブレークポイントに達するまで実行されます。「デバッガウィンドウ」と「出力ウィンドウ」には、ConverterMIDlet プログラムの現在の状態が反映されます。「デバッガウィンドウ」は、設定されているブレークポイントを表示するだけでなく、現在の

セッションやその状態 (停止中、ブレークポイントで停止など)、現在アクティブな変数とその値を表示します (プログラムがコードの該当部分を通過すると、エミュレータデバイススキンの情報も表示されます)。



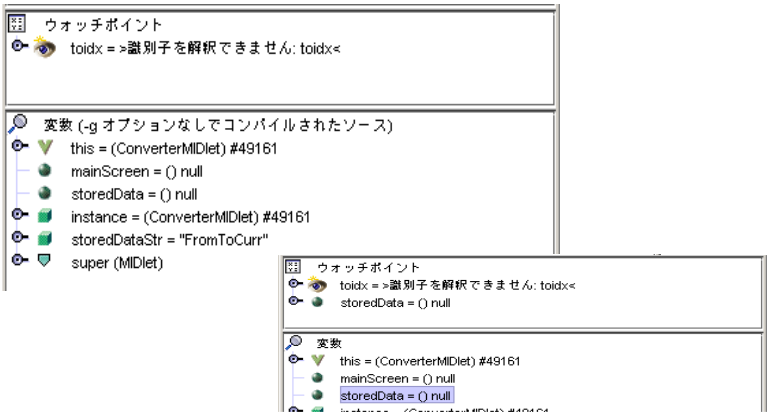
変数値の調査と設定

「デバッガウィンドウ」から、プログラムの変数に現在設定されている値を調査したり、変数にウォッチポイントを設定したり、変数の値を変更したりできます。

変数の値を設定するには、次の操作を行います。

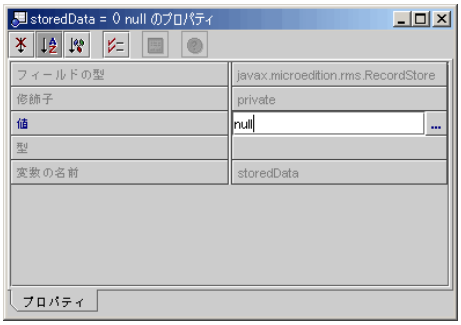
1. storedData 変数に固定ウォッチポイントを設定します。「デバッガウィンドウ」の「変数」ビューで設定したい変数を右クリックし、コンテキストメニューから「固定ウォッチポイントを作成」を選択します。

   「デバッガウィンドウ」の「変数」ビューに、プログラムの変数とともにその現在の値が表示されます。固定ウォッチポイントは、「変数」ビューに表示されている変数に作成することができます。そして「ウォッチポイント」ビューには、固定ウォッチポイントが作成された変数が表示されます。つまり、storedData に固定ウォッチポイントを作成すると、「ウォッチポイント」ビューと「変数」ビューの両方に storedData が表示されます。



2. 変数の値を変更します。「変数」ビューで変更したい変数を右クリックし、そのプロパティシートを開いて新しい値を入力します。

   変数の値を変更することによって、プログラムの実行にどのような影響があるのかを調べることができます。



ステップ実行オプションの設定

メソッドにステップインしたときのデバッガの動作を制御することができます。ただし、メソッドにステップインするとデバッガからソースコードにアクセスできなくなることがあります。ステップイン時のアクションは以下から選択できます。

- メソッドからステップアウトする
- デバッグプロセスを停止する
- ソースコードを検索する

「オプション」ウィンドウで、これらの状況におけるデバッガの対処方法を制御します。「オプション」ウィンドウで「デバッグと実行」->「デバッガ設定」ノードを開きます。「Java」ノードを選択し、そのプロパティシートにある「ステップイン時のアクション」プロパティに適切な値を設定します。図 2-3 を参照してください。

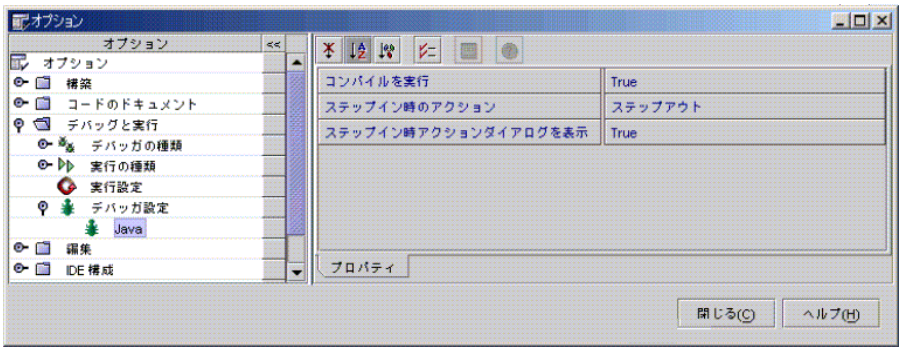


図 2-3　Java のデバッガ設定

ソースのないメソッドにステップインしたときには、プログラムを停止してどのようなアクションを取るか問い合わせるダイアログを表示するように設定することもできます。このオプションを設定するには、「ステップイン時アクションダイアログを表示」プロパティを「True」に設定します。

プログラムのステップ実行

プログラムをステップ実行するには、「デバッグ」メニューのオプションを使用します。ステップ実行は、デバッガ上で動作しているプログラムが一時停止するか、ブレークポイントで停止したときに、実行を再開する 1 つの方法です。「デバッグ」メニューには、プログラムをステップ実行するための 3 つのオプション (ステップオーバー、ステップイン、ステップアウト) が用意されています。

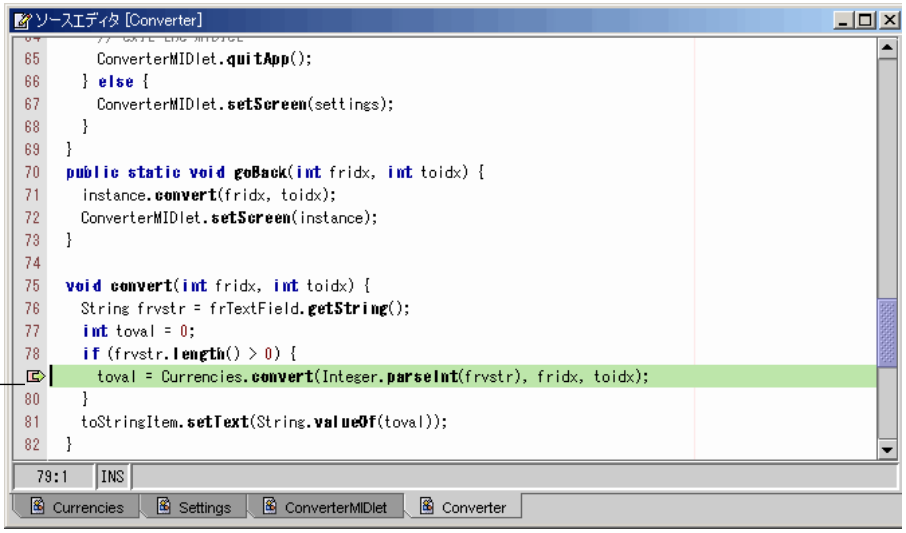
これらのオプションはそれぞれ次のソース行を実行しますが、実行した行がメソッド呼び出しであるかどうかによって動作が異なります。行がメソッド呼び出しの場合、「ステップオーバー」はメソッドの個々の命令をステップ実行することなくメソッド全体を実行します。「ステップイン」は、メソッドの最初の文を実行する直前で停止します。「ステップアウト」は前述の 2 つとは少し異なり、ソース行がメソッドの一部である場合にメソッドの残りの行を実行し、メソッドの呼び出し元に制御を返します。図 2-4 は、プログラムがブレークポイントに達したときに「デバッグ」メニューから使用できるオプションを示しています。

図 2-4　ブレークポイントにおける「デバッグ」メニューのオプション

プログラムをステップ実行するには、次の操作を行います。

1. 「デバッグ」->「ステップイン」を選択して、コードの次の行を実行します。そのコード行がメソッド呼び出しの場合は、メソッドの最初の行に進みます。

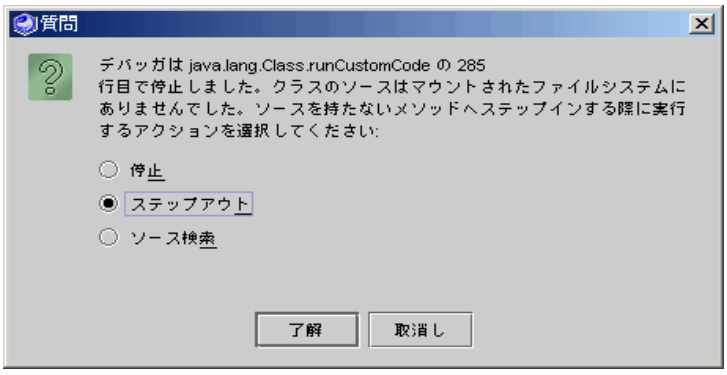
   ソースエディタに、ブレークポイントで停止している Converter クラス内のコード行が強調表示されます。このプログラムは、Currencies の `converter` メソッドを呼び出そうとしているところです。「デバッグ」->「ステップイン」を選択して、このメソッドの最初の行に進みます。



   メソッドのソースコードを検出できない場合、デバッガはそれに応じたメッセージを表示します。

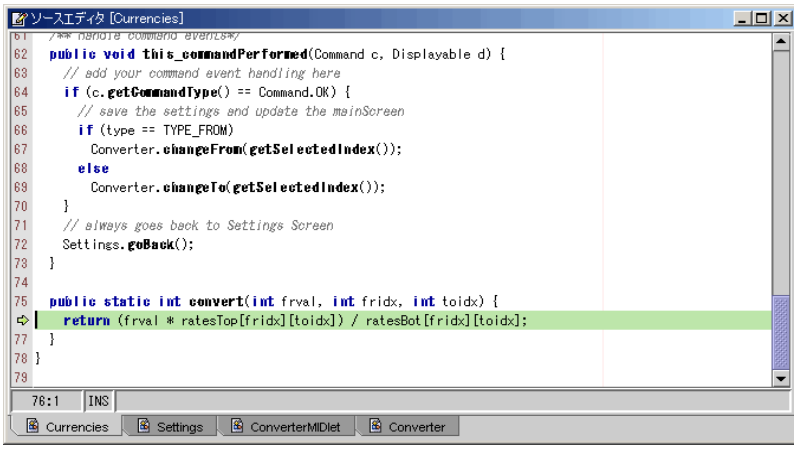
   その場合は、プログラムの実行を停止したり、問題のメソッドからステップアウトしたり、デバッガにソースコードの検出を行わせたりできます。「ステップアウト」オプション (デフォルト) はメソッドを最後まで実行して、そのメソッドの呼び出し元に制御を返します。この場合、「ステップアウト」は「デバッグ」->「ステップアウ

ト」を使用するのと同じことです。



2. デバッガがメソッドのソースコードを検出できた場合は、ソースエディタにそのコードが開かれ、ステップ実行した行が強調表示されます。「デバッグ」->「ステップアウト」でメソッドを最後まで実行します。

   プログラムは Currencies の `converter` メソッドで停止します。ここでは、このメソッドの残りのコードをステップ実行することも、メソッドからステップアウトし、1 回の操作でメソッドを最後まで実行することもできます。



3. 「デバッグ」->「ステップオーバー」を選択し、次の初期化メソッド呼び出しを実行します。このとき、メソッドの命令が個々にステップ実行されることはありません。

   Mobile Edition IDE はメソッドを実行して、コードの次の行に進みます。「ソースエディタ」ウィンドウにその行が示されます。

呼び出しスタックの利用

「デバッガウィンドウ」には「呼び出しスタック」ビューがあります。「呼び出しスタック」ビューはプログラムの現在行を示し、その地点に達するまでの、現在のスレッド内のメソッド呼び出しが階層表示されます。最後に実行されたメソッドは、矢印付きでスタックの先頭に表示され、最初のメソッドはスタックの一番下に表示されます。

たとえば Converter プログラムが Converter クラスの行番号 81 の位置で停止したと仮定します。「デバッガウィンドウ」の「呼び出しスタック」ビューのスタックの先頭には、最後に実行されたメソッドの `myconverter.Converter.converter` が表示されます。またソースエディタには、Converter の行番号 81 で実行が停止したことが示されます。図 2-5 を参照してください。

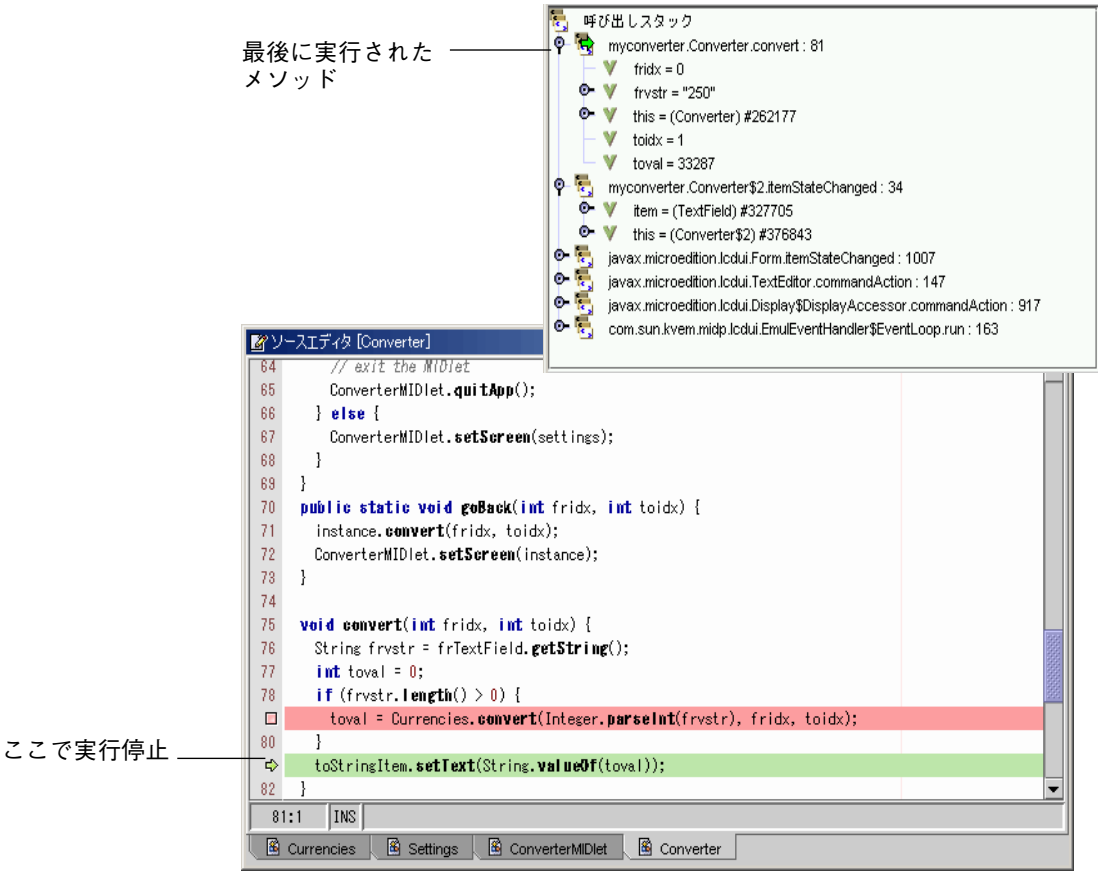
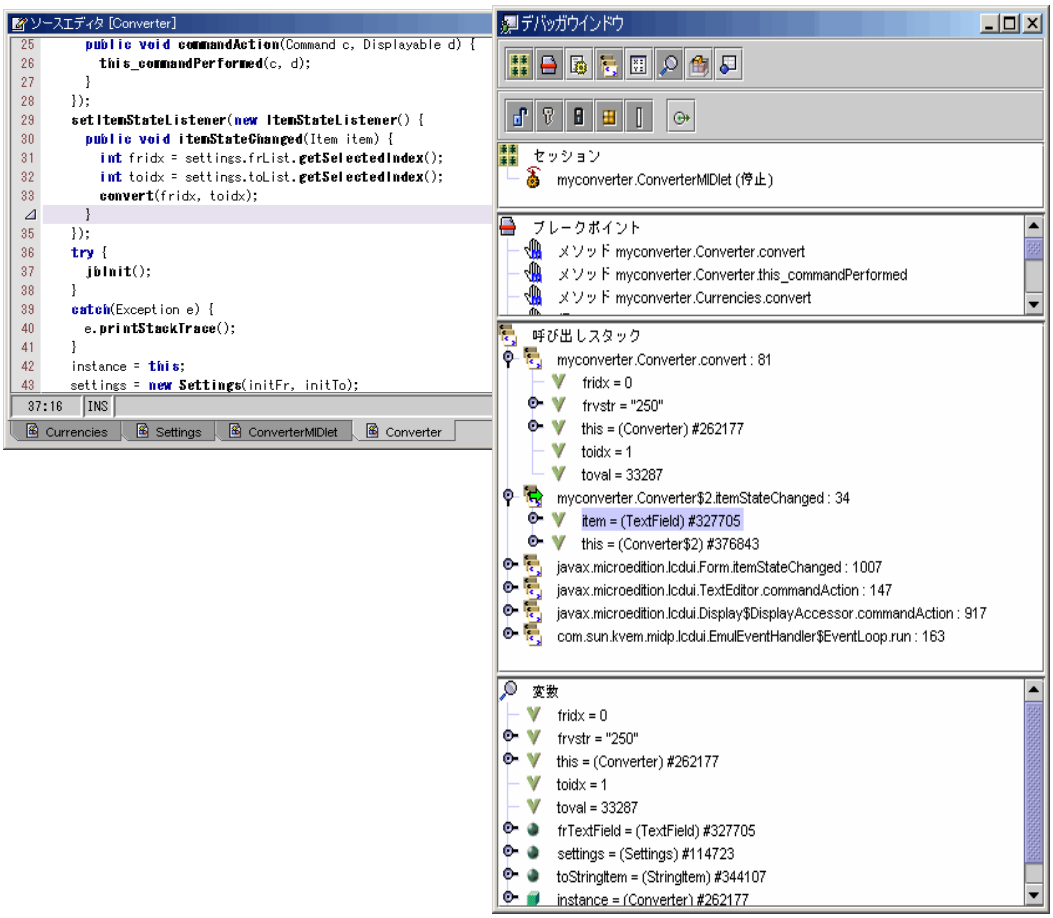


図 2-5　「呼び出しスタック」ビューとソースコード

呼び出しスタックを使用するには、次の操作を行います。

- 「デバッグ」->「呼び出し元メソッドへ」を選択し、呼び出しスタックの下方向 (すなわち、スタックの一番下の最初のメソッドの方向) に移動します

  「呼び出し元メソッドへ」オプションを実行すると、`myconverter.Converter.converter` メソッドを呼び出したメソッドである `myconverter.Converter$2.itemStateChanged` に移動します。呼び出しスタックのポインタはメソッド呼び出し 1 つぶん下に移動し、ソースエディタには、Converter クラスを呼び出した Converter クラスの行番号 34 にマークが表示されます。このとき「呼び出しスタック」ビューから変数の値を調べたり、変更したりすることができます。「呼び出しスタック」ビューには、メソッドに入ったときにスコープ内にあった変数が、その値とともに表示されます。



変数値の変更は、その変更を行うメソッド呼び出しのスコープに限定されます。実行が継続されると、プログラムはその新しい値を使用します。

プログラムの実行の継続

- 停止位置からプログラムの実行を継続するには、「デバッグ」->「継続」を選択します。

  次のブレークポイントに達するか、終了するまでプログラムが実行されます。

プログラムの一時停止

- プログラムの実行を停止して再びデバッグモードに入るには、「デバッグ」->「一時停止」を選択します。

  この機能は、たとえばブレークポイントに達する前にプログラムがユーザーからの入力待ちになっているときなどに役立つことがあります。

デバッグセッションの終了

- デバッグセッションを終了するには、「デバッグ」->「完了」を選択します。

第3章

高度な機能

この章では、以下のような少し高度な機能を取り上げます。

- アプリケーションのプロジェクトを作成する
- Mobile Edition IDE に新しいエミュレータを追加する
- OTA (Over-the-Air) アプリケーションのダウンロードをサポートする
- アプリケーションを配備する

プロジェクトの操作

プロジェクトが明示的に設定されなかった場合、IDE はデフォルトのプロジェクトを使用してすべての作業内容をプロジェクトにまとめます。プロジェクトには、MIDlet アプリケーションの生成に必要なすべてのファイルがまとめられます。ファイルをプロジェクトにまとめることによって、すべてのファイルを 1 つの単位として操作することができます。たとえばプロジェクトをコンパイルするということは、そこに含まれるすべての MIDlet ファイルと Java ソースファイルをコンパイルすることを意味します。

最も大切なことは、プロジェクトには MIDlet の設定もまとめられることです。そうした設定には、コード補完やデフォルトコンパイラ、実行方法、デバッガの設定などがあります。

プロジェクトの作成や管理は、メインウィンドウの「プロジェクト」メニューを使用して行うことができます。

現在のプロジェクトのファイルを表示するには、次の操作を行います。

- 「エクスプローラ」ウィンドウの「プロジェクト」タブをクリックします。

「プロジェクト」タブには、現在のプロジェクトの名前が表示されます。

新規プロジェクトの作成

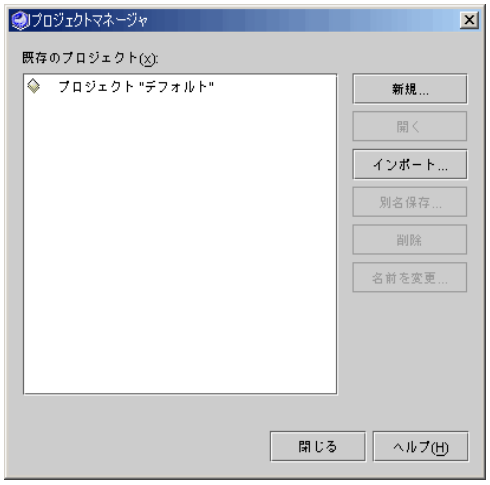
新規プロジェクトを作成するには、次の操作を行います。

1. メインウィンドウで「プロジェクト」->「プロジェクトマネージャ」を選択します。

   「プロジェクトマネージャ」ウィンドウが開きます。

   この「プロジェクトマネージャ」ウィンドウには、既存のプロジェクトが表示されます。(少なくとも 1 つのデフォルトプロジェクトがあり、「プロジェクト " デフォルト "」という名前が付けられています。)「プロジェクトマネージャ」ウィンドウでは、プロジェクトを開いたり、開いていないプロジェクトを削除したり、既存のプロジェクト名を変更したりできます。

2. 「プロジェクトマネージャ」ウィンドウから「新規」を選択します。

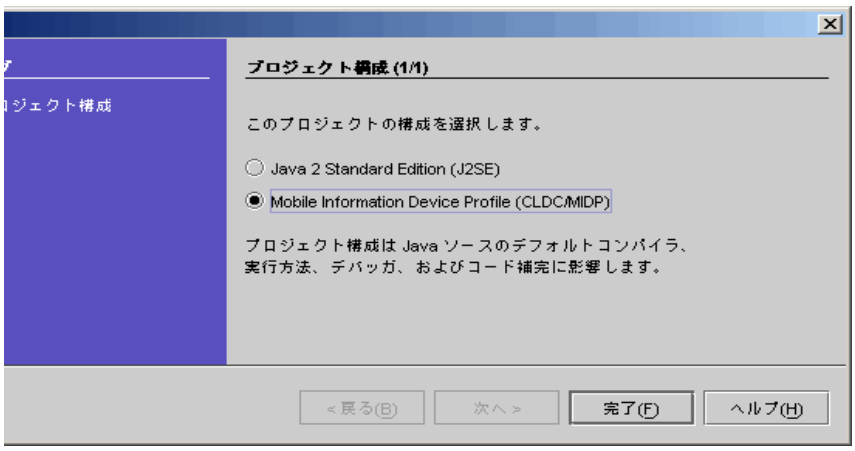


3. 「新規プロジェクトを作成」ダイアログに、新規プロジェクト名として `CurrencyConverter` を入力し、「了解」をクリックします。

   IDE によって CurrencyConverter というプロジェクトが新規作成されます。プロジェクト構成ウィザードが開きます。

4. プロジェクト構成ウィザードで「Mobile Information Device Profile (CLDC/MIDP) 」を選択して、「完了」をクリックします。

   プロジェクトを構成すると、デフォルトのコンパイラと実行方法、デバッガ、Java ソースコード補完が設定されています。各プロジェクトは専用の構成を持つことができます。

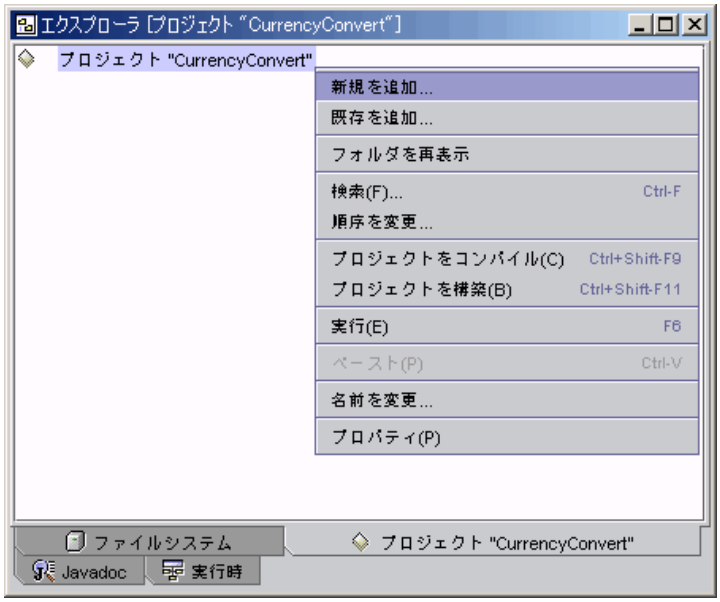


5. 「ファイルシステム」タブを選択します。「ファイルシステム」ノードを右クリックし、「マウント」->「ローカルディレクトリ」を選択して、プロジェクトに含めるソースファイルまたは他のオブジェクトを含むファイルシステム (ディレクトリ) をマウントします。

   このときローカルディレクトリの新規ウィザードが開いて、マウントするファイルシステムを選択することができます。ファイルシステムをマウントして、プロジェクトマネージャがその内容を利用できるようにする必要があります。

6. 「エクスプローラ」ウィンドウの「プロジェクト」タブを選択します。プロジェクト名を右クリックし、「新規を追加」を選択してプロジェクトに新規コンポーネントを追加するか、「既存を追加」を選択して既存のコンポーネントを追加します。

   エクスプローラの「プロジェクト」タブには、現在開いているプロジェクト名が表示されます。プロジェクトを初めて作成した場合、Mobile Edition IDE はそのプロジェクトを開きます。プロジェクトに新規コンポーネントを追加する場合は、最初に、「MIDP」->「MIDlet」や「MIDP」->「MIDletSuite」などの使用可能なテンプレートのリストからテンプレートを選択します。テンプレートを選択すると、そのコンポーネントを作成する手順を案内する適切なウィザードが表示されます。



7. コードをコンパイルまたは構築して、実行します。

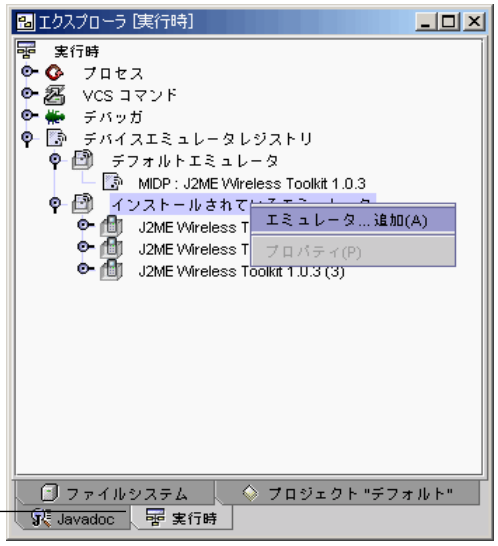
   アプリケーションをコンパイル・構築したり、実行する場合は、プロジェクトのコンテキストメニューを使用します。

エミュレータの設定

Mobile Edition IDE は、携帯電話などの無線機器で使用する MIDP アプリケーションの開発を支援します。対象機器でアプリケーションを実際に実行してみることは、アプリケーションの開発およびテストでは重要です。このため、Mobile Edition IDE はデバイスエミュレータレジストリを管理し、このレジストリからアプリケーションをテストするエミュレータを選択できるようにしています。デバイスエミュレータレジストリはインストール済みの使用可能なエミュレータの一覧であり、現在デフォルトとして設定されているエミュレータがわかるようになっています。

デバイスエミュレータレジストリにエミュレータを追加するには、次の操作を行います。

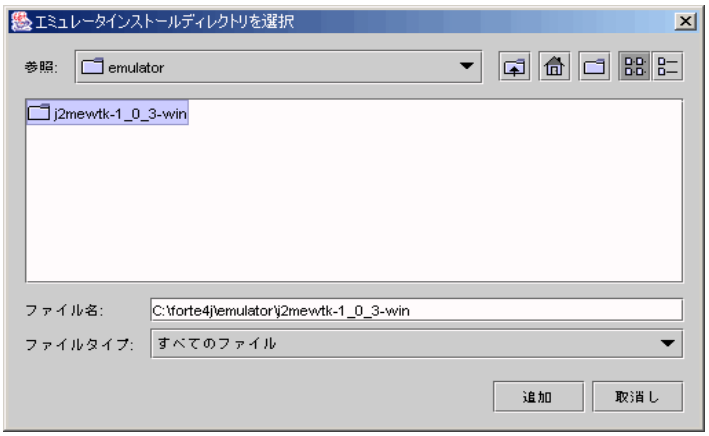
1. エミュレータベンダーの指示に従って、エミュレータをシステムにインストールします。
2. 「エクスプローラ」ウィンドウの「実行時」タブで「デバイスエミュレータレジストリ」ノードを開き、「インストールされているエミュレータ」を右クリックして、「エミュレータ ... 追加」を選択します。



実行時タブ

3. 「エミュレータインストールディレクトリを選択」ダイアログから、インストールしたエミュレータが含まれているディレクトリに移動し、「追加」をクリックします。

   Mobile Edition に用意されている J2ME Wireless Toolkit の場合、ディレクトリは *ffj_install*¥emulator¥j2mewtk-1_0_3-*xxx*. です。



4. エミュレータの追加インスタンスを追加するには、「インストールされているエミュレータ」ノードを開き、インストールされているエミュレータを右クリックして、「エミュレータインスタンス 追加」を選択します。

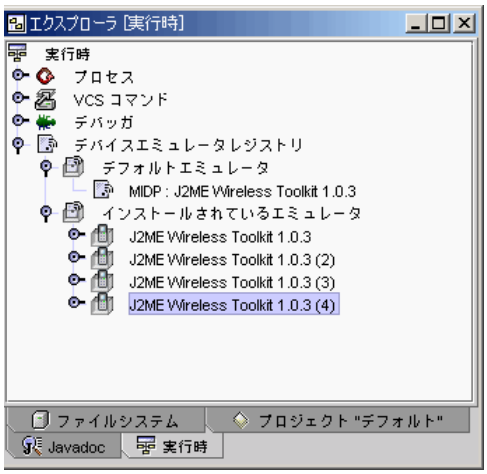
   エミュレータの追加インスタンスを追加することによって、プロパティ、あるいはそのエミュレータインスタンス用に実行される MIDlet を表示するデフォルトのエミュレータデバイスを変更することができます。つまり、こうして追加インスタンスを追加すれば、アプリケーションのテスト条件を変更するたびにデバイスエミュレータレジストリに移動しなくても、MIDlet スイートのプロパティシートの「実行」タブから適切なデバイスエミュレータを選択して、アプリケーションを実行することができます。

5. 作成した 2 つ目のインスタンスを右クリックして「プロパティ」を選択します。「デフォルトデバイス」フィールドをクリックして、コンボボックスから「MinimumPhone」を選択します。

   これで MinimumPhone が、J2ME Wireless Toolkit エミュレータの 2 つ目のインスタンス用のデフォルトデバイスになりました。異なるエミュレータに限らず、同じエミュレータの複数のバージョンをインストールすることもできます。

6. エミュレータのインスタンスをさらに 2 つ追加して、そのインスタンスごとにデフォルトのデバイスを設定します。

   これで、J2ME Wireless Toolkit エミュレータの 3 つ目と 4 つ目のインスタンスを追加したことになります。これらのバージョンにはそれぞれ 3、4 の番号が割り当てられます。



7. 「エクスプローラ」ウィンドウの「ファイルシステム」タブにあるMIDlet スイートノードを右クリックし、「プロパティ」を選択します。

8. 「実行」タブから「エミュレータ」フィールドをクリックします。

   コンボボックスのリストから、MIDlet スイートの実行とテストに使用するデバイスエミュレータを選択することができます。MIDlet スイートを実行して、エミュレータによって結果がどのように変わるか確認してください。

OTA (Over-the-Air) アプリケーションのダウンロードのサポート

OTA (Over-the-Air) とは、ユーザー主導プロビジョニング技術です。この技術の呼び名は、特殊なポータルサイトあるいはアプリケーションサーバーからアプリケーションをダウンロードする、携帯電話など無線機器の機能を表しています。ホストサー

バーやポータルサイトと無線機器との間では、物理的な接続なしにダウンロードが行われます。ユーザーはOTA によって、ほぼあらゆる時間、あらゆる場所から「無線 (over the air) 」で目的のサービスにアクセスすることができます。

Mobile Edition IDE は OTA そのものをサポートしているわけではありませんが、MIDlet スイートに OTA を実現するための手順が用意されています。この手順に従って、サービスを提供するサーバーに MIDlet を配備し、その MIDlet スイートのダウンロードページを作成できます。

OTA ダウンロード機能を実現するには、次の操作を行います。

1. 内部コンパイラの「デバッグ」プロパティを False に設定し、MIDlet コードをデバッグ情報なしでコンパイルします。

   この再コンパイルの目的は、パッケージ化して配備する前に MIDlet コードを最適化することにあります。内部コンパイラの「デバッグ」プロパティを False に設定してソースコードを再コンパイルすることを忘れないでください。内部コンパイラは、「ツール」->「オプション」ウィンドウの「構築」->「コンパイラの種類」ノードにあります。60 ページの「MIDlet スイートの配備」を参照してください。

2. JAR ファイルを再構築します。

   25 ページの「MIDlet スイートのパッケージ化」を参照し、JAR ファイルの再構築を行います。

3. JAD ファイルにサービスを提供する適切な MIME タイプを追加し、 Web サーバーを構成します。

   次の URL を参照し、サーバーに適切な MIME タイプが設定されているかを確認してください。

   `http://`*`host`*`:`*`port`*`/servlet/org.netbeans.modules.j2me.jam.JAMServlet/`*`path_to_jad`*

   たとえば次のように URL を設定します。

   ```
   http://localhost:8083/servlet/org.netbeans.modules.j2me.jam.
   JAMServlet/converter.jad
   ```

   MIDP アプリケーションは、Mobile Edition IDE の内部 HTTP サーバーからダウンロードすることもできます。MIME タイプについての詳細は、OTA の仕様を参照してください。この仕様は、以下のサイトにあります。

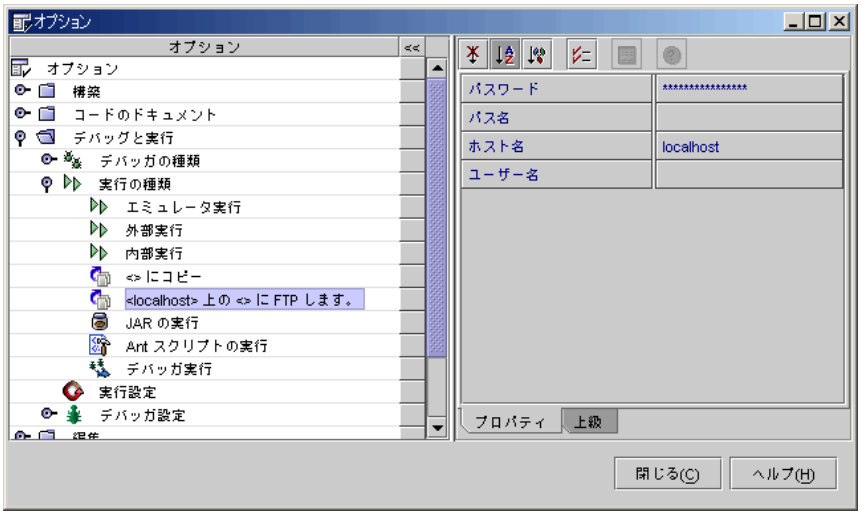
   `http://java.sun.com/products/midp/OTAProvisioning-1.0.pdf`

4. MIDlet スイートのダウンロードページを作成します。

   お好みの HTML ツールでダウンロードページを作成します。

   次に、ファイル転送プロトコル (FTP) を使用して MIDlet スイートをコピーし、別の場所に配備します。つまり、スイートと付属するアプリケーション記述子ファイルを指定した FTP サイトに送信します。FTP で MIDlet スイートを配備するには、まず、実行方法オプションとして FTP のコピーサービスを設定する必要があります。

5. メインウィンドウから「ツール」->「オプション」を選択し、「デバッグと実行」->「実行の種類」ノードを開きます。続いて、コピーの実行方法として「<localhost> 上の <> に FTP します。」を選択し、FTP のプロパティに値を入力します。



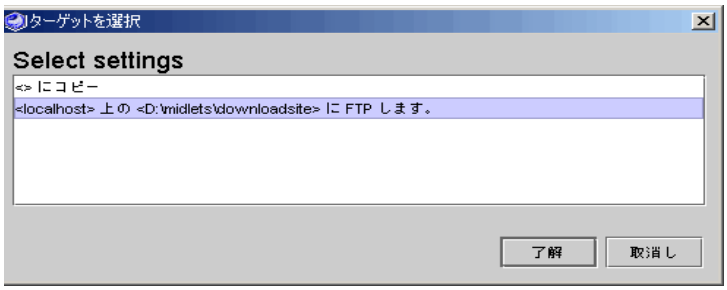
6. 4 つの FTP のプロパティに値を設定します。

   FTP プロパティは以下のように設定してください。

   - ホスト名: ファイルのコピー先の FTP サイトの名前を設定します。
     例: `localhost`
   - パスワード: FTP サイトのパスワードを設定します。
   - パス名: ファイルのコピー先の FTP サイトの場所を設定します。
     例: `D:¥midlets¥downloadsite`
   - ユーザー名: FTP サイトへのアクセスを許可するユーザー名を設定します。
     例: `MobileUser`

   これら 4 つの値を入力したら、「閉じる」をクリックします。

7. アプリケーションを配備するには、「エクスプローラ」ウィンドウで、MIDlet スイートを右クリックします。「ツール」->「ファイルをコピー」を選択し、「ターゲットを選択」ダイアログが開いたら、「<localhost> 上の <D:\midlets\downloadsite> に FTP します。」を選択して、「了解」をクリックします。



指定した FTP サイトにアプリケーションが送信されます。

8. MIDlet アプリケーションを実際のデバイスにダウンロードします。

現在、OTA アプリケーションをエミュレータにダウンロードできません。

MIDlet スイートの配備

アプリケーションを配備する際は、デバッグを無効にしてアプリケーションを再コンパイルします。これにより、最終コードのサイズが最小化されます。通常、この作業はデバッグを完了した後に行います。プログラムのデバッグ方法については、第 2 章を参照してください。

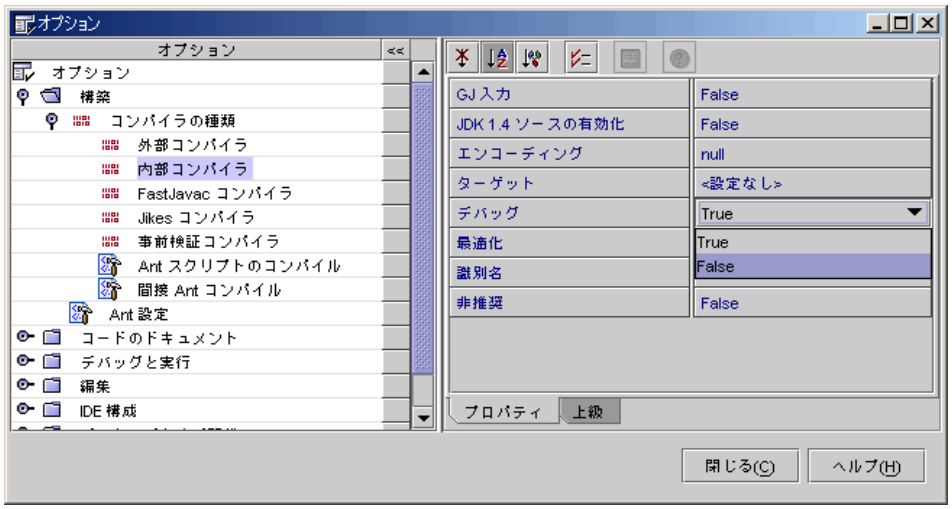
アプリケーションの開発・テスト中は、Mobile Edition IDE の事前検証コンパイラを設定して、その Java コンパイラが内部コンパイラを使用するようにしてください。この設定によって、デバッグ機能を利用できるようになります。アプリケーションのテストが完了したら、内部コンパイラのデバッグ機能を無効にし、コードを再コンパイルします。

デフォルトでは、事前検証コンパイラは内部コンパイラを使用します。これは、プログラムを明示的にデバッグしなかった場合でも、デバッグ機能を無効にしてコードを再コンパイルする必要があることを意味します。

MIDlet スイートを配備するには、次の操作を行います。

1. メインウィンドウから「ツール」->「オプション」を選択し、「オプション」ウィンドウで「構築」->「コンパイラの種類」を開きます。「内部コンパイラ」をクリックしてそのプロパティシートを表示します。

   プロパティシートで「デバッグ」プロパティのコンボボックスをクリックし、その値を「False」に設定してください。



2. すべてのソースコードを再コンパイルして JAR ファイルを更新します。

   ソースコードをコンパイルする前に、メインウィンドウから「構築」->「生成物を削除」または「構築」->「すべての生成物を削除」を選択して、コンパイラによってすべてのクラスが再コンパイルされるようにすることを推奨します。アプリケーションがコンパイルされて、デバッグ情報なしでパッケージ化されます。これでアプリケーションを配備できます。

付録 A

サンプルアプリケーションのソースコード

この付録では、サンプルの ConverterMIDlet アプリケーションを構成する 4 つのファイルのソースコードをまとめています。

- ConverterMIDlet.java
- Converter.java
- Currencies.java
- Settings.java

ConverterMIDlet.java のソース

コード例 A-1 ConverterMIDlet.java

```
package myconverter;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.rms.*;

public class ConverterMIDlet extends MIDlet {
  static  ConverterMIDlet instance;
  private Converter       mainScreen;

  private static String storedDataStr = "FromToCurr";
  private RecordStore   storedData;
```

コード例 A-1　ConverterMIDlet.java (続き)

```
  /**MIDlet を作成 */
  public ConverterMIDlet() {
    this.instance = this;
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }

  /** メインメソッド */
  public void startApp() {
    int fridx = 0;
    int toidx = 1;
    try {
      storedData = RecordStore.openRecordStore(storedDataStr,
true);
      if (storedData.getNumRecords() > 0) {
        byte[] data = storedData.getRecord(1);
        fridx = (int)data[0];
        toidx = (int)data[1];
      }
    } catch (RecordStoreException e) {
    }
    mainScreen = new Converter(fridx, toidx);
    Display.getDisplay(this).setCurrent(mainScreen);
  }

  /**MIDlet の一時停止処理 */
  public void pauseApp() {
  }

  /**MIDlet の破壊処理 */
  public void destroyApp(boolean unconditional) {
    try {
      byte[] data = new byte[2];
      data[0] =
(byte)mainScreen.settings.frList.getSelectedIndex();
```

コード例 A-1　ConverterMIDlet.java (続き)

```
      data[1] =
(byte)mainScreen.settings.toList.getSelectedIndex();
      if (storedData.getNumRecords() > 0)
        storedData.setRecord(1, data, 0, 2);
      else
        storedData.addRecord(data, 0, 2);
    } catch (Exception e) {
    }
  }

  /**MIDlet を終了 */
  public static void quitApp() {
    instance.destroyApp(true);
    instance.notifyDestroyed();
    instance = null;
  }
  public static void setScreen(Displayable d) {
    Display.getDisplay(instance).setCurrent(d);
  }

  private void jbInit() throws Exception {
  }
}
```

Converter.java のソース

コード例 A-2　Converter.java

```
package converter;

import javax.microedition.lcdui.*;

public class Converter extends Form {
  static Converter  instance = null;
  TextField  frTextField = new TextField("", "", 1, 1);
  StringItem toStringItem = new StringItem("", "");

  Settings  settings  = null;
```

コード例 A-2　Converter.java (続き)

```
  /**Displayable を作成 */
  public Converter(int initFr, int initTo) {
    super("Converter");
    // この Displayable がコマンドイベントを待機するように設定
    setCommandListener(new CommandListener()  {
      public void commandAction(Command c, Displayable d) {
        this_commandPerformed(c, d);
      }
    });
    setItemStateListener(new ItemStateListener() {
      public void itemStateChanged(Item item) {
        int fridx = settings.frList.getSelectedIndex();
        int toidx = settings.toList.getSelectedIndex();
        convert(fridx, toidx);
      }
    });
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
    instance = this;
    settings = new Settings(initFr, initTo);
    changeFrom(initFr);
    changeTo(initTo);
  }

  /**Component の初期化 */
  private void jbInit() throws Exception  {
    toStringItem.setLabel("(to)");
    frTextField.setLabel("(from)");
    frTextField.setConstraints(TextField.NUMERIC);
    frTextField.setMaxSize(10);
    // Exit コマンドを追加
    addCommand(new Command("Exit", Command.EXIT, 1));
    addCommand(new Command("Settings", Command.SCREEN, 1));
    this.append(frTextField);
    this.append(toStringItem);
  }
```

コード例 A-2　Converter.java (続き)

```
  /** コマンドイベントの処理 */
  public void this_commandPerformed(Command c, Displayable d) {
    if (c.getCommandType() == Command.EXIT) {
      // MIDlet を終了
      ConverterMIDlet.quitApp();
    } else {
      ConverterMIDlet.setScreen(settings);
    }
  }
  public static void goBack(int fridx, int toidx) {
    instance.convert(fridx, toidx);
    ConverterMIDlet.setScreen(instance);
  }

  void convert(int fridx, int toidx) {
    String frvstr = frTextField.getString();
    int toval = 0;
    if (frvstr.length() > 0) {
      toval = Currencies.convert(
Integer.parseInt(frvstr), fridx, toidx);
    }
    toStringItem.setText(String.valueOf(toval));
  }
  static void changeFrom(int newidx) {
    instance.frTextField.setLabel(Currencies.currencies[newidx]);
  }
  static void changeTo(int newidx) {

instance.toStringItem.setLabel(Currencies.currencies[newidx]);
  }
```

Currencies.java のソース

コード例 A-3 Currencies.java

```
package converter;

import javax.microedition.lcdui.*;

public class Currencies extends List {
  static String[] currencies = {
    new String("US $"),      // US ドル
    new String("JP \u00a5"), // 円
    new String("Euro")
  };
  static int[][] ratesTop = {
    {  100, 13315,   112},
    {  100,   100,   100},
    {  100, 11830,   100}
  };
  static int[][] ratesBot = {
    {  100,   100,   100},
    {13315,   100, 11830},
    {  112,   100,   100}
  };
  static int  TYPE_FROM  = 0,
              TYPE_TO    = 1;
  private int type;

  /**Displayable を作成 */
  public Currencies(String name, int type, int initidx) {
    super(name + "Currencies", List.EXCLUSIVE);
    // この Displayable がコマンドイベントを待機するように設定
    setCommandListener(new CommandListener()  {
      public void commandAction(Command c, Displayable d) {
        this_commandPerformed(c, d);
      }
    });
    try {
      jbInit();
    }
```

コード例 A-3　Currencies.java (続き)

```
    catch(Exception e) {
      e.printStackTrace();
    }
    this.type = type;
    setSelectedIndex(initidx, true);
  }

  /** コンポーネントの初期化 */
  private void jbInit() throws Exception  {
    for (int i = 0; i < currencies.length; i++) {
      this.append(currencies[i], null);
    }
    //addCommand(new Command("Back", Command.BACK, 2));
    addCommand(new Command("Save", Command.OK,   1));
  }

  /** コマンドイベントの処理 */
  public void this_commandPerformed(Command c, Displayable d) {
    // コマンドイベント処理をここに追加
    if (c.getCommandType() == Command.OK) {
      // 設定を保存して mainScreen を更新
      if (type == TYPE_FROM)
        Converter.changeFrom(getSelectedIndex());
      else
        Converter.changeTo(getSelectedIndex());
    }
    // 必ず Settings 画面に戻る
    Settings.goBack();
  }

  public static int convert(int frval, int fridx, int toidx) {
    return (frval * ratesTop[fridx][toidx]) /
ratesBot[fridx][toidx];
  }
}
```

Settings.java のソース

コード例 A-4 Settings.java

```
package converter;

import javax.microedition.lcdui.*;

public class Settings extends List {
  private static Settings instance = null;
  Currencies frList = null;
  Currencies toList = null;

  /**Displayable を作成 */
  public Settings(int initFr, int initTo) {
    super("Settings", List.IMPLICIT);
    // この Displayable がコマンドイベントを待機するように設定
    setCommandListener(new CommandListener()  {
      public void commandAction(Command c, Displayable d) {
        this_commandPerformed(c, d);
      }
    });
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
    instance = this;
   frList = new Currencies("From", Currencies.TYPE_FROM, initFr);
    toList = new Currencies("To",   Currencies.TYPE_TO,   initTo);
  }

  /** コンポーネントの初期化 */
  private void jbInit() throws Exception  {
    append("1. From Currencies", null);
    append("2. To Currencies", null);
    addCommand(new Command("Back", Command.BACK, 2));
  }
```

コード例 A-4　Settings.java (続き)

```
  /** コマンドイベントの処理 */
  public void this_commandPerformed(Command c, Displayable d) {
    // コマンドイベント処理をここに追加
    if (c.getCommandType() == Command.BACK) {
      Converter.goBack(frList.getSelectedIndex(),
toList.getSelectedIndex());
    } else {
        if (getSelectedIndex() == 0)
            ConverterMIDlet.setScreen(frList);
        else
            ConverterMIDlet.setScreen(toList);
    }
  }

  public static void goBack() {
    ConverterMIDlet.setScreen(instance);
  }
}
```